Department of Computer Science
Series of Publications A
Report A-2013-2

Probabilistic, Information-Theoretic Models for Etymological Alignment

Hannes Wettig

To be presented, with permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XIV of the University Main Building, on February 9th 2013 at 12 o'clock noon.

University of Helsinki
Finland

Supervisor
Petri Myllymäki and Roman Yangarber,
University of Helsinki, Finland

Pre-examiners
Steven de Rooij,
Centrum Wiskunde & Informatica (CWI), Amsterdam, Netherlands
Timo Honkela,
Aalto University School of Science, Finland

Opponent
Erik Aurell,
Skolan för dataveteskap och kommunikation (KTH), Sweden
and Helsinki University of Technology (TKK), Finland

Custos
Petri Myllymäki
University of Helsinki, Finland

Contact information

Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

Email address: postmaster@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/
Telephone: +358 9 1911, telefax: +358 9 191 51120

ISSN 1238-8645
ISBN 978-952-8588-8 (paperback)
ISBN 978-952-8589-5 (PDF)
Computing Reviews (1998) Classification: G.3, H.1.1, I.2.6, I.2.7
Helsinki 2013
Helsinki University Print

Probabilistic, Information-Theoretic Models for Etymological Alignment

Hannes Wettig

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
wettig@hiit.fi

PhD Thesis, Series of Publications A, Report A-2013-2
Helsinki, January 2013, 202 pages
ISSN 1238-8645
ISBN 978-952-8588-8 (paperback)
ISBN 978-952-8589-5 (PDF)

Abstract

This thesis starts out by reviewing Bayesian reasoning and Bayesian network models. We present results related to discriminative learning of Bayesian network parameters. Along the way, we explicitly identify a number of problems arising in Bayesian model class selection. This leads us to information theory and, more specifically, the minimum description length (MDL) principle. We look at its theoretic foundations and practical implications. The MDL approach provides elegant solutions for the problem of model class selection and enables us to objectively compare any set of models, regardless of their parametric structure. Finally, we apply these methods to problems arising in computational etymology. We develop model families for the task of sound-by-sound alignment across kindred languages. Fed with linguistic data in the form of cognate sets, our methods provide information about the correspondence of sounds, as well as the history and ancestral structure of a language family. As a running example we take the family of Uralic languages.

Computing Reviews (1998) Categories and Subject Descriptors:
G.3 [Mathematics of Computing] Probability and Statistics.
H.1.1 [Information Systems] Models and Principles — Systems and Information Theory.
I.2.6 [Computing Methodologies] Artificial Intelligence — Learning.

I.2.7 [Computing Methodologies] Artificial Intelligence — Natural Language Processing.

General Terms:
Data Analysis, Probabilistic Modeling, Information Theory, Natural Language Processing.

Additional Key Words and Phrases:
Bayesian Networks, Logistic Regression, Minimum Description Length Principle, Kolmogorov Complexity, Normalized Maximum Likelihood, Etymology, Language Alignment, Phylogenetic Trees.

Foreword

Contents

List of Included Publications

Publication I *H.Wettig, P. Grünwald, T.Roos, P. Myllymäki, H.Tirri:*
When Discriminative Learning of Bayesian Network Parameters Is Easy
18th International Joint Conference on Artificial Intelligence (IJCAI '03).

Publication II *H. Wettig, P. Kontkanen and P. Myllymäki:*
Calculating the Normalized Maximum Likelihood Distribution for Bayesian Forests
IADIS International Journal on Computer Science and Information Systems 2 (2007).

Publication III *H. Wettig and R. Yangarber:*
Probabilistic Models for Aligning Etymological Data
18th Nordic Conference of Computational Linguistics (NODALIDA 2011).

Publication IV *H. Wettig, S. Hiltunen and R. Yangarber:*
MDL-based Models for Aligning Etymological Data
Conf. on Recent Advances in Natural Language Processing (RANLP 2011).

Publication V *H. Wettig, K. Reshetnikov and R. Yangarber:*
Using context and phonetic features in models of etymological sound change
EACL Joint Workshop of LINGVIS & UNCLH 2012.

Publication VI *H. Wettig, J. Nouri, K. Reshetnikov and R. Yangarber*
Information-Theoretic Methods for Analysis and Inference in Etymology
Fifth Workshop on Information Theoretic Methods in Science and Engineering (WITMSE 2012).

For detailed references, summaries and contributions of the current author see pages 131–133.

List of Abbreviations

AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
DAG	Directed Acyclic Graph
ESS	Equivalent Sample Size; *the parameter prior for Bayesian network models, which spreads pseudo-counts evenly across the data space*
FAN	Forest-Augmented Naive Bayes
fNML	factorized Normalized Maximum Likelihood
i.i.d.	independent and identically distributed; *the assumption, that data come from a stationary distribution, from which we repeatedly draw independent samples*
KL	KL-divergence: Kullback-Leibler divergence
LR	Logistic Regression
MAP	Maximum A Posteriori; *the model of maximal posterior probability*
MB	Markov Blanket
MDL	Minimum Description Length
MarLi	Marginal Likelihood

ML	Maximum Likelihood
NB	Naive Bayes
NCD	Normalized Compression Distance
NED	Normalized Edit Distance
nfp($\mathcal{C}$)	Number of Free Parameters (of a model class $\mathcal{C}$)
NML	Normalized Maximum Likelihood
PCB	Prefix CodeBook
SCB	Suffix CodeBook
sNML	sequential Normalized Maximum Likelihood
TAN	Tree-Augmented Naive Bayes
UPGMA	Unweighted Pair Group Method with Arithmetic Mean
UTM	Universal Turing Machine
XIC	Information Criteria *such as AIC, BIC and so on*

Chapter 1

Introduction

"Essentially, the world is probabilistic."

'Data modeling' is a term which, in all its generality, applies to the subject of this thesis. It means that, given some data, we learn a machine that describes the data in a generalizing manner. This machine, or model, may then reveal properties of the data not immediately evident and can be used for inference—prediction of entities that are not given, but are assumed to come from the same generating process.

Another—even more general—term frequently used in this context is 'data mining'. Nowadays, data can be collected automatically, cheaply and therefore in large amounts, hoping it will be useful in some way, some day. This calls for methods to process this data. Wikipedia states that "the overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use". This is exactly what we will be doing with etymological data in Chapter 4 of this work, but of course this requires specification. Data mining as such simply means "to do something interesting with data".

Data modeling, as opposed to data mining, includes the notion of a model, a machine to describe the data. Hence, in data modeling, we need to have an idea of what it is that we expect—or hope—to find in the data. Instead of simply scanning for conspicuous statistics, we have a model to describe the *entire* data, a simulated process to encode or generate it. The choice of an appropriate model is naturally guided by the data at hand, but also requires some initial understanding of the process that has generated the data. For any given data, multiple such choices may seem reasonable, which calls for an objective means of model (class) selection.

All interesting modeling problems involve uncertainty. It is of no prac-

tical relevance, whether the data truly come from a random process, or whether the model itself introduces uncertainty through generalization. In fact, it is a philosophical question whether true randomness exists in the first place. But whether a process is random by nature or whether we are simply unaware of the underlying determinism, in either case we need to deal with uncertainty in modeling it.

Mathematical probability is a natural means to describe—and process—this uncertainty. Our approach will be, still very generally speaking, that of probabilistic data modeling. Although the term 'probability' is being used in many different ways, all of them describe a measure of belief and uncertainty. Subjectivists, for instance, use probabilities simply to describe their personal, subjective view on things. Frequentists in turn regard probabilities as relative frequencies of repeatable random events. Bayesians combine the two by first defining a *prior*, a subjective view on what to expect before observing any data, and then augmenting it with the observed data frequencies to form the *posterior*. The posterior is the (Bayesian) belief of what the data-generating distribution most probably looks like, after having seen the data. This distribution can then be used to predict unobserved data entities, such as future data. The posterior depends on both the initial belief as described by the prior, and the given training data. In information theory, the probability of some data is considered to be two-to-the-minus number of bits needed to describe that data unambiguously, possibly subject to normalization.

We consider classes of *parametric* models. A *model class* $\mathcal{C}$ is thus a set of models (probability distributions or densities) sharing the same parametric form, a model $\mathcal{M} = \mathcal{C}(\boldsymbol{\Theta})$ is defined by the model class $\mathcal{C}$ it belongs to, together with a parameter vector $\boldsymbol{\Theta}$ specifying a distribution (or density) from this class.

In model class selection problems we also speak of a *model family* $\mathcal{F}$. By this we mean a collection of model classes, from which we want to choose a suitable one, guided by the given data. For example, we may start out with the model family of all polynomials of finite degree. A model class $\mathcal{C}_n$ would then include all polynomials of given degree n and a model would become a polynomial $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ with given coefficients a_i. But a model family need not be nested like this. It may also, for instance, be the set of possible Bayesian networks—directed acyclic graphs on a given set of variables that encode independence assumptions of the corresponding models. More generally, a model family may be any collection of model classes, which can be of numerous types and structures.

How is the model in the above example probabilistic? In general, to fit

data of size n exactly we need a polynomial of degree $n-1$, therefore there will be one parameter (coefficient) for each data item. This leads to sharply oscillating functions which are not well-suited to make predictions about areas inbetween data points. This can easily be seen by removing some data point and fitting a polynomial of degree $n-2$. Typically, the value of this polynomial at the position of the removed data will greatly differ from the held-out value. Similarly, when fitting a polynomial of degree $n-1$, the coefficient values are highly sensitive to noise, such as measurement inaccuracy. Since we have not actually simplified the data, but merely rewrote the n data points as n coefficients, the model does not generalize.

For this reason we hardly ever aim to model the data exactly, overly complex models generalize poorly. We need to choose a model class that truly simplifies the data, which typically means fitting it imperfectly. We then determine the model parameters (here: the coefficients of the polynomial) according to some objective function, e.g. goodness of fit. For polynomial fit, we often minimize the sum of squares of the residuals. Implicitly, we thereby assume the data to be normally distributed around the point indicated by the polynomial. Explicitly or implicitly, the model $\mathcal{M} = \mathcal{C}_n(\Theta = (\mathbf{a}, \sigma^2))$ includes a variance parameter σ^2, to define a conditional probability distribution $P(y|x, \mathcal{M})$, a normal distribution of variance σ^2 with its mean given by the polynomial with coefficients $\mathbf{a}$ evaluated at x.

Comfortably, all data appearing in the course of this work are of discrete nature, all distributions will be multinomial. This simplifies many things. The parameters of the Bayesian classifiers of Chapter 2 are specified by local counts, priors take on a similar form, and the parametric complexity of a class of Bayesian forest models in Chapter 3 is guaranteed to be finite. Probabilities *sum*—rather than *integrate*—to one and form a distribution directly, without the need to specify the precision to which the data is being modeled. The only probability *densities* that appear are those that apply to (continuous) model parameters.

The course of this work roughly follows the topics of the six included publications, referred to as Publications I–VI. However, we draw a larger picture, filling in gaps between the publications and giving motivation step-by-step. This thesis tells a story, a thread of the author's work in research, of which the attached publications only represent some aspects. The body of this thesis is meant to be as intuitive—easy to read—as possible, while still making all the points that seem essential and explaining all concepts that are relevant from the point of view of understanding the contents of this work. There is a tradeoff in writing, between complexity and being

comprehensive, much like the tradeoff between complexity and data fit in the modeling and coding problems appearing in this work. But in writing, complexity is not measured in number of free parameters, nor in number of bits or words, but more subjectively by how well the author's wife manages to follow.

Chapter 2 introduces the reader to Bayesian reasoning and the related topic of Bayesian network models. This probabilistic framework can be seen as the foundation of modern probability theory and is appealing in many ways. As it separates subjective beliefs—the prior—from learned probability ratios in an explicit way, the data analyst is forced to formulate his or her expectations clearly.

There are also some drawbacks to this approach, which we explicitly identify. For one, Bayesian network models are not well-suited for supervised, discriminative learning tasks such as classification. Publication I gives theoretical evidence, and Section 2.7 provides empirical backup for this claim.

Secondly, the formulation of a reasonable prior can be quite challenging indeed, as subjective belief hardly ever translates into a prior (on the model parameters) directly. When we fix a model class that is simple with respect to the data size, then in many cases this prior has little influence on the posterior, as the sheer amount of data causes the learned probability ratio to outweigh it. However, the situation becomes very different when we want to perform model class *selection*. As the complexity of the most suitable model class can be seen to be a function of the data size, there seems to never be 'a sufficient amount' of data available. It has been shown that in many cases the choice of the most suitable model class by Bayesian means *greatly* depends on the chosen prior [Silander 2007].

Finally, model class selection not only depends on the prior chosen for the model parameters, but we also need to formulate a prior distribution over the model classes under consideration. As rule of thumb, more complex models should be less likely a priori, in order to prevent the learning algorithm from overfitting the training data, which would result in poor generalization capability.

The standard Bayesian approach to model class selection is the use of *Bayes factors* [Berger 1985], which enable us to compare model classes according to the probability they assign to the given data. In hypothesis testing these are simply likelihood ratios, the Bayes factor tells how much more likely a hypothesis has become by influence of the observed data. This does not solve the problem of defining a prior distribution over the hypotheses to choose from, but it does separate this issue from the in-

ference problem. But for the model *class* selection problem it gets more complicated. The Bayes factor is now the average probability assigned to the data by all models in the class, a weighted sum or integral over the respective parameter spaces. Averaging requires a (parameter) prior to average with respect to, which has strong influence on the selection that is made. There is some sort of automatic complexity penalty built in to this approach, as a larger (e.g. higher-dimensional) parameter space decreases the likelihood average, given that its mass is concentrated in a small area. In practice, however, this alone cannot entirely prevent overfitting (poor generalization).

Another way of dealing with the situation in which we choose among model classes of differing complexity is to add a complexity penalty term to the objective. This leads to the so-called information criteria, such as the Akaike Information criterion (AIC, [Akaike 1974]), the Bayesian Information Criterion (BIC, [Schwarz 1978]) and so on. These penalty terms however, typically amount to counting the free parameters of a model, preceded by some weight. This is not necessarily a good measure of complexity, especially when comparing model classes of different nature.

An entirely different solution to these problems offers the Minimum description Length (MDL) principle, which brings us to Chapter 3. We employ Kraft's inequality to regard the probability of any data as the two-to-the-minus number of bits needed to decodably encode it—the shorter the code, the higher the probability. This information theoretic paradigm has its roots in Kolmogorov theory [Kolmogorov 1965, Li 1997]. Kolmogorov Complexity—the length of the shortest input to a universal Turing machine to produce a given string and halt—is, in a very specific theoretic way, the optimal way to define this codelength, which then translates into probabilities. Unfortunately, it has been proven to be incomputable [Li 1997].

MDL methods computably approximate the Kolmogorov complexity. The simplest way of achieving this are the so-called two-part codes. We first encode the model we want to use, and subsequently the data by means of this model. Encoding in this context simply means describing in a compact manner such that the original—model or data—can be recovered from this description. While the length of the first part of this description can also be seen as a complexity penalty term much like that of the information criteria mentioned above, in most cases this 'prior' can be formulated in an easy, natural and intuitive way. At the same time, the decodability requirement acts as a constant sanity check, making the researcher's life much easier.

But often we can do better than to use two-part codes. Normalized Maximum Likelihood (NML, [Shtarkov 1987, Rissanen 1987]) is a one-part

code, defining a distribution over all possible data of given structure and size, with respect to a chosen model class directly. It does not involve any prior or model codelength and can—where it exists and computation is feasible—be used to compare model classes of utterly different structures. NML minimizes the worst-case *regret*, the additional description length as compared to the best model in the class that we could have chosen only with hindsight. Under the assumption that the data come from a universal distribution in the Kolmogorov sense as discussed in Section 3.3, this implies average-case minimal description length. Of course, such assumption may well be questioned. But in most cases, assuming a universal distribution is closer to the truth than any prior distribution we are able to guess (and formulate).

The fundamental problems with NML are issues of its efficient computability. While two-part codes are—in many cases—easy to compute, and Kolmogorov Complexity cannot be computed at all, NML lies somewhere inbetween. It is superior to, but harder to compute than two-part codes, and inferior to Kolmogorov Complexity but in exchange computable. For some simple model classes it is also *efficiently* computable. Publication II investigates efficient computation of the NML distribution for a family of Bayesian network models, the so-called Bayesian forests. Since the NML distribution has many desirable properties, we want to be able to employ it in as many cases as possible, which makes NML computability issues an interesting topic for the scientific community.

MDL, both in the form of a two-part code and using NML, has been successfully applied to a wide range of practical applications. These include—but are not limited to—histogram density estimation [Kontkanen 2007b], image denoising [Roos 2005a], clustering [Kontkanen 2006] and DNA sequence compression [Korodi 2005].

In the work presented in this thesis, we have added etymology to this list, the study of the history of words. Specifically, we have developed models for the problem of etymological *alignment*, which we use to investigate processes of phonetic sound change in natural languages. Probabilistic and information-theoretic methods have previously been applied in many areas of linguistics, including morphology [Creutz 2007], syntax [Stolcke 1994] and topic modeling [Blei 2003]. In recent years we have also seen a new interest in computational historical linguistics, as phylogenetic methods have been transferred from evolutionary biology onto this field [Greenhill 2009]. Our MDL approach models etymological data on the level of single sounds, but may also be used to infer language phylogenies.

Chapter 4, as well as the attached Publications III–VI, present MDL

methods for the analysis of etymological data, which comes to us in the form of *cognate sets*, collections of related words from a family of kindred natural languages. Our methods can be applied to any such data. As a running example in this work we use the StarLing Data on the Uralic language family. While Chapters 2 and 3 give us theoretic foundation for these methods, Chapter 4 is the application part of this thesis.

Being interested in the changing of sounds across languages, we must align words from different languages to obtain information on which symbols or sounds do in fact correspond. Our models provide such alignment in an automated fashion and generate rules of phonetic change that are valid throughout a set of languages. We also use the models and induced alignments to build phylogenetic trees, depicting the inferred history of language separation within the family. This approach is not only an interesting, novel application of MDL, but also offers new insight to linguists working in the field. The corresponding software is currently being made publicly available for anyone to use on their favourite language family and will appear on the project homepage at `etymon.cs.helsinki.fi`.

Chapter 5 provides a summary of this work, draws conclusions and discusses directions of current and future research. Following the references, there is a brief summary of the included publications, indicating the contributions of the current author. Reprints of these publications are to be found at the very end of this book.

Chapter 2

Bayesian Reasoning

"All models are wrong, but some are Bayesian."

Bayesian reasoning has got its name from the English mathematician (and Presbyterian minister) Thomas Bayes (1701–1761), who formulated what was to become known as **Bayes' Rule** as a solution to a problem of 'inverse probability'. **Bayes' Theorem** in turn, a generalized version of this result, is being accounted to Pierre-Simon Laplace (1749–1827). But it was not until about 1950 that the term **Bayesian** has come into use, and only in the 1980s have Bayesian methods begun to spread. There are a number of good books on the subject, such as [Berger 1985] and [Bernardo 1994], to name only a couple.

2.1 Bayes' Rule

Bayes' Rule is the central foundation of Bayesian reasoning. It is a direct consequence of the **chain rule of probabilities** which states that the **joint** probability of a set of random variables can be written as a chain of **conditional** probabilities.

Rule 1 (Chain Rule) *For a set* $\mathbf{A} = \{A_1, \ldots, A_M\}$ *of random variables* A_i*, their joint probability can be rewritten as*

$$P(\mathbf{A}) = P(A_1)P(A_2|A_1)\cdots P(A_M|A_1,\ldots,A_{M-1}) = \prod_{i=1}^{M} P(A_i|A_1,\ldots,A_{i-1}). \quad (2.1)$$

The same is true for any set $\mathbf{A} = \{A_1, \ldots, A_M\}$ *of* events *with corresponding probabilities* $P(A_i)$.

Throughout this thesis we will use boldfaced symbols to denote sets, vectors or matrices and thereby visually separate them from single numbers or entities.

Since $\mathbf{A}$ is an unordered set, we are free to renumber the A_i, change the ordering, and thus for random variables A and B we can write

$$P(A, B) = P(A)P(B|A) = P(B)P(A|B). \tag{2.2}$$

In MDL, where we **encode** events (or random variables) A and B—or a larger set $\mathbf{A}$—this simply means that we may do so in any order we choose. Bayesians use the above to **invert** probabilities. This is Bayes' Rule, which is most frequently encountered in the following shape.

Rule 2 (Bayes' Rule) *For any event* A*, and an event* B *with non-zero probability* $P(B)$*, we have*

$$P(A|B) = \frac{P(B|A)}{P(B)}P(A). \tag{2.3}$$

This observation can be utilized in various ways. For one, it couples the probability of a model $\mathcal{M}$ given data $\mathbf{D}$ to its reverse—the likelihood of $\mathbf{D}$ under $\mathcal{M}$:

$$P(\mathcal{M}|\mathbf{D}) = \frac{P(\mathbf{D}|\mathcal{M})}{P(\mathbf{D})}p(\mathcal{M}), \tag{2.4}$$

and further, as $P(\mathbf{D})$ is independent of $\mathcal{M}$,

$$P(\mathcal{M}|\mathbf{D}) \propto P(\mathbf{D}|\mathcal{M})p(\mathcal{M}). \tag{2.5}$$

That is, the conditional probability of model $\mathcal{M}$ given data $\mathbf{D}$ is proportional to the likelihood of $\mathbf{D}$ under $\mathcal{M}$ times the **prior** probability of $\mathcal{M}$, which we choose to visually separate from all other probabilities by using a lower case p. An easy—but problematic—choice for this prior is the so-called **uniform prior** $p(\mathcal{M}) \propto \mathbf{1}$. In case there either are no known preferences on the set of models $\mathcal{M} \in \mathcal{C}$, or these preferences cannot be formulated as a prior distribution, this is sometimes seen as the non-informative

choice, as it means 'to me any model is—a priori—as good as any other'. Under the uniform prior (2.5) simplifies to

$$P(\mathcal{M}|\mathbf{D}) \propto P(\mathbf{D}|\mathcal{M}), \tag{2.6}$$

that is, the probability of a model $\mathcal{M}$ given data $\mathbf{D}$ is directly proportional to the probability it assigns to that data.

But the models of a class are indexed by a set of parameters, and thus the chosen prior is also a distribution over the parameters of a model. Therefore, '*the* uniform prior' can be uniform only with respect to a chosen parametrization. A solution to this problem was offered by [Jeffreys 1946], and *Jeffreys' prior* is often regarded to be truly non-informative. In any case, in Bayesian reasoning there is no way around the formulation of a prior, for without it there can be no **posterior**, the probability of a model $\mathcal{M}$ given data $\mathbf{D}$.

It is important to note that the prior has to be given *before* any data has been observed. The frequently used term 'data prior' is a contradiction in terms; using the same data to formulate a prior and to calculate the posterior from it is circular reasoning.

With a uniform prior, the posterior is proportional to the likelihood of $\mathbf{D}$ under $\mathcal{M}$:

$$P(\mathcal{M}|\mathbf{D}) \propto P(\mathbf{D}|\mathcal{M}). \tag{2.7}$$

So what does this mean? Under the assumption that our data come from a distribution which lies within our chosen model class $\mathcal{C}$, when we have specified a prior distribution over all models $\mathcal{M} \in \mathcal{C}$, this gives us a posterior distribution over all models, consisting of probabilities for each $\mathcal{M}$ to be the one that generated $\mathbf{D}$. This posterior combines our prior belief with the evidence that the data provide. We play the so-called **prior-to-posterior game**, where the data likelihood transforms the prior into a posterior distribution over the models in the class under consideration.

Under the frequentist assumption of independent, identically distributed (i.i.d.) data from a random source, it can be shown that in this way—given that our prior assigns non-zero probability to the correct model—we will, with increasing data size, eventually find it, or approximate it to arbitrary precision. Moreover, if the generating distribution lies outside of the model class under consideration, we still minimize the **Kullback-Leibler Divergence** between the true and the estimated distributions in the limit [Gelman 1995]. In both cases, the actual prior chosen is irrelevant, as the data likelihood overrides it when we throw an unlimited amount of data at the model.

However, it is questionable whether a 'true distribution' needs to exist at all. On one occasion the author had the pleasure to witness, Jorma Rissanen illustrated this point by drawing a number of chalk dots on the blackboard, then asking the audience which distribution they assessed these had come from. Furthermore, in-the-limit results are of little use in practical applications with limited data availability. They can act as a sanity check, as any method should at least consistent, i.e., find the optimal model in case data availability is not an issue. But for data of limited size the prior can play an important role, and its formulation is not an easy task.

In this context, one often hears George E. P. Box's famous quote 'all models are wrong, but some are useful' [Box 1979], which is of course hard to disagree with. In the following, we will keep a close eye on the aspect of usefulness.

2.2 Marginal Likelihood

It has been noted, that the most probable model in a model class, the **maximum a posteriori** (MAP) model, is not the most suitable—or useful—for the task of *prediction*. Take as an example the data to be a single toss of a biased coin. The corresponding model class $\mathcal{C}$ is the set of **Bernoulli** models, indexed by the single free parameter Θ, defining the probability of heads facing up. Assuming a uniform prior over the values of $\Theta \in [0, 1]$, the MAP model will then be either $\mathcal{M} = \mathcal{C}(\Theta = 0)$ or $\mathcal{M} = \mathcal{C}(\Theta = 1)$, depending on whether we observe tails or heads. Using this model for prediction of the second toss would mean giving probability one to the outcome being the same as the first. Clearly, this does not reflect any belief we could reasonably argue.

Instead, when predicting, we can use *all* models in the class, integrating over all parameters/models. For an i.i.d. sample $\mathbf{D} = \{d_1, \ldots, d_n\}$, the **marginal likelihood** (MarLi, see, e.g. [MacKay 2002]) is given by

$$P(\mathbf{D}|\mathcal{C}) = \int_{\mathbf{\Theta}} P(\mathbf{D}|\mathcal{M} = \mathcal{C}(\mathbf{\Theta}))p(\mathcal{M} = \mathcal{C}(\mathbf{\Theta}))d\mathbf{\Theta}. \tag{2.8}$$

Whenever $\mathcal{C}$ is fixed, we may drop it from the notation and simply write

$$P(\mathbf{D}) = \int_{\mathbf{\Theta}} P(\mathbf{D}|\mathbf{\Theta})p(\mathbf{\Theta})d\mathbf{\Theta}. \tag{2.9}$$

For the binomial distribution, **conjugate priors** are given by a **Beta distribution**, and, for multinomial distributions over more than two values, by its generalized form, the **Dirichlet distribution**, see [MacKay 2002]. By conjugate we mean that the prior and the posterior distributions are of the same form. Therefore, we can play the prior-to-posterior game any number of times and include more data into the posterior as it arrives. By the chain rule (Rule 1), the posterior is invariant to the split and ordering by which we add data:

$$\begin{aligned} P(\mathbf{\Theta}, \mathbf{D} = \{\mathbf{D}_1, \mathbf{D}_2\}) &= P(\mathbf{D}_1, \mathbf{D}_2|\mathbf{\Theta})p(\mathbf{\Theta}) \overset{i.i.d.}{=} P(\mathbf{D}_1|\mathbf{\Theta})P(\mathbf{D}_2|\mathbf{\Theta})p(\mathbf{\Theta}) \\ &= P(\mathbf{D}_2|\mathbf{\Theta})P(\mathbf{\Theta}, \mathbf{D}_1) = P(\mathbf{D}_1|\mathbf{\Theta})P(\mathbf{\Theta}, \mathbf{D}_2). \end{aligned} \tag{2.10}$$

Also, for the posterior we have

$$P(\mathbf{\Theta}|\mathbf{D}_i) \propto P(\mathbf{\Theta}|\mathbf{D}_i)P(\mathbf{D}_i) = P(\mathbf{\Theta}, \mathbf{D}_i) \tag{2.11}$$

since the data probability $P(\mathbf{D}_i)$ is constant wrt. the chosen model.

For conjugate priors, where the posterior $P(\boldsymbol{\Theta}|\mathbf{D}_1)$ is of the same form as the prior $p(\boldsymbol{\Theta})$, we can therefore regard it as a new prior before observing the second batch $\mathbf{D}_2$. For multinomial distributions, when we choose a Dirichlet prior, we always remain within the world of Dirichlet distributions.

The prior p, which is uniform with respect to the standard parametrization of a multinomial distribution, is conjugate and given by

$$p \sim Dir(1, \ldots, 1). \tag{2.12}$$

In the above coin tossing example, the uniform (flat) prior therefore is

$$p(\Theta, 1-\Theta) \sim Dir(\alpha_{heads}, \alpha_{tails}) = Dir(1, 1) \tag{2.13}$$

and the marginal likelihood solves to

$$P(\mathbf{D}) = \frac{c(heads)!c(tails)!}{(c(heads) + c(tails) + 1)!}, \tag{2.14}$$

where $c(heads)$ and $c(tails)$ are the counts observed in $\mathbf{D}$.

Prediction of the next sample d_{n+1} turning up heads becomes (with $\alpha_{heads} = \alpha_{tails} = 1$)

$$\begin{aligned} P(d_{n+1} = heads|\mathbf{D}) &= \frac{P(\mathbf{D}, d_{n+1} = heads)}{P(\mathbf{D})} \\ &= \frac{(c(heads) + c(tails) + 1)!\ c(heads + 1)!\ c(tails)!}{(c(heads) + c(tails) + 2)!\ c(heads)!\ c(tails)!} \\ &= \frac{c(heads) + 1}{c(heads) + c(tails) + 2}, \end{aligned} \tag{2.15}$$

and thereby the probability of seeing the same outcome a second time, after having observed just one toss of the biased coin, under the uniform prior becomes $\frac{2}{3}$.

In general for a K-valued multinomial, taking on values in $\{1, \ldots, K\}$, assuming any Dirichlet prior $\alpha \sim Dir(\alpha_1, \ldots, \alpha_K)$ the marginal likelihood has the following form [Heckerman 1995].

$$P(\mathbf{D}) = \int_{\boldsymbol{\Theta}} P(\mathbf{D}|\boldsymbol{\Theta})p(\boldsymbol{\Theta})d\boldsymbol{\Theta} = \frac{\prod_k \Gamma(c(k) + \alpha_k)}{\Gamma(\sum_k (c(k) + \alpha_k))} \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \tag{2.16}$$

Here, the $c(k)$ are the observed counts of each value k in $\mathbf{D}$, and Γ is the gamma function.

Prediction of an unobserved value d takes on the following simple form:

$$P(d = k|\mathbf{D}) = P(d = k|\mathbf{c}) = \frac{c(k) + \alpha_k}{\sum_k (c(k) + \alpha_k)}. \tag{2.17}$$

This takes on the same form as the MAP model or, in fact, any model in the class of multinomials. Therefore, we can speak of the **marginal likelihood model**, which emulates the use of all models in the class. While the parameters of the MAP model are just the observed relative frequencies, the parameters of the marginal likelihood (MarLi) model are the relative **augmented** frequencies, to which we have added the pseudo-counts given by the Dirichlet prior. The count vector $\mathbf{c}$ completely (and minimally) determines both MAP and MarLi models.

But marginal likelihood is not only a tool for prediction. Without it, the data likelihood $P(\mathbf{D}|\mathcal{C})$—given just the model *class* with no parameters—is undefined and meaningless. The Bayesian answer is to integrate over all possible parameters, weighted by their prior probability. If these prior probabilities are conjugate, i.e. in the case of multinomials follow a Dirichlet distribution, then this can be done elegantly and efficiently. Now we can, by Bayes' rule, also calculate the reverse—$P(\mathcal{C}|\mathbf{D})$—the probability that the data has been generated by a model $\mathcal{M} \in \mathcal{C}$, as we shall further explore in Section 2.4.

Remark 1 (on prequential probability)
The marginal likelihood for multinomial distributions as given by Equation 2.16 can also be seen as the **prequential** *(for 'predictive sequential' [Dawid 1984]) probability. The rationale goes as follows. We interpret the Dirichlet prior as pseudo-counts, e.g. for the uniform prior we pretend that we have seen one of each kind in advance. For a data set of size n we then play the prior-to-posterior game n times. For example, for a series of n coin tosses $d_1, \ldots, d_n$ consisting of $c(heads)$ heads and $c(tails)$ tails, the first outcome will always get probability $\frac{1}{2}$, the second either $\frac{1}{3}$ or $\frac{2}{3}$, as we have added the first toss to the pseudo-counts, and so on.*

If we order the series heads first, tails last, the sequence of probabilities $P(d_j|d_1, \ldots, d_{j-1})$ becomes

$$\frac{1}{2}, \frac{2}{3}, \ldots, \frac{c(heads)}{c(heads)+1}, \frac{1}{c(heads)+2}, \ldots, \frac{c(tails)}{c(tails)+c(heads)+1}$$

and the probability of the complete sequence—the product of the above terms—is given by Equation 2.14. Reordering the sequence only permutes the enumerators, and thus leaves the product unchanged. Therefore the marginal likelihood can be interpreted as the product of the sequence of predictive probabilities $P(d_j|d_1, \ldots, d_{j-1})$.

2.3 Bayesian Network Models

So far we have dealt only with a single multinomial variable. The situation gets more interesting, when there are multiple, potentially interdependent variables involved. Suppose that data $\mathbf{D}$ consists of m multinomial variables $\mathbf{X} = \{X_1, \ldots, X_m\}$, where each X_i can take on values in $\{1, \ldots, K_i\}$. Then by the chain rule their joint probability distribution is given by a product of conditional probability distributions

$$P(\mathbf{X}) = \prod_{i=1}^{m} P(X_i | X_1, \ldots, X_{i-1}), \tag{2.18}$$

where of course we are free to choose the ordering of the X_i.

Each such conditional probability distribution $P(X_i|X_1, \ldots, X_{i-1})$ is represented by a table of size $\prod_{i'=1}^{i} K_{i'}$ listing the single probabilities $P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$, which are the parameters of the corresponding model. This easily becomes a large number of parameters. Counting the *free* parameters, for the joint distribution (2.18) we get an overall number of $\prod_{i=1}^{m} K_i - 1$. When data availability is limited—as it tends to be—already for a very moderate amount of variables there clearly is too little support for each parameter; the *full* model is overly complex and we need to simplify it.

Bayesians do so by introducing **independence assumptions**. Each conditional probability distribution is approximated by a distribution conditioned only on a subset $Pa(i) \subseteq \{X_1, \ldots, X_{i-1}\}$ of the preceding variables, such that we have

$$P_{\mathcal{B}}(\mathbf{X}) = \prod_{i=1}^{m} P(X_i | Pa_i^{\mathcal{B}}). \tag{2.19}$$

These independence assumptions can then be depicted in a **Bayesian Network** $\mathcal{B}$—which we have included in the above equation as an index— such that $Pa_i^{\mathcal{B}}$ is the **parent set** of X_i in $\mathcal{B}$. As the network $\mathcal{B}$ completely defines the parametric structure of the associated model class $\mathcal{C}_{\mathcal{B}}$ we can identify the two. To simplify notation, in the following we may write $\mathcal{B}$ in place of $\mathcal{C}_{\mathcal{B}}$—e.g., $\mathcal{B} \in \mathcal{F}$ in model class selection— $\mathcal{B}(\mathbf{\Theta})$ instead of $\mathcal{C}_{\mathcal{B}}(\mathbf{\Theta})$ for the instantiated model, and so on.

More formally, a Bayesian Network is a **directed acyclic graph** (DAG) on the relevant variables of a problem which is interpreted as in Equation 2.19. Acyclicity ensures that—after suitably reordering the variables X_i—we indeed have $Pa_i \subseteq \{X_1, \ldots, X_{i-1}\}$ for all i [Pearl 1988], such that there can be no circular reasoning.

To make this more clear, let us look at an example—the **burglar alarm** problem introduced in [Pearl 1988]—which is one of the most commonly used in tutorials and educational material.

Example 1 (Burglar alarm) *Suppose you are at work and your neighbour gives you a call (C) to inform you that the burglar alarm (A) at your home has gone off. So, thinking there has been a burglary (B), you jump into your car and drive home. On the way, you listen to the news (N) reporting an earthquake (E) in the area where you live. This changes your degree of belief about a burglary having actually occurred.*

There are five relevant variables involved: C, A, B, N and E. They are all binary (boolean), taking on values in $\{t, f\}$, short for $\{true, false\}$. A corresponding Bayesian Network $\mathcal{B}$ that we might want to use is given in Figure 2.1.

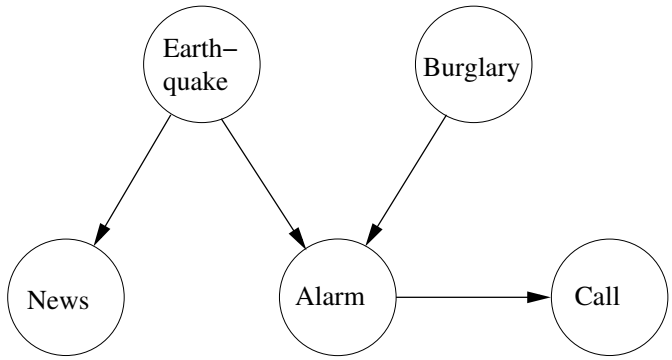

Figure 2.1: A Bayesian Network $\mathcal{B}$ for the burglar alarm problem.

The joint probability distribution then simplifies to

$$P_{\mathcal{B}}(E, B, N, A, C) = P(E)P(B)P(N|E)P(A|E, B)P(C|A). \qquad (2.20)$$

$P(E)$ depends on the area where you live and so does $P(B)$, $P(A|E, B)$ depends on the quality of your alarm system, and so on. Let us now define the model, by writing down the probability tables for each of these conditional distributions in Table 2.1. Note that this is a significant simplification to arbitrary joint distributions represented by a fully connected Bayesian network and specified by a probability table of size $2^5 = 32$. The full table lists 31 free parameters (the 32^{nd} given by the preceding 31 as

$P(E)$	
earthquake	
t	f
0.003	0.997

$P(B)$	
burglary	
t	f
0.002	0.998

$P(N\|E)$		
	news	
earthquake	t	f
t	0.7	0.3
f	0.001	0.999

$P(A\|E,B)$			
		alarm	
earthquake	burglary	t	f
t	t	0.95	0.05
t	f	0.5	0.5
f	t	0.9	0.1
f	f	0.005	0.995

$P(C\|A)$		
	call	
alarm	t	f
t	0.8	0.2
f	0.004	0.996

Table 2.1: Subjective conditional probabilities for the burglar alarm problem.

one minus their sum), while Table 2.1, using network $\mathcal{B}$ from Figure 2.1, only lists 10.

From these tables all individual conditional probabilities can be read off, e.g., the probability your neighbour will actually call when the alarm sounds is 0.8, the probability that the alarm will (falsely) alert in case there is an earthquake but no burglary is 0.5, and so on. These probabilities are predefined and subjective. Were we to learn the model from given training data, these probabilities would be the relative observed frequencies for the MAP model and the relative augmented frequencies—including the pseudo-counts given by a Dirichlet prior—for the marginal likelihood model.

We can now calculate the probability you would give to the event of a burglary at your home after your neighbour has called by summing over the unobserved values e of E, n of N and a of A as well as normalizing over the values b of B.

$$\begin{aligned} P_{\mathcal{B}}(B=t|C=t) &= \frac{\sum_{e,n,a} P(e)P(B=t)P(n|e)P(a|e,B=t)P(C=t|a)}{\sum_{e,b,n,a} P(e)P(b)P(n|e)P(a|e,b)P(C=t|a)} \\ &= \frac{\sum_{e,a} P(e)P(B=t)P(a|e,B=t)P(C=t|a)}{\sum_{e,b,a} P(e)P(b)P(a|e,b)P(C=t|a)} \approx 13.6\% \end{aligned} \quad (2.21)$$

and similarly the belief, after the earthquake news has been broadcast changes to

$$P_{\mathcal{B}}(B = t|N = t, C = t) \approx 5.4\%. \tag{2.22}$$

The news about the earthquake has influenced our belief about there having been a burglary. If there had not been a call from the neighbour about the alarm sounding, this would not have been the case, as earthquake (E) and burglary (B) are modeled as being independent events in $\mathcal{B}$. Let us now rewind and introduce some definitions.

Definition 1 (d-separatedness) *Let $\mathcal{B}$ be a Bayesian network on variables $\mathbf{X} = \{X_1, \ldots, X_m\}$ and $\mathbf{Z} \subseteq \mathbf{X}$ be a subset of these variables. Then any two variables $X_i, X_j \notin \mathbf{Z}$ are said to be* **d-separated** *by $\mathbf{Z}$ in $\mathcal{B}$, if every* trail *(path in the underlying undirected network) T from X_i to X_j is* **blocked** *by $\mathbf{Z}$, that is, we encounter at least one of the following situations:*

- *T contains a partial trail $X_u \rightarrow X_v \leftarrow X_w$ such that either X_v or a descendant of X_v is an element of $\mathbf{Z}$*
- *T contains a partial trail X_u—X_v—X_w such that at least one of the two arcs points away from X_v, and $X_v \notin \mathbf{Z}$*

Otherwise, X_i and X_j are said to be **d-connected** *in $\mathcal{B}$ given $\mathbf{Z}$.*

Equivalently, X_i and X_j are d-separated (d for 'dependence') by $\mathbf{Z}$ in $\mathcal{B}$ if, and only if, for any $\mathcal{B}$-model they are **conditionally independent** given $\mathbf{Z}$. In other words, for any set of parameters $\mathbf{\Theta}$ of $\mathcal{B}$ we have

$$P(X_i, X_j|\mathbf{Z}, \mathcal{B}(\mathbf{\Theta})) = P(X_i|\mathbf{Z}, \mathcal{B}(\mathbf{\Theta}))P(X_j|\mathbf{Z}, \mathcal{B}(\mathbf{\Theta})). \tag{2.23}$$

Therefore, d-separatedness is a means to read off conditional independence assumptions from a network $\mathcal{B}$.

In the burglar alarm example, N is d-separated from B given $\mathbf{Z} = \emptyset$ (burglary and earthquake—or the news about one—are initially independent events), but d-connected given $\{C\}$, the call providing evidence for the alarm A. Therefore, after the phone call has been received, the radio news can influence our belief in the event of a burglary.

Definition 2 (collider) *A substructure $X_u \rightarrow X_v \leftarrow X_w$ of a Bayesian network $\mathcal{B}$ is called a* **collider** *(inverted fork).*

A collider thus blocks a trail, if no evidence for its middle node X_v, i.e. for none of its descendants nor X_v itself, is given (in $\mathbf{Z}$). In all other situations, a path is blocked by a variable X_v which *is* given.

Definition 3 (v-structure) *A substructure $X_u \rightarrow X_v \leftarrow X_w$ is called a* **v-structure** *(unshielded collider) if there is no direct arc between X_u and X_w in $\mathcal{B}$.*

Lemma 1 *When checking for conditional independence, it suffices to consider v-structures instead of colliders.*

Proof Let T be a trail between X_i and X_j which is not blocked by $\mathbf{Z}$, and X_u, X_v, X_w form a shielded collider along T with tip X_v. It follows that $X_u \notin \mathbf{Z}$, $X_w \notin \mathbf{Z}$ and there is evidence about X_v in $\mathbf{Z}$ (X_v or at least one of its descendants is in $\mathbf{Z}$).

Then also the trail $T \setminus \{X_v\}$, which short cuts from X_u to X_w, is not blocked, since both X_u and X_w lie outside of $\mathbf{Z}$, but have a descendant which lies inside of $\mathbf{Z}$. Therefore neither of the situations of Definition 1 applies to a partial trail of $T \setminus \{X_v\}$ consisting of three nodes with middle node X_u or X_w. ■

Definition 4 (network equivalence) *Bayesian networks $\mathcal{B}$ and $\mathcal{B}'$ are said to be* **equivalent***, if they encode the same conditional independence assumptions.*

Lemma 2 *Bayesian networks $\mathcal{B}$ and $\mathcal{B}'$ are equivalent, if and only if their underlying undirected graphs are identical and both have the same v-structures.*

This is a direct consequence of Lemma 1.

Example 2 *The networks depicted in Figures 2.1 and 2.2 are equivalent. The arc between variables 'earthquake' and 'news' has been reversed, but this changes no v-structures. Although the latter seems false on an intuitive level, it encodes the exact same conditional independence assumptions. Bayesian networks do* not *encode* causal *dependencies.*

Definition 5 (Markov blanket) *The* **Markov Blanket** *$MB_{\mathcal{B}}(X)$ of a node X in $\mathcal{B}$ is the set of all its parents, all its children, and all parents of any of its children in $\mathcal{B}$.*

The Markov Blanket of X consists of exactly those nodes on which X is dependent—not d-separated—in the fully instantiated case. That is, for $X \neq Y \in \mathbf{X}$ and $\mathbf{Z} := \mathbf{X} \setminus \{X, Y\}$, it holds that

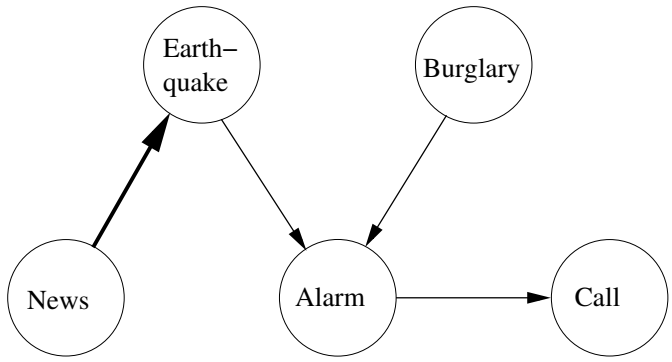

Figure 2.2: An alternative Bayesian network for the burglar alarm problem, equivalent to the network of Figure 2.1

$$Y \notin MB_{\mathcal{B}}(X) \Leftrightarrow [P(X|Y, \mathbf{Z}, \mathcal{B}(\mathbf{\Theta})) = P(X|\mathbf{Z}, \mathcal{B}(\mathbf{\Theta})) \text{ for all } \mathbf{\Theta}]\,. \quad (2.24)$$

In equivalent networks all nodes have identical Markov blankets.

Using network $\mathcal{B}$ from Figure 2.1 with parameters $\mathbf{\Theta}$ given by Table 2.1 we can calculate the joint probability of any instantiation (e, b, n, a, c) of the variables E, B, N, A, C. Using Bayes' rule we can also—by marginalizing over unobserved values—calculate all conditional probabilities. After receiving the phone call we had $P(B = t|C = t, \mathcal{M} = \mathcal{B}(\mathbf{\Theta})) \approx 13.6\%$, and after listening to the news about the earthquake, the probability of a burglary had dropped to $P(B = t|C = t, N = t, \mathcal{M} = \mathcal{B}(\mathbf{\Theta})) \approx 5.4\%$. This phenomenon, in which the branches of a v-structure become dependent in the case where we have evidence for the tip of the v-structure, is called **explaining away**. We have explained away the burglary by finding another reasonable cause—the earthquake—for the behaviour of their common child, the alarm.

2.4 Bayesian Model Class Selection

Model class selection is the problem of, given some data $\mathbf{D}$, finding a suitable model class $\mathcal{C}$ to describe it. In the Bayesian approach, this means calculating the probability $P(\mathcal{C}|\mathbf{D})$, which by Bayes' rule is

$$P(\mathcal{C}|\mathbf{D}) = \frac{P(\mathbf{D}|\mathcal{C})P(\mathcal{C})}{P(\mathbf{D})}. \tag{2.25}$$

The marginal data likelihood $P(\mathbf{D}|\mathcal{C})$ is defined in Equation 2.8, $P(\mathcal{C})$ is another prior distribution, which we need to define over all model classes $\mathcal{C} \in \mathcal{F}$ in the model family under consideration, and the term $P(\mathbf{D})$ is independent of $\mathcal{C}$ and thus serves as a normalizing constant.

One may then interpret $P(\mathcal{C}|\mathbf{D})$ as a degree of belief that the distribution $\mathcal{M}$ that has generated $\mathbf{D}$ is a member of $\mathcal{C}$, and conclude that the maximizing $\mathcal{C}$ is the most probable class.

This approach, simple as it sounds, is problematic in a number of ways, as we argue in the following. To make our criticism more explicit, we take as an example the model family consisting of all Bayesian networks on m multinomial variables, out of which we are to choose a suitable structure $\mathcal{B}$ for a given data set $\mathbf{D}$ of size n. The Bayesian approach to model class selection as such does not limit us to Bayesian network models, and it is more of a coincidence that both are being labeled 'Bayesian'. However, considering Bayesian network models is a good means to identify the problems arising in Bayesian model class selection.

Let $\mathbf{X} = \{X_1, \ldots, X_m\}$ be the data space consisting of m multinomial variables X_i of corresponding cardinalities K_i and $\mathbf{D}$ be an n-fold i.i.d. sample, i.e. an $n \times m$-matrix, where each entry $d_{ji} \in \{1, \ldots, K_i\}$ is the value that sample $\mathbf{d_j}$ (j^{th} row) takes on at variable X_i (i^{th} column). We search for the most probable Bayesian network $\mathcal{B}$, the one that maximizes Equation 2.25.

Each Bayesian network $\mathcal{B}$ is parametrized by a vector $\mathbf{\Theta}^{\mathcal{B}}$ with components of the form $\Theta^{\mathcal{B}}_{x_i|pa_i} = P(X_i = x_i|Pa_i = pa_i, \mathcal{M} = \mathcal{B}(\mathbf{\Theta}^{\mathcal{B}}))$, which are the entries of the conditional probability tables such as the ones listed in Table 2.1. The joint data likelihood of data set $\mathbf{D}$ given network structure $\mathcal{B}$ becomes

$$P(\mathbf{D}|\mathcal{B}(\mathbf{\Theta}^{\mathcal{B}})) \overset{i.i.d.}{=} \prod_{j=1}^{n} P(\mathbf{d_j}|\mathcal{B}(\mathbf{\Theta}^{\mathcal{B}})) = \prod_{j=1}^{n}\prod_{i=1}^{m} \Theta^{\mathcal{B}}_{d_{ji}|pa_i(\mathbf{d_j})}, \tag{2.26}$$

where $pa_i(\mathbf{d_j})$ is the instantiation of the parent set Pa_i of variable X_i in $\mathcal{B}$ which is given by the j^{th} sample $\mathbf{d_j}$.

If we now define a Dirichlet parameter prior for each conditional distribution $P(X_i|Pa_i = pa_i, \mathcal{B}(\mathbf{\Theta}^{\mathcal{B}})) = \Theta^{\mathcal{B}}_{\cdot|pa_i}$ such that

$$\Theta^{\mathcal{B}}_{\cdot|pa_i} \sim Dir(\alpha_{1|pa_i}, \ldots, \alpha_{K_i|pa_i}) \tag{2.27}$$

we also have the marginal data likelihood

$$\begin{aligned} P(\mathbf{D}|\mathcal{B}) &= \int_{\mathbf{\Theta}^{\mathcal{B}}} P(\mathbf{D}|\mathcal{B}(\mathbf{\Theta}^{\mathcal{B}}))p(\mathbf{\Theta}^{\mathcal{B}}|\mathcal{B})d\mathbf{\Theta}^{\mathcal{B}} \\ &= \prod_{i=1}^{m} \prod_{pa_i \in Pa_i} \left(\frac{\prod_{k=1}^{K_i} \Gamma(c(k|pa_i) + \alpha_{k|pa_i})}{\Gamma(\sum_{k=1}^{K_i}(c(k|pa_i) + \alpha_{k|pa_i}))} \frac{\Gamma(\sum_{k=1}^{K_i} \alpha_{k|pa_i})}{\prod_{k=1}^{K_i} \Gamma(\alpha_{k|pa_i})} \right). \end{aligned} \tag{2.28}$$

Whereas in Equation 2.16 we only had one multinomial distribution, we now take the product over all conditional distributions appearing in $\mathcal{B}$. Again, $c(k|pa_i)$ are the observed data counts and Γ is the Gamma function.

Let us now look at the problems arising, when we seek the 'most probable' Bayesian network for given data.

Problem 1 (the generating distribution assumption)
In order to define a distribution over the model classes $\mathcal{C} \in \mathcal{F}$, consisting of the probabilities that a model $\mathcal{M} \in \mathcal{C}$ has generated the data $\mathbf{D}$, we need to be sure that in fact such model exists.

It is a philosophical question, whether or not *any* generating distribution exists. But let us assume so. We still need to be sure that it is part of a model class in $\mathcal{F}$.

In our structure learning example, any multivariate multinomial distribution lies in $\mathcal{B}_{full}$, the class corresponding to any fully connected network (all of which are equivalent). But we have also made the i.i.d. assumption. If it does not hold—the data source was not completely stationary after all—it follows that the generating distribution lies outside of $\mathcal{F}$, which renders the search for the *most probable* model or model class meaningless.

Unless we are certain of our assumptions, i.e. we have generated the data ourselves, all we can hope for is to find a *useful* network structure, one that will give us good predictive performance.

Problem 2 (overlapping model classes)
The model classes in $\mathcal{F}$ may be overlapping, i.e. some of the models may be contained in multiple classes. In this case, $P(\mathcal{C}|\mathbf{D})$ should not be a distribution, but a superdistribution—sum to more than one—since models in the areas of overlap contribute their corresponding probability mass to more than just one class.

This is true for the structure learning problem, not only because of network equivalence. Observe, that adding arcs to a network $\mathcal{B}_1$ to arrive at a network $\mathcal{B}_2$ only makes the model class larger (reduces the conditional independence assumptions), such that we have $\mathcal{B}_1 \subset \mathcal{B}_2$ (as sets of distributions). More severely

Problem 3 (undefined data probability)
The term $P(\mathbf{D})$ in Equation 2.25 is undefined as such.

We can, however, interpret it as

$$P(\mathbf{D}|\mathcal{F}) = \sum_{\mathcal{B}\in\mathcal{F}} P(\mathbf{D}|\mathcal{B})P(\mathcal{B}), \tag{2.29}$$

that is, as a normalizing constant. But usually the model classes in $\mathcal{F}$ are not disjoint, as is the case for the family of Bayesian network structures.

Problem 4 (large number of model classes)
Under the assumption that $P(\mathbf{D}|\mathcal{F})$ is meaningfully defined by Equation 2.29, we may still not be able to compute it, if the number of model classes in $\mathcal{F}$ is large.

The number of Bayesian network structures is huge already for a reasonably small number of variables. There is only one structure for a single variable, for two variables there are three networks (two of which are equivalent), and for five variables—the situation of the burglar alarm example—there are already 29.281 different structures. The number of networks grows superexponentially with the number of variables, which can be seen by only considering chains of the form $X_1 \to X_2 \to \cdots \to X_m$. For each ordering of the variables there is one such chain, which means that there are already $m!$ chains for m variables. Restricting to equivalence classes of structures does not change the situation either, since for each m there are only two equivalent chains. For 10 variables we have more than 10^{19} network structures, and for 20 variables this number grows to over 10^{73} [Robinson 1977]. The number of equivalence classes is of the same order [Gillispie 2001].

This means that, in general, $P(\mathbf{D}|\mathcal{F})$ is beyond computational capacity. We can, however, still compute the relative probability for any two networks. In other words, we can still search for the best structure $\mathcal{B}$. Unfortunately, the absolute probabilities $P(\mathcal{B}|\mathbf{D},\mathcal{F})$ will be very small. By Bayesian reasoning, the most probable network will therefore almost certainly be wrong.

Summarizing over Problems 1–4, we find that it makes no sense to talk about the *most probable* model class. But can we still search for the most *useful* class, the one that predicts best? The answer to this question—naturally—greatly depends on the priors we choose.

Problem 5 (the parameter prior)
For each model class (Bayesian network structure $\mathcal{B}$) we need to define a suitable (Dirichlet) parameter prior.

As discussed in Sections 2.1 and 2.2, defining a reasonable parameter prior α is not an easy task. Since usually we do not have expert domain knowledge available, or are unable to transform it into Dirichlet distributions, a frequently made choice is the so-called 'uniform' prior $\alpha_{\cdot|pa_i} \sim Dir(1, \ldots, 1)$, which assigns equal probability to all parameter vectors $\Theta^{\mathcal{B}}$. As the uniform prior also assigns equal probability to all data sets of size 1, that is, to the first sample in the prequential game, it is often thought to be 'non-informative', see, e.g. [Hill 1997].

However, we need to remember that 'uniformity' is a property only defined with respect to a given parametrization. In fact it turns out that the 'uniform' prior, in general, yields different marginal likelihood for a data set $\mathbf{D}$ when using different, but *equivalent* Bayesian networks (Definition 4), which encode the exact same independence assumptions and consist of the exact same models (joint probability distributions). Equivalent network structures differ in the way they are parametrized and therefore the 'uniform' prior can have a different meaning depending on the actual representative one chooses for a given equivalence class, see Figure 2.3.

Figure 2.3: Two equivalent Bayesian network structures $\mathcal{B}_1$ (left) and $\mathcal{B}_2$ (right). For the 'uniform' parameter prior the associated marginal likelihood distributions are different, iff the cardinalities of the multinomials are: $K_1 \neq K_2 \Leftrightarrow P(.|\mathcal{B}_1) \neq P(.|\mathcal{B}_2)$. This can be seen most easily by interpreting the Dirichlet prior as pseudo-counts.

Jeffreys' prior [Jeffreys 1946] is invariant to reparametrization, yet being used less frequently, largely for technical reasons. Its main drawback is that it does not take on the Dirichlet form and hence is not conjugate. Both

prior and posterior take on forms which cannot typically be formulated in closed form and need to be approximated.

It can be shown, that the only Dirichlet priors which are invariant to this type of parameter transformation are those that, when viewed as pseudo-counts, correspond to actual data samples, non-integer counts allowed. Therefore the only 'non-informative' Dirichlet priors, in the forementioned sense, are the so-called **equivalent sample size** (ESS) priors given by

$$\alpha^{ESS}(S) : \Theta_{\cdot|pa_i} \sim Dir\left(\frac{S}{\mathcal{K}_i}, \ldots, \frac{S}{\mathcal{K}_i}\right) \quad \text{with} \quad \mathcal{K}_i := K_i \prod_{i' \in Pa_i} K_{i'}. \quad (2.30)$$

Using this type of parameter prior, we are left to choose only a single parameter, the size S of the assumed pseudo-data, which is then being spread evenly across the data space $\mathbf{X}$.

Unfortunately, there is no natural choice for S, nor is the resulting prior non-informative. As it turns out, the choice of the most probable network is extremely sensitive to this parameter [Silander 2007].

Problem 6 (the class prior)
We need to define a suitable prior distribution $P(\mathcal{C})$ over all classes in the family $\mathcal{F}$.

In structure learning, once more for lack of better knowledge, a uniform prior over the network structures is often assumed. Due to the large number of structures, all of them are therefore assigned very low prior probability. Sometimes, also a uniform prior over the equivalence classes is being used, which leads to similar results.

If we search for the *most probable* class, then we should assign probability one to the equivalence class of fully connected networks, as it contains all multivariate multinomial distributions. If, however, we look for a *useful* network, we hope for something much simpler, as it will have larger support for its parameters and generalize to unseen cases better.

Problem 7 (overfitting)
A too complex model class will overfit the observed data, generalize poorly and not be useful at all.

The principle of **Ockham's razor** tells us to choose the *simplest* hypothesis explaining the observations, see, e.g. [Angluin 1983].

Bayesian model class comparison is classically based on **Bayes Factors**, see [Berger 1985], the ratio of marginal likelihoods. However, the choice of model class according to this ratio may strongly depend on the parameter prior being used, as demonstrated in [Silander 2007]. Also observe that the marginal likelihood does not a priori favour simple model classes. In fact, for the empty data $\mathbf{D} = \{\emptyset\}$ it is equal to one for any model class, which can be observed by plugging zero counts $\mathbf{c} \equiv 0$ into Equation 2.16. Therefore, complexity is often being penalized in a more explicit way.

So how can we measure complexity or simplicity in order to apply Ockham's razor? With Bayesian network we do have some idea of what 'simple' means. At least we know that adding arcs makes a network more complex, as it will represent a larger set of distributions. But what does 'explain' mean in this context? Each network $\mathcal{B}$ assigns a probability $P(\mathbf{D}|\mathcal{B})$ to the given data, which is strictly positive, but usually quite small. While in the sense of Ockham's razor a hypothesis either explains the data or does not, for us 'explanation' is measured continuously. The higher the assigned probability, the better the explanation.

Clearly, there is a tradeoff between data fit and generalization capability. To this end, the so-called **information criteria** have been developed, the most prominent ones being the **Akaike Information Criterion** (AIC, [Akaike 1974]) and the **Bayesian Information Criterion** (BIC, [Schwarz 1978]). They make this tradeoff explicit by optimizing—instead of the model class *probability* $P(\mathcal{C}|\mathbf{D})$—a score of the form

$$XIC(\mathcal{C}|\mathbf{D}) = \text{data fit}(\mathbf{D}|\mathcal{C}) - \text{penalty}_{XIC}(\mathcal{C}), \tag{2.31}$$

XIC standing for 'any information criterion'.

The *data fit* term typically—and theoretically correctly—is the minus-logarithm of the **maximum likelihood** (ML) $\hat{P}(\mathbf{D}|\mathcal{C})$, defined as

$$\hat{P}(\mathbf{D}|\mathcal{C}) = P(\mathbf{D}|\mathcal{C}(\hat{\mathbf{\Theta}}(\mathbf{D}))), \tag{2.32}$$

where

$$\hat{\mathbf{\Theta}}(\mathbf{D}) = \underset{\mathbf{\Theta}}{\operatorname{argmax}}\, P(\mathbf{D}|\mathcal{C}(\mathbf{\Theta})) \tag{2.33}$$

is the set of parameters which, when we instantiate $\mathcal{C}$ with them, assigns the maximum probability to $\mathbf{D}$. The maximum likelihood parameters $\hat{\mathbf{\Theta}}(\mathbf{D})$ need not be unique for $\hat{P}(\mathbf{D}|\mathcal{C})$ to be defined. XIC scores are purely model class selection criteria and using the maximum likelihood to measure data fit does not imply we should also use a maximum likelihood model $\mathcal{C}(\hat{P}(\mathbf{D}))$ for prediction.

The *penalty* term differs for the various information criteria. Its purpose is to measure expected generalization capability. For AIC this penalty equals the number of free parameters $\mathrm{nfp}(\mathcal{C})$ in the model class,

$$\mathrm{penalty}_{AIC}(\mathcal{C}) = \mathrm{nfp}(\mathcal{C}), \tag{2.34}$$

while for BIC also the data size n is relevant,

$$\mathrm{penalty}_{BIC}(\mathcal{C}, n) = \frac{\log n}{2}\mathrm{nfp}(\mathcal{C}). \tag{2.35}$$

Seemingly, Equation 2.31 couples two entirely different things, the number of free parameters of a model class and the negative logarithm of a probability. But in fact the different penalty terms are approximations of quantities of the same form. Both AIC and BIC (among others) have been derived from asymptotic behaviour where the data size n goes to infinity and have some desirable properties *in the limit*.

While asymptotics are good as a sanity check, in the real-world situation of limited data availability we do not know how well the information criteria will work, that is, how useful a model class chosen in this way will turn out to be.

2.5 Supervised Learning Tasks

Let us now return to the task of parameter learning in Bayesian network models. We have already seen that defining a suitable parameter prior is problematic. Another problem arises, when we are presented with a **supervised** (discriminative) learning task [Vapnik 1998]. This means that we are not interested in the joint likelihood of a data sample, but only in some aspects of it. In the multivariate multinomial data domain the simplest—and most frequently encountered—supervised learning task is that of **classification**. For clarity of notation we add a *class variable* X_0 of cardinality K_0 to the set of *predictors* $\{X_1, \ldots, X_m\}$ to be defined as in the preceding.

A **classifier** then is a *conditional* distribution $P(X_0|X_1, \ldots, X_m)$.

Traditionally, classifiers are seen as a functions $f : X_1, \ldots, X_m \to X_0$, but here we deviate from this definition, since we consider *probabilistic* models. We therefore require a classifier to return a distribution instead of a single class. Of course, we can do that as well: return the class that gets the highest probability (see Section 2.7). More generally, a supervised learning task may also be to model a larger subset of the domain variables, or some other aspects of the data space. For simplicity, we only consider classification tasks here. The class variable (or, in general, the objective of the supervised learning task) *supervises* parameter learning, by telling us which aspects of the data are important to us.

It has been recognized, that for supervised prediction tasks such as classification, we should also use a supervised (discriminative) learning algorithm, such as conditional likelihood maximization [Greiner 2001, Ng 2001, Greiner 1997, Kontkanen 2001, Friedman 1997]. Nevertheless, in most related applications, model parameters are still determined using unsupervised methods, such as joint likelihood maximization and ordinary Bayesian methods.

Remember, that Bayesian networks model *joint* probability distributions over the data space $\mathbf{X} = \{X_0, \ldots, X_m\}$ by decomposing it into $m+1$ local, conditional distributions. Their parameters, regardless of whether we choose to use marginal likelihood or maximum a posteriori, are determined with respect to the joint distribution. Of course we can calculate the conditional distribution of the class given the predictor variables from

the joint

$$P(x_0|x_1,\ldots,x_m) = \frac{P(x_0,x_1\ldots,x_m)}{P(x_1,\ldots,x_m)} = \frac{P(x_0,x_1\ldots,x_m)}{\sum_{x_0'=1}^{K_0} P(x_0',x_1,\ldots,x_m)}, \quad (2.36)$$

and this is exactly what is typically done, leading to Bayesian network classifiers, see [Friedman 1997]. But this type of use does not correspond to the design of these models, since the joint data likelihood when used for parameter selection does not reflect performance in classification. This is like driving your tractor to work. It can be done, but tractors were designed for something else and there are better vehicles to get you to work.

Then why is it, that joint models, such as Bayesian classifiers, are still being used for supervised tasks? The main reason may be the (most often erroneous) assumption, that the chosen model class is 'correct'. Employing asymptotics of unlimited data availability as a sanity check, we pass the test if in fact data $\mathbf{D}$ is an i.i.d. sample drawn from a distribution in $\mathcal{B}$. This type of situation is visualized in Figure 2.4.

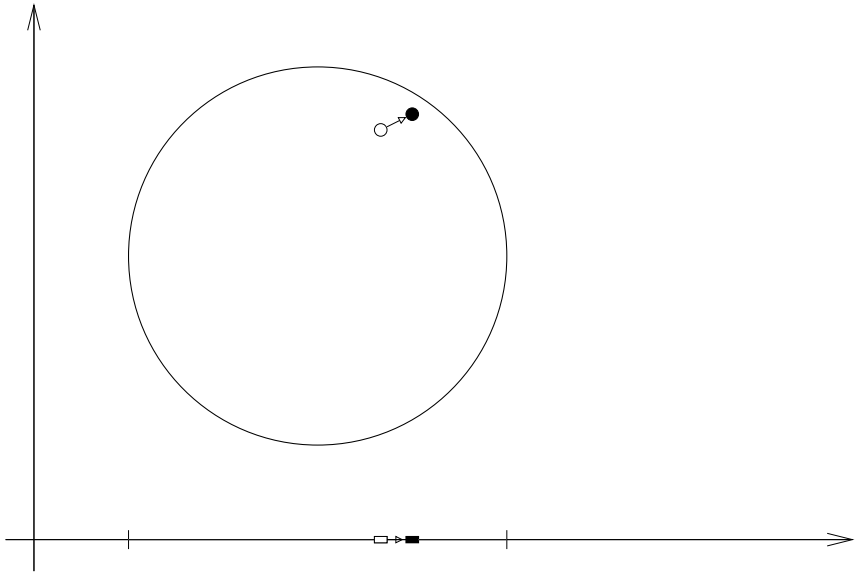

Figure 2.4: If the generating distribution (solid small circle) lies within the model class (large circle), then the learned joint model (open small circle) will approach it with growing data size to arbitrary precision with probability one. The conditional distributions (projection to the x-axis) will behave in the same way.

But if the generating distribution (assuming it exists) lies outside of $\mathcal{B}$

the situation becomes different. The conditional distribution obtained from the joint distribution in $\mathcal{B}$ which is closest to the true distribution (in terms of KL-divergence) need *not* be the best *classifier* in the class. Therefore, even with unlimited data availability, we may never get close to the most useful model, i.e. the one that minimizes the *conditional* KL-divergence from the true model. Intuition for this is given in Figure 2.5.

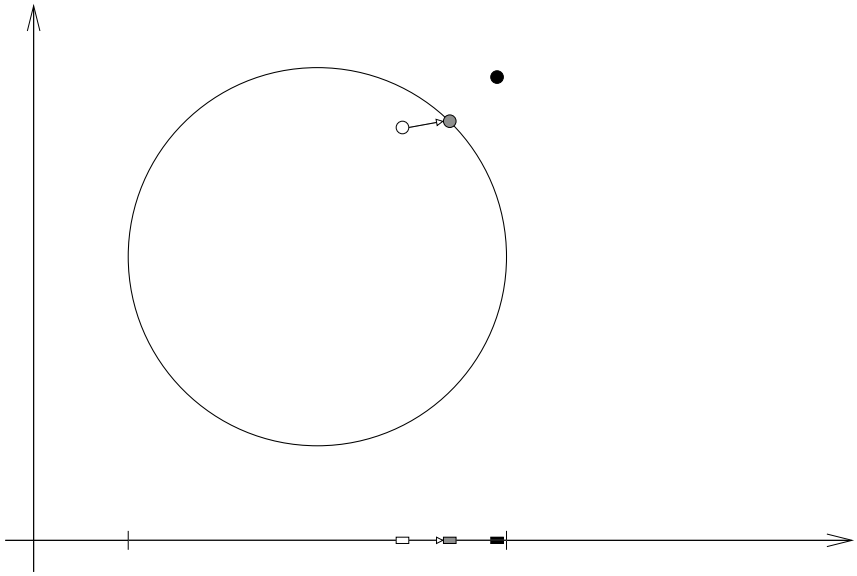

Figure 2.5: If the generating distribution (solid small circle) lies outside of the model class (large circle) then, with growing data size, the learned joint model (open small circle) will approach (under fairly weak assumptions, see [Cover 1991]) the distribution closest to it within the model class (gray circle). Conditioning this distribution (projecting to the x-axis) may *not* mean approaching the best *conditional* model.

The second reason why Bayesian network classifiers often determine their parameters by learning the joint distribution instead of the conditional, is the difficulty in finding the global maximum of the conditional likelihood. Publication I investigates the situations in which the model parameters can be efficiently learned in a discriminative fashion and the situations in which this is hard to accomplish. Section 2.6 summarizes these results as well as closes a gap which had been left by the original paper. While Publication I provides a sufficient condition under which a Bayesian network classifier is equivalent to a logistic regression model, here we are able to prove that this condition is also necessary.

2.6 Discriminative Parameter Learning

Given a network structure $\mathcal{B}$, we now investigate whether we can learn its parameters in a discriminative fashion, that is, with respect to the conditional distribution that is our objective. We do so by mapping $\mathcal{B}$ to a class of logistic regression models.

A model $\mathcal{B}(\boldsymbol{\Theta})$ with $\boldsymbol{\Theta} = \{\Theta_{x_i|pa_i}\}$ defines the conditional likelihood of the class X_0 as

$$P^{\mathcal{B}}(x_0|x_1,\ldots,x_m) = \frac{P^{\mathcal{B}}(x_0,x_1,\ldots,x_m)}{\sum_{x_0'=1}^{K_0} P^{\mathcal{B}}(x_0',x_1,\ldots,x_m)} = \frac{\prod_{i=0}^{m}\Theta_{x_i|pa_i(\mathbf{x})}}{\sum_{x_0'=1}^{K_0}\prod_{i=0}^{m}\Theta_{x_i|pa_i(\mathbf{x}')}}$$
$$= \frac{\Theta_{x_0|pa_0(\mathbf{x})}\prod_{i:X_0\in Pa_i}\Theta_{x_i|pa_i(\mathbf{x})}}{\sum_{x_0'=1}^{K_0}\Theta_{x_0'|pa_0(\mathbf{x}')}\prod_{i:X_0\in Pa_i}\Theta_{x_i|pa_i(\mathbf{x}')}}, \quad (2.37)$$

where we set $\mathbf{x}' = (x_0', x_1, \ldots, x_m)$, and $pa_i(\mathbf{x})$ is the instantiation of Pa_i given by $\mathbf{x}$. Note that the variables appearing in the rightmost expression are only X_0 and its Markov blanket. This is due to the fact that all other terms—the ones not involving x_0 (resp. x_0')—cancel. Equation 2.37 defines the $\mathcal{B}$-classifier. We denote the set of conditional distributions obtained using $\mathcal{B}$ in this way as $\mathcal{B}_{cond} = \{\mathcal{B}_{cond}(\boldsymbol{\Theta})\}$.

Logistic regression models, e.g. [McLachlan 1992, p.255], are of similar shape. Let $X_0 = \{1,\ldots,K_0\}$ and let $Y_1,\ldots,Y_K$ be real-valued random variables. The *multiple logistic regression model with dependent variable* X_0 *and covariates* $Y_1,\ldots,Y_K$ is defined as the set of conditional distributions

$$P^{LR}(x_0 \mid y_1,\ldots,y_K,\beta) := \frac{\exp\sum_{k=1}^{K}\beta_{x_0|k}\ y_k}{\sum_{x_0'=1}^{K_0}\exp\sum_{k=1}^{K}\beta_{x_0'|k}\ y_k} \quad (2.38)$$

where the model parameters $\beta_{x_0|k}$ are allowed to take on any value in $\mathbb{R}$. This defines a conditional model parametrized in $\mathbb{R}^{K_0\cdot K}$.

Now, for $i \in \{0\}\cup\{i : X_0 \in Pa_i\}$ (the class and its children in $\mathcal{B}$), $x_i \in \{1,\ldots,K_i\}$ and pa_i in the set of parent configurations of X_i, let

$$Y_{x_i|pa_i} := \begin{cases} 1 & \text{if } X_i = x_i \text{ and } Pa_i = pa_i \\ 0 & \text{otherwise.} \end{cases} \quad (2.39)$$

Subsequently, for $\mathcal{B}$-parameters $\boldsymbol{\Theta} = \{\Theta_{x_i|pa_i}\}$ we set

$$\beta_{x_i|pa_i} = \log\Theta_{x_i|pa_i}. \quad (2.40)$$

The indicator variables $Y_{x_i|pa_i}$ can be lexicographically ordered and renamed $1, \ldots, K$. The corresponding parameters $\beta_{x_i|pa_i}$ are of suitable form—as x_0 is either x_i itself or a member of pa_i—but need to be renamed accordingly. This shows that we have transformed the Bayesian network model $\mathcal{B}_{cond}(\boldsymbol{\Theta})$ into a logistic regression model, which we denote $LR_{\mathcal{B}}(\beta)$. Moreover, both models encode the same conditional distribution, as can easily be verified.

This proves **Theorem 1** of Publication I, namely

$$\mathcal{B}_{cond} \subseteq LR_{\mathcal{B}}. \tag{2.41}$$

At first sight, it would seem that we would even have equality here. Can we not transform any model $LR_{\mathcal{B}}(\beta)$ back into a model $\mathcal{B}_{cond}(\boldsymbol{\Theta})$ by setting $\Theta_{x_i|pa_i} = \exp \beta_{x_i|pa_i}$? Unfortunately, this is not the case in general. If of course, the β's come from Θ's, transformed by taking their logarithms as above, this can be done. However, for a general set $\beta \in \mathbb{R}^{K_0 \cdot K}$ (for suitable K), the Θ's will violate the sum-to-one constraints they are bound to by the fact that they form local, conditional probability distributions. That is, in order for $\boldsymbol{\Theta}$ to be a set of parameters defining a model $\mathcal{B}_{cond}(\boldsymbol{\Theta})$ we must have

$$\text{For all } \; i \in \{0\} \cup \{i : X_0 \in Pa_i\} \; \text{ and all } \; pa_i \in Pa_i : \sum_{x_i=1}^{K_i} \Theta_{x_i|pa_i} = 1. \tag{2.42}$$

But what Theorem 1 does give us, is the fact that any class of conditional Bayesian network models $\mathcal{B}_{cond}$ is contained in a (possibly larger) class of logistic regression models $LR_{\mathcal{B}}$. In this model class, learning parameters that maximize the conditional likelihood

$$\prod_{j=1}^{n} P^{LR}(d_{jo}|d_{j1}, \ldots, d_{jm}, \beta) \tag{2.43}$$

for given data $\mathbf{D} = (d_{ji})_{\substack{j=1..n \\ i=0..m}}$ is relatively easy. By **Theorem 2** of Publication I, the conditional log-likelihood

$$\sum_{j=1}^{n} \log P^{LR}(d_{jo}|d_{j1}, \ldots, d_{jm}, \beta) \tag{2.44}$$

is a concave function of the parameters β. Together with the fact that the parameter space $\mathbb{R}^{K_0 \cdot K}$ is convex we know that there can be no local

maxima that are not at the same time also global maxima. Choosing a strictly log-concave prior $p(\beta)$ on β, and maximizing

$$\sum_{j=1}^{n} \log P^{LR}(d_{jo}|d_{j1}, \ldots, d_{jm}, \beta) + \log p(\beta) \tag{2.45}$$

instead of Equation 2.44, we get a strictly concave objective, which can be optimized locally, e.g. by hillclimbing methods. However, the model $LR_{\mathcal{B}}(\beta)$ found in this way may *not* correspond to a model $\mathcal{B}_{cond}(\mathbf{\Theta})$ for any $\mathbf{\Theta}$ satisfying (2.42).

Theorem 3 of Publication I identifies the situations in which this does *not* happen. If $\mathcal{B}$ satisfies the following condition, then for each model $LR_{\mathcal{B}}(\beta)$ in $LR_{\mathcal{B}}$ there exists a parameter set $\mathbf{\Theta}$, such that $\mathcal{B}_{cond}(\mathbf{\Theta})$ encodes the same conditional distribution.

Condition 1. For all i such that $X_0 \in Pa_i$, there exists $X_{i'} \in Pa_i$ such that $Pa_i \subseteq Pa_{i'} \cup \{X_{i'}\}$.

The proof of Theorem 3 can be found in Publication 1. An alternative, equivalent definition of Condition 1 is given in [Roos 2005b]. It states that, when we restrict $\mathcal{B}$ to the Markov blanket of X_0 and connect all parents of X_0 to arrive at a network $\mathcal{B}^*$ (which can always be done without introducing cycles, see [Lauritzen 1996]), then any two nodes having a common child in must be connected in $\mathcal{B}^*$.

In simpler terms, Condition 1 demands that any two parents of any child X_i of the class X_0 must be connected, unless they are both also parents of X_0 itself. The converse therefore is

Converse of Condition 1. There exists a child X_i of X_0 with parents X_j and X_k, such that

- X_j and X_k are not connected in $\mathcal{B}$, and
- X_j is not a parent of X_0

Figure 2.6 depicts examples of four classes of Bayesian networks, for which Condition 1 is satisfied. Naive Bayes (NB) models assume that all predictors are independent once the class x_0 is known, see [Rish 2001]. In the corresponding network all children of the class have only one parent, the class itself. A diagnostic classifier [Kontkanen 2001] is defined by a network structure in which the class does not have children at all. The network

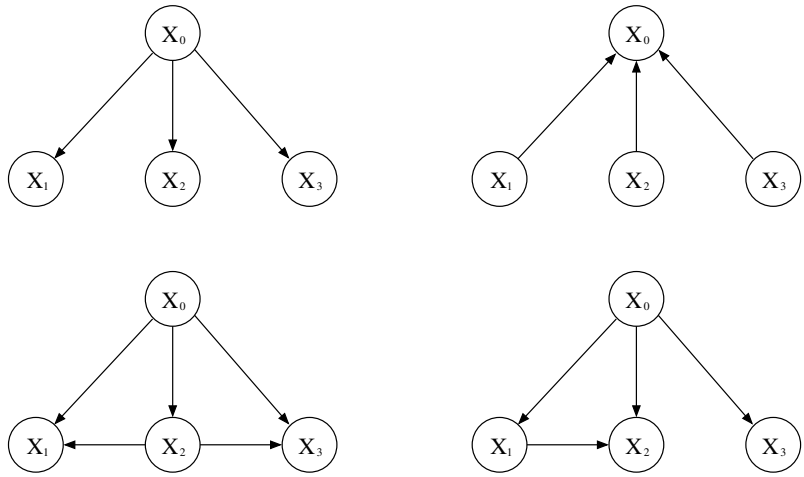

Figure 2.6: Examples of four types of Bayesian network satisfying Condition 1: Naive Bayes (NB, top left), a diagnostic classifier (top right), tree-augmented naive Bayes (TAN, bottom left) and forest-augmented naive Bayes (FAN, bottom right).

specifying a tree-augmented naive Bayes (TAN) classifier [Friedman 1997] is that of naive Bayes, to which arcs have been added that form an out-tree on the predictors. Similarly, forest-augmented naive Bayes (FAN) is NB augmented by a forest on the predictors, i.e. a collection of out-trees. TAN and FAN have in common, that any predictor X_i has the class X_0 as a parent and either no other parent, in which case $Pa_i \subseteq Pa_0 \cup \{X_0\} = \{X_0\}$, or exactly one other parent $X_{i'}$, in which case $Pa_i \subseteq Pa_{i'} \cup \{X_{i'}\}$. Therefore, by Theorem 3, the conditional versions of all these model classes are equivalent to a class of logistic regression models with freely varying parameters.

Theorem 3 has proven, that Condition 1 is sufficient for model class equivalence of the Bayesian network classifier to a suitably defined class of logistic regression models. But is it also necessary? Figure 2.7 depicts three types of networks, which do *not* satisfy the condition. Publication I proves in its Theorem 4 that for the network marked 'Type I' there in fact exists a data set $\mathbf{D}$ for which the conditional log-likelihood (Equation 2.46) does exhibit local, non-global maxima. This not only suggests that it cannot be optimized locally, but moreover that the classifier is not equivalent to *any* LR model, see below.

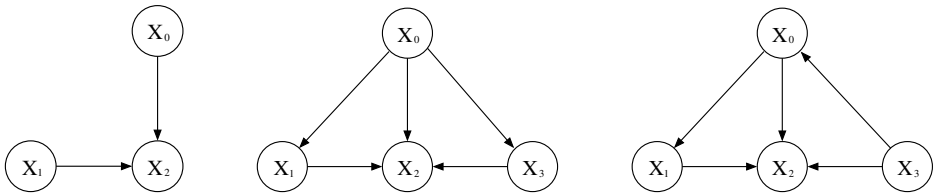

Figure 2.7: Three types of network structures *not* satisfying Condition 1.

Here, we prove a stronger result, which Publication I had left for 'future work'. We claim that in fact Condition 1 is necessary and therefore we have

Theorem 4 (previously unpublished) *For a Bayesian network $\mathcal{B}$, there exists a class of logistic models $LR_{\mathcal{B}}$ that consists of the same conditional distributions $\mathcal{B}_{cond} = LR_{\mathcal{B}}$ if* and only if *Condition 1 holds.*

Proof (sketch). The 'if' part is Theorem 3. It remains to show 'and only if'. We prove that, if $\mathcal{B}$ does not satisfy Condition 1, i.e. the Converse holds, then there exist data $\mathbf{D}$ for which the conditional log-likelihood

$$\sum_{j=1}^{n} \log P^{\mathcal{B}}(d_{jo}|d_{j1}, \ldots, d_{jm}, \mathbf{\Theta}) \tag{2.46}$$

exhibits multiple peaks. This not only suggests that we cannot find the global maximum by local search methods such as hillclimbing, but also that the classifier is not equivalent to *any* LR model. For if it were, then the log-transform (Equation 2.40) would preserve the peaks of the conditional log-likelihood in contradiction to Theorem 2.

Let $\mathcal{B}$ be a Bayesian network for which the 'Converse of Condition 1' holds, X_0 being the class, X_i a child of X_0, X_j and X_k parents of X_i such that $X_j \notin Pa_0$ and X_j and X_k are not connected in $\mathcal{B}$.

If $X_0 = X_j$ or $X_0 = X_k$, or either of them is not connected to the class, then there is a subgraph in $\mathcal{B}$ of Type I in Figure 2.7. If in turn this is not the case, then X_0, X_j and X_k are three distinct nodes, and both X_j and X_k are connected to X_0. By assumption, X_j is not a parent of X_0, which means it must be a child of the class. If X_k is likewise, then we have a subgraph of Type II, otherwise we find a subgraph of Type III in Figure 2.7. In fact, it is easy to see that Types II and III are equivalent. It suffices to restrict attention to these subgraphs, since any data for the variables of a

subgraph displaying multiple log-likelihood peaks can be extended to data for all of $\mathcal{B}$'s variables with the same property, e.g. by setting all values of any vector for variables not appearing in the subgraph to the same value. Also, we restrict to considering binary variables, without loss of generality. Note, that for larger variable cardinalities the additional values need not appear in $\mathbf{D}$, as the counter example data is of our own construction.

For Type I our case has been proven in Publication I. We defined a data set

$$\mathbf{D}_I = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 2 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \tag{2.47}$$

consisting of four data vectors, the columns corresponding to binary variables X_0, X_1 and X_2 in this order. We then could show, that there are four local, non-connected suprema of the conditional log-likelihood of $\mathbf{D}_I$. With a little trick, we could also make these suprema maxima, each having a different value. For the technical details please refer to Publication I, let us here look at what happens on a more intuitive level.

Observe that the only dependency this data displays is the equality of values in X_0 and X_1 for all four vectors, while all combinations of values in X_1 and X_2 appear. But there is no edge in $\mathcal{B}_{(Type\ I)}$ to directly model this dependency. On the other hand, X_2 is arbitrary. For each vector with $X_2 = 1$ we also have its counterpart, differing from it only in that it has $X_2 = 2$. The Bayesian classifier can now exploit its ability to explain away (cf. Sec. 2.3) and use the conditional probability table at X_2 to introduce the observed dependency between X_0 and X_1 into the conditional model. While doing so, the joint likelihood decreases, but the conditional goes up. We 'sacrifice' probability at X_2, which plays no part in the objective of the classification task.

For Types II and III we take a similar approach. We define

$$\mathbf{D}_{II/III} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 2 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 2 & 2 & 2 & 1 \\ 1 & 2 & 2 & 2 \end{bmatrix}; \tag{2.48}$$

columns corresponding to variables X_0, X_1, X_2 and X_3. In $\mathbf{D}_{II/III}$ we find

all combinations of values for X_1, X_2 and X_3 but the class takes on value

$$X_0 = \begin{cases} 1 & \text{if } X_1 = X_3 \\ 2 & \text{if } X_1 \neq X_3 \end{cases}. \tag{2.49}$$

Therefore the class depends on none of the predictors directly, but we can explain away the class based on the value of '$X_1 = X_3$?' through X_2. In effect, we have reduced Type II/III to Type I, where now '$X_1 = X_3$?' plays the role of X_1 in Type I. The rest of the proof is analogue to the proof of (the weaker) Theorem 4 in Publication I. We omit the technicalities. ■

We have shown, that learning the parameters of a Bayesian network classifier via the conditional likelihood, rather than the joint, can be done efficiently whenever the corresponding network structure satisfies Condition 1. If it does not, then there can be multiple local, non-global optima, which potentially makes finding the global optimum a hard task. It remains to show that indeed, the resulting classifier is to be prefered over the vanilla solution. We investigate this empirically in the following section.

2.7 Empirical Evaluation

We study the usefulness of discriminative parameter learning using the naive Bayes classifier as an example. The corresponding network structure (an example is given in Figure 2.6, top left) satisfies Condition 1, and therefore the classifier is equivalent to logistic regression with suitably defined covariates. The following results have been published in a technical report accompanying Publication I, [Wettig 2002a].

Note, that the conditional log-likelihood as a function of the standard Bayesian network parameters $\boldsymbol{\Theta}$ cannot have local, non-global maxima, since the log-transform (Equation 2.40) would preserve them and we know that there are no such maxima in the conditional log-likelihood viewed as a function of the LR parameters β. However, in the original parametrization the conditional log-likelihood is *not* concave and, if we restrict the parameter space to a convex subset, we can no longer be sure that there will be no unconnected local maxima. The log-transform mends this situation by concavifying the objective, see [Wettig 2002a]. Also, the sum-to-one constraints seem inconvenient in optimization, and therefore we choose to maximize the conditional log-likelihood in the LR parameter space. Empirical evidence, giving additional reason to prefer the LR parametrization over standard Bayesian, has been reported in [Greiner 2002].

We maximize the objective by iteratively optimizing one parameter $\beta_{x_0|k}$ at a time with growing precision, and terminate the search when the norm of the gradient at the current solution has dropped below a predefined threshold. This was sufficiently fast for our purposes, for a comparison of more sophisticated methods see, e.g., [Minka 2001].

The vanilla NB classifier uses the uniform prior wrt. its standard parametrization. To be fair, we tried to define the prior $p(\beta)$ on the LR parameters to be as close as possible to this. To that end, we transform the β's back into the standard parameter space—by taking their exponentials and normalizing suitably— and then take their product

$$p(\beta) := \prod_{k=1}^{K} \prod_{x_0=1}^{K_0} \frac{\exp \beta_{x_0|k}}{\sum_{x_0'} \exp \beta_{x_0'|k}}. \tag{2.50}$$

This is what we obtain when we interpret the uniform NB prior as pseudo-counts and look at the predictors one-at-a-time. Of course, this is not actually the same thing as a Bayesian uniform prior on the joint distributions, but since the LR model *does not define* any joint distribution, this seemed to be the best we can do.

We compared the NB and LR models using 32 real-world data sets from

Data	n	m	K	0/1-loss NB	0/1-loss LR	log-loss NB	log-loss LR
Mushrooms	8124	21	234	95.57	**100.00**	0.131	**0.002**
Page Blocks	5473	10	420	94.74	**96.29**	0.172	**0.102**
Abalone	4177	8	1008	23.49	**25.95**	2.920	**2.082**
Segmentation	2310	19	917	94.20	**97.01**	0.181	**0.118**
Yeast	1484	8	1060	55.59	**57.75**	1.155	**1.140**
German Credit	1000	20	244	**75.20**	74.30	0.535	**0.524**
TicTacToe	958	9	56	69.42	**98.33**	0.544	**0.099**
Vehicle Silhouettes	846	18	1116	63.95	**72.22**	1.731	**0.682**
Annealing	798	31	740	93.11	**99.00**	0.161	**0.053**
Diabetes	768	8	178	**76.30**	75.78	0.488	**0.479**
Breast C. (Wisc.)	699	10	222	**97.42**	96.42	0.260	**0.105**
Australian Credit	690	14	232	**86.52**	85.94	0.414	**0.334**
Balance Scale	625	4	63	92.16	**93.60**	0.508	**0.231**
Congr. Voting	435	16	104	90.11	**96.32**	0.632	**0.102**
Mole Fever	425	32	408	**90.35**	88.71	**0.213**	0.241
Dermatology	366	34	804	97.81	97.81	**0.042**	0.079
Ionosphere	351	33	402	92.31	**92.59**	0.361	**0.171**
Liver	345	6	236	64.06	**68.70**	0.643	**0.629**
Primary Tumor	339	17	903	48.97	**49.26**	1.930	**1.769**
Ecoli	336	7	760	80.36	**81.85**	**0.518**	0.562
Soybean	307	35	2527	85.02	**90.23**	0.647	**0.314**
Heart D. (Clevel.)	303	13	510	**58.09**	55.78	1.221	**1.214**
H.D. (Hungarian)	294	13	210	**83.33**	82.99	0.562	**0.444**
Breast Cancer	286	9	88	**72.38**	70.98	0.644	**0.606**
Heart D. (Statlog)	270	13	162	**85.19**	83.33	0.422	**0.419**
Thyroid Disease	215	5	198	**98.60**	94.88	**0.054**	0.132
Glass Identification	214	10	804	**70.09**	69.63	0.913	**0.809**
Wine Recognition	178	13	573	**97.19**	96.63	**0.056**	0.169
Hepatitis	155	19	274	79.35	**82.58**	0.560	**0.392**
Iris Plant	150	4	255	94.00	**94.67**	**0.169**	0.265
Lymphography	148	18	240	85.81	**86.49**	0.436	**0.375**
Postoperative	90	8	75	**67.78**	66.67	0.840	**0.837**

Table 2.2: Leave-one-out cross-validation results

the UC Irvine Machine Learning Repository (`archive/ics.uci.edu/ml/`). Continuous data were discretized, the exact method and discretized data sets can be found at `www.cs.Helsinki.FI/u/pkontkan/Data/`. We performed leave-one-out cross-validation, n times holding out one data sample at a time and training on the remaining $n-1$. We report, for both model classes, the 0/1-loss as percentage of correct classifications (when we return

the most probable class) and the log-loss, i.e. the average natural logarithm of the probability assigned to the correct class. Apart from the prior, the latter is exactly what we are optimizing on the training data with the LR model. The former is correlated, and commonly reported for classification tasks, but—also due to its discrete nature—we do expect it to behave less decisively.

The results we have obtained in this way are listed in Table 2.2, where n denotes data size, m the number of predictors and K the number of parameters—which is equal for both model classes—as we had before. Winning scores are boldfaced.

We observe, that in 26 out 32 cases the discriminative method has produced lower log-loss. On all larger data sets, it consistently outperformed the NB classifier, in several cases by a large margin. On six of the smaller data LR actually lost to NB, but by much smaller margin. As expected, the situation is less clear-cut for the 0/1-loss, but LR still wins 18:13 (with one draw), again larger data working in favour of LR. Similar observations have been reported in [Ng 2001].

We presume that this behaviour means that LR, as it better fits the training data, is also more prone to overfitting. But this is not to say that on small data sets optimizing the joint likelihood is to be prefered. In fact, unsupervised—joint—learning is no automatism to prevent overfitting. Instead, we propose a different approach. In logistic regression, it is easy to *prune* parameters from the model, effectively just fixing them to zero. In Bayesian network models, we can remove arcs from the network structure to obtain a simpler, better generalizing model class, e.g. predictor selection in naive Bayes classification. This amounts to the removal of a *whole set* of parameters. But logistic regression is more flexible than that. We can remove *any* set of parameters—not just sets that correspond to arc removal—even just a single parameter. One possible criterion for parameter selection is the size of its support, as suggested in [Wettig 2002b]. Here we remove a parameter $\beta_{x_i|pa_i}$ (set it to zero), whenever its support $|\{1 \leq j \leq n : d_{ji} = x_i \text{ and } pa_i(\mathbf{d}_j) = pa_i\}| \leq T$ in $\mathbf{D}$ does not exceed a given threshold T.

An even more elegant solution is the use of the so-called α-prior (Laplace prior, L_1-prior) defined by

$$\log p_\alpha(\beta) := \alpha \sum_{k=1}^{K} \sum_{x_0=1}^{K_0} |\beta_{x_0|k}|. \tag{2.51}$$

An example of its behaviour as compared to that of the 'transformed uniform' prior we have used in our experiments is plotted in Figure 2.8.

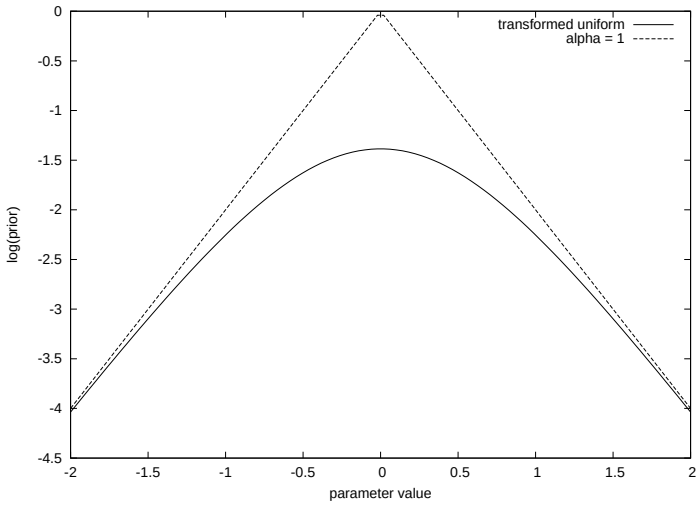

Figure 2.8: The logarithms of the prior probabilities $p_\alpha(\beta_{1|k}, \beta_{2|k})$ with $\alpha = 1$, and the 'transformed uniform' (2.50) of a pair of parameters predicting a binary class evaluated at $\beta_{1|k} = -\beta_{2|k}$.

We see that for large parameter values the two priors essentially agree, but the α-prior much more clearly prefers values near zero. In fact, test runs we have performed on the same data used here (plus some others) have shown that a model trained using the α-prior usually becomes quite sparse. In typical cases with hundreds of parameters and data size ranging from the hundreds to the thousands, most parameter values will be zero exactly, and thus have been pruned away. Exactly how sparse a model will be obviously depends on the value chosen for α. This way of automatic parameter selection clearly improves prediction, especially with small data size. We have not published these results, as others have beat us to it, e.g. [Cawley 2007].

2.8 Summary

This chapter has given an introduction to Bayesian reasoning and, more specifically, to Bayesian network models. We have seen that Bayesian network models can be a very neat way of expressing our subjective beliefs, as long as we have a good understanding of the problem domain. That is, we need to be able to explicitly encode reasonable independence assumptions as a network structure and instantiate its parameters locally as conditional probability tables. Then we can calculate the resulting joint (global) probability distribution and—by conditioning—any conditional distributions we might be interested in, which need no longer be local within the network.

Learning from data, however, is much harder. First of all, the conditional distributions that have been calculated from the joint need not be optimal with respect to the training data. Learning the joint distribution is always a compromise, as we optimize all conditional distributions simultaneously. Any specific such conditional may thus be far from optimal. It is hardly the fault of Bayesian network models, that they are frequently being used for discriminative learning tasks such as classification. We have shown, that in many cases there is no need to do so, either. This is where the author's contributions of this chapter lie. In Section 2.6 we have explicitly identified the situations in which we can—and those in which we cannot—(easily) learn the parameters of a Bayesian network classifier optimally in a discriminative way instead of suboptimally from the joint. We did so by transforming the classifier into a class of logistic regression models. This can always be done, and this transformation is always one-to-one, but only under 'Condition 1' on the network structure is it also onto. The condition holds, intuitively speaking, whenever the Bayesian classifier does not explain away the class. As we would have guessed, logistic regression models *cannot* explain away. But any other type dependencies a Bayesian classifier is able capture can also be represented by a logistic regression model, which is easier to handle and more flexible.

Another problem associated with parameter learning in Bayesian networks is the need for a prior distribution. We need this parameter prior, in order to obtain a posterior distribution; the training data transform the prior into a posterior. And while it can be seen as an asset that Bayesian network models force us to make our prior beliefs explicit, in practice it is typically unclear how to transform these beliefs into a prior distribution *over the parameters*. In fact, the only computationally convenient prior is Dirichlet, which is far from being intuitive. We are—more or less—safe, as long as we have a fixed, simple network structure and loads of data. But when we want to learn the network *structure* from data the situation

becomes awkward, as in structure learning the parameter prior plays a decisive role. The main reason for this being that, with respect to the model complexity, we never seem to have a sufficient amount of data available, as in model class selection complexity is a function of the data size. Therefore, the amount of data serving as support for any given parameter will remain strictly limited.

In many important areas, such as admission of pharmaceutic products and courts of law, classic hypothesis testing using t-tests with p-values is still the norm. Over the last decades, however, it has become increasingly accepted that Bayesian model class selection should be prefered, as it provides more objective, less biased answers [Kruschke 2012]. But also this approach faces its problems, as listed and discussed in Section 2.4. One of them being the 'X-Files assumption' ("the truth is out there"), the conjecture that the data generating distribution exists, and lies inside the considered model family. And while we do not claim that there exist no Bayesian solutions to these problems, it often is beneficial to take an utterly different approach.

In the following chapter, we introduce the **Minimum Description Length** (MDL) Principle, which offers an elegant framework for the task of model class selection. Many of the problems that the Bayesian approach has to deal with simply will not arise, to others there will be simple and straightforward solutions.

Chapter 3

Information Theory

"The truth is *in* there."

3.1 The Minimum Description Length Principle

Information theory *quantifies* information, measuring it in bits. Classic information theory, developed by the American mathematician Claude E. Shannon (1916–2001) [Shannon 1948], measures the information content of a data generating source as the *expected* number of bits needed to communicate a sample from it over a noiseless channel. This entity is called the (Shannon) **entropy** (denoted H after Boltzmann's H-theorem) and is, for a multinomial random variable X with cardinality K, defined by

$$H(X) = -\sum_{k=1}^{K} P(X = k) \log P(X = k). \tag{3.1}$$

Figure 3.1 shows a plot of the binary entropy $H_2(X)$ as a function of $P(X = 1) = 1 - P(X = 2)$. Entropy can also be viewed as a measure of uncertainty. It is maximized for the uniform distribution with value $\log K$ and minimized for the completely specified case (probabilities equal zero or one), when there is no uncertainty about the outcome and the entropy is zero. For example, tossing a fair coin contains one bit of information per throw, and a coin toss with known outcome delivers no additional information. The information content of the throw of a biased coin lies inbetween these extrema and is measured by the entropy.

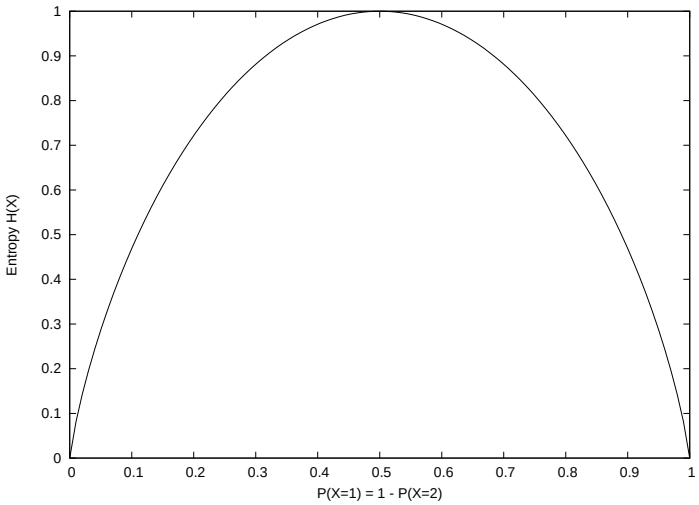

Figure 3.1: Plot of the binary entropy.

Note, that the entropy is a function of a *distribution* Q, so sometimes it is more intuitive to write $H(Q)$ in place of $H(X)$. The following theorem links the entropy to codelength.

Theorem 5 (Shannon's source coding theorem, [Shannon 1948]) *Let Q be a distribution over a finite set Σ, and L a* uniquely decodable code *(e.g. a* prefix code*) that for $\sigma \in \Sigma$ assigns a code word of length $|L(\sigma)|$. If L is optimal in that it minimizes the expected codelength $\mathbb{E}_Q[|L(\sigma)|]$ then we have*

$$H(Q) \leq \mathbb{E}_Q[|L(\sigma)|] \leq H(Q) + 1. \tag{3.2}$$

In practice, we will disregard the '+1'. When encoding n entities $\sigma_1, \ldots, \sigma_n$, we can achieve

$$nH(Q) \leq \mathbb{E}_Q \left[\sum_{j=1}^{n} |L(\sigma_j)| \right] \leq nH(Q) + 1, \tag{3.3}$$

and therefore rounding up to an integer number of bits has no significance in our applications. Or, as Peter Grünwald puts it, 'Non-integer Codelengths

Are Harmless' [Grünwald 2007, p.95][1]. We crudely allow real-valued codelengths.

In the year following the publication of Shannon's theorem, Leon G. Kraft—refering to Shannon's work—proved a stronger result, which has become known as **Kraft's inequality**.

Theorem 6 (Kraft's inequality, [Kraft 1949])
Let each symbol from an alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_K\}$ be encoded by a uniquely decodable code with corresponding codeword lengths $l_1, \ldots, l_K$ then

$$\sum_{i=1}^{K} 2^{-l_i} \leq 1. \tag{3.4}$$

Conversely, for a set of integers (read: numbers) $l_1, \ldots, l_K$ satisfying this inequality, there exists a uniquely decodable code for Σ with these codelengths.

Remark 2 *This result also applies to countably infinite alphabets.*

Remark 3 *Kraft originally proved the theorem only for prefix codes. The generalization to any uniquely decodable code is due to Brockway McMillan [McMillan 1956]. For this reason, the above is sometimes also referred to as the* Kraft-McMillan inequality*.*

Remark 4 *We use a capital L to denote the actual encoding $L(\sigma)$ of an entity σ. Its length—a real-valued number of bits—will be denoted by a lower case l: $l(\sigma) = |L(\sigma)|$.*

We can now, for any probability distribution $P = (p_1, \ldots, p_K)$ over the alphabet Σ define a uniquely decodable code with corresponding codeword lengths $l_i = -\log p_i$, satisfying Kraft's inequality. In fact, there is no need to explicitly define it, it suffices to know that such code exists. Of course, the expected codelength per symbol will be $H(P)$, the entropy of the distribution.

But Kraft's inequality gives us more than that. We can also turn any uniquely decodable code into a probability distribution, by setting $p_i = C^{-1} \cdot 2^{l_i}$, where $C \leq 1$ is a normalizing constant needed in case (3.4) is strict, i.e.

$$\sum_{i=1}^{K} 2^{-l_i} = C < 1. \tag{3.5}$$

[1] A rumour goes that, for the second edition of the book, this will be changed into 'Mostly Harmless'.

If equality does hold, then $C = 1$ plays no role.

Therefore, codelengths and probabilities are essentially the same. Coding and modeling are equivalent. The shorter the codelength, the higher the probability the code assigns to a data. With this observation, learning from data becomes finding regularities in it. These regularities help us to compress the data, and also to predict future data.

This is the **Minimum Description Length** (MDL) Principle, introduced by Jorma Rissanen [Rissanen 1978]. When the (two-to-the-minus) size of a compressed file is being viewed as a probability, then minimum description length means maximum probability.

But 'probability' takes on a slightly different meaning now. The probability distribution defined by any model does convert into an encoding scheme. And if the model is 'correct'—i.e., it equals the data generating distribution—then, by Shannon's theorem, the corresponding code is optimal in that it minimizes the expected codelength of data coming from this source. This expectation is the entropy of the source. But, when we *learn* a model from data, i.e. choose a model from a model class which best fits the data, then the resulting code is *not* uniquely decodable without knowledge of that choice.

A good way to describe the situation is the way Shannon looked at it. Picture a sender, a (noiseless) channel, and a receiver. The sender encodes data $\mathbf{D}$, the encoded message $L(\mathbf{D})$ is transmitted over a channel (which is expensive and there is no flat-rate; we want to keep the message short), and the receiver needs to be able to recover the original data from this message. Figure 3.2 visualizes this concept.

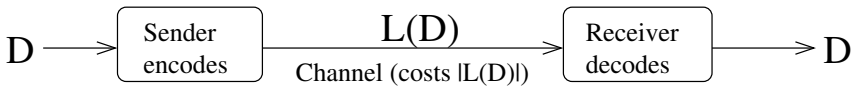

Figure 3.2: Schematic representation of the channel coding game.

This scheme forces us to be very explicit about three things:

1. Is the message decodable? This is a very good sanity check, which keeps us from cheating. If we cannot recover $\mathbf{D}$ from $L(\mathbf{D})$, then we have forgotten to transmit some vital information. As a result, Kraft's inequality may not hold and therefore the code may not be transformable into a probability (sub-) distribution.

In standard modeling, there is no such sanity check. Of course, every

model is a probability distribution and thereby forced to sum to one. But the actual distribution used is often implicit, e.g., we can hardly recover a model from a file of size equal to an information criterion penalty term such as (2.34) or (2.35).

2. Is this the shortest we can do? Of course, we really have no way of knowing, see Section 3.3. But we can always scan the message $L(\mathbf{D})$ for remaining regularities and, if we find any, include them in our code to make the message shorter. Observe that also any standard compressor such as `gzip`, `bzip2` and the like defines a code, as the compressed files are uniquely decodable. Therefore, we can always compare our codelength $|L(\mathbf{D})|$ against, say, the length of the file '`D.gz`'. We can also use standard compressors on the message $L(\mathbf{D})$. If the message further compresses then we must have overlooked some regularity.

There is no such 'remainder' to scan for additional regularity in probabilistic modeling, nor are other methods, such as standard compressors, directly comparable.

3. What have we agreed upon? In order for the receiver to be able to interpret the message, she and the sender must have agreed upon an encoding scheme. They have to meet at least once, e.g. on an open channel in the internet, and talk about what it is exactly that will be sent, in which order, and how it is going to be encoded. In other words, we need to be very clear about what is given and what is assumed. For example, sender and receiver may agree upon an i.i.d. assumption and that the data may arrive in any order. Of course, this is not enough. There has to be common knowledge about what the data structure is, which model family is being used, any side information and so on. Everything that is needed to interpret the message must have been agreed upon *in advance*. When we are being this explicit, it is easy to check whether this agreement matches the problem, whether we are actually sending the information we want to find regularities in.

Extreme cases of this are the null agreement (e.g., 'I will send you a selfextracting file. Run it and it will produce everything you need to know') and full specification (e.g., 'I will send you data of size n for the burglar alarm problem, using the network from Figure 2.1 with probabilities from Table 2.1'). Of course, the latter does not make much sense, we have not actually learned anything or found any regularities *in the data*. Learning is always a problem of selection, we need to find a model class and/or a model, that describes the data well or, equivalently, a coding scheme

that compresses the data well. Usually, the nature of the sender-receiver agreement will lie somewhere inbetween the two extremes. In any case, we are fine as long as we remember to never agree on anything, which may depend on data that is yet to be sent.

The 'fully specified' example illustrates, that any model whatsoever is also a code. But the main area of application for the MDL principle lies in model class selection. Observe that, at this point, we have already solved the first three problems related to Bayesian model class selection discussed in Section 2.4.

Solution to Problem 1 (no generating distribution assumption)
We do not assume that data $\mathbf{D}$ *have been generated by* any *distribution. We are simply compressing it, wherever it might have come from. If we do a good job, the resulting code will be* useful*, but there is no such thing as a* correct *code.*

Solution to Problem 2 (overlapping model classes)
Since we only want to find a useful *code, we do not need to worry about this. Of course, there may be many ways to encode the same data* $\mathbf{D}$ *and therefore, using the shortest encoding we can find, usually renders an* incomplete *code, i.e. Kraft's inequality will be strict. However, the normalizing constant*

$$C = \sum_{\mathbf{D}' \sim \mathbf{D}} 2^{-l(\mathbf{D}')} < 1, \tag{3.6}$$

is a value associated only with the code L*. So if* L *allows the use of alternative model classes, minimizing* $l(\mathbf{D})$ *will still pick the best, most compressing one. By* $\mathbf{D}' \sim \mathbf{D}$ *we mean that* $\mathbf{D}'$ *and* $\mathbf{D}$ *are of the same (previously agreed upon) format, e.g. matrices of the same dimensions with values from the same alphabets.*

Solution to Problem 3 (defined data codelength)
$P(\mathbf{D}) = C^{-1}2^{-l(\mathbf{D})}$ *is always defined, even though* C *may not be known.*

The simplest way to implement the MDL Principle are the so-called **two-part codes**. The following section explains this concept and gives illustrating examples.

3.2 Two-Part Codes

The basic idea of **two-part coding**, given a model class $\mathcal{C}$, is to first encode the model parameters $\boldsymbol{\Theta}$ and subsequently the actual data $\mathbf{D}$ using distribution $\mathcal{C}(\boldsymbol{\Theta})$:

$$l_{\mathcal{C}}^{2p}(\mathbf{D}) = l(\boldsymbol{\Theta}) + l(\mathbf{D}; \mathcal{C}(\boldsymbol{\Theta})). \tag{3.7}$$

We may then choose the best model class as the one that minimizes this combined codelength.

Taking the negative of Equation 3.7 (now to be maximized) makes it look a lot like the information criteria (2.31),

$$-l_{\mathcal{C}}^{2p}(\mathbf{D}) = \log P(\mathbf{D}|\mathcal{C}(\boldsymbol{\Theta})) - l(\boldsymbol{\Theta}). \tag{3.8}$$

Therefore we can regard $l(\boldsymbol{\Theta})$ as a complexity penalty term. If we choose to encode the parameters in a given (previously agreed upon) range to given precision, then we arrive at a penalty term that only depends on the *number* of (free) parameters, as we have for AIC. If range and/or precision depend on the data size, then so does the penalty term, as it does for BIC. But where the XIC were just criteria that have fallen from the skies of asymptotics, we now have the decodability requirement to meet.

At this point, it becomes clear why it makes no sense to simply select the model that assigns the highest probability $P(\mathbf{D}|\mathcal{C}(\hat{\boldsymbol{\Theta}}))$ to the data. This would be disregarding the term $l(\boldsymbol{\Theta})$, the length of the encoding of the used model, which is a measure of its complexity. We cannot recover $\mathbf{D}$ from a message of length $-\log P(\mathbf{D}|\mathcal{C}(\hat{\boldsymbol{\Theta}}))$, as—summed over all possible data—these code lengths violate Kraft's inequality.

Remark 5 *We use the term 'two-part code' whenever we encode the model parameters separately, following the notation of [Grünwald 2007]. Otherwise we speak of 'one-part codes'. This terminology may be somewhat confusing, as both types of encoding may involve multiple parts, a one-part code may consist of two or more parts and a two-part code of, say, five. In the literature (including attached Publications III–VI), there is no unanimous way of using 'two-part coding' to denote 'separate parameter encoding'. For obvious reasons, this is sometimes referred to as 'naive MDL', e.g. [Djurić 1998].*

Let us now look at a few examples.

Two-part encoding using a Bayesian network $\mathcal{B}$

Let, as in Section 2.4, $\mathbf{X} = \{X_1, \ldots, X_m\}$ be the data space consisting of m multinomial variables X_i of corresponding cardinalities K_i and

$\mathbf{D} = (d_{ji})_{\substack{j=1..n \\ i=1..m}}$ consist of n samples $\mathbf{d_j}$, with m entries $d_{ji} \in \{1, \ldots, K_i\}$ each. Given a Bayesian network $\mathcal{B}$ on the variables $X_i \in \mathbf{X}$, we order the X_i such that for all i we have $Pa_i \subseteq \{X_1, \ldots, X_{i-1}\}$. We want to first encode the model parameters $\mathbf{\Theta} = (\Theta_{x_i|pa_i})$ and subsequently $\mathbf{D}$ using $\mathcal{B}(\mathbf{\Theta})$.

Since the parameters are continuous, we can only encode them to finite precision. However, the MAP parameters are relative frequencies, so for our purposes it is enough to encode these frequencies, the counts of $\mathbf{D}$ with respect to $\mathcal{B}$. Slightly deviating from standard two-part coding, we encode $\mathbf{D}$ columnwise, for each variable X_i first encoding the corresponding parameters, and then its n values $d_{1i}, \ldots, d_{ni}$.

Assuming that the data size n is known to the receiver, the counts at X_1 can be communicated using

$$\log \binom{n + K_1 - 1}{K_1 - 1} \tag{3.9}$$

bits, since $\binom{n+K_1-1}{K_1-1}$ is the number of weak compositions of n into K_1 parts, see [Andrews 1976], or the number of different count vectors at X_1 for data of size n. The first column of $\mathbf{D}$ can then be transmitted in

$$-\sum_{k=1}^{K_1} c_k^1 \log \Theta_k^1 = -\sum_{k=1}^{K_1} c_k^1 \log \frac{c_k^1}{n} = -\sum_{k=1}^{K_1} c_k^1 \log c_k^1 + n \log n \tag{3.10}$$

bits, where $\mathbf{c^1} = (c_1^1, \ldots, c_{K_1}^1)$ is the vector of counts at X_1 encoded above, which transforms into MAP parameters $\Theta_k^1 = \frac{c_k^1}{n}$. For zero counts, we take on the convenient convention that $0 \log 0 = 0$.

For the remaining variables $X_2, \ldots, X_M$, which may have parents in $\mathcal{B}$, we need to encode $\mathcal{K}_i = \prod_{i':X_i \in Pa_{i'}} K_{i'}$ sets of parameters (counts) $(\Theta^i_{k|pa_i})_{k=1..K_i}$ $((c^i_{k|pa_i})_{k=1..K_i})$, which takes

$$\log \binom{c(pa_i) + K_i - 1}{K_i - 1} \tag{3.11}$$

bits for each instantiation $pa_i \in Pa_i$. Note, that the parent counts $c(pa_i)$ are known to the receiver at this point. Coding the i^{th} column of $\mathbf{D}$ using these MAP parameters then takes

$$\begin{aligned} &-\sum_{pa_i \in Pa_i} \sum_{k=1}^{K_i} c^i_{k|pa_i} \log \frac{c^i_{k|pa_i}}{c(pa_i)} \\ &\qquad = \sum_{pa_i \in Pa_i} \left(-\sum_{k=1}^{K_i} c^i_{k|pa_i} \log c^i_{k|pa_i} + c(pa_i) \log c(pa_i) \right) \end{aligned} \tag{3.12}$$

bits. The overall codelength becomes

$$l^{2p}(\mathbf{D}; n, \pi, \mathcal{B}) = \sum_{i=1}^{m} \sum_{pa_i \in Pa_i} \log \binom{c(pa_i) + K_i - 1}{K_i - 1} + \sum_{i=1}^{m} \sum_{pa_i \in Pa_i} \left(- \sum_{k=1}^{K_i} c^i_{k|pa_i} \log c^i_{k|pa_i} + c(pa_i) \log c(pa_i) \right), \quad (3.13)$$

where for any X_i with $Pa_i = \emptyset$ we assume there is a single instantiation pa_i with $c(pa_i) = n$. Sender and receiver have previously agreed upon data size n, the ordering of the variables π and the network structure $\mathcal{B}$. For this reason we have included them in the codelength term $l^{2p}(\mathbf{D}; n, \pi, \mathcal{B})$, separated from the data to be encoded by a semicolon. Read: 'Two-part codelength of $\mathbf{D}$ *using* n, π and $\mathcal{B}$'.

Note that the 'penalty term' in the upper line of Equation 3.13 depends on the actual data, not only on its size. This is not dangerous. In MDL, any clever way of encoding is permitted. Which is not to say that the above is very clever, it merely serves as an example of what one *could* do.

The parameter prior we have implicitly used is uniform in that any count vector $(c^i_{k|pa_i})_{k=1..K_i, pa_i \in Pa_i}$ for each variable X_i with parent instantiation pa_i is encoded with constant length. On the other hand, it gives point mass only to parameters corresponding to data counts, while assigning zero probability to any other parameters.

Two-part encoding using any Bayesian network

Let us now assume, that sender and receiver have only agreed on the fact that they will use *a* Bayesian network to encode data from domain $\mathbf{X}$, but not on its actual structure. The sender wants to be free to, *after* having seen the data $\mathbf{D}$, choose some network $\mathcal{B}$ that yields short codelength.

We first encode the data size n in a **selfdelimiting** way, [Li 1997, p.79], which can be done in $2 \log n + 1$ bits by first sending $\log n$ ones, then a single zero followed by $\log n$ bits to encode n itself. In general, there is a shorter selfdelimiting description of an integer n. With the iterated logarithm, we can achieve $l(n) = \log(n) + \mathcal{O}(\log \log n)$, see [Li 1997, p.80][2]. Next, we encode the network structure. As there are superexponentially many such structures, but for our limited data size we expect to be using a relatively

[2] a simpler way to achieve an encoding of same order length is to encode $\log n$ in blocks of k bits and reserving one block as a special string to mean 'end of this part', then sending this special string, followed by the encoding of n of length $\log n$. In this way, for any $k \geq 2$, we get a codelength of $k \left(\frac{\log \log n}{\log(2^k - 1)} + 1 \right) + \log n$.

sparse network, it seems beneficial not to use a uniform distribution. Instead, we may simply list all arcs, first giving the number $0 \leq |Pa_i| \leq m-1$ of parents using $\log m$ bits per node and then specifying which parents are present using $\log \binom{m-1}{|Pa_i|}$ bits for node X_i. Of course, in this way we can also encode networks which are not DAGs, and therefore we could define a tighter code if we bothered. A suitable ordering π of the variables can be retrieved by the receiver from the structure. We get an overall codelength of

$$l^{2p}(\mathbf{D}) \;=\; 2\log n \,+\, 1 \,+\, m\log m \,+\, \sum_i \log \binom{m-1}{|Pa_i|} \,+\, l^{2p}(\mathbf{D}; n, \pi, \mathcal{B}). \tag{3.14}$$

One-part encoding using a Bayesian network
Of course, we could have also employed the marginal likelihood, instead of encoding the parameters explicitly. Here, sender and receiver agree on a set of pseudo-counts α, for example an ESS prior as in Equation 2.30, and then play the prequential game of Remark 1 in Section 2.2. That is, the data samples are transmitted one at a time, and after each transmission both sender and receiver update their counts (including the pseudo-counts α) and always use the relative counts observed so far as the probability distribution to encode/decode the next sample. The resulting codelength is the negative of the marginal log-likelihood

$$\begin{aligned} l^{1p}(\mathbf{D}; n, \pi, \mathcal{B}, \alpha) = \sum_{i=1}^{m} \sum_{pa_i \in Pa_i} \Bigg(&- \sum_{k=1}^{K_i} \log \Gamma(c(k|pa_i) + \alpha_{k|pa_i}) \\ &+ \log \Gamma(\sum_{k=1}^{K_i} (c(k|pa_i) + \alpha_{k|pa_i})) - \log \Gamma(\sum_{k=1}^{K_i} \alpha_{k|pa_i}) + \sum_{k=1}^{K_i} \log \Gamma(\alpha_{k|pa_i}) \Bigg), \end{aligned} \tag{3.15}$$

cf. Equation 2.28.

Two-part encoding using logistic regression
For classification, it makes sense to only transmit the class labels of given data $\mathbf{D}$ from a domain $\mathbf{X} = \{X_0, \ldots, X_m\}$, where X_0 is the class variable. We therefore search for a code of length $l^{2p}_{LR}(\mathbf{d^0}; \mathbf{d^1}, \ldots, \mathbf{d^m})$, where $\mathbf{d^i}$ is the ith column of matrix $\mathbf{D}$. From the predictor data, the receiver also knows the number n of class labels to be transmitted. We further assume, that we have agreed on the candidate set $\mathbf{Y} = Y_1, \ldots, Y_K$ of covariates that may be included in the logistic regression model.

In order to minimize the total length of the transmitted code, we want to choose a sparse model, avoiding to encode unnecessarily many parameters, cf. Section 2.7. If we are using K' parameters out of the given K candidates, we can encode our choice in

$$l(K') = \log(K+1) + \log \binom{K}{K'} \tag{3.16}$$

bits, first encoding K' according to a uniform distribution over $\{0, \ldots, K\}$ and then giving the actual indices of the non-zero parameters. For convenience let us—without loss of generality—assume that the chosen indices are $\{0, \ldots, K'\}$.

How to optimally encode continuous parameters is discussed in, e.g., [Gao 2000]. As we have had some good experience with the α-prior, we choose to encode the β's with respect to a discretized Laplace (double exponential) distribution located around zero with scale one. We have

$$P(\beta|\varepsilon) = \frac{1}{1 - \frac{1}{2}\left(\exp\frac{\varepsilon}{2} - \exp\frac{-\varepsilon}{2}\right)} \frac{1}{2} \left(\exp\frac{\varepsilon}{2} - \exp\frac{-\varepsilon}{2}\right) \exp(-|\beta|), \tag{3.17}$$

where ε is the precision we use, the first factor comes from the fact that the receiver knows that a parameter encoded in this way will be non-zero, and the rest is the integral

$$\int_{x=\beta-\frac{\varepsilon}{2}}^{\beta+\frac{\varepsilon}{2}} \frac{1}{2}\exp(-|x|)dx = \frac{1}{2}\left(\exp\frac{\varepsilon}{2} - \exp\frac{-\varepsilon}{2}\right)\exp(-|\beta|) \tag{3.18}$$

of the chosen distribution over an ε-interval around the value of β. The corresponding codelength $l(\beta;\varepsilon)$ is the minus logarithm of (3.17). We further choose ε to be the inverse of an integer, which needs to be encoded selfdelimitingly in $l(\varepsilon) = 2\log\frac{1}{\varepsilon} + 1$ bits.

The overall codelength becomes

$$\begin{aligned} &l^{2p}_{LR}(\mathbf{d^0}; \mathbf{d^1}, \ldots, \mathbf{d^m}) \\ &= l(K') + l(\varepsilon) + \sum_{k=1}^{K'} l(\beta_k;\varepsilon) - \sum_{j=1}^{n} \log P^{LR}(d_{j0} \mid Y_1(\mathbf{d_j}), \ldots, Y_{K'}(\mathbf{d_j}), \beta) \end{aligned} \tag{3.19}$$

where all β_k are multiples of ε and the final term is the conditional log-likelihood under the logistic regression model given by Equation 2.38. It is now up to the sender to optimize the model with respect to this cost function.

For all the examples mentioned here, we have made arbitrary choices. For instance, for the β's of the logistic regression model we could have chosen a different distribution, such as a Laplace distribution with variable, separately encoded scale. More examples will be given in Chapter 4.

We now look at the remaining open problems of model class selection raised in Section 2.4.

Solution to Problems 4, 6 and 7 (class encoding, overfitting)
As we have seen, even when the number of classes in $\mathcal{F}$ is large, there are ways to encode what we expect *to be using in considerably fewer bits than $\log|\mathcal{F}|$, e.g. encoding Bayesian network structures by explicitly listing its arcs. Note, that also for sparse logistic regression models, $K' \ll K$, we have $\log(K+1)+\log\binom{K}{K'} \ll K$, where K is the codelength we need to choose any of the 2^K subsets of the parameters using a uniform distribution. The 'complexity' of the class is now measured in bits, as is the data log-likelihood, and therefore the two parts of the code are directly comparable. The combined codelength trades off data fit against complexity* automatically*, quantifying Ockham's razor and preventing overfitting.*

Solution to Problem 5 (the parameter prior, postponed)
We have made some progress by encoding the parameters to specified precision. The larger the data, the higher the precision, as the relative weight of the parameter codelength decreases. However, we are still encoding the parameters with respect to an assumed distribution, effectively a prior. And while we can now compare different such distributions by the resulting total codelength, in many cases we can still do much better. Section 3.4 introduces the **Normalized Maximum Likelihood** *(NML) distribution, which no longer needs a prior at all.*

There are always many ways to encode data of any sort, the best being the one that yields the shortest codelength. It makes sense to ask oneself, which is the *optimal* codelength. The length of the shortest decodable description of a given data is known as the **Kolmogorov Complexity** and it comes with extensive theory. In the following section, we briefly review its main concepts and the implications relevant to this work.

3.3 Kolmogorov Complexity

A good introduction to the theory of Kolmogorov Complexity is provided by [Li 1997], which we use as the standard reference throughout this section. In the following, references to this book will be limited to page numbers. For an introduction, the book is rather lengthy (790 pages!), due to the fact that the theory of Kolmogorov complexity is very extensive. For this reason, this section is going to be *very* brief, hand-wavy at times. For instance, we will not go into the definition of prefix complexity, the selfdelimiting version of the Kolmogorov Complexity, even though we are using it in the definition of the universal distribution. Nonetheless, we include all definitions and results we will need for motivation of the remainder of this work.

Definition 6 (Identification of strings and natural numbers, p.12)

We identify strings and natural numbers by the following bijection

$$\begin{aligned} \mathbb{B}^* &\longrightarrow \mathbb{N} \\ s_n \dots s_0 &\longmapsto \sum_{i=0}^{n} (s_i + 1) 2^i \end{aligned} \tag{3.20}$$

This definition differs from standard binary representation (which is not a bijection) in that leading zeroes do add numerical value. It provides a natural way of enumerating strings of any length, where the empty string ϵ corresponds to the number zero. It also enables us to define the Kolmogorov complexity (Definition 8) for natural numbers as well as for strings.

Definition 7 (Universal Turing Machine, p.30)
A **Turing Machine** *is a device that manipulates symbols, which can be zero, one or blank, on a one-dimensional tape according to a finite program—i.e. set of rules—with a read/write head. At the beginning of a run, the tape is filled with blanks, except for a finite interval to the right of the read/write head of the machine. The string in this area, consisting of zeroes and ones, is called the* **input**. *If the machine halts, then the (finite) string found on the tape is the* **output**. *A* **Universal Turing Machine** *(UTM) is a Turing machine that can emulate the behaviour of any other Turing machine.*

Example 3 *The author's favourite programming language* ANSI C *(as most other programming languages) can be viewed as a UTM, since any Turing machine can be encoded in it. It clarifies thinking to regard universal Turing machines to simply be programming languages.*

Definition 8 (Kolmogorov Complexity, p.106)
The **Kolmogorov Complexity** $C_U(\mathbf{s})$ *of a string* $\mathbf{s}$ *with respect to a universal Turing machine* U *is defined as the length of the shortest input—or program—*π *to* U *that prints* $\mathbf{s}$ *and then halts. We write*

$$C_U(\mathbf{s}) = \min_{\pi:U(\pi)=\mathbf{s}} |\pi|. \tag{3.21}$$

Kolmogorov complexity is defined only with respect to a UTM U. However, its dependence on U is bounded by an additive constant, regardless of the length $l(\mathbf{s})$ of a string $\mathbf{s}$.

Lemma 3 (Uniqueness of the $C_U(\mathbf{s})$ up to a constant, p.111)
For any two UTMs U *and* U' *and all strings* $\mathbf{s} \in \mathbb{B}^*$ *we have*

$$C_{U'}(\mathbf{s}) - c_{U',U} \leq C_U(\mathbf{s}) \leq C_{U'}(\mathbf{s}) + c_{U,U'}. \tag{3.22}$$

The constants $c_{U',U}$ and $c_{U,U'}$ only depend on the universal Turing machines U and U' and are given by the length of a program in language U to interpret U' and vice versa. For this reason, we can speak of *the* Kolmogorov complexity $C(\mathbf{s})$, remembering that it is defined only up to an additive constant.

Example 4 *The string* $\mathbf{s} = \{01\}^{1000}$ *of length* $\ell(\mathbf{s}) = 2000$ *has low Kolmogorov complexity* $C(\mathbf{s}) \ll 2000$ *since there is a short program to produce it:*

```
for(i=0;i<1000;i++)printf("01");
```

For the same reason the number $n = 2^{1000}$ *has low complexity, i.e. contains little information.*

Lemma 4 (Upper bound to $C(\mathbf{s})$, p.108)
There is a constant c *such that for all strings* $\mathbf{s}$ *of length* $\ell(\mathbf{s})$

$$C(\mathbf{s}) \leq \ell(\mathbf{s}) + c. \tag{3.23}$$

This constant obviously depends on the chosen UTM, but can be assumed to be small. The above is easy to see, as the program

```
printf("%s",s);
```

provides the required.

Lemma 5 (Almost all strings are incompressible, p.117)
The fraction of strings which are compressible by at least k *bits is smaller than* $2^{-(k-1)}$.

Consider the set of strings of length n. There are 2^n such strings, but only at most $2^{n-(k-1)} - 1$ descriptions of length $\leq n - k$. This proves the point for any n, and therefore also for the set of all strings.

This means that, if we generate a random string, e.g. by tossing a fair coin, it will almost certainly be incompressible. In contrast to that, our experience tells us that many data we come across in real life are highly regular and therefore compressible by a large amount. The world is far from random. Certainly, this holds for natural languages, which are highly structured, as well as contain a lot of redundancy, as discussed in [Shannon 1948]. We will exploit this fact in Chapter 4 of this work.

Remark 6 *For any string* $\mathbf{s}$*, its shortest description* $L_{min}(\mathbf{s})$ *(of length* $C(\mathbf{s})$*) is incompressible. Otherwise, there would be a shorter description of length* $C(L_{min}(\mathbf{s}))$*. Therefore, any compressible description of* $\mathbf{s}$ *is suboptimal. Shortest descriptions look random.*

Lemma 6 ($C(\mathbf{s})$ **is incomputable, p.127)**
The Kolmogorov complexity $C(\mathbf{s})$ *of a given string* $\mathbf{s}$ *cannot be computed.*

Consider the expression

```
The smallest integer which cannot be described in ten words.
```

Seemingly, this is a contradiction. Have we not just described this number in ten words? The proof deduces, that the above cannot be a description. Hence, we cannot *compute* the number from it. More formal proof is provided by the referenced book.

But how can this be? Given a UTM U, can we not simply run it on all inputs in parallel, until the shortest program to produce a given string $\mathbf{s}$ has halted? In fact, in this way we can compute a decreasing sequence of upper bounds for $C_U(\mathbf{s})$.

Lemma 7 (Semicomputability of $C_U(\mathbf{s})$**)**
The Kolmogorov Complexity $C_U(\mathbf{s})$ *can be approximated from above.*

This can be seen as follows. Let a UTM U and a string $\mathbf{s}$ be given. Run $U(0)$ for one time step. Then run $U(0)$ and $U(1)$ for one time step each. Continue to add the next input n and run each program $U(0), \ldots, U(n)$ for another time step, until one of them halts with output $U(i) = \mathbf{s}$. Any such program provides an upper bound for $C_U(\mathbf{s})$ and we can abort all runs of $U(n)$ with $n > i$. We only need to execute the programs that remain running to see whether an even shorter description—a new upper bound

for $C_U(\mathbf{s})$—can be found. Eventually, this bound will be tight and we will have found the shortest description of $\mathbf{s}$.

But does this not contradict Lemma 6? Have we not just described a way to compute $C_U(\mathbf{s})$? Unfortunately, this is not the case. Yes, eventually we will find the shortest description, but we will not *know* that we did. The problem is, that the above procedure never terminates, as some of the programs never stop running. In effect, they get stuck in infinite loops.

Lemma 8 (The Halting Problem, p.34)
Given a Turing machine T, its halting set is defined as the set of inputs n for which $T(n)$ eventually halts, $H_U = \{n : T(n) \text{ halts}\}$. There is no program to decide for a given number n whether $n \stackrel{?}{\in} H_U$.

Some of the programs of length $< C_U(\mathbf{s})$ will therefore run indefinitely. And since we cannot know whether any of them will halt (with output $\mathbf{s}$), we never know, whether an even shorter description exists.

This is very much the dilemma in which an MDL researcher finds himself every day. We have found *a* description of a given data, but are uncertain of its actual quality. It is soothing to know that we are not alone, we simply *cannot* know whether a shorter description exists. But at least we have an objective measure of quality, namely the codelength. So we can always compare and happily shout out: 'mine is shorter than yours!'

Definition 9 (The Universal Distribution, p.273)
We can define the **Universal Distribution** *with respect to a UTM U to be*

$$P_U(\mathbf{s}) \propto 2^{-K_U(\mathbf{s})},$$

where $K_U(\mathbf{s})$ is the **prefix complexity** [p.202]*, the length of the shortest selfdelimiting program for U that prints $\mathbf{s}$ and halts.*

Of course, this distribution cannot be computed. But it has some remarkable properties. First of all, most random strings coming from it are in fact compressible. Remember that by Lemma 5 almost all strings are incompressible, yet a universal distribution assigns high probability exactly to those strings that *can* be compressed. Therefore it reflects our experience that many data we *encounter* are highly regular.

Also, the universal distribution dominates all computable distributions up to a multiplicative constant, which depends on the length of the program π for U that generates that distribution.

We can imagine the universe as a universal Turing machine U which runs on random inputs π from a distribution[3] defined by $P(\pi) \propto 2^{-\ell(\pi)}$. What we *observe* will then be the outputs $U(\pi)$ for those inputs $\pi \in H_U$ on which the machine halts.

There is another remarkable property of the[4] universal distribution: it equalizes worst- and average-case.

Lemma 9 (Worst-case and Average-case are identical, p.290)
With respect to a universal distribution, the average-case performance of any algorithm with respect to any objective is the same as its worst-case performance.

The reason for this is, loosely speaking, that being the worst-case input to the algorithm is almost a description—of constant length—of a string $\mathbf{s}$. To specify such $\mathbf{s}$, we must only provide its length. For any given length, the universal distribution reserves a constant fraction of probability mass for the worst-case, dragging the average-case down to the same level.

Example 5 (Quicksort, p.291)
The sorting algorithm quicksort *is known to have a worst-case running time of $\mathcal{O}(n^2)$, quadratic in the size n of its input. Its average-case performance (with respect to a uniform distribution over inputs of size n) is of complexity $\mathcal{O}(n \log n)$. It has been noticed though, that the worst-case behaviour does surface, and more often than one would expect. Under the assumption that real-world data come from a universal distribution, Lemma 9 explains this fact.*

[3] In order to define this probability distribution, $\ell(\pi)$ needs to be such that $\sum_{\pi=0}^{\infty} 2^{-\ell(\pi)} < \infty$, e.g., a selfdelimiting (naive) description.

[4] we can speak of *the* universal distribution, defined up to a multiplicative constant, in the same way that we can speak of *the* Kolmogorov complexity, cf. Lemma 3.

3.4 Normalized Maximum Likelihood

As we have seen in the previous section, there are two major drawbacks to Kolmogorov Complexity. Firstly, it is only defined up to an additive constant, which depends on the chosen universal Turing machine. This constant may be of substantial weight, when—instead of asymptotics—we are interested in the complexity of actual, real-world data of limited size. Secondly, Kolmogorov Complexity is incomputable. And while it can be approximated from above, Lemma 7 supplies no way of doing so *in practice*.

Therefore, instead of using the theoretically optimal minimum description length $C(\mathbf{s})$, which uses a *universal* Turing machine, we need to retract to something less universal—but more practical. In the following, we introduce the **Stochastic Complexity**, which is defined with respect to a model class of our choice. It can be used to objectively choose among model classes. Once the model class has been fixed, it also defines a probability distribution, the **Normalized Maximum Likelihood (NML)**.

In a sense yet to be specified, this is the optimal way of describing the data, discovering its regularities and predicting unobserved entities. The quality of this approach will depend—up to the usual constant we deal with in the theory of Kolmogorov complexity—only on the suitability of the model family under consideration.

In 1987, Yuri Shtarkov has proposed the following minimax problem:

$$\min_{Q\in\mathcal{C}} \max_{|\mathbf{D}|=n} \log \frac{\hat{P}(\mathbf{D}|\mathcal{C})}{Q(\mathbf{D})} = \min_{Q\in\mathcal{C}} \max_{|\mathbf{D}|=n} \Big(-\log Q(\mathbf{D}) - \Big(-\log \hat{P}(\mathbf{D}|\mathcal{C}) \Big) \Big), \tag{3.24}$$

[Shtarkov 1987], where $\hat{P}(\mathbf{D}|\mathcal{C})$ is the maximum likelihood (2.32).

Definition 10 *Given a model class $\mathcal{C}$, data $\mathbf{D}$ and a distribution Q we call the quantity*

$$\log \frac{\hat{P}(\mathbf{D}|\mathcal{C})}{Q(\mathbf{D})} = -\log Q(\mathbf{D}) - \min_{Q'\in\mathcal{C}} \big(-\log Q'(\mathbf{D}|\mathcal{C}) \big) \tag{3.25}$$

the **regret** *of Q for $\mathbf{D}$ relative to $\mathcal{C}$.*

The regret is the number of excess bits we need to encode $\mathbf{D}$ using Q, instead of the best possible distribution $\hat{P}(\mathbf{D}|\mathcal{C})$ in $\mathcal{C}$, which we cannot know *before* seeing the data.

Let us stare at (3.24) for a while. Given model class $\mathcal{C}$, we play the minimax game by picking a distribution Q, against an opponent who picks a data set $\mathbf{D}$ of size n and a distribution $\hat{P} \in \mathcal{C}$. We move first, and pick

any distribution Q (which need not be a member of $\mathcal{C}$). Then our opponent picks data $\mathbf{D}$ and encodes it with hindsight, using the best compressing model $\hat{P}(\mathbf{D}|\mathcal{C}) \in \mathcal{C}$ there is in the class. We, in turn, use Q to encode $\mathbf{D}$, a distribution we had to pick before we knew what we were going to be using it for. We try to minimize the regret—the difference of the two codelengths—, i.e. the number of bits we need more than the max-player. Our opponent tries to maximize the regret. When picking Q, we therefore minimize the **worst-case** regret.

Shtarkov also provided the unique solution to (3.24), the Normalized Maximum Likelihood (NML) distribution (a.k.a. the Shtarkov distribution) given by

$$P_{\mathrm{NML}}^{\mathcal{C}}(\mathbf{D}) = \frac{\hat{P}(\mathbf{D}|\mathcal{C})}{R(\mathcal{C}, n)}, \tag{3.26}$$

where the normalizing constant $R(\mathcal{C}, n)$ depends only on model class $\mathcal{C}$ and data size n. The regret $\log R(\mathcal{C}, n)$ is therefore the same for any data of given size, and is also being called the **parametric complexity** of $\mathcal{C}$ for data size n. It is easy to see that this in fact does solve the minimax problem. Any distribution Q differing from $P_{\mathrm{NML}}^{\mathcal{C}}$ is bound to have larger regret for *some* data. Since both Q and $P_{\mathrm{NML}}^{\mathcal{C}}$ are distributions—i.e., sum to unity—there must be some data $\mathbf{D}$ such that $Q(\mathbf{D}) < P_{\mathrm{NML}}^{\mathcal{C}}(\mathbf{D})$ and therefore Q's worst-case regret is larger than that of $P_{\mathrm{NML}}^{\mathcal{C}}$.

On the other hand, there are also data $\mathbf{D}'$ for which the opposite is true and Q achieves lower regret. Hence, $P_{\mathrm{NML}}^{\mathcal{C}}$ is not only optimal in the worst-case, but also the worst in the best case. The NML distribution spreads the regret evenly across all data. It is the only distribution that has constant regret, while any other distribution has larger regret on some, and smaller regret on other data.

The denominator—normalizing the maximum likelihood—is given by

$$R(\mathcal{C}, n) = \sum_{|\mathbf{D}'|=n} \hat{P}(\mathbf{D}'|\mathcal{C}). \tag{3.27}$$

The codelength

$$-\log P_{\mathrm{NML}}^{\mathcal{C}}(\mathbf{D}) = -\log \hat{P}(\mathbf{D}|\mathcal{C}) + \log R(\mathcal{C}, n) \tag{3.28}$$

the NML distribution assigns to data $\mathbf{D}$ is called the **stochastic complexity** of $\mathbf{D}$ relative to $\mathcal{C}$, [Rissanen 1987].

Remark 7 *This definition of the NML distribution is sufficient for the multivariate multinomial data domain* $\mathbf{X}$ *we consider in this thesis. In general,* $P_{NML}(\mathbf{D}|\mathcal{C})$ *can also be a probability* density*, in which case* $R(\mathcal{C}, n)$

is given by an integral over all appropriate data. In many such cases—as well as in many cases of countably infinite data range—$R(\mathcal{C}, n)$ can be infinite and we have to restrict the data domain in order to define the stochastic complexity. But where it is defined, it still solves the minimax problem (which may turn into an inf-sup *problem), see* [Rissanen 2007]. *For the scope of this work, the above is well-defined.*

Remark 8 *The NML distribution, like the marginal likelihood distribution for Bayesian networks, mimics the behaviour of* all *distributions $\mathcal{M} \in \mathcal{C}$. But not with respect to an assumed prior distribution, but worst-case optimally relative to the model class. Unlike the marginal likelihood, the NML distribution does not share the parametric structure of $\mathcal{C}$, and therefore it lies outside of the model class it mimics. In fact, it is a* **non-parametric** *distribution altogether. Nonetheless we can use it for prediction, by using the* **plug-in predictor**

$$P^{\mathcal{C}}_{NML}(\mathbf{d_{n+1}}|\mathbf{D}) = \frac{P^{\mathcal{C}}_{NML}(\mathbf{D} \cup \mathbf{d_{n+1}})}{\sum_{\mathbf{d'_{n+1}}} P^{\mathcal{C}}_{NML}(\mathbf{D} \cup \mathbf{d'_{n+1}})} = \frac{\hat{P}(\mathbf{D} \cup \mathbf{d_{n+1}}|\mathcal{C})}{\sum_{\mathbf{d'_{n+1}}} \hat{P}(\mathbf{D} \cup \mathbf{d'_{n+1}}|\mathcal{C})}. \quad (3.29)$$

Solution to Problem 5 (no parameter prior)
Being non-parametric, the NML distribution very elegantly avoids the problem of finding a suitable parameter prior: it does not need any*.*

In many cases, the NML distribution is the best we can do when we are using model class $\mathcal{C}$. Not knowing the data generating process, we aim to minimize the *expected* codelength. But with respect to which distribution? If there were a generating distribution and further it would be known to us, then there would be nothing to do. By Shannon's theorem, this distribution itself is the one we should use to encode the data, the expected codelength being its entropy. Instead, we may want assume that data come from a *universal* distribution which, of course, cannot be computed. For this reason, we retract to the use of a model class $\mathcal{C}$, of which we hope that it is able to capture the regularities appearing in $\mathbf{D}$, or at least a large portion of them. Worst-case optimality assures, that we have taken into account *all regularities* that $\mathcal{C}$ can capture. Remember that, with respect to a universal distribution, the worst-case *is* the average-case.

Worst-case optimality with respect to the regret therefore means average-case minimal codelength with respect to the universal distribution. While

the assumption that data come from a universal distribution is debatable, it still is a natural choice when there is no better candidate available. It is also a safe choice, as the universal distribution dominates any (computable) prior distribution we might have chosen. This, of course, only up to a constant, which depends on both the prior distribution it is compared against and the UTM with respect to which it is defined. Successful applications of the NML distribution in histogram density estimation [Kontkanen 2007b], image denoising [Roos 2005a], clustering [Kontkanen 2006], DNA sequence compression [Korodi 2005] and other areas have given evidence for this.

Naturally, there will also be regularities that $\mathcal{C}$ cannot capture, and a bad choice of model class yields a bad NML distribution. But NML can also be used for model class *selection*, by choosing the class within a family $\mathcal{F}$ which minimizes

$$l^{\mathcal{C}}_{\mathrm{NML}}(\mathbf{D}|n) = l(\mathcal{C}) - \log P^{\mathcal{C}}_{\mathrm{NML}}(\mathbf{D}). \tag{3.30}$$

As in Section 3.2, we need to first encode the model class itself, before the receiver knows the NML distribution we will be using.

The model class $\mathcal{C}$ chosen in this way is the one which best suits $\mathbf{D}$ and therefore, also the corresponding NML distribution $P^{\mathcal{C}}_{\mathrm{NML}}$ is the most meaningful among all NML distributions for $\mathcal{F}$.

3.5 More Properties of the NML Distribution

Invariance to parameter transformation
By definition—which does not employ the parametric form of model class $\mathcal{C}$— the NML distribution is invariant to any sort of parameter transformation. For instance, it is automatically identical for all Bayesian network structures belonging to the same equivalence class.

The regret as a penalty term
Let us unfold Equation 3.30 to

$$l^{\mathcal{C}}_{\mathrm{NML}}(\mathbf{D}|n) = l(\mathcal{C}) - \log \hat{P}(\mathbf{D}|\mathcal{C}) + \log \sum_{|\mathbf{D}'|=n} \hat{P}(\mathbf{D}'|\mathcal{C}) \tag{3.31}$$

and observe that it looks—apart from the sign, as now we are minimizing—a lot like Equation 2.31, the Bayesian penalized model class selection criterion. The NML cost (i.e. the negative penalty) then is

$$\mathrm{cost}_{\mathrm{NML}}(\mathcal{C}|n) = l(\mathcal{C}) + \log \sum_{|\mathbf{D}'|=n} \hat{P}(\mathbf{D}'|\mathcal{C}). \tag{3.32}$$

The first term—$l(\mathcal{C})$—is not actually related to the NML distribution, it simply comes from the encoding of the model class we have chosen. The second term—the parametric complexity (regret) of $\mathcal{C}$—measures the complexity of $\mathcal{C}$ in a very sensible way, as the sum over the maximum likelihoods assigned to any data (of given size) we might have observed. The NML model selection criterion (3.31) therefore chooses a model class $\mathcal{C}$ that allows for good fit of $\mathbf{D}$ *relative* to the fit of *any* data $\mathbf{D}'$.

The regret, as does the penalty term of the Bayesian information criterion, depends on the data size n. In fact, BIC has been viewed as an approximation to the stochastic complexity, see [Kontkanen 2003], but found to be inferior to other approximations. Also, BIC disregards the class encoding term $l(\mathcal{C})$, which can make a crucial difference [Roos 2009].

Conditional NML
For supervised learning tasks, where the receiver is assumed to be aware of the predictor data $\mathbf{d^1}, \ldots, \mathbf{d^m}$ (cf. Eq. 3.19) we can define the conditional NML distribution

$$P^{\mathcal{C}}_{\mathrm{NML}}(\mathbf{d^0}|\mathbf{d^1}, \ldots, \mathbf{d^m}) = \frac{\hat{P}(\mathbf{d^0}|\mathbf{d^1}, \ldots, \mathbf{d^m}, \mathcal{C})}{\sum_{|\bar{\mathbf{d}}^0|=n} \hat{P}(\bar{\mathbf{d}}^0|\mathbf{d^1}, \ldots, \mathbf{d^m}, \mathcal{C})}, \tag{3.33}$$

where $\hat{P}(\mathbf{d^0}|\mathbf{d^1},\ldots,\mathbf{d^m})$ is now the conditional maximum likelihood. In this way, we get a model selection criterion

$$\begin{aligned} &l_{\mathcal{C}}^{\mathrm{NML}}(\mathbf{d^0}|\mathbf{d^1},\ldots,\mathbf{d^m}) \\ &= l(\mathcal{C}) - \log \hat{P}(\mathbf{d^0}|\mathbf{d^1},\ldots,\mathbf{d^m},\mathcal{C}) + \log \sum_{|\bar{\mathbf{d}}^{\mathbf{0}}|=n} \hat{P}(\bar{\mathbf{d}}^{\mathbf{0}}|\mathbf{d^1},\ldots,\mathbf{d^m},\mathcal{C}) \end{aligned} \tag{3.34}$$

where the sum of the regret only goes over that part of the data in which we are interested.

NML to deal with sampling bias

The situation in which the data itself has an influence on the fact whether or not it will be observed by us, is known as the **sampling bias**. In such case, we often still want to model the underlying process that generated the data in the first place, excluding the sampling bias. Then we should—when we want to minimize the expected MDL codelength—weight the data $\mathbf{D}$ by their probability $w(\mathbf{D})$ of being observed, to arrive at an unbiased (weighted) NML distribution

$$P_{\mathrm{NML}}^{\mathcal{C},w}(\mathbf{D}) = \frac{w(\mathbf{D})\hat{P}(\mathbf{D}|\mathcal{C})}{\sum_{\mathbf{D}'} w(\mathbf{D}')\hat{P}(\mathbf{D}'|\mathcal{C})}. \tag{3.35}$$

This, of course, requires that we have a fairly good understanding of the nature of the involved sampling bias. [Grünwald 2009] investigates such problem, in which statistical data is to be used as evidence in a court of law. Part of this same data had led to the trial in the first place, introducing sampling bias: had the data been different, the case might have never been raised. Grünwald studies several different approaches, all leading to the conclusion that the evidence that had led to police investigation should have lesser weight in court than the evidence gathered during the investigation.

The basic assumption has been, that the police got involved only when the observed data $\mathbf{D}$ had become sufficiently extreme, that is, a function $f(\mathbf{D})$ had exceeded some threshold T. The point of time when that had happened gives us a fairly good estimate of T. The biasing weights are therefore binary,

$$w(\mathbf{D}) = \begin{cases} 1 & \text{if } f(\mathbf{D}) \geq T \\ 0 & \text{else} \end{cases}. \tag{3.36}$$

The unbiased NML distribution is then similar to the standard NML, only differing from it by a restriction in the sum of the regret

$$P_{\mathrm{NML}}^{\mathcal{C},T}(\mathbf{D}) = \frac{\hat{P}(\mathbf{D}|\mathcal{C})}{\sum_{\mathbf{D}':f(\mathbf{D}')\geq T} \hat{P}(\mathbf{D}'|\mathcal{C})}. \tag{3.37}$$

The larger the threshold T, the smaller the T-restricted regret. The T-restricted NML distribution assigns higher probability to $\mathbf{D}$ with increasing T, as it is a distribution over fewer possible data the higher the threshold becomes. Higher probability in turn means less decisive evidence in a court. This is in accordance with Grünwald's findings.

Computability

Unlike the universal distribution, NML is computable or, in case of continuous data, can be approximated to arbitrary precision. But its computation is demanding, in many cases forbiddingly so. First, we need to be able to efficiently compute the maximum likelihood $\hat{P}(\mathbf{D}|\mathcal{C})$ and then, for the regret, an exponential sum (resp. multidimensional integral) over all data we might have observed. This means that straightforward computation takes exponential time. In some cases, however, we can calculate the NML distribution in polynomial time, in a less brute-force way.

We have seen that we can hardly expect to ever get closer to the universal distribution than with NML. But whether we can compute the NML distribution efficiently, still depends on the model class under consideration. Section 3.6 reviews recent advances in NML computation.

3.6 Computing the NML Distribution

Computation through the sufficient statistics
We cannot efficiently compute the parametric complexity straightforwardly, as it involves an exponential sum or a multidimensional integral. However, the maximum likelihood under model class $\mathcal{C}$ depends on the data $\mathbf{D}$ only through its counts, see Section 2.2. Grouping the possible data $\mathbf{D}'$ of size n into sets corresponding to the same count vectors can therefore enable us to calculate the regret more efficiently.

In this way, [Rissanen 2000] has developed an NML criterion for linear regression and applied it to denoising problems. While the (worst-case) regret itself is infinite in this case, multiple levels of bounding the involved integral and renormalization lead to a criterion which is independent of the chosen bounds, as their influence on the resulting codelength is constant with respect to the model class. Calculation of this criterion can be done in linear time. Moreover, for orthonormal basis functions, it can be shown that the class which optimizes this criterion retains, out of n coefficients, the largest k for some $0 \leq k \leq n$. Therefore we can find the NML-optimal model in time $\mathcal{O}(n \log n)$, the computational complexity of ordering the coefficients. This has been exploited in speech signal [Rissanen 2000] and image [Roos 2005a] denoising.

For a single multinomial variable of cardinality K (cf. Section 2.2), [Kontkanen 2003] has proposed an algorithm that computes the NML distribution with time complexity $\mathcal{O}(n^{K-1})$. The same paper discovered a recursive formula, which improves this result to $\mathcal{O}(n^2 \log K)$.

For Bayesian networks of multiple multinomial variables, the situation becomes more complicated. Unlike the marginal likelihood (Eq. 2.28), the NML distribution does not factorize into local terms for each variable. However, for the naive Bayes network structure (e.g. top left network in Figure 2.6), [Kontkanen 2005] introduces an algorithm of time complexity $\mathcal{O}(n^2 \cdot \log K_0)$, where K_0 is the cardinality of the root node, again using a recursion formula. Once more, the same paper immediately improves upon its own result, with a fast Fourier transform to $\mathcal{O}(n \log n \cdot K_0)$.

Publication II of this thesis developed an algorithm to compute the normalized maximum likelihood for Bayesian forests, Bayesian networks in which any node can have at most one parent (cf. Section 2.6). Its time complexity is $\mathcal{O}(n^{K^*-1})$, where $K^* = \max_{i:\exists(j:i=Pa_j)}(K_i \cdot K_{Pa_i})$. Therefore the algorithm is polynomial in the data size n, but the degree of this polynomial depends on the maximal product of the cardinalities of an inner node (one that has a parent and at least one child) and its parent. This clearly is tolerable only when these cardinalities are small, e.g. for binary

variables. The reason for this behaviour lies in the non-factorizing nature of the regret. The bottleneck of the algorithm is at the inner nodes of the forest. Here, the problem of summing over all possible data $\mathbf{D}'$ of size n—weighted by their contribution to the regret—is equivalent to the problem of counting non-negative integer matrices with given marginals (row- and column-sums). The latter has been proven to be #P-hard, [Dyer 1997], which casts strong doubt on the existence of an algorithm that would be polynomial with respect to not only n, but also the cardinalities K_i of the involved multinomials.

Computation using generating functions

In the year following the appearance of Publication II, [Mononen 2008] developed a slightly faster algorithm for the same problem. However, its time complexity is of similar order and the algorithm is *not* polynomial in all input quantities. Interestingly, Mononen's algorithm takes an entirely different approach, employing generating functions [Flajolet 2009].

The idea to apply generating functions to NML calculation problems is due to Petri Kontkanen, who developed a linear-time algorithm for computation of the (single variable) multinomial NML [Kontkanen 2007a]. This has been exploited in histogram density estimation in [Kontkanen 2007b]. For another application, [Mononen 2007]—slightly—improved the time complexity of NML computation for the naive Bayes network structure to $\mathcal{O}(n^2)$, no longer depending on the cardinalities of the involved variables.

Approximations

While efficient algorithms for exact computation of the stochastic complexity have been developed—and successfully applied—for a number of relatively simple model classes, it remains a fact that NML computation is infeasible for many model classes that we would like to use. This gives rise to the question, whether in these cases we can at least calculate good approximations to this score.

As mentioned earlier, one can view the Bayesian information criterion (BIC, [Schwarz 1978]) as such an approximation. [Kontkanen 2003] compares it to two more refined approximations—Rissanen's asymptotic expansion [Rissanen 1996] and the Szpankowski approximation—in the cases of a single multinomial variable (with varying cardinalities) and the naive Bayes network structure. The latter approximation Kontkanen derives based on Szpankowski's theorem on the redundancy rate for memoryless sources [Szpankowski 2001], once more using the generating function trick. In all reported situations, the Szpankowski approximation turns out to be most

accurate, while Rissanen's asymptotic expansion is a better approximation than BIC.

For the case of general Bayesian networks—including the special case of Bayesian forests—the problem with NML computation is the fact that the sum of the regret does not factorize into local scores at each variable. [Myllymäki 2008] proposes to use the **factorized Normalized Maximum Likelihood** (fNML), an approximation to NML which encodes the data columnwise and normalizes each column separately, resulting in a codelength of

$$l^{\mathcal{C}}_{\mathrm{fNML}}(\mathbf{D}|n) = \sum_{i=1}^{m} l^{\mathcal{C}}_{\mathrm{NML}}(\mathbf{d_i}|\mathbf{d_{Pa_i}}, n), \tag{3.38}$$

with the definitions of Equation 3.26. This score can easily be computed, and is reported to perform favourably as compared to the marginal likelihood score. It remains unclear, how close to NML its factorized version fNML comes, since NML itself cannot be computed.

A similar approach is taken in [Silander 2009]. The **sequential Normalized Maximum Likelihood** (sNML), encodes the data rowwise and normalizes each sample separately:

$$l^{\mathcal{C}}_{\mathrm{sNML}}(\mathbf{D}|n) = \sum_{j=1}^{n} l^{\mathcal{C}}_{\mathrm{NML}}(\mathbf{d_j}|\mathbf{d_1}, \ldots, \mathbf{d_{j-1}}, n). \tag{3.39}$$

Sequential NML can be seen as a prequential score in the same way that Bayesian marginal likelihood can. But unlike the marginal likelihood, sNML *does* depend on the ordering of the encoded data samples, even when the chosen model class makes the i.i.d. assumption.

We may also combine fNML and sNML ('fsNML'), and normalize after the encoding of each matrix entry. In all cases, each normalizing sum only goes over the parts of the data that appear to the left of the conditioning bar, which greatly simplifies computation.

The idea to normalize in smaller chunks, in order to make the resulting score efficiently computable, makes NML-derived scores applicable to a wide range of model classes. Sequential NML, for instance, has been successfully applied to regression problems [Rissanen 2010]. When deviating from the standard definition of NML in this way, the resulting codelength is no longer worst-case optimal nor does the regret (or, rather, the series of local regrets) remain independent of the data at hand. For this reason, we can no longer speak of the parametric complexity *of a model class*. However, methods derived in this fashion seem to work very well in practice.

3.7 Summary

We have seen, how Shannon's source coding theorem and Kraft's inequality tie probabilities to codelenghts. Maximum probability becomes minimum codelength. The MDL principle is based on this observation, transforming the problem of data modeling into a problem of data compression.

Already in its simplest form—the so-called two-part codes—the MDL principle solves most of the problems we have encountered in Bayesian model class selection. Others are greatly simplified, made more explicit. Two-part codes offer a theoretically sound approach to data encoding, and enable us to avoid many pitfalls common in data modeling. We can compare any models (codes), regardless of their parametric structure, simply by comparing their compression capabilities. This can be done using only the data at hand, without any assumptions regarding the source they might have come from.

When we require an encoding to be universal in the sense that it can describe any given data (of appropriate form), then the shortest codelength we can achieve is, by definition, the Kolmogorov complexity. As its definition involves the use of a universal Turing machine, which may freely be chosen, Kolmogorov complexity is only defined up to an additive constant. Furthermore, it is incomputable. The theory behind it, however, has some important practical implications. It leads to the NML distribution, which often gives us the shortest codelength we can achieve in practice. This non-parametric distribution simulates the use of all models within a given model class—without being a member of it—using no prior on the model parameters. It can be also used as an objective tool to choose among competing model classes. Model class selection then boils down to the definition of a suitable model family to choose from, and computation of the stochastic complexity.

The NML distribution can be computed efficiently only for a few relatively simple model classes. But these classes do appear in real-world problems—often as subproblems—and a number of successful NML applications have been reported. In case we cannot compute the stochastic complexity exactly, this is no reason to throw in the towel. A variety of approximations have been developed and put to the test with encouraging results.

The following chapter introduces applications of MDL methods in the field of computational etymology, the study of the history of words. We are interested in phonetic sound change over time, across a family of kindred languages.

Chapter 4

Etymology

"Eurgh!"
—Arthur Dent, using a babel fish for the first time.

4.1 Motivation

Etymology is the study of the history of words. Our focus is on cross-language change of sounds, the way phonetics have developed over time within a family of languages descending from a common ancestor. Typically, this ancestor language is not known to us, as it dates far back into ancient past. The data serving as input to our methods takes on the form of **cognate sets**, groups of related words from kindred languages, as described in more detail in Section 4.2.

We introduce several models to investigate and evaluate these cognate sets. Our main point of departure is **alignment** of etymological data, i.e., identifying the symbols or sounds that correspond. In etymology, the alignment problem is different from alignment in Machine Translation [Och 2003], which seeks correspondence between words in the source and target languages. While there is some ambiguity also on the level of words, sounds appear in a much wider range of context. Moreover, in our problem setting we do not have a dictionary available to help in mapping sounds. Instead, we want to discover these correspondences, together with the contexts they appear in.

Given a raw set of etymological data, we aim to find the best alignment, in a sense yet to specified. Motivation for this starting point will be given in Section 4.3. As it turns out, our alignment models also include the rules of sound correspondence we are after, as it is these rules that enable us to

compress the data. The models, model classes and families we develop to this end are described in detail in Sections 4.4 and 4.5.

Sets of etymological data are found in digital etymological databases, such as ones we use for the Uralic language family. A database is typically organized into *cognate sets*. Each element of such cognate set is a word in one of the member languages of the family under consideration, all cognates forming such set are posited to be derived from a common origin and thus to be genetically related. This origin is some (unknown) word form in the assumed common ancestor language. The cognate sets are formed by linguists, and often there is debate about their correctness. While some word forms are clearly related, others are more distant, believed to belong to the set by some linguists, rejected by others. Each cognate set is—simplifyingly—assumed to have evolved from a protoform in a common ancestor language in a straight line, not to have been lost and reintroduced by borrowing. The languages in the family form a tree that describes their history of separation. This is a standard assumption and lean words, where detected, are not being included in the cognate sets.

Computational etymology poses several problems, including the discovery of *regular sound correspondences* across languages in a given language family and determination of genetic relations among groups of languages, both of which are subject of this work. Problems we only brush are the discovery of cognate sets and reconstruction of unobserved word forms. The latter splits further into diachronic reconstruction, i.e. reconstruction of protoforms for a hypothetical ancestor language, and synchronic reconstruction of word forms that are missing from a known language.

As we develop our alignment models at the sound or symbol level, we explicitly model correspondence of sounds. In the process of evaluation of these models, we also arrive at modeling relationships among entire languages within the family. Section 4.7 describes ways of inferring phylogenetic language trees, describing the assumed genetic interrelations between the languages in the family, from the models we have learned. Construction of phylogenies is studied extensively, e.g., by [Nakhleh 2005, Ringe 2002, Barbançon 2009]. Their work differs from ours in that it operates on manually precompiled sets of *characters*, which capture divergent features of languages within the family, whereas we operate at the level of sounds within words and cognate sets.

The problem of cognate discovery is addressed in [Bouchard-Côté 2009, Kondrak 2004, Kessler 2001]. Here we consider the etymological data—cognate sets—to be given. Different data sets may include different—even conflicting—cognate sets. We start out from a set of etymological data (or

more than one such set) for a language family *as given*. Our focus is on the principle of *recurrent sound correspondence*, as in much of the literature, including [Kondrak 2002, Kondrak 2003]. This means that we assume that whenever a sound has changed in some language, then this has happened in some specific context throughout the entire language.

Reconstruction, such as the protoforms given in [Rédei 1991], seems to be the most demanding of the fore-mentioned problems. To this date, it has been done purely by hand, essentially by making educated guesses. Our imputation method, introduced in Section 4.6, provides a first approach to automatically reconstruct missing word forms of known languages. Reconstruction of protoforms is a subject of future research. We do believe, that the MDL principle, together with a suitably adjusted imputation procedure, can provide a means to this end.

Comparative and historical linguists have worked on problems of etymology for centuries. Their methods have been applied to various language families with immense labour. The obtained results have been subject to debate and disagreement, as every linguist has introduced her personal expertise and—inevitably—bias and subjectivity.

Computational linguistics provide means to analyze data far more efficiently. Moreover, only some major language families have extensively been studied from the etymological perspective, while many others have not. Language families such as Indo-European have received more attention than others and have been studied in greater detail, mainly because there is more relevant data that has been collected, and this data has been available to scholars for a longer time. For language families that have been paid lesser attention, automatic analysis will allow linguists to obtain results quickly, providing a foundation for further, more detailed investigation. We have been the first to study the Uralic language family by computational means, but no longer are we alone [Honkola 2013].

Minimizing description length also adds a level of objectivity to the analysis, as we can measure performance in bits. Of course the data itself, as well as the choice of the model family under consideration, remain choices to be made—an inescapable source of subjectiveness. Our methods may uncover previously unrecognized regularities, but even in case they only validate previously established theories, this is a useful result. Because computational approaches differ in nature from traditional linguistic methods, a matching result always serves as a non-trivial, independent confirmation of correctness of traditional methods, or adds its voice to an open debate.

Computational methods can provide valuable tools for the etymological

community. From a linguistic point of view, the methods can be judged by how well they model certain aspects of etymology, and by whether the automatic analysis produces results that match theories established by manual analysis. Imputation—reconstruction of previously deleted cognates—also provides an intuitive method of model quality evaluation, much like cross-validation in statistics.

From an information-theoretic point of view, we evaluate a model (class) by the codelength it achieves. We will see that codelength correlates well with the above linguistic criteria, proving our approach to be consistent.

4.2 The Data

Our methods are applicable to any reasonable collection of cognate sets from any language family. In this work we focus on the Uralic language family, using the StarLing Uralic database [Starostin 2005] originally based on [Rédei 1991]. So far, we have also run our algorithms on *Suomen Sanojen Alkuperä* (SSA, 'The Origin of Finnish Words', [Itkonen 2000]), a Finnish-centered etymological dictionary, and the StarLing Turkic database. For simplicity, we present our results—as well as all example alignments—only for StarLing Uralic, on which we will take a closer look in the following.

The StarLing Uralic database consists of 1898 cognate sets containing cognates from 15 languages. In this work, we restrict the data to the 10 languages from the finno-ugric branch, namely **Estonian** (EST), **Finnish** (FIN), **Khanty** (KHN), **Komi** (KOM), **Mansi** (MAN), **Mari** (MAR), **Mordovian** (MRD), **Saami** (SAA), **Udmurt** (UDM) and **Hungarian** (UGR). Each cognate set is derived from an entry in [Rédei 1991], an example of which is given in Figure 4.1. For each language, there may be multiple word forms from different dialects given. The data is mostly stemmed, containing relatively little extraneous morphological material. This is not the case with all data.

jokse- 'laufen; ? läufig od. brünstig sein, sich paaren FU

?[Finn. *juokse-* 'laufen; fließen; geschwind vorwärtsgehen, fortschreiten; gerinnen (Milch); brünstig sein' *jouksuaika* 'Laufzeit, Brunst-, Brunftzeit' (*aika* 'Zeit'); est. *jookse-* 'laufen, rennen; fließen, lecken; zufallen, zuteil werden; belaufen (v. Begatten der Tiere, verächtl. auch von Menschen gebraucht)', *kala jookseb* 'die Fische gehen (zum Laichen) dem Fluß hinauf'] | ? lapp. 1. N *juk'sâ-* ~ *jǫk'sâ- -vs-* 'reach, overtake, obtain; hit', L *jåksâ-* 'einholen (mit Akk.), gelangen bis', K (479) T *juksï-*, Kld. *juχse-* 'einholen, erreichen, 2. (T. I. Itk., WbKKlp. 67) Ko. Suonikylä *jǫχše-* 'verschwinden, sich entfernen' | ? ung. *ív-* (1662: *juvnak*, Lipp.: Cal. 5: NySz.; 1689: *vivástul*, Teleki: FLél. 53: NySz; dial. *iv-, ví-, vív,- vij-*) 'laichen (Fisch)'.

Das ung. Wort gehört nur dann hierher, wenn seine ursprüngliche Bedeutung 'laufen' war. Zum Bedeutungswandel 'laufen' → 'läufig od. brünstig sein, sich paaren, laichen' vgl. ung. *űz* 'jagen, treiben, verfolgen' ~ *üzekedik* 'stieren, bocken, rammeln, brünstig sein'; wog. *χājt-* 'laufen' ~ 'brünstig sein', tschuw. *tʃ̆up-* 'laufen' ~ 'brünstig od. läufig sein'; dt. *laufen* ~ *belaufen, läufig*.

Als lapp. Entsprechung können alternativ zwei Wörter in Frage kommen.

Setälä: JSFOu. 14/3:15; Wiklund: MSFOu. 10:272; *s*-Laute 47; SzófSz.; Nyíri: Alföldi Tudományos Gyüjtemény 2:228—34, SKES; FUV; MSzFgrE; TESz.

Figure 4.1: Original data entry corresponding to the Finnish word stem *juokse-* ('to run') in [Rédei 1991].

StarLing compiles these entries into a more digestible format, an example is given in Figure 4.2. Entries subject to disagreement among linguists are indicated by (one or more) question marks. In the example

of Figures 4.1–4.2, the Hungarian form 'ív' is questionable according to [Rédei 1991]. The StarLing database, however, marks all other entries as uncertain.

Number: 189
Proto: *jokse-
English meaning: to run; be rutty, copulate
German meaning: laufen; ? läufig od. brünstig sein, sich paaren
Finnish: juokse- 'laufen; fließen; brünstig sein' ?
Estonian: jookse- 'laufen, rennen; belaufen; (von Fischen) zum Laichen dem Flus hinaufgehen' ?
Saam (Lapp): juk'sâ- ~ jǫk'sâ- -vs- (N) 'reach, overtake, obtain', jåksâ- (L) 'einholen, gelangen bis', juksî- (T), juχse- (Kld.) 'einholen, erreichen';
Hungarian: ív-, dial. iv-, ví-, vív-, vij- 'laichen (Fisch)'

Figure 4.2: Preprocessed data entry as found in [Starostin 2005].

Finally, we simplify this further by arranging the data into a matrix, parts of which are shown in Figure 4.3. We ignore question marks (for now) and restrict each language to its most prominent dialect[1]. The resulting matrix has many missing entries, as cognate sets need not contain entries for all languages in the family. In fact, there are very few that do. Missing word forms are marked by dashes. Each language has its own alphabet, which can be either print or phonetic alphabets, depending on the database. We prefer the latter, since we are interested in sound change, not orthographic conventions.

ID	EST	FIN	KHN	KOM	MAN	MAR	MRD	SAA	UDM	UGR
1	aas	-	-	aʒ́	-	-	-	-	aź	aszó
2	ahka	ahka	aš	-	-	ošə	ašo	-	-	-
3	-	-	-	voĺ	alā	-	-	-	vaĺ	ágy
4	aja	aja	-	voj	wujt	-	-	vuoggje	uj	-
⋮										
186	hŭüüp	hyypiä	jëwəɣ	-	jäpəw	-	-	jĭipp	-	-
187	-	-	ȧt	-	jīt	jət	-	-	-	-
188	jõgi	joki	jŏχ	ju	jē	joɣə	jov	jǫkkâ	ju	jó
189	jookse	juokse	-	-	-	-	-	juk'sâ	-	ív
190	-	-	-	-	jām	-	-	-	-	indul
191	jõudsin	joutsen	-	juś	josch	jükšə	lokśij	njuk'čâ	juś	-
⋮										
1894	-	-	-	-	aj	-	-	-	-	ajtó
1895	-	-	-	-	ŭmətɔ̆l	-	-	-	-	óv
1896	-	-	-	-	üńəwl	-	-	-	-	enyelëg
1897	huul	huuli	lul	-	-	-	-	sullâ	-	-
1898	-	-	-	-	χujt	-	-	-	-	hajt

Figure 4.3: Part of the cognate matrix serving as input to our methods.

[1] We have also run our algorithms on data including the multiple dialects, but the aim here is to just give one example of a possible input.

Our basic assumption is, that phonetic change is a regular process. In other words, if along the way from an ancestor to the observed language some sound has undergone change, say, an 'a' changing into an 'o', then this happens throughout the language. Any such change may be context-dependent, but never random. But if the underlying process is in fact deterministic, then why use probabilistic models in the first place?

First of all, there are several sources of noise present in the data. It has been compiled by linguists with subjective views, and in many cases these views do conflict. There are also 'dubious' entries, indicated with a question mark (or multiple ones), which may be due to a weak semantic link, a violation of expected phonetic regularity, or a little of both. Even where cognate correspondence is undisputed, it may still only be partial. There may be morphological information contained in a word form, that simply does not correspond. The StarLing data has been stemmed (mostly), but that still leaves the problem of *ossification*. By this we mean that a morph that once carried meaning of its own has over time become an integral part of a word and therefore can no longer be stemmed away.

Secondly, we cannot expect to be able to recover *all* relevant, deterministic rules of sound change from the data. Over thousands of years many such changes will have occurred, back and forth, each single change conditioned on context which might be long lost, as it has undergone changes of its own. An additional source of complexity in language evolution is known as *blocking*. Some phonetic change occuring in a language may have exceptions where some word form would become indistinguishable from another existing word, in which case the unaltered form is preserved. Clearly, we cannot learn such mechanism from the data we are using. Moreover, the data is rather small. For some language pairs there are only a few hundred cognate sets with entries for both languages. Some symbols within a phonetic alphabet may only appear once or twice in the whole data.

These considerations give motivation to use probabilistic models for this data. Our models need to be capable of allowing exceptions from each rule implicit to them, since rules can be expected to hold only to some degree.

4.3 The Alignment Problem

While etymological datasets, containing cognate sets, are readily available, the *alignment* at the sound level between two or more word forms in a cognate set is almost never given explicitly, as is the case with StarLing.

Instead, in linguistic handbooks one finds general *rules of derivation* from a parent language to daughter languages. The handbook, in turn, typically provides a handful of examples of the application of each rule, at best also mentioning how pervasive the rule is, i.e., whether there are exceptions—examples that contradict the rule. The actual alignment for most related forms in the database are therefore implicit to these rules.

This creates several immediate problems. First, the user of the database is left to her own devices to determine precisely how any two or more words in a cognate set are related—which rules of derivation give rise to the observed forms? By this we mean a **full sound-by-sound accounting** of relationships in the observed word forms, leaving no sound unexplained.

Secondly, the presented word forms may contain morphological material that is etymologically unrelated. Some databases give 'dictionary' forms, which usually contain extraneous morphological affixes, and thereby obscure which subpart of a given word form stand in etymological relationship with other members in the cognate set, and which do not. While some affixes may be 'obvious', many are not. As mentioned in Section 4.2, the StarLing Uralic database is relatively clean in this respect, but it does contain a lot of ossified affixes, parts of the given cognates that need not etymologically relate.

Most seriously, the posited rules are usually insufficient to explain the totality of the observed data. This is due to the complex nature of the underlying derivation processes, which limits the ability of database creators to capture the regularities in full, and to make them explicit as rules. The given rules may also have exceptions that are not mentioned (or acknowledged) in the handbooks, yielding incomplete explanation.

In fact, rules given in handbooks are usually idealized abstractions, covering only the *common* examples of relationship. Typically, the rules described apply cleanly only in the simplest and most straightforward examples, while in much of the data the rules have unexplained exceptions, or fail to hold altogether. While some amount of exceptions or failure may be insufficient grounds to categorically dismiss a rule, we must also account for the exceptions.

A central idea behind our automatic analysis is to extract the rules of correspondence directly from the data, with no side information given. We use etymological handbooks such as [Lytkin 1973, Sinor 1997] only to verify

our results, not to guide learning. Therefore our results are independent of linguistic bias, as far as this bias is not already part of the data. In other words, all rules we learn are inherent in the given corpus. Unlike, e.g., [Kondrak 2004] we also utterly disregard the semantics of the involved cognates.

Given a set of cognates, we know (or assume) that its members are genetically related, in some way and at least in some parts. But this relation is not given on a sound-by-sound level, only on the level of complete words. Since we are interested in phonetic change, we need to first find out which sounds correspond to each other. This is the alignment problem. In its simplest form, we align single sounds for a pair of languages. We equate sounds and symbols, assuming phonetic notation. We speak of *source* and *target* languages, even though in many cases our models—being symmetric—do not require this distinction. Let Σ be the source and T the target alphabets, $|\Sigma|$ and $|T|$ their respective sizes.

At the symbol level, an alignment is then a pair $(\sigma : \tau) \in \Sigma \times T$ consisting of a source symbol σ and a target symbol τ. We call such pair an *event* in order to distinguish it from an *alignment* of single cognates or the whole corpus. Some symbols may align with themselves ($\sigma = \tau$), while others may have undergone changes during the time the two related languages have been evolving separately ($\sigma \neq \tau$). Clearly, with this type of one-to-one alignment alone we cannot align a source word $\boldsymbol{\sigma}$ of length $|\boldsymbol{\sigma}|$ with a target word $\boldsymbol{\tau}$ of a different length $|\boldsymbol{\tau}| \neq |\boldsymbol{\sigma}|$. We also need *insertions* and *deletions*, for which reason we augment both alphabets with an empty symbol, denoted by a dot. We denote the augmented alphabets $\Sigma_{\cdot}$ and $T_{\cdot}$.

Typically, a sound will behave in a number of different ways depending on the context it appears in. For example, some symbol σ from the source alphabet maps to the same sound $\tau_1 = \sigma$ in the target alphabet in most cases, but changes into a different sound $\tau_2 \neq \sigma$ inbetween vowels and gets deleted at the end of a word. The number of different choices for each sound can be assumed to be small.

We can now align word pairs such as (*vuosi*, *al*), meaning "year" in Finnish and Khanty, in any of the following

v	*u*	*o*	*s*	*i*
|	|	|	|	|
a	*l*	.	.	.

v	*u*	*o*	*s*	*i*
|	|	|	|	|
.	.	*a*	*l*	.

v	*u*	*o*	.	*s*	*i*
|	|	|	|	|	|
.	*a*	.	*l*	.	.

as well as another 58 ways we have counted.

The rightmost alignment, for example, consists of the symbol pairs (v:.),

(u:a), (o:.), (.:l), (s:.) and (i:.). Our objective is to find the best alignment for each pair (or set) of cognates in the data.

Applying the MDL principle, the goodness criterion will be the overall number of bits needed to encode the aligned data. We will verify in several ways that this in fact yields good models, as evaluated from a linguistic point of view.

The models presented in the following have been introduced in Publications III–VI. However, we take the freedom to deviate from these papers—both in content and in notation—whenever this seems to make the concepts more clear or more recent results suggest to do so.

4.4 Baseline Model and Extensions

Following the MDL paradigm, we need to encode the complete data $\mathcal{D}$ in order to communicate it via a channel. As it is not meaningful to speak of an alignment without the underlying (unaligned) data $\mathbf{D}$, we also speak of the alignment $\mathcal{D}$. For a pair of languages, we want to minimize the code length of the ordered pairs of cognates together with the alignments we have chosen for each. Sending the pairs in order is an arbitrary choice with no effect on the alignments, the codelength differs from that of unordered transmission by the additive constant $\log n!$, where n is the number of cognates that source and target language share. For the receiver to be able to decode the message, the word boundaries must also be encoded. To this end we add the special character '$' to both alphabets, meaning 'end of word', the augmented alphabets we denote by $\Sigma_{\$.}$ and $T_{\$.}$. For the simplest of our model classes, which we call the 'baseline model', this character will only appear in the cognate-terminating event ($: $). In some cases we will also need a symbol to denote the beginning of each word, we choose the character '#'. For the baseline model, we do not need to encode the cognate-initial event (# : #), as we know where it will occur: at the beginning of the code, as well as after each event ($: $) as long as we have not yet communicated all n cognate pairs, but not elsewhere.

4.4.1 Baseline Model

Our simplest model class for the alignment problem considers the event space $\mathcal{E} = |\Sigma_.| \times |T_.| \cup (\$: \$) \setminus (. : .)$ as a multinomial variable of cardinality $K = |\mathcal{E}| = |\Sigma_.| \cdot |T_.|$. The end-of-word symbol can only align with itself, while simultaneous deletion and insertion is not meaningful in this context, as it could only increase codelength.

We encode the events using marginal likelihood with uniform prior. For an alignment $\mathcal{D} = ((\boldsymbol{\sigma}_1 : \boldsymbol{\tau}_1), \ldots, (\boldsymbol{\sigma}_n : \boldsymbol{\tau}_n))$ of $\mathbf{D} = ((\boldsymbol{\sigma}_1, \boldsymbol{\tau}_1), \ldots, (\boldsymbol{\sigma}_n, \boldsymbol{\tau}_n))$ with event counts $c(e)$ we therefore obtain a codelength of

$$l^K_{MarLi}(\mathcal{D}) = -\sum_{e \in \mathcal{E}} \log c(e)! + \log\left(|\mathcal{D}| + K - 1\right)! - \log(K-1)!, \qquad (4.1)$$

where

$$|\mathcal{D}| = \sum_{e \in \mathcal{E}} c(e) \qquad (4.2)$$

is the total number of events in $\mathcal{D}$, cf. Equation 2.16. Implicitly we have assumed that the number n of common cognates is known to the receiver, such that the end of the message is unambiguous during decoding. Of

course, we could also start the message with a selfdelimiting encoding of n. This would add a constant to the codelength. Since this has no influence on the resulting model, we choose to ignore this term.

4.4.2 Learning Procedure

We learn an alignment of the corpus—all cognate pairs present in both languages under consideration—in the following way. Starting from a random alignment of all cognate pairs we iteratively realign one cognate pair $(\boldsymbol{\sigma}_i, \boldsymbol{\tau}_i)$ at a time. We first subtract all involved events from the counts. These decreased counts we denote $\mathbf{c}^{-i}$ and, for notational ease, the aligned data excluding the i^{th} cognate pair $\mathcal{D}^{-i}$. We then align $(\boldsymbol{\sigma}_i, \boldsymbol{\tau}_i)$ optimally using dynamic programming [Bellman 1957] and reinsert the new events into the count vector. Note that this alignment is optimal only with respect to the 'frozen' model, based on $\mathcal{D}^{-i}$. For the complete $\mathcal{D}$-model it is optimal in most cases, but in some cases only a good approximation to the optimum.

Dynamic programming requires that we find intermediate states, for which it does not matter (in terms of cost) how we got there. In our case, these states mark a point at which we have encoded the first u symbols of the source word and the first v symbols of the target word. We arrange these states in an alignment matrix A of size $(|\boldsymbol{\sigma}_i| + 2) \cdot (|\boldsymbol{\tau}_i| + 2)$. Each cell stores the minimal achievable codelength of the corresponding partial alignment, recursively defined by

$$\begin{array}{rcll} a_{00} & = & 0 & \\ a_{u0} & = & a_{u-1,0} + L((\sigma_u : .)|\mathcal{D}^{-i}) & \text{for } u \geq 1 \\ a_{0v} & = & a_{0,v-1} + L((. : \tau_v)|\mathcal{D}^{-i}) & \text{for } v \geq 1 \\ a_{uv} & = & \min \left\{ \begin{array}{rcl} a_{u-1,v} & + & L((\sigma_u : .)|\mathcal{D}^{-i}) \\ a_{u,v-1} & + & L((. : \tau_v)|\mathcal{D}^{-i}) \\ a_{u-1,v-1} & + & L((\sigma_u : \tau_v)|\mathcal{D}^{-i}) \end{array} \right\} & \text{for } u, v \geq 1 \end{array} \tag{4.3}$$

and for $u, v \geq 1$ also which of the three choices (deletion, insertion, one-to-one alignment) achieves the minimum. For fixed i, we denote the word lengths be $N = |\boldsymbol{\sigma}_i|$ and $M = |\boldsymbol{\tau}_i|$. The bottom rightmost cell $a_{N+1,M+1}$ can only be reached from cell a_{NM}. This transition corresponds to event $(\$: \$)$. Cells $a_{u,M+1}$ for $u = 1..N$ and $a_{N+1,v}$ for $v = 1..M$ are disallowed, as any path through them would correspond to an alignment containing an event involving an end-of-word symbol together with some other symbol. Cell $a_{N+1,M+1}$ then contains the cost of the best alignment, backtracking through A gives the alignment itself. Figure 4.4 visualizes the alignment matrix.

	#	τ_1	. . .	τ_{v-1}	τ_v	. . .	τ_M	\$
#	0							x
σ_1								x
. . .								x
σ_{u-1}				$a_{u-1,v-1}$	$a_{u-1,v}$			x
σ_u				$a_{u,v-1}$	$\searrow$ $\downarrow$ $\rightarrow$ a_{uv}			x
. . .								x
σ_N								x
\$	x	x	x	x	x	x	x	$\searrow$ ■

Figure 4.4: Alignment matrix A. Each cell a_{uv} stores the codelength of the most probable alignment and a pointer to the cell we have come from to achieve it. Word beginnings are marked '#', word endings '$' and cells marked 'x' are disallowed.

The cost of moving from one cell to another (to the right, down, or both) is given by the increase in codelength caused by adding the corresponding event e. These differences in codelength are

$$l(e|\mathcal{D}^{-i}) = l(\mathcal{D}^{-i} \cup \{e\}) - l(\mathcal{D}^{-i}) \tag{4.4}$$

$$= -\log(c^{-i}(e) + 1) + \log(|\mathcal{D}^{-i}| + K), \tag{4.5}$$

corresponding to transition probabilities

$$P(e|\mathcal{D}^{-i}) = 2^{-l(e|\mathcal{D}^{-i})} = \frac{c^{-i}(e) + 1}{|\mathcal{D}^{-i}| + K}. \tag{4.6}$$

We realign all cognate pairs and update the model accordingly until the algorithm converges. Because this greedy method tends to get stuck in local minima, we use Simulated Annealing [Kirkpatrick 1983] with a suitable cooling schedule. Since the focus of this dissertation is on modeling, not optimization, we will not go into detail here. This learning scheme is common to all modeling approaches presented in this thesis. For some models, we have to make appropriate modifications which we will describe as needed.

4.4.3 Sanity Checking

We check the obtained codelengths against standard compressors (gzip, bzip2) and find that our models achieve shorter encoding, cf. Figure 4 of Publication IV. In this test, the input to the standard compressors has been the unaligned data. Our models also encode the alignments, which accounts for additional information. On the other hand, we can exploit the relatedness of the cognates, while standard compressors cannot. Since our models compress more, we conclude that this regularity outweighs the overhead from encoding the alignments. This is hardly surprising, but nonetheless soothing to observe.

The resulting alignments can also be checked by hand, using etymological handbooks and common sense. We find that in fact they already do make a lot of sense. There are no gold standards available for the alignment problem, since it is unclear what an optimal alignment should look like. Therefore, we cannot easily assign a score (apart from the codelength) to measure the quality of results. But the baseline model does find the vast majority of obvious correspondences.

Another way of looking at the results is through the $|\Sigma_\cdot| \times |T_\cdot|$ count matrices, examples of which are shown in Figures 2 and 3 of Publication III. We hope for these matrices to be sparse, as handbooks typically list only a small number of possible correspondences for each sound. Of course we also want to hit the right ones. Vowels should align with vowels, consonants with consonants, and in general sounds aligning with each other should be phonetically reasonably similar. For languages with similar alphabets we hope for a strong 'diagonal'. Further, there should not be too many deletions and insertions.

Judged by these linguistic criteria, the baseline model is definitely on the right track. Starting from complete randomness, the codelength criterion guides the algorithm to meaningful alignments.

Finally, we wanted to know how resistant these models are to noise. To this end, we performed the following experiment. We aligned Finnish with itself, independently introducing noise into both the source and the target level. The contaminated data was produced by sweeping through the data on each level once, at each point adding a random symbol—uniformly drawn from $\Sigma = T$—with probability ν and reading off a symbol from the original data with probability $1 - \nu$. We hoped to be able to align each (original) source symbol σ with the same symbol $\tau = \sigma$ on the target level, while the introduced noise symbols would be dealt with by deletions and insertions. Up to a noise level of $\nu = 0.85$, our models displayed the desired behaviour almost to perfection. For higher levels of noise the amount of

correctly found correspondences quickly decreased. This result proves the baseline model to be highly noise-resistant.

After these encouraging initial results with the baseline model, we now look at several extensions and improvements we have implemented.

4.4.4 NML

We want and expect the count matrix to be sparse, since typically a sound in the source language can correspond to only a small number of sounds on the target side, the context determining to which of these. In such case, the normalized maximum likelihood distribution yields slightly shorter codelength than the marginal likelihood, which is why we prefer to use it. With this simple model, it is easy to do the swap. The new codelength is

$$l^K_{NML}(\mathcal{D}) = -\sum_{e\in\mathcal{E}} c(e)\log c(e) + \log R(K, |\mathcal{D}|), \tag{4.7}$$

where $R(K, |\mathcal{D}|)$ is the multinomial regret, a special case of Equation 3.27.

Note that, instead of the constant data size n, here we have the total number of events, which does vary for different alignments due to insertions and deletions. Therefore, the parameters used during realignment are not given by the maximum likelihood as for the plug-in predictor (Eq. 3.29), and the change in codelength does not simplify. We have

$$\begin{aligned} -\log P(e|\mathcal{D}^{-i}) &= L(\mathcal{D}^{-i}\cup e) - L(\mathcal{D}^{-i}) \\ &= (c^{-i}(e)+1)\log(c^{-i}(e)+1) - c^{-i}(e)\log c^{-i}(e) \\ &\qquad + \log R(K, |\mathcal{D}^{-i}|+1) - \log R(K, |\mathcal{D}^{-i}|). \end{aligned} \tag{4.8}$$

Fortunately, we can precalculate the regrets for all possible total numbers of events and store them in a look-up table, instead of recalculating them every time we realign.

Switching from marginal likelihood to NML consistently decreases codelength. However, looking at the alignments and the count matrix, it is hard to tell the difference.

4.4.5 Codebook

As mentioned above, we expect the count matrix to be sparse. Therefore, it is beneficial to encode the positions of its non-zero entries separatedly, and subsequently the aligned data given this knowledge. Note that in the notation of this thesis—unlike that of Publications III and IV—this is not a

two-part code. Here we view the choice of the non-zero positions to define a model class. Let $\mathcal{E}^+ \subseteq \mathcal{E}$ be the set of positions of non-zero entries, their number be $K^+ = |\mathcal{E}^+|$. Then $\mathcal{E}^+$ is to be seen as the model class—or codebook—we use to encode $\mathcal{D}$. But before we can do so, we need to communicate $\mathcal{E}^+$, which can be done in

$$l(\mathcal{E}^+) = \log(K+1) + \log \binom{K}{K^+} \tag{4.9}$$

bits. We can then consider $\mathcal{D}$ to come from a multinomial of cardinality K^+.

The overall codelength—with the additional index 'CB' for codebook—then becomes

$$\begin{aligned} l_{CB,MarLi}(\mathcal{D}) &= l(\mathcal{E}^+) + l^{K^+}_{MarLi}(\mathcal{D}) \\ &= \log(K+1) + \log \binom{K}{K^+} \\ &\quad - \sum_{e \in \mathcal{E}^+} \log c(e)! + \log\big(|\mathcal{D}| + K^+ - 1\big)! - \log(K^+ - 1)!, \end{aligned} \tag{4.10}$$

in the marginal likelihood case, and

$$\begin{aligned} l_{CB,NML}(\mathcal{D}) =& l(\mathcal{E}^+) + l^{K^+}_{NML}(\mathcal{D}) \\ &= \log(K+1) + \log \binom{K}{K^+} - \sum_{e \in \mathcal{E}^+} c(e) \log c(e) + \log R(K^+, |\mathcal{D}|) \end{aligned} \tag{4.11}$$

in the case of NML encoding.

The alignment procedure remains unchanged. Only the transition costs (or probabilities) become slightly more complicated in cases where we have to add a new entry to the codebook. Without writing it down explicitly, be it mentioned that in both cases—$e \in \mathcal{E}^+$ and $e \notin \mathcal{E}^+$—the transition probability is given by Equation 4.4 as the increase in codelength of adding the corresponding event to $\mathcal{D}^{-i}$.

Using such codebook significantly decreases codelength. At the same time, the count matrices sparsify further. This is hardly surprising, since adding a new *event type* to the codebook comes with additional cost. But linguistic analysis (e.g., using handbooks) also reveals that the learned alignments have greatly improved. We take this as evidence of the adequacy of our approach.

4.4.6 Distinguishing Between Kinds of Events

Let us first some clarify some notation. We have spoken of *events* of being a single instance of some core alignment, say, (p:b). *Event types* in our notation are all such alignments throughout the aligned data $\mathcal{D}$, that is, all instances the (p:b)-entry in the alignment matrix corresponds to. In the following, we also need *event kinds*, sets of alignment types that share a certain characteristic. We consider four different kinds, namely one-to-one alignments, deletions, insertions and word boundaries. The data space $\mathcal{E}$ then splits into four parts, $\mathcal{E} = \mathcal{E}_{1-1} \dot\cup \mathcal{E}_{del} \dot\cup \mathcal{E}_{ins} \dot\cup \mathcal{E}_{\$}$.

In general, for $\mathcal{K}$ kinds, we have $\mathcal{E} = \dot\bigcup_{\kappa=1}^{\mathcal{K}} \mathcal{E}_\kappa$. We denote the sizes of these subsets of the data space $K_\kappa = |\mathcal{E}_\kappa|$, such that $K = \sum_{\kappa=1}^{\mathcal{K}} K_\kappa$. Similarly, the event types actually occuring split into $\mathcal{E}^+ = \dot\bigcup_{\kappa=1}^{\mathcal{K}} \mathcal{E}_\kappa^+$ with sizes $K_\kappa^+ = |\mathcal{E}_\kappa^+|$. We do not expect different kinds of events to behave similarly. That is, the codebook is likely to contain quite different fractions of event types for the different kinds. Therefore it is beneficial to separately encode $\mathcal{K}$ codebooks—one for each kind—of sizes K_κ^+. Encoding the data given these codebooks remains unchanged, yielding the new (NML) codelength

$$\begin{aligned} l_{CBs,NML}(\mathcal{D}) &= \sum_{\kappa=1}^{\mathcal{K}} l(\mathcal{E}_\kappa^+) + l_{NML}^{K^+}(\mathcal{D}) \\ &= \sum_{\kappa=1}^{\mathcal{K}} \left(\log(K_\kappa + 1) + \log \binom{K_\kappa}{K_\kappa^+} \right) - \sum_{e \in \mathcal{E}^+} c(e) \log c(e) + \log R(K^+, |\mathcal{D}|) \end{aligned} \tag{4.12}$$

with the index 'CBs' for (separate) codebooks, plural.

Once more, this reduces codelength, sparsifies the count matrix, and also improves the resulting alignments as seen from a linguist point of view.

4.4.7 Multiple Sound Alignment

So far, our models have not taken into account any of the context information on which phonetic rules generally condition. A first approach to do so is to align multiple sounds at a time. This is etymologically motivated, e.g. diphthongs in one language may correspond to single vowels in another, and has previously been done in [Kondrak 2003, Bouchard-Côté 2007].

While our MDL models find many 'true' multiple sound alignments (no gold standard exists, but some seem obvious), the actual strength of this approach lies in the context information it can capture. In order to exploit as much context information as possible in this way, we also model both

word boundaries explicitly, since they play an important role as conditioning context. As before, we denote the beginning of a word by '#', its end by '$'. Encoding both initial and terminal word boundaries is a slight extension to what has been described in Publications III and IV. The MDL cost will be defined in such a way, that this does not lead to excess codelength.

We still regard the event space $\mathcal{E}$ as a multinomial—when aligning d symbols at a time it has cardinality $K = |\mathcal{E}| \approx |\Sigma|^d |T|^d$. We choose to align at most two symbols at a time, such that this number is roughly a million. With the amount of data we have available, $d \geq 3$ seems unreasonable. An example of a two-to-two alignment—the word pair $(tuomi, toom)$ meaning 'bird cherry' in Finnish and Estonian—is as follows.

$$\begin{array}{cccc} \#t & uo & m & i\$ \\ | & | & | & | \\ \#t & oo & m & \$ \end{array}$$

Here we have captured diphthong-to-long-vowel rule $(uo : oo)$, which can be seen as a correspondence truly involving two sounds on the source side. The other two multiple alignments (initial unvoiced plosive remains unvoiced $(\#t : \#t)$ and terminal i vanishes) involve context information.

We distinguish between 16 different kinds of possible event types, as listed below:

$$\left\{ \begin{array}{cccc} (\# : \#) & (\#\sigma : \#) & (\# : \#\tau) & (\#\sigma : \#\tau) \\ (\sigma : .) & (. : \tau) & (\sigma_1\sigma_2 : .) & (. : \tau_1\tau_2) \\ (\sigma : \tau) & (\sigma_1\sigma_2 : \tau) & (\sigma : \tau_1\tau_2) & (\sigma_1\sigma_2 : \tau_1\tau_2) \\ (\$: \$) & (\sigma\$: \$) & (\$: \tau\$) & (\sigma\$: \tau\$) \end{array} \right\}. \tag{4.13}$$

The corresponding event subspaces are of largely differing sizes, event types involving two symbols on both source and target side outnumbering the simpler types. On the other hand, we expect few of these types ever to occur, since regular sound change requires relatively few rules. Therefore it becomes vital that we encode the codebooks separately, as described in Section 4.4.6.

The only real change to the cost function (apart from the definition of the event space) comes from the fact that we encode both word boundaries explicitly. But at each point of time during decoding, the receiver knows whether a new word is starting (after having read a '$') or not. Therefore we can split data $\mathcal{D}$ into two parts $\mathcal{D} = \mathcal{D}_{\#} \dot{\cup} \mathcal{D}_{\to\$}$, where the first part consists of kinds of the top row in (4.13) and the latter of the rest. We can then encode these parts separately, and the receiver will be able to

merge them back together. Denote the event space and codebook sizes accordingly, $K = K_{\#} + K_{\rightarrow\$}$ and $K^{+} = K^{+}_{\#} + K^{+}_{\rightarrow\$}$. The NML codelength then becomes

$$l_{2-2}(\mathcal{D}) = \sum_{\kappa=1}^{16} l(\mathcal{E}^{+}_{\kappa}) + l^{K^{+}_{\#}}_{NML}(\mathcal{D}_{\#}) + l^{K^{+}_{\rightarrow\$}}_{NML}(\mathcal{D}_{\rightarrow\$}). \tag{4.14}$$

The alignment procedure, in essence, remains the same. However, the steps we are allowed to take in the alignment matrix of Figure 4.4 are now not only single steps right, down and diagonal, but also double steps into the same directions as well as right-and-down knight moves. Therefore, the minimum of Equation 4.3 is now taken over eight values. A word boundary may now be part of an aligned pair of symbols. The transition costs are once more given by Equation 4.4.

Two-to-two alignment further decreases codelength. It finds context-dependent rules of sound correspondence that could not be captured aligning only single sounds. Among them, e.g. for Finnish:Estonian, diphthongs *uo*, *yö* to long vowels *oo*, *öö*, unvoiced plosives k, p, t remain unvoiced word-initially, elsewhere change into their voiced counterparts g, b, d, and so on. The count matrix becomes too large to look at as such. We choose to only look at the one-to-one alignments in matrix form and at all others as a list. These lists are reasonably short, containing little 'garbage'—entries that are not linguistically justified. Furthermore, all event types occuring more than once do make sense intuitively. The (one-to-one) count matrix becomes a lot more sparse, as the captured context rules reduce its entropy.

4.4.8 Separate Encoding of Affixes

To deal with the problem of poorly stemmed and/or ossified data described in Section 4.2, we need a means of encoding affixes without *aligning* them. Aligning parts of words that are not genetically related inevitably skews the distribution of events, and in doing so may keep a model from finding etymologically correct alignments in other parts of the corpus. These unrelated parts of cognates can be seen as noise in the data. From this point of view, we are simply facing a typical noisy data problem.

From the MDL perspective, noise is simply data that does not compress. When modeling data, noise often does have structure and is therefore compressible, but this structure is not of the type we are interested in. In our case, these 'nuisance affixes' share phonetic regularities of the language they belong to, but not the cross-language regularities we find by means of alignment. They can be encoded more compactly with a 'naive', single-language model.

The MDL way to separate noise from signal is to build two alternative models. One of them captures the regularities we wish to extract from the data, the other is a naive data model to account for parts of the data that do not display these regularities. The data is then encoded by a mixture of both, thereby separating them. This approach has been successfully applied to denoising problems, [Rissanen 2000, Roos 2005a]. We take a similar approach, briefly described in Section 6 of Publication IV.

We define affix codebooks PCB (prefix codebook) and SCB (suffix codebook), encoding non-relating cognate parts in a naive, non-aligning way. We encode separate codebooks for source and target language, each in the same way. For example, the codelength of the source language prefix codebook is

$$l(PCB_{source}) = l(|PCB_{source}|) + l_{NML}^{|\Sigma|+1}(PCB_{source}), \tag{4.15}$$

where $l(|PCB_{source}|)$ is the length of a selfdelimiting encoding of the number of prefixes in the codebook, and $l_{NML}^{|\Sigma|+1}(PCB_{source})$ is the stochastic complexity of the codebook viewed as coming in an i.i.d. manner from a multinomial distribution of cardinality $|\Sigma| + 1$. The prefixes need to be encoded including their boundaries, hence the cardinality of the multinomial: the size of the source alphabet plus one for the boundary symbol.

We choose to encode the codebooks such that they reflect the order in which they appear in the aligned data, with possible repetitions. We therefore do not need to encode the the actual affixes when encoding the data, after the affix codebooks have been transmitted. We do, however, need to communicate whether an affix is present at any point, on source, target or both sides, which can be done by introducing special events. This makes the code of the affix-free data slightly longer, but in sum we gain.

Finding the optimal alignment of a pair of cognates can still be done with the dynamic programming procedure as described in Section 4.4.2. In addition to the previously mentioned steps through the alignment matrix, we are now allowed to do 'hyper-jumps', that is, enter and exit the matrix at any given cell, at the associated cost coming from the implied change in codelength. An example of an affix-stemmed alignment is the following alignment of $(takki, takista)$ in Finnish and Estonian, assumed original meaning 'to be attached to'.

$\#t$	a	kk	i		\$
|	|	|	|		|
$\#t$	a	k	i	suffix	'sta\$'

'Denoising' the data in this way, makes the learned rules of correspondence cleaner. Although affixes are far from random and do contain information, this is not the information we are interested in, information about phonetic *correspondence*. Therefore—in this context—we may regard them as noise. The gain in codelength obviously depends on the corpus, for the StarLing Uralic database it is much smaller than for SSA. It is interesting as such, which parts of a cognate will be 'stemmed away' in this automated fashion.

4.4.9 Multilingual Alignment

Ultimately, we want to be able to reconstruct ancestor languages. To this end, we need to be able to align more than two languages simultaneously, three at the very least, two descendant languages plus their common ancestor. In principle, our models generalize to d-fold alignment—simultaneous alignment of d languages. Events become d-tuples of symbols, one from each involved language, and the alignment matrix becomes d-dimensional. But there is a fundamental problem to this approach. While the event space grows exponentially in d, the number of cognate sets with entries in all d languages decreases rapidly. In order to learn anything of use, we must use *all* correspondence data available, which means that many cognates will be missing from one or more languages involved.

But how do we treat missing data? And how do we prevent the counts from becoming much smaller, when the data spreads over a larger event space, providing deteriorating support to an increasing number of parameters?

Our approach is to define the cost function as the sum over each involved pair of languages,

$$l(\mathcal{D}^d) = \sum_{s<t} l(\mathcal{D}^2_{st}), \tag{4.16}$$

where $\mathcal{D}^2_{st}$ is the two-dimensional, aligned data restricted to source language Λ_s and target language Λ_t. Whenever either of the cognates $\boldsymbol{\sigma}_i$ and $\boldsymbol{\tau}_i$ is missing, the corresponding item i is not part of $\mathcal{D}^2_{st}$. This definition is valid for any of the aforementioned model classes and, as all of these are symmetric by nature, it does not depend on the ordering of the languages in $\mathcal{D}^d$.

The alignment matrix does become d-dimensional, therefore growing exponentially with d. The number of cells we can transition from to arrive at a cell a_{uv} (cf. Figure 4.4) is now (at most) $2^d - 1$ when aligning single symbols, and $3^d - 1$ when allowing two symbols to be aligned together.

The requirements for dynamic programming to produce the optimal d-dimensional alignment are still given, but the minimum in the analogue of Equation 4.3 is now being taken over a larger number of values.

Section 4.7 introduces phylogenetic language trees. As we choose them to be binary—each observed language forms a leaf, while each protolanguage has exactly two children—we always deal with three languages at a time. For this reason, it suffices for our purposes to set $d = 3$, which ensures efficient computation.

The cost function (4.16) is, strictly speaking, not a codelength. It is a sum of interdependently constrained codelengths, since we demand the two-fold alignments to be *compatible*. To illustrate this, let us look at the following example.

$$\begin{array}{cccc} v & uo & s & i \\ | & | & | & | \\ v & oo & s & . \\ | & | & | & | \\ . & a & l & . \end{array}$$

is a three-fold alignment of the triplet $(vuosi, voos, al)$ ('year') in Finnish, Estonian and Khanty. Three pairwise alignments of the same cognates might read as follows.

$$\begin{array}{cccc} v & uo & s & i \\ | & | & | & | \\ v & oo & s & . \end{array} \qquad \begin{array}{cccc} v & uo & s & i \\ | & | & | & | \\ . & a & l & . \end{array} \qquad \begin{array}{ccc} v & oo & s \\ | & | & | \\ . & a & l \end{array}$$

But, as these three are of different length, they do not correspond to any three-fold alignment and are therefore *incompatible*. In fact, we have to slightly modify the terms of the sum in (4.16), and reallow events of type $(. : .)$, simultaneous insertion and deletion. Incompatibility may also appear in more subtle situations.

We observe, that the pairwise codelengths involved in (4.16) are slightly larger than what we can achieve without the compatibility constraint. In this respect, three-fold (or more-fold) alignment differs from all other extensions to the baseline model we have introduced, all of which aim at decreasing codelength. Here, we simply build a tool for reconstruction. This tool—a collection of compatible models—utilizes all available data. Cognates present in only some of the languages still contribute their share to the regularities captured by this collection, indirectly influencing all pairwise distributions.

4.5 Featurewise Context Modeling

In Section 4.4 we have presented our baseline model and a series of freely combinable extensions to it. These models perform surprisingly well when evaluated against any of the techniques described in Section 4.4.3 above or Sections 4.6–4.7 below. They perform well, even though there are two important features of the data they cannot capture.

4.5.1 Larger Context

First of all, one-to-one alignment cannot find any *context dependent* rules. Two-to-two alignment only takes a very limited context into account, namely at most one of the neighbouring symbols. However, many etymological rules of correspondence condition on the context of a sound less locally. For instance, vowel harmony in Finnish demands that front vowels $\{\ddot{a}, \ddot{o}, y\}$ may not mix with back vowels $\{a, o, u\}$ in the same (uncompounded) word. This is a *long distance* phonological assimilation process and the models of Section 4.4 are unable to capture this type of rule.

The two-to-two correspondence model was a first (and successful) approach to take context into consideration, but it cannot easily be extended to larger context. In Publications V and VI, we develop a family of context-aware models, which take on a different structure from what we have seen so far. To our knowledge, the models described in the following are the first to ever capture longer range context in etymological data.

4.5.2 Phonetic Features

A second shortcoming of the models presented so far is that they operate on the symbol level, while phonetic rules typically operate only on some aspect—or *feature*—of a sound. As a result, we had to learn a number of rules, when in fact there was only one underlying phonetic process involved. For example, the diphthong-to-long-vowel rule in Finnish:Estonian had to be learned twice: $(uo : oo)$ and $(y\ddot{o} : \ddot{o}\ddot{o})$. This results in excess codelength as well as requires more data (support) in order to be captured by a model.

In the following, we operate on the *feature space*, where each symbol is represented as a vector of phonetic features. Each symbol participating in an alignment has a **Type** feature, a value in $\{K, V, ., \#\}$ denoting consonant, vowel, absence by deletion/insertion and word boundary. All consonants and vowels then have 4 features, each with a varying number of possible values as listed in Table 4.1.

The question may arise, why can the models presented in Section 4.4 not

consonant features

symbol	meaning	values
M	**Manner**	plosive, nasal, lateral, trill, fricative, sibilant, glide, affricate.
P	**Place**	bilabial, labiodental, dental, retroflex, velar, uvular.
X	**Voicedness**	– , +.
S	**Secondary**	none, palatalized, labialized, aspirated.

vowel features

symbol	meaning	values
V	**Vertical**	high, mid-high, mid-low, low.
H	**Horizontal**	front, central, back.
R	**Rounding**	– , +.
L	**Length**	reduced, very short, short, half-long, long.

Table 4.1: Consonant and vowel features of articulation.

operate on features instead of symbols? This is due to the fact that they cannot incorporate context. All the context they can see is the symbol itself (plus at most one neighbouring symbol for the two-to-two model), but they can see all of its features, although not separately. Take away this information and we cannot expect them to learn very much at all.

But for the context-aware models described below the situation is different. They are able to look at the relevant aspects of both intra-symbol and inter-symbol context. Each model class consists of a set of decision trees, one for each feature, encoding which aspects of the context are relevant to a specific feature of articulation. These models once more encode a pair of languages, but can also be extended to three or more languages in the way we have described in Section 4.4.9.

There is a number of possible variants for the context tree model classes, we give an example of a set of choices below.

4.5.3 Context Trees

We encode the separate features, in some fixed order, first source then target language. The relevant context is given by the model class, which holds a tree for each feature and each language level, source and target. The *decision nodes* of these trees correspond to a query of the value of a relevant feature at some position of either the source or the target cognate. We call the triplet (P, L, F) a *context*, where P is one of the positions that

the model may query, L is the *level* (either source or target) and F is one of the possible features.

We choose to encode the symbols (i.e., their features) in the order they come. We can only query symbols (and features of the present symbol) already encoded. Also, we are blind to past insertions and deletions. This restriction needs to be made to ensure we can optimally align using dynamic programming[2] as described in Section 4.4.2. Further, we encode the features in a specific order, starting with the **Type** feature.

The positions we allow are **Self** (of which we can only query features already encoded), **Previous Symbol** (which is not a dot), **Previous Consonant**, **Previous Vowel**, as well as **Self or Previous Consonant** and **Self or Previous Vowel** (previous iff **Self** is not a consonant/vowel). The definition of the set of positions a tree can query is crucial. Our choice is not the only one that is possible, nor is it the only one we have tried. It simply seems to be reasonable and working well.

For the consonant positions we can only query consonant features, vowel features for vowel positions. Both return $'n/a'$ if no appropriate symbol can be found, i.e., we run into the word boundary first. For the undecided positions **Self** and **Previous Symbol** we can query **Type** and any other feature, which may then also return $'n/a'$, if the symbol found at that position and level is of inadequate type to the query.

Each decision node/query splits the tree into a number of subtrees equal to the number of possible answers. In order to keep the trees from growing too wide, we also allow binary queries of the form (P, L, F, V), where V is a given value and the return value is either $true$ or $false$.

Each feature F at level L of a symbol σ or τ then takes a path through the decision nodes, starting from the root and arriving at the leaf node corresponding to its relevant context, as the feature tree defines it. Each leaf node contains a distribution over the values of F, as a multinomial of cardinality $|F|$, the number of values for that feature. This distribution is used to encode $F(\sigma)$, respectively $F(\tau)$. Figure 4.5 shows a context tree, encoding the context relevant to feature V (voicedness) in Estonian, conditioned on Finnish.

Each path from the root to a leaf of such tree encodes a context rule. For instance, following the leftmost branch from the root lets us arrive at a leaf immediately. The corresponding rule states that, if a Finnish consonant is voiced (Finnish Itself Voiced = $\oplus$), then so is its Estonian counterpart,

[2] A more recent implementation of the algorithm does not require this restriction, as long as we are allowed to query the **Type** feature only for the present and immediately preceding symbol.

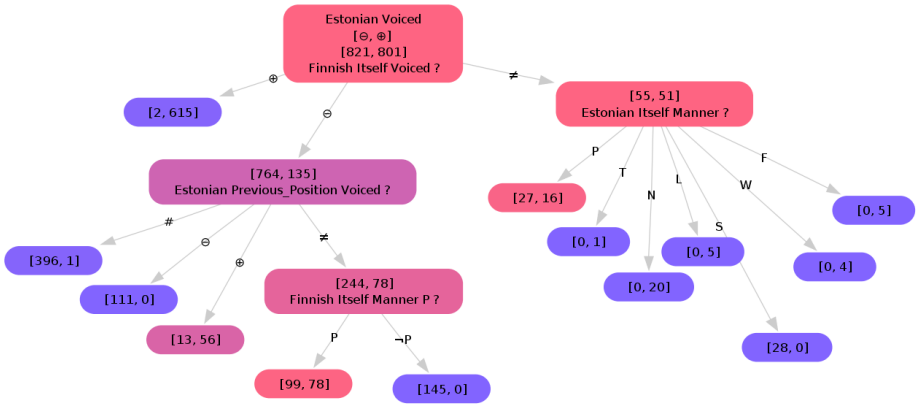

Figure 4.5: Context tree for the consonant feature 'Voicedness' in Estonian, conditioned on Finnish.

with high probability (615 observed instances against 2).

4.5.4 Codelength

We encode the model class—the trees—in a naive way, by simply listing for each node whether it is a leaf or not using one bit and, if it is not, which triplet (P, L, F) or quadruplet (P, L, F, V) is being queried. Encoding this decision costs the logarithm of the number of choices we can make here. This number depends on the feature to be encoded, and with our choice of contexts that can be queried amounts to a little more than eight bits on average. For a model class $\mathcal{C}$, a collection of 18 trees (two levels times **Type** plus four consonant and four vowel features), the corresponding codelength $l(\mathcal{C})$ is then the number of total nodes plus—roughly—eight times the total number of decision nodes.

This encodes the model class. Encoding the aligned data $\mathcal{D}$ given the model class can be done at the leaves of each tree. At every point in time during encoding—as well as during decoding—we are aware of the context already encoded. This context together with the model class (the structures of the trees) determines the leaf into which $F(\sigma)$ (or $F(\tau)$) falls. We can therefore encode the data at the leaves, one leaf at a time, and the decoder will know in which order to merge these feature events back together in

order to recover the original data. The total codelength is then

$$l_{context}(\mathcal{D}, \mathcal{C}) = l(\mathcal{C}) + \sum_{L} \sum_{F} \sum_{\ell} l_{NML}^{|F|}(\mathcal{D}_{|L,F,\ell}). \tag{4.17}$$

For the data part of the code, we sum over both levels L, all phonetic features F, and all leaves ℓ of the corresponding context tree in $\mathcal{C}$. For each leaf we apply the NML code to the data $\mathcal{D}_{|L,F,\ell}$—restricted to the current level, feature and leaf—as a multinomial of cardinality $|F|$.

4.5.5 Learning

To learn the model, we once more start from a random alignment of all cognates. We then build the decision trees as described below, to arrive at an initial model. This we use to realign all cognates as we have done in Section 4.3. Not being able to query dots (deletions/insertions) in the decision nodes ensures, that at any cell in the alignment matrix of Figure 4.4 the path we took to arrive at this cell has no influence on future transition costs, the requirement needed to ensure optimality of the alignment found by means of dynamic programming. After realigning a single pair of cognates, we only update the counts at the leaves of all feature trees, not rebuild those trees, which would slow the algorithm too much. We only rebuild the feature trees after realignment of the complete corpus. We alternate between tree rebuilding and corpus realignment until convergence. Again, to avoid premature convergence to a poor, but locally optimal solution, we employ simulated annealing with a suitable cooling schedule.

We learn the trees—given a complete alignment of the corpus—by iteratively splitting the feature data according to the cost-optimal decision in a greedy fashion. For given level L and feature F, we start out by storing all corresponding feature events e at the root node of the tree, e.g. for the voicedness feature V in Estonian (aligned to Finnish, cf. Figure 4.5) we store data with counts

+	801
-	821

In this example, there are 1.622 occurrences of Estonian consonants in the data, 801 of which are voiced. The best split the algorithm found was on '(Source, Self, V)', resulting in three leaves (or roots of new subtrees). The data now splits according to this context into three subsets with counts

+	
+	615
-	2

-	
+	135
-	764

n/a	
+	51
-	55

For each of these new nodes we split further, until no further drop in total codelength can be achieved. A split costs about eight plus the number of decision branches in bits, the achieved gain is the drop in NML cost obtained from splitting the data.

4.5.6 Evaluation

As the model classes have become much more potent in that they can take larger context into account, the codelength decreases considerably. As we still align on the symbol level—even though we encode each symbol by its feature vector—we may again plot the count matrix. When we did so, we noticed some strange behaviour. While for some runs on some data the corresponding count matrix looks sensible (sparse), for others it does not, far from it. Looking at the actual alignments reveals the problem: in the latter cases the algorithm finds shifted alignments, such as

$$\begin{array}{ccccc} v & u & o & s & i \\ | & | & | & | & | \\ . & v & o & o & s \end{array}$$

In these cases it does so consistently, which does not hurt in terms of codelength. For any symbol, the most relevant context information is obviously in the symbol which it correctly corresponds to. Encoding source first, then target given the source, the information about the according feature of the corresponding symbol is available to the target, but not to the source. Shifting the alignment to the left reverses this situation. Shifting to the right (as in the example above) makes some of the source future available to the target. Of course, shifting the available context window does not make additional information available in sum. Overall, we do not gain from shifting (on average), but it does no harm, either. However, shifted alignments make the models less readable, as we have to undo the shift in our heads before we can read off rules from the context trees. Moreover, as stated above, the count matrices no longer display the desired structure. Obviously, this behaviour is unintended.

In order to still obtain a sound-by-sound alignment, we have recently developed a post-processing method that has produced encouraging initial results. The rationale behind it is that the largest portion of the information relevant to some sound should be in the corresponding sound on the opposite level. We measure the information gain at each node along the root-to-leaf path for every sound and use dynamic programming to find, for each word pair, the alignment that maximizes the sum of information gains over all pairs of sounds involved in it.

We can now look at the count matrices and we find they do make sense. But they only show some aspects of the models, as they do not capture the context information as the models do. The context rules can be read off the feature trees themselves. Every path from the root of a tree that ends in a leaf with a distribution of low entropy provides us a rule. For instance, the rule that unvoiced consonants in Finnish remain unvoiced in Estonian word-initially, can be represented by the path (in the Voicedness-tree for Estonian, see Figure 4.5)

$$(Source, Self, V) \xrightarrow{-} (Target, Previous, Type) \xrightarrow{\#} \begin{pmatrix} + & | & 1 \\ - & | & 396 \end{pmatrix} \quad (4.18)$$

where the leaf distribution has very low entropy, 396 out of 397 instances of word-initial Estonian consonants corresponding to an unvoiced consonant in Finnish are unvoiced. Leaf nodes with high entropy represent unexplained variation.

4.5.7 Exploiting Monolingual Rules

Checking these rules, we also notice that some of them are not of the type we had hoped to find. While many are non-trivial rules of etymological correspondence, others are much more bland, such as the following rule for Finnish place of articulation of consonants.

$$(Source, Self, M) \xrightarrow{trill} \begin{pmatrix} bilabial & | & 0 \\ labiodental & | & 0 \\ dental & | & 52 \\ retroflex & | & 0 \\ velar & | & 0 \\ uvular & | & 0 \end{pmatrix} \quad (4.19)$$

This simply means that all Finnish trills are dentally articulated. Trivial, as the only trill in the Finnish alphabet is 'r'. In fact, rules of this type can be encoded more compactly by a *sound map*, an example of which is given in Table 4.2.

Manner→	plosive		nasal		lateral		trill		fricative		sibilant		glide		affricate	
Voicedness→	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
Place↓																
bilabial		p	m													
labiodental									v							
dental	d	t	n		l		r					s				
retroflex																
velar		k	ŋ										j			
uvular										h						

Table 4.2: A map of Finnish consonants.

The sound maps for the involved languages can safely be assumed to be known to the receiver of the encoded data. A first—yet to be implemented—approach to exploit this information in encoding is the following. When building a feature tree, simply erase all feature events from the data stored at the root node, which are completely specified using the previously encoded features of the same symbol together with the sound map. There is no additional information needed to decode these instances and hence we can save some bits here. At the same time, the feature trees should become smaller, better readable. Also, when encoding the data at a leaf, we can ignore impossible values. For example, there are no retroflexes in Finnish. When the cardinality of the multinomial feature variable can be decreased in this way, the regret shrinks and the code gets shorter. Unfortunately, not all situations are as clear cut. Some events will be specified only partially. How to deal with this type of situation is a subject of future research.

Another interesting direction for the future is to combine separate models for each involved language together with a pure correspondence model. Context models as described above can be just as easily built for a single language, simply by building the trees without any alignment. The allowed contexts to be queried are then restricted to a single level. In this way, we can find phonetic rules within the modeled language, e.g. for **Type** in Finnish the rule

$$\begin{aligned}&(Source, Previous, Type) \overset{consonant}{\longrightarrow} \\ \longrightarrow\ &(Source, Previous\ Vowel, Type) \overset{n/a}{\longrightarrow} \left(\begin{array}{c|c} vowel & 1332 \\ consonant & 0 \\ \# & 0 \end{array}\right)\end{aligned} \tag{4.20}$$

stating that word-initially, there cannot be two consecutive consonants. Taking this information out of the correspondence model would yield smaller trees, which are better interpretable and provide larger data support at each leaf.

4.6 Imputation

We have introduced a series of MDL methods, producing codes of decreasing length. Information-theoretically, it is interesting to see how the linguistic assumption of regularity of phonetic change helps us to compress the given data. We have also evaluated these models by the rules of etymological correspondence of sounds they are able to find. Finally, we have checked the alignments they produce by hand. As there exists no gold standard for the alignment problem—and we do not know how any such standard could be defined—we are left to our own devices in judging their quality. While we find that both the discovered rules and the alignments found by the models closely follow the length of the codes associated to them, this way of evaluating 'by hand' remains somewhat unsatisfactory.

For this reason, we have developed a more intuitively appealing, cross-validation type procedure of evaluation we call *imputation*. For a given model and a language pair—e.g., Finnish/Estonian—we hold out one word pair at a time and train the model on the remaining data. Then we show the hidden Finnish word to the model and let it guess, or *impute*, the corresponding Estonian form. We then compute the Levenshtein edit distance [Levenshtein 1966] between the imputed word and the withheld Estonian word, the minimum number of basic editing operations needed to produce one from the other. We impute all words of the target language in this way, sum the edit distances, and normalize by the number of symbols in the correct target data. This gives us the **Normalized Edit Distance** (NED). Table 1 of Publication V lists the (symmetrized) NED scores achieved by the context model on all language pairs in the StarLing Uralic database. Figure 6 of Publication VI compares the NED scores achieved by the context model to those achieved by the baseline model with codebook.

For the baseline model (and its extensions), we can use a dynamic programming procedure similar to that used for alignment to obtain an imputed word form which, when the true, erased word is replaced with it, results in the shortest overall codelength. The alignment matrix now simplifies to a vector of length equal to the length of the given word in the source language, plus two for the word boundaries. Allowed transitions are steps of size one (one or two for a two-to-two model) and the associated cost is given by the minimum cost over all events e which involve the corresponding symbol(s) on the source side. The target symbol(s) involved in the transitions of the cost-optimal path form the imputed word. In the same way, we can also impute conditioned on cognates in multiple languages, in which case the dimension of the imputation matrix equals the number of given word forms.

Following the same scheme with a context model does not necessarily produce the codelength-optimal imputation. Since the imputed target word contributes to the context—and therefore a partial imputation can influence future transition costs—the optimality requirement of dynamic programming is violated. Our solution to this problem is the following. At any point i during imputation, corresponding to having read off $\sigma_1, \ldots, \sigma_i$ from the given source word, we have stored the B best partial imputations. We now read off the next symbol σ_{i+1} in all $|T_\cdot|$ possible ways by encoding $\tau \in T_\cdot$ using the model and all B stored contexts. Out of these $B \cdot |T_\cdot|$ new solutions, we again keep the B ones with minimal cost. Theoretically, this does not guarantee optimal imputation, but in practice, for only one cognate given, we find that $B = 100$ suffices. We see this by checking that, say, $B = 1000$ yields the same solutions. In this way we can efficiently impute with the context model, even if optimality is no longer guaranteed.

Codelength and NED correlate very well. This is an encouraging indicator for codelength minimization to be a good approach. Our methods *do not* optimize NED directly, but operate on the codelength instead. Yet they produce models of low NED, a simple and intuitive measure of a model's quality.

Imputation will become a necessary tool in reconstruction of unobserved ancestor languages, which is the ultimate goal of our research. Even if we do not yet know how to automatically reconstruct protoforms (and nor does anyone else, to our knowledge), we feel that imputation provides a promising approach to this task, which we further discuss in Chapter 5.

The normalized edit distance also defines a measure of closeness between related languages, which we will use to construct *phylogenetic language trees*, as we describe in the following Section.

4.7 Phylogenetic Language Trees

Another way to verify the soundness of our approach is to let the models induce phylogenetic trees of the involved languages, representing the structure of their genetic relatedness. These can then be compared to trees found in the literature. Examples of such trees, for the languages of our focus, are shown in Figure 5 of Publication IV (adapted from [Anttila 1989]) and Figure 1 of Publication V (adapted from the Encyclopedia Britannica). Although the two agree in large parts, there are some differences. A large collection of phylogenetic trees for this language family—and others—can be found at [Multitree 2009], displaying even more different views on the subject. The evolution of the Uralic language family is subject to debate among linguists. So once again, there is no gold standard to compare our results to, but we intend to add a voice to the discussion.

Phylogenetic trees are typically binary, rooted and inferred from distance matrices. The observed languages serve as the leaves of such tree, the inner nodes are unobserved, common ancestor languages to the leaves of the subtree rooted in them. Binary trees suite our purposes well, as the maximum number of neighbours we need to consider in reconstruction is minimal for them. Several algorithms to learn binary trees from given pairwise distances are readily available, such as the Unweighted Pair Group Method with Arithmetic Mean (UPGMA, [Murtagh 1984]), NeighborJoin [Saitou 1987] and the Quartet Tree method [Cilibrasi 2006]. The latter two actually produce unrooted trees, but there is a standard way to root them. Adding a 'garbage node'—one with distance to all other nodes greater than any inter-language distance—provides a pointer to the position of the root. There are also several ways to define the language distances used to infer a tree.

Normalized edit distance is one obvious candidate. As this measure is asymmetric, we need to symmetrize the distance matrix, e.g. by taking the arithmetic mean over the distances in both directions. Figure 7 of Publication VI shows a phylogenetic language tree induced by NeighborJoin from the symmetrized normalized edit distances obtained by the context model.

To the information theorist, a more natural way to define pairwise languages distances is via the codelength itself. Not only does it involve the the (minus log-) probability of the correct target word (under the optimal alignment) instead of the Levenshtein distance to the imputed version. As the latter is discrete, it can be regarded less informative. Codelength also involves the complexity of the learned rules of correspondence between the two languages. The means we use to turn codelengths into a distance mea-

sure is the **Normalized Compression Distance** (NCD) [Cilibrasi 2007]. For Languages **S** and **T**, with corresponding common cognates $\mathcal{D}_{\mathbf{S}}$ and $\mathcal{D}_{\mathbf{T}}$, it is defined as

$$NCD(\mathcal{D}_{\mathbf{S}}, \mathcal{D}_{\mathbf{T}}) = \frac{l(\mathcal{D}_{\mathbf{S}} : \mathcal{D}_{\mathbf{T}}) - \min\{l(\mathcal{D}_{\mathbf{S}}), l(\mathcal{D}_{\mathbf{T}})\}}{\max\{l(\mathcal{D}_{\mathbf{S}}), l(\mathcal{D}_{\mathbf{T}})\}}, \tag{4.21}$$

where $l(\mathcal{D}_{\mathbf{S}} : \mathcal{D}_{\mathbf{T}})$ is the codelength achieved by the model we are using, and $l(\mathcal{D}_{\mathbf{S}})$ and $l(\mathcal{D}_{\mathbf{T}})$ are the codelengths achieved by a monolingual version of the same model. All models we have considered here can easily be restricted to encode a single language instead of two or more. Figure 6 of Publication IV shows a tree that was built by UPGMA with the NCD matrix for the baseline model. Figure 4.6 of this thesis, in turn, depicts a tree induced by NeighborJoin based on the normalized compression distances calculated using the context model, representing the phylogenetic structure of the Turkic language family.

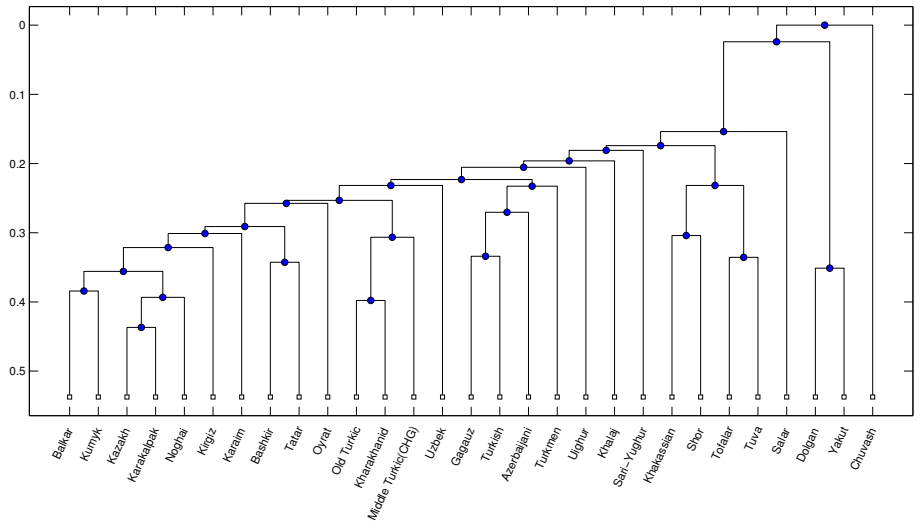

Figure 4.6: Phylogenetic language tree for the Turkic language family, induced by NeighborJoin from NCDs based on the context model of Section 4.5.

The phylogenetic languages trees induced by our methods are strikingly similar to the ones that can be found in the literature. The baseline model still makes a clear mistake by misplacing (the correctly found pair) *Mansi* and *Khanty*. The context model, however, makes no obvious mistakes, the

resulting trees lie within the variation among expert opinions, with equality for some.

We find this to be strong evidence for the validity of our approach. After all, we have induced these trees from raw data alone, using no prior assumptions, no semantic knowledge and no linguistic expertise in training the models.

Chapter 5

Summary and Current Work

"We can only see a short distance ahead,
but we can see plenty there needs to be done."
—Alan Turing, Computing Machinery and Intelligence.

The starting point of this thesis has been Bayesian reasoning, in particular its application to the task of model class selection. We have identified a number of problems related to this approach, such as the questionable assumption about the existence of a data generating distribution. Even if such distribution exists, it will not usually be a member of a model class under consideration, which can be especially harmful in supervised learning tasks such as classification. We further have to define prior distributions, both over the model classes we want to choose from and the models within each class. It is hard to do so in an objective way and, even in cases where we do have prior knowledge about the problem domain, the formulation of a prior can prove to be very difficult.

An elegant way around these problems is being offered by information theory, which enables us to look at data with no assumptions regarding their source. Following the minimum description length (MDL) principle, we regard the problem of maximizing probability as one of minimizing codelength, that is, data compression. Compression then is purely data-driven. As with any approach, we must first to define a suitable model class or model family to choose a class from. Then we define a method to describe the data as compactly as possible, an encoding scheme that gives good compression rate. This description method can be seen as corresponding to a prior of some sort, but typically this choice is more intuitive and straight-forward than the formulation of a prior.

The theory of Kolmogorov complexity provides proof for the fact that

there can be no automated procedure to optimally choose a model class. Regardless of the approach we are taking, we have to define a model family by hand, and choose an appropriate class from that. The MDL model class selection criterion is compression rate, we choose the class that achieves the shortest description length for the data at hand. But we can never be sure whether the class found in this way is optimal, model classes outside the family under consideration may achieve better compression rates. On the other hand, MDL methods make no prior assumptions about the nature of the data, its generating distribution or the suitability of the chosen model family or any of its members.

The problem of selecting a model within a chosen class or, equivalently, parameter estimation has turned into the problem of defining an encoding scheme. Where efficiently computable, we have a generic choice available, namely the normalized maximum likelihood (NML) distribution. Compared to the best model in the class under consideration, it achieves minimal excess codelength in the worst-case over all possible data. Under the assumption that data come from a universal distribution, Kolmogorov theory proves that the NML distribution also yields the shortest code on average, at least up to an additive constant depending on the universal Turing machine (UTM) with respect to which the universal distribution is defined. This assumption, while it may be called into question as such, is justified by the fact that any universal distribution, again up to a constant factor, dominates all other distributions.

This makes the NML distribution a valuable tool in data modeling. Unfortunately, computing this distribution is often infeasible in practice. But this does not hold for a number of relatively simple model classes. Publication II investigates the boundaries of NML computability for a specific family of Bayesian network models.

In case we cannot efficiently compute the NML distribution, we can often still use it for subproblems. Approximations of various type have also been reported to yield good model class selection criteria in an increasing number of applications. But with or without NML, we can always compare different encoding schemes objectively by looking at the resulting codelengths. Comparability is among the greatest strengths of the MDL approach, as it enables us to compare models of utterly different type and parametric structure by their compression capability.

Based on these findings, in Publications III–VI, we develop and evaluate a series of model families for the alignment problem arising in etymological data analysis. This is a novel approach, as for the first time information-theoretic methods are being applied to this problem. We utilize the MDL

principle and compress the raw data together with the alignment we are searching. In order to do so, we have to find and exploit the regularities of phonetic correspondence in the given data. This enables us to not only find good alignments, but also directly read off these rules from the models. The rules we find are probabilistic by nature. Although the basic, linguistically motivated assumption behind the problem is that inter-language change of sounds is a regular process, the data we are using contain noise from a variety of sources. Therefore, we cannot expect to find rules to completely explain the data in a deterministic manner.

Rules of phonetic variation are typically context-dependent, where the context relevant to a rule may restrict to the immediate neighbourhood of the sound in concerns, but often also involves longer-distance dependencies. The context model introduced in Section 4.5 is able to capture many such dependencies. It builds feature-wise decision trees, which explicitly encode the context it sees relevant to a rule. An obvious direction of further research is to make an even larger context available to these trees. For instance, we can model vowel harmony in Finnish only to some degree. Since in our present implementation, we can only query information about the vowel immediately preceding each position, we cannot 'see through' vowels $\{i, e\}$ behaving neutrally with respect to this rule. Phonetic rules may also condition on aspects on the syllable level, such as stress. At present, we are unable to capture these rules.

A problem we face in this research lies in evaluation of the obtained results. Neither does a gold standard exist for the alignment problem, nor are there obvious, objective ways to compare the performance of our algorithms to other methods. We develop several ways to give evidence for the validity of our approach and the quality of our results. From an information-theoretic perspective, the achieved compression rate measures the quality of any model. Using codelength as a criterion, we can compare our models among each other. But we have no outside milestone to compare them against. This calls for further, linguistically motivated evidence to indicate the quality of our results.

Checking phonetic rules discovered by the models by hand, or even inspecting the obtained alignments on the cognate level, is troublesome and hardly leads to any objective criterion for comparison. On the cognate level, we introduce an imputation procedure that supplies an intuitive quality measure: the normalized edit distance (NED) between the true and the model-imputed word forms. On the corpus level, we let our models induce phylogenetic trees of the involved languages, which we then compare to expert-built trees found in the literature. These trees can be constructed

by various algorithms that typically take a matrix of pair-wise language distances as input. We can produce such matrices using NED, but also directly from the codelengths. Normalized compression distance (NCD) provides the desired means here.

In all tests we have performed our models do well. On top of that, the codelength—the MDL criterion we optimize during learning—correlates strongly with any of the other criteria we have used in evaluation. Figure 5.1 shows a schematic diagram of the evaluation methods we have used, the bottom level containing the evaluation nodes.

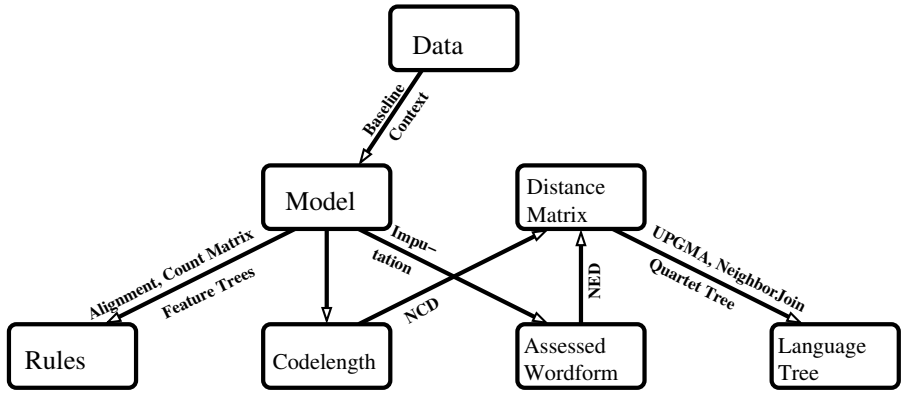

Figure 5.1: Flow chart of the used evaluation methods.

Another direction, as mentioned in Section 4.5.7, is to separate phonetic rules within each single language—including information about the alphabet the language is using—from the correspondence model. This should improve learning, as the models become simpler and better supported by the data. Also, this would improve model readability, giving a clearer picture of the involved phonetic processes. However, this requires an encoding scheme that can combine the rules encoded in the multiple models to ends of codelength minimization. At present it is unclear how to achieve this.

A more immediate application of our methods is that of cognate evaluation. Many of the entries in the corpora are marked uncertain. And while our models cannot evaluate the semantic aspects involved in cognate discovery, they *do* define a measure of phonetic similarity. We can modify the codelength-based distance measure NCD to apply on the cognate level, conditioned on the learned models of a larger data. For cognates $\boldsymbol{\sigma} \in \mathbf{S}$

and $\boldsymbol{\tau} \in \mathbf{T}$ we define

$$NCD(\boldsymbol{\sigma}, \boldsymbol{\tau}) = \frac{l((\boldsymbol{\sigma} : \boldsymbol{\tau})|\mathcal{M}(\mathcal{D}_{\mathbf{S}} : \mathcal{D}_{\mathbf{T}})) - \min\{l(\boldsymbol{\sigma}|\mathcal{M}(\mathcal{D}_{\mathbf{S}})), l(\boldsymbol{\tau}|\mathcal{M}(\mathcal{D}_{\mathbf{T}}))\}}{\max\{l(\boldsymbol{\sigma}|\mathcal{M}(\mathcal{D}_{\mathbf{S}})), l(\boldsymbol{\tau}|\mathcal{M}(\mathcal{D}_{\mathbf{T}}))\}}, \quad (5.1)$$

where $\mathcal{M}(\mathcal{D}_{\mathbf{S}})$, $\mathcal{M}(\mathcal{D}_{\mathbf{T}})$ and $\mathcal{M}(\mathcal{D}_{\mathbf{S}} : \mathcal{D}_{\mathbf{T}})$ are the mono- and bilingual models learned from the given corpus. Ranking the cognate pairs according to this score, we may then decide to remove some improbable cases to arrive at a cleaner data set. Furthermore, this measure of degree of phonetic correspondence—as the model sees it—will be of linguistic interest in itself.

Finally, as mentioned earlier, we would ultimately like to be able to reconstruct ancestor languages which have been lost and can no longer be observed. We are able to generate phylogenetic language trees, good guesses of the history of language separation. The leaf nodes of these trees represent the languages of our corpus, while the inner nodes are unobserved. Alternatingly building models of correspondence between neighbouring nodes and re-estimating the word forms of the unobserved languages by means of imputation, we hope to be able to reconstruct the history all the way up to—in our case—'Ur-Uralic'.

References

[Akaike 1974] H. Akaike. *A new look at the statistical model identification.* IEEE Transactions on Automatic Control, vol. 19, no. 6, pages 716–723, 1974.

[Andrews 1976] G.E. Andrews. *The Theory of Partitions.* Addison-Wesley Publishing Company, Reading, MA, 1976.

[Angluin 1983] D. Angluin & C.H. Smith. *Inductive Inference: Theory and Methods.* ACM Computing Surveys, vol. 15, no. 3, pages 237–269, 1983.

[Anttila 1989] R. Anttila. *Historical and Comparative Linguistics.* John Benjamins, New York, 1989.

[Barbançon 2009] F.G. Barbançon, T. Warnow, D. Ringe, S.N. Evans & L. Nakhleh. *An experimental study comparing linguistic phylogenetic reconstruction methods.* In Proceedings of the Conference on Languages and Genes, pages 887–896, UC Santa Barbara, June 2009. Cambridge University Press.

[Bellman 1957] R. Bellman. *Dynamic Programming.* Dover Publications, Incoporated, 1957. republished 2003.

[Berger 1985] J.O. Berger. *Statistical Decision Theory and Bayesian Analysis.* Springer, 2 edition, 1985.

[Bernardo 1994] J.M. Bernardo & A.F.M Smith. *Bayesian theory.* John Wiley, 1994.

[Blei 2003] D.M. Blei, A.Y. Ng & M.I. Jordan. *Latent Dirichlet Allocation.* Journal of Machine Learning Research, vol. 3, pages 993–1022, 2003.

[Bouchard-Côté 2007] A. Bouchard-Côté, P. Liang, T. Griffiths & D. Klein. *A Probabilistic Approach to Diachronic Phonology.* In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 887–896, Prague, June 2007.

[Bouchard-Côté 2009] A. Bouchard-Côté, T. Griffiths & D. Klein. *Improved Reconstruction of Protolanguage Word Forms.* In Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL09), 2009.

[Box 1979] G.E.P. Box. *Robustness in the strategy of scientific model building.* In R.L. Launer & G. Wilkinson, editors, Robustness in Statistics, pages 201–236. Academic Press, 1979.

[Cawley 2007] G.C. Cawley, N.L.C. Talbot & M. Girolami. *Sparse Multinomial Logistic Regression via Bayesian L1 Regularisation.* In B. Schölkopf, J. Platt & T. Hofmann, editors, Advances in Neural Information Processing Systems 19 (NIPS-06), volume 10. The MIT Press, 2007.

[Cilibrasi 2006] R. Cilibrasi & P.M.B. Vitányi. *A New Quartet Tree Heuristic for Hierarchical Clustering.* CoRR, 2006. `http://arxiv.org/abs/cs/0606048`.

[Cilibrasi 2007] R. Cilibrasi & P.M.B. Vitányi. *The Google Similarity Distance.* IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 3, pages 370–383, 2007.

[Cover 1991] T. Cover & J. Thomas. *Elements of Information Theory.* John Wiley & Sons, New York, NY, 1991.

[Creutz 2007] M. Creutz & K. Lagus. *Unsupervised Models for Morpheme Segmentation and Morphology Learning.* ACM Transactions on Speech and Language Processing, vol. 4, no. 1, pages 3:1–3:34, 2007.

[Dawid 1984] A.P. Dawid. *Statistical Theory: The Prequential Approach.* Journal of the Royal Statistical Society A, vol. 147, pages 278–292, 1984.

[Djurić 1998] P.M. Djurić. *Asymptotic MAP criteria for model selection.* IEEE Transactions on Signal Processing, vol. 46, no. 10, pages 2726–2735, 1998.

[Dyer 1997] M.E. Dyer, R. Kannan & J. Mount. *Sampling contingency tables.* Random Structures and Algorithms, vol. 10, no. 4, pages 487–506, 1997.

[Flajolet 2009] P. Flajolet & R. Sedgewick. *Analytic Combinatorics.* Cambridge University Press, 2009.

[Friedman 1997] N. Friedman, D. Geiger & M. Goldszmidt. *Bayesian Network Classifiers.* Machine Learning, vol. 29, pages 131–163, 1997.

[Gao 2000] Q. Gao, M. Li & P.M.B. Vitanyi. *Applying MDL to learn best model granularity.* Artificial Intelligence, vol. 121, no. 1–2, pages 1–29, 2000.

[Gelman 1995] A. Gelman, J. Carlin, H. Stern & D. Rubin. *Bayesian Data Analysis.* Chapman & Hall, 1995.

[Gillispie 2001] S.B. Gillispie. *Enumerating Markov equivalence classes of acyclic digraph models.* In J. Breese & D. Koller, editors, Proceedings of the 17th International Conference on Uncertainty in Artificial Intelligence (UAI'01), pages 171–177. Morgan Kaufmann Publishers, 2001.

[Greenhill 2009] S.J. Greenhill & R.D. Gray. *Austronesian Language Phylogenies: Myths and Misconceptions about Bayesian Computational Methods.* In A. Adelaar & A. Pawley, editors, Festschrift for Robert Blust, pages 375–397. Pacific Linguistics, Canberra, 2009.

[Greiner 1997] R. Greiner, A. Grove & D. Schuurmans. *Learning Bayesian Nets that Perform Well.* In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97), pages 198–207, Providence, August 1997.

[Greiner 2001] R. Greiner & W. Zhou. *Discriminant Parameter Learning of Belief Net Classifiers*, 2001. from http://www.cs.ualberta.ca/∼greiner/.

[Greiner 2002] R. Greiner & W. Zhou. *Structural Extension to Logistic Regression: Discriminant Parameter Learning of Belief Net Classifiers.* In Proceedings of the Eighteenth Annual National Conference on Artificial Intelligence (AAAI-02), pages 167–173, Edmonton, August 2002.

[Grünwald 2007] P. Grünwald. *The Minimum Description Length Principle.* MIT Press, 2007.

[Grünwald 2009] P. Grünwald. *Ignoring Data in Court: an Idealized Decision-Theoretic Analysis*, 2009. `homepages.cwi.nl/∼pdg/publicationpage.html`.

[Heckerman 1995] D. Heckerman, D. Geiger & D.M. Chickering. *Learning Bayesian Networks: The Combination of Knowledge and Statistical Data.* Machine Learning, vol. 20, no. 3, pages 197–243, September 1995.

[Hill 1997] R.M. Hill. *Applying Bayesian Methodology with a Uniform Prior to the Single Period Inventory Model.* European Journal of Operational Research, vol. 98, no. 3, pages 555–562, 1997.

[Honkola 2013] T. Honkola, O. Vesakoski, K. Korhonen, J. Lehtinen, K. Syrjänen & N. Wahlberg. *Cultural and Climatic Changes Shape the Evolutionary History of the Uralic Languages.* Journal of Evolutionary Biology (submitted), 2013.

[Itkonen 2000] E. Itkonen & U.-M. Kulonen. *Suomen Sanojen Alkuperä (The Origin of Finnish Words).* Suomalaisen Kirjallisuuden Seura, Helsinki, Finland, 2000.

[Jeffreys 1946] H. Jeffreys. *An invariant form for the prior probability in estimation problems.* Proc. Roy. Soc. A, vol. 186, pages 453–461, 1946.

[Kessler 2001] B. Kessler. *The Significance of Word Lists: Statistical Tests for Investigating Historical Connections Between Languages.* The University of Chicago Press, Stanford, CA, 2001.

[Kirkpatrick 1983] S. Kirkpatrick, D. Gelatt & M.P. Vecchi. *Optimization by simulated annealing.* Science, vol. 220, no. 4598, pages 671–680, May 1983.

[Kolmogorov 1965] A.N. Kolmogorov. *Three Approaches to the Quantitative Definition of Information.* Problems of Information Transmission, vol. 1, no. 1, pages 1–7, 1965.

[Kondrak 2002] G. Kondrak. *Determining recurrent sound correspondences by inducing translation models.* In Proceedings of COLING 2002: 19th International Conference on Computational Linguistics, pages 488–494, Taipei, August 2002.

[Kondrak 2003] G. Kondrak. *Identifying Complex Sound Correspondences in Bilingual Wordlists.* In A. Gelbukh, editor, Computational Linguistics and Intelligent Text Processing (CICLing-2003), pages 432–443, Mexico City, February 2003. Springer-Verlag Lecture Notes in Computer Science, No. 2588.

[Kondrak 2004] G. Kondrak. *Combining Evidence in Cognate Identification.* In Proceedings of the Seventeenth Canadian Conference on Artificial Intelligence (Canadian AI 2004), pages 44–59, London, Ontario, May 2004. Lecture Notes in Computer Science 3060, Springer-Verlag.

[Kontkanen 2001] P. Kontkanen, P. Myllymäki & H. Tirri. *Classifier Learning with Supervised Marginal Likelihood.* In J. Breese & D. Koller, editors, Proceedings of the 17th International Conference on Uncertainty in Artificial Intelligence (UAI'01), pages 277–284. Morgan Kaufmann Publishers, 2001.

[Kontkanen 2003] P. Kontkanen, W. Buntine, P. Myllymäki, J. Rissanen & H. Tirri. *Efficient Computation of Stochastic Complexity.* In C. Bishop & B. Frey, editors,

Proceedings of the Ninth International Conference on Artificial Intelligence and Statistics, pages 233–238. Society for Artificial Intelligence and Statistics, 2003.

[Kontkanen 2005] P. Kontkanen & P. Myllymäki. *A Fast Normalized Maximum Likelihood Algorithm for Multinomial Data.* In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05), 2005.

[Kontkanen 2006] P. Kontkanen, P. Myllymäki, W. Buntine, J. Rissanen & H. Tirri. *An MDL Framework for Data Clustering.* In P. Grünwald, I.J. Myung & M. Pitt, editors, Advances in Minimum Description Length: Theory and Applications. The MIT Press, 2006.

[Kontkanen 2007a] P. Kontkanen & P. Myllymäki. *A linear-time algorithm for computing the multinomial stochastic complexity.* Information Processing Letters, vol. 103, no. 6, pages 227–233, 2007.

[Kontkanen 2007b] P. Kontkanen & P. Myllymäki. *MDL Histogram Density Estimation.* In M. Meila & S. Shen, editors, Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, March 2007.

[Korodi 2005] G. Korodi & I. Tabus. *An efficient normalized maximum likelihood algorithm for DNA sequence compression.* ACM Trans. Inf. Syst., vol. 23, no. 1, pages 3–34, 2005.

[Kraft 1949] L.G. Kraft. *A Device for Quantizing, Grouping, and Coding Amplitude-Modulated Pulses.* Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1949.

[Kruschke 2012] J.K. Kruschke. *Bayesian Estimation Supersedes the* t *Test.* Journal of Experimental Psychology: General, 2012.

[Lauritzen 1996] S. Lauritzen. *Graphical Models.* Oxford University Press, 1996.

[Levenshtein 1966] V. Levenshtein. *Binary Codes Capable of Correcting Deletions, Insertions and Reversals.* Soviet Physics Doklady, vol. 10, no. 8, pages 707–710, 1966.

[Li 1997] M. Li & P.M.B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer Verlag, 1997.

[Lytkin 1973] V.I. Lytkin. *Voprosy Finno-Ugorskogo Jazykoznanija (Issues in Finno-Ugric Linguistics)*, volume 1–3. Nauka, Moscow, 1973.

[MacKay 2002] D.J.C. MacKay. *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, New York, 2002.

[McLachlan 1992] G.J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition.* John Wiley & Sons, New York, 1992.

[McMillan 1956] B. McMillan. *Two inequalities implied by unique decipherability.* IRE Transactions on Information Theory, vol. 2, no. 4, pages 115–116, 1956.

[Minka 2001] T. Minka. *Algorithms for maximum-likelihood logistic regression.* Technical report 758, Carnegie Mellon University, Department of Statistics, 2001. Revised Sep. 2003.

[Mononen 2007] T. Mononen & P. Myllymäki. *Fast NML Computation for Naive Bayes Models.* In V. Corruble, M. Takeda & E. Suzuki, editors, Proceedings of the 10th International Conference on Discovery Science, October 2007.

[Mononen 2008] T. Mononen & P. Myllymäki. *Computing the NML for Bayesian Forests via Matrices and Generating Polynomials.* In Proceedings of the IEEE Information Theory Workshop, Porto, Portugal, May 2008.

[Multitree 2009] Multitree. *A digital library of language relationships*, 2009. Ypsilanti, MI: Institute for Language Information and Technology (LINGUIST List), Eastern Michigan University. `http://multitree.org`.

[Murtagh 1984] F. Murtagh. *Complexities of Hierarchic Clustering Algorithms: the state of the art.* Computational Statistics Quarterly, vol. 1, no. 1, pages 101–113, 1984.

[Myllymäki 2008] P. Myllymäki, T. Roos, T. Silander, P. Kontkanen & H. Tirri. *Factorized NML Models.* In P. Grünwald, P. Myllymäki, I. Tabus, M. Weinberger & B. Yu, editors, Festschrift in Hounour of Jorma Rissanen, chapter 11. 2008.

[Nakhleh 2005] L. Nakhleh, D. Ringe & T. Warnow. *Perfect Phylogenetic Networks: A New Methodology for Reconstructing the Evolutionary History of Natural Languages.* Language (Journal of the Linguistic Society of America), vol. 81, no. 2, pages 382–420, 2005.

[Ng 2001] A.Y. Ng & M.I. Jordan. *On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes.* Advances in Neural Information Processing Systems, vol. 14, pages 605–610, 2001.

[Och 2003] F.-J. Och & H. Ney. *A Systematic Comparison of Various Statistical Alignment Methods.* Computational Linguistics, vol. 29, no. 1, pages 19–51, 2003.

[Pearl 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[Rédei 1991] K. Rédei. *Uralisches etymologisches WÃ¶rterbuch.* Harrassowitz, Wiesbaden, 1988–1991.

[Ringe 2002] D. Ringe, T. Warnow & A. Taylor. *Indo-European and computational cladistics.* Transactions of the Philological Society, vol. 100, no. 1, pages 59–129, 2002.

[Rish 2001] I. Rish. *An Empirical Study of the Naive Bayes Classifier.* In IJCAI'01 Workshop on Empirical Methods in Artificial Intelligence, 2001.

[Rissanen 1978] J. Rissanen. *Modeling by shortest data description.* Automatica, vol. 14, pages 445–471, 1978.

[Rissanen 1987] J. Rissanen. *Stochastic Complexity.* Journal of the Royal Statistical Society, vol. 49, no. 3, pages 223–239 and 252–265, 1987.

[Rissanen 1996] J. Rissanen. *Fisher Information and Stochastic Complexity.* IEEE Transactions on Information Theory, vol. 42, no. 1, pages 40–47, January 1996.

[Rissanen 2000] J. Rissanen. *MDL Denoising.* IEEE Transactions on Information Theory, vol. 46, no. 7, pages 2537–2543, 2000.

[Rissanen 2007] J. Rissanen. *Information and Complexity in Statistical Modeling.* Springer, 2007.

[Rissanen 2010] J. Rissanen, T. Roos & P. Myllymäki. *Model Selection by Sequentially Normalized Least Squares.* Journal of Multivariate Analysis, vol. 101, no. 4, pages 839–849, April 2010.

[Robinson 1977] R. Robinson. *Counting unlabeled acyclic Digraphs.* In C. Little, editor, Combinatorial Mathematics, numéro 622 in Lecture Notes in Mathematics. Springer-Verlag, 1977.

[Roos 2005a] T. Roos, P. Myllymäki & H. Tirri. *On the Behavior of MDL Denoising.* In Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS), pages 309–316, 2005.

[Roos 2005b] T. Roos, H. Wettig, P. Grünwald, P. Myllymäki & H. Tirri. *On Discriminative Bayesian Network Classifiers and Logistic Regression.* Machine Learning, vol. 59, no. 3, pages 267–296, 2005.

[Roos 2009] T. Roos, P. Myllymäki & J. Rissanen. *MDL Denoising Revisited.* IEEE Transactions on Signal Processing, vol. 57, no. 9, pages 3347–3360, September 2009.

[Saitou 1987] N. Saitou & M. Nei. *The neighbor-joining method: a new method for reconstructing phylogenetic trees.* Molecular biology and evolution, vol. 4, no. 4, pages 406–425, 1987.

[Schwarz 1978] G. Schwarz. *Estimating the dimension of a model.* Annals of Statistics, vol. 6, pages 461–464, 1978.

[Shannon 1948] C.E. Shannon. *A Mathematical Theory of Communication.* The Bell System Technical Journal, vol. 27, no. 4, pages 623–656, October 1948.

[Shtarkov 1987] Yu.M. Shtarkov. *Universal sequential coding of single messages.* Problems of Information Transmission, vol. 23, pages 3–17, 1987.

[Silander 2007] T. Silander, P. Kontkanen & P. Myllymäki. *On Sensitivity of the MAP Bayesian Network Structure to the Equivalent Sample Size Parameter.* In R. Parr & L. van der Gaag, editors, Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence, pages 360–367. AUAI Press, 2007.

[Silander 2009] T. Silander, T. Roos & P. Myllymäki. *Locally Minimax Optimal Predictive Modelling with Bayesian Networks.* Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS '09), pages 504–511, 2009.

[Sinor 1997] D. Sinor, editor. *The Uralic Languages: Description, History and Foreign Influences (Handbook of Uralic Studies).* Brill Academic Publishers, Wiesbaden, August 1997.

[Starostin 2005] S.A. Starostin. *Tower of Babel: Etymological Databases*, 2005. `http://newstar.rinet.ru/`.

[Stolcke 1994] A. Stolcke. *Bayesian Learning of Probabilistic Language Models.* Technical report, 1994.

[Szpankowski 2001] W. Szpankowski. *Average case analysis of algorithms on sequences.* John Wiley & Sons, 2001.

[Vapnik 1998] V. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, New York, 1998.

[Wettig 2002a] H. Wettig, P. Grünwald, T. Roos, P. Myllymäki & H. Tirri. *On Supervised Learning of Bayesian Network Parameters*. Technical report 2002–1, Helsinki Institute for Information Technology (HIIT), 2002. http://cosco.hiit.fi/Articles/hiit2002-1.ps.

[Wettig 2002b] H. Wettig, P. Grünwald, T. Roos, P. Myllymäki & H. Tirri. *Supervised Naive Bayes Parameters*. In P. Ala-Siuru & S. Kaski, editors, Intelligence, The Art of Natural and Artificial: Proceedings of the 10th Finnish Artificial Intelligence Conference, pages 72–83. Finnish Artificial Intelligence Society, 2002.

Original Publications

Summary and Contributions

This doctoral dissertation is based on the following six publications, referred to as Publications I-VI. These are reprinted in the following. In all six papers, I am the first author, which reflects my significant contribution to each of them. It follows a listing with brief summary and contribution details.

Publication I

H.Wettig, P. Grünwald, T.Roos, P. Myllymäki, H.Tirri:
When Discriminative Learning of Bayesian Network Parameters Is Easy
Pp. 491-496 in Proceedings of the 18th International Joint Conference on Artificial Intelligence, edited by G.Gottlob and T.Walsh. Morgan Kaufmann, 2003.

We investigate the relation between Bayesian network classifiers and logistic regression. We find that, in many cases which we explicitly identify, the two are equivalent in the sense that they encode the exact same set of conditional distributions. We further argue that the latter is to be preferred as it instantiates its model parameters in a discriminative fashion. Empirical results backing up our claim did not fit into this (six page) publication, but are included in Chapter 2 of this dissertation.

Idea and implementation are of my signature. Theorems 3 and 4 are due to Teemu Roos and Peter Grünwald. Theorem 4 appearing in Section 2.6, a stronger result than that of Theorem 4 of the paper, is my work.

Publication II

H. Wettig, P. Kontkanen and P. Myllymäki:
Calculating the Normalized Maximum Likelihood Distribution for Bayesian Forests
Pp. 1–12 in IADIS International Journal on Computer Science and Information Systems Vol. 2 (2007), No. 2 (October).
Also published in:
Proceedings of the IADIS International Conference on Intelligent Systems and Agents 2007. Lisbon, Portugal, 2007 (Outstanding paper award).

This publication presents an algorithm to compute the normalized maximum likelihood (NML) distribution for tree-structured Bayesian Networks in polynomial time. The degree of this polynomial depends on the cardinality of the multinomials involved, which is unfortunate. There is, however, strong evidence for the inevitability of this, both in this publication and in [Mononen 2008], which—a year later—took an entirely different approach to the same problem.

The problem was suggested to me by Petri Myllymäki and Petri Kontkanen. Realization and implementation are of my doing.

Publication III-VI

H. Wettig and R. Yangarber:
Probabilistic Models for Aligning Etymological Data
Proceedings of the 18th Nordic Conference of Computational Linguistics NODALIDA 2011. Editors: Bolette Sandford Pedersen, Gunta Nešpore and Inguna Skadiņa. NEALT Proceedings Series, Vol. 11 (2011), viii-ix.

H. Wettig, S. Hiltunen and R. Yangarber:
MDL-based Models for Aligning Etymological Data
RANLP-2011: Conference on Recent Advances in Natural Language Processing (2011), Hissar, Bulgaria.

H. Wettig, K. Reshetnikov and R. Yangarber:
Using context and phonetic features in models of etymological sound change
EACL Joint Workshop of LINGVIS & UNCLH 2012, Avignon, France.

Publication VI

H. Wettig, J. Nouri, K. Reshetnikov and R. Yangarber
Information-Theoretic Methods for Analysis and Inference in Etymology
Pp. 52–55 in Proceedings of the Fifth Workshop on Information Theoretic Methods in Science and Engineering (WITMSE 2012)
Edited by Steven de Rooij, Wojciech Kotłowski, Jorma Rissanen, Petri Myllymäki, Teemu Roos and Kenji Yamanishi.

These four publications report a continuance of ongoing research. Naturally, there is considerable overlap to their content, as each subsequent publication builds up on the preceding ones. We introduce a series of models for the alignment of kindred words within a family of natural languages. We optimze MDL-based codelength criteria, which do not utilize any language-specific prior knowledge, resulting in utterly generic methods. The corresponding encoding schemes range from context-unaware modeling on the symbol level to feature-wise modeling using context trees. We present our results and find that it is difficult to objectively evaluate them, as there is no golden standard alignment available. In fact, any attempt to define such standard would inevitably result in fierce debate among linguists. Therefore, we take on a number of approaches to verify the relevance of our findings. Of course—from the MDL perspective—short data encoding is a quality measure in a right of its own. We show that, indeed, code-length correlates well with other, linguistically intuitive scores.

The alignment problem was put before me by Roman Yangarber, who also provided the linguistic expertise. All encoding schemes and algorithms are of my design. The research software has been implemented by Suvi Hiltunen and Javad Nouri, and some of the data pre-processing has been taken care of by Kirill Reshetnikov.

Publication I

Hannes Wettig, Peter Grünwald, Teemu Roos, Petri Myllymäki and Henry Tirri

When Discriminative Learning of Bayesian Network Parameters Is Easy

Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI '03)
Pp. 491-496
edited by G.Gottlob and T.Walsh
Morgan Kaufmann, 2003.

When Discriminative Learning of Bayesian Network Parameters Is Easy

Hannes Wettig⋆, Peter Grünwald◦, Teemu Roos⋆, Petri Myllymäki⋆, and Henry Tirri⋆

⋆ **Complex Systems Computation Group (CoSCo)**
Helsinki Institute for Information Technology (HIIT)
University of Helsinki & Helsinki University of Technology
P.O. Box 9800, FIN-02015 HUT, Finland
{*Firstname*}.{*Lastname*}*@hiit.fi*

◦ **Centrum voor Wiskunde en Informatica (CWI)**
P.O. Box 94079
NL-1090 GB Amsterdam, The Netherlands.
Peter.Grunwald@cwi.nl

Abstract

Bayesian network models are widely used for discriminative prediction tasks such as classification. Usually their parameters are determined using 'unsupervised' methods such as maximization of the joint likelihood. The reason is often that it is unclear how to find the parameters maximizing the conditional (supervised) likelihood. We show how the discriminative learning problem can be solved efficiently for a large class of Bayesian network models, including the Naive Bayes (NB) and tree-augmented Naive Bayes (TAN) models. We do this by showing that under a certain general condition on the network structure, the discriminative learning problem is exactly equivalent to logistic regression with unconstrained convex parameter spaces. Hitherto this was known only for Naive Bayes models. Since logistic regression models have a concave log-likelihood surface, the global maximum can be easily found by local optimization methods.

1 Introduction

In recent years it has been recognized that for discriminative prediction tasks such as classification, we should use a 'supervised' learning algorithm such as conditional likelihood maximization [Friedman *et al.*, 1997; Ng and Jordan, 2001; Kontkanen *et al.*, 2001; Greiner and Zhou, 2002]. Nevertheless, for Bayesian network models the parameters are customarily determined using ordinary methods such as maximization of the joint (unsupervised) likelihood. One of the main reasons for this discrepancy is the difficulty in finding the global maximum of the conditional likelihood. In this paper, we show that this problem can be solved, as long as the underlying Bayesian network meets a particular additional condition, which is satisfied for many existing Bayesian-network based models including Naive Bayes (NB), TAN (tree-augmented NB) and 'diagnostic' models [Kontkanen *et al.*, 2001].

We consider domains of discrete-valued random variables. We find the maximum conditional likelihood parameters by logarithmic reparametrization. In this way, each conditional Bayesian network model is mapped to a logistic regression model, for which the likelihood surface is known to be concave. However, in some cases the parameters of this logistic regression model are not allowed to vary freely. In other words, the Bayesian network model corresponds to a subset of a logistic regression model rather than the full model.

Our main result (Thm. 3 below) provides a general condition on the network structure under which, as we prove, the Bayesian network model is mapped to a full logistic regression model with freely varying parameters. Therefore, in the new parametrization the conditional log-likelihood becomes a concave function of the parameters that under our condition are allowed to vary freely over the convex set $\mathbb{R}^k$. Now we can find the global maximum in the conditional likelihood surface by simple local optimization techniques such as hill climbing.

The result still leaves open the possibility that there are *no* network structures for which the conditional likelihood surface has local, non-global maxima. This would make our condition superfluous. Our second result (Thm. 4 below) shows that this is not the case: there are very simple network structures that do not satisfy our condition, and for which the conditional likelihood can exhibit local, non-global maxima.

Viewing Bayesian network (BN) models as subsets of logistic regression models is not new; it was done earlier in papers such as [Heckerman and Meek, 1997a; Ng and Jordan, 2001; Greiner and Zhou, 2002]. Also, the concavity of the log-likelihood surface for logistic regression is known. Our main contribution is to supply the condition under which Bayesian network models correspond to logistic regression with *completely freely varying parameters*. Only then can we guarantee that there are no local maxima in the likelihood surface. As a direct consequence of our result, we show for the first time that the supervised likelihood of, for instance, the tree-augmented Naive Bayes (TAN) model has no local maxima.

This paper is organized as follows. In Section 2 we introduce Bayesian networks and an alternative, so-called L-parametrization. In Section 3 we show that this allows us to consider Bayesian network models as logistic regression models. Based on earlier results in logistic regression, we conclude that in the L-parametrization the supervised log-likelihood is a concave function. In Section 4 we present our main results, giving conditions under which the two parametrizations correspond to exactly the same conditional

distributions. Conclusions are summarized in Section 5; proofs of the main results are given in Appendix A.

2 Bayesian Networks and the L-model

We assume that the reader is familiar with the basics of the theory of Bayesian networks, see, e.g., [Pearl, 1988].

Consider a random vector $X = (X_0, X_1, \ldots, X_{M'})$, where each variable X_i takes values in $\{1, \ldots, n_i\}$. Let $\mathcal{B}$ be a Bayesian network structure over X, that factorizes $P(X)$ into

$$P(X) = \prod_{i=0}^{M'} P(X_i \mid Pa_i), \tag{1}$$

where $Pa_i \subseteq \{X_0, \ldots, X_{M'}\}$ is the parent set of variable X_i in $\mathcal{B}$.

We are interested in predicting some *class variable* X_m for some $m \in \{0, \ldots, M'\}$ conditioned on all X_i, $i \neq m$. Without loss of generality we may assume that $m = 0$ (i.e., X_0 is the class variable) and that the children of X_0 in $\mathcal{B}$ are $\{X_1, \ldots, X_M\}$ for some $M \leq M'$. For instance, in the so-called Naive Bayes model we have $M = M'$ and the children of the class variable X_0 are independent given the value of X_0. The Bayesian network model corresponding to $\mathcal{B}$ is the set of all distributions satisfying the conditional independencies encoded in $\mathcal{B}$. It is usually parametrized by vectors $\Theta^{\mathcal{B}}$ with components of the form $\theta^{\mathcal{B}}_{x_i|pa_i}$ defined by

$$\theta^{\mathcal{B}}_{x_i|pa_i} := P(X_i = x_i \mid Pa_i = pa_i), \tag{2}$$

where pa_i is any configuration (set of values) for the parents Pa_i of X_i. Whenever we want to emphasize that each pa_i is determined by the complete data vector $x = (x_0, \ldots, x_{M'})$, we write $pa_i(x)$ to denote the configuration of Pa_i in $\mathcal{B}$ given by the vector x. For a given data vector $x = (x_0, x_1, \ldots, x_{M'})$, we sometimes need to consider a modified vector where x_0 is replaced by x'_0 and the other entries remain the same. We then write $pa_i(x'_0, x)$ for the configuration of Pa_i given by $(x'_0, x_1, \ldots, x_{M'})$.

We let $\mathcal{M}^{\mathcal{B}}$ be the set of *conditional* distributions $P(X_0 \mid X_1, \ldots, X_{M'}, \Theta^{\mathcal{B}})$ corresponding to distributions $P(X_0, \ldots, X_{M'} \mid \Theta^{\mathcal{B}})$ satisfying the conditional independencies encoded in $\mathcal{B}$. The conditional distributions in $\mathcal{M}^{\mathcal{B}}$ can be written as

$$P(x_0 \mid x_1, \ldots, x_{M'}, \Theta^{\mathcal{B}}) = \frac{\theta^{\mathcal{B}}_{x_0|pa_0(x)} \prod_{i=1}^{M'} \theta^{\mathcal{B}}_{x_i|pa_i(x)}}{\sum_{x'_0=1}^{n_0} \theta^{\mathcal{B}}_{x'_0|pa_0(x)} \prod_{i=1}^{M'} \theta^{\mathcal{B}}_{x_i|pa_i(x'_0,x)}}, \tag{3}$$

extended to N outcomes by independence.

Given a complete data-matrix $D = (x^1, \ldots, x^N)$, the *conditional log-likelihood*, $S^{\mathcal{B}}(D; \Theta^{\mathcal{B}})$, with parameters $\Theta^{\mathcal{B}}$ is given by

$$S^{\mathcal{B}}(D; \Theta^{\mathcal{B}}) := \sum_{j=1}^{N} S^{\mathcal{B}}(x^j; \Theta^{\mathcal{B}}), \tag{4}$$

where

$$S^{\mathcal{B}}(x; \Theta^{\mathcal{B}}) := \ln P(x_0 \mid x_1, \ldots, x_{M'}, \Theta^{\mathcal{B}}). \tag{5}$$

Note that in (3), and hence also in (4), all $\theta^{\mathcal{B}}_{x_i|pa_i}$ with $i > M$ (standing for nodes that are neither the class variable nor any of its children) cancel out, since for these terms we have $pa_i(x) \equiv pa_i(x'_0, x)$ for all x'_0. Thus the only relevant parameters for determining the conditional likelihood are of the form $\theta^{\mathcal{B}}_{x_i|pa_i}$ with $i \in \{0, \ldots, M\}$, $x_i \in \{1, \ldots, n_i\}$ and pa_i any configuration of parents Pa_i. We order these parameters lexicographically and define $\mathbf{\Theta}^{\mathcal{B}}$ to be the set of vectors constructed this way, with $\theta^{\mathcal{B}}_{x_i|pa_i} > 0$ and $\sum_{x_i=1}^{n_i} \theta^{\mathcal{B}}_{x_i|pa_i} = 1$ for all $i \in \{0, \ldots, M\}$, x_i and all values (configurations) of pa_i. Note that we require all parameters to be *strictly* positive.

The model $\mathcal{M}^{\mathcal{B}}$ does not contain any notion of the joint distribution: Terms such as $P(X_i \mid Pa_i)$, where $0 < i \leq M'$, are undefined and neither are we interested in them. Our task is prediction of X_0 *given* $X_1, \ldots, X_{M'}$. Heckerman and Meek call such models *Bayesian regression/classification* (BRC) models [Heckerman and Meek, 1997a; 1997b].

For an arbitrary conditional Bayesian network model $\mathcal{M}^{\mathcal{B}}$, we now define the so-called L-model, another set of conditional distributions $P(X_0 \mid X_1, \ldots, X_{M'})$. This model, which we denote by $\mathcal{M}^L$, is parametrized by vectors Θ^L in some set $\mathbf{\Theta}^{\mathbf{L}}$ that closely resembles $\mathbf{\Theta}^{\mathcal{B}}$. Each different $\mathcal{M}^{\mathcal{B}}$ gives rise to a corresponding $\mathcal{M}^L$, although we do not necessarily have $\mathcal{M}^{\mathcal{B}} = \mathcal{M}^L$. For each component $\theta^{\mathcal{B}}_{x_i|pa_i}$ of each vector $\Theta^{\mathcal{B}} \in \mathbf{\Theta}^{\mathcal{B}}$, there is a corresponding component $\theta^L_{x_i|pa_i}$ of the vectors $\Theta^L \in \mathbf{\Theta}^{\mathbf{L}}$. The components $\theta^L_{x_i|pa_i}$ take values in the range $(-\infty, \infty)$ rather than $(0, 1)$. Each vector $\Theta^L \in \mathbf{\Theta}^{\mathbf{L}}$ defines the following conditional distribution:

$$P(x_0 \mid x_1, \ldots, x_{M'}, \Theta^L) := \frac{(\exp \theta^L_{x_0|pa_0(x)}) \prod_{i=1}^{M} \exp \theta^L_{x_i|pa_i(x)}}{\sum_{x'_0=1}^{n_0} (\exp \theta^L_{x'_0|pa_0(x)}) \prod_{i=1}^{M} \exp \theta^L_{x_i|pa_i(x'_0,x)}}. \tag{6}$$

The model $\mathcal{M}^L$ is the set of conditional distributions $P(X_0 \mid X_1, \ldots, X_{M'}, \Theta^L)$ indexed by $\Theta^L \in \mathbf{\Theta}^{\mathbf{L}}$, extended to N outcomes by independence. Given a data-matrix D, let $S^L(D; \Theta^L)$ be the conditional log-likelihood with parameters Θ^L, defined analogously to (4) with (6) in place of (3).

Theorem 1. $\mathcal{M}^{\mathcal{B}} \subseteq \mathcal{M}^L$.

Proof. From (3) and (6) we get that Θ^L defined by setting $\theta^L_{x_i|pa_i} = \ln \theta^{\mathcal{B}}_{x_i|pa_i}$ for all i, x_i and pa_i, indexes the same conditional distribution as $\Theta^{\mathcal{B}}$. ∎

In words, all the conditional distributions that can be represented by parameters $\Theta^{\mathcal{B}} \in \mathbf{\Theta}^{\mathcal{B}}$ can also be represented by parameters $\Theta^L \in \mathbf{\Theta}^{\mathbf{L}}$. The converse of Theorem 1, i.e., $\mathcal{M}^L \subseteq \mathcal{M}^{\mathcal{B}}$, is true only under some additional conditions on the network structure, as we explain in Section 4. First we take a closer look at the L-model.

3 The L-model Viewed as Logistic Regression

Although L-models are closely related to and in some cases formally identical to Bayesian network models, we can also think of them as predictors that combine the information of the attributes using the so-called *softmax* rule [Heckerman and Meek, 1997b; Ng and Jordan, 2001]. In statistics, such models have been extensively studied under the name of *logistic regression* models, see, e.g. [McLachlan, 1992, p.255].

More precisely, let $X_0 = \{1, \ldots, n_0\}$ and let $Y_1, \ldots, Y_k$ be real-valued random variables. The *multiple logistic regression model with dependent variable X_0 and covariates $Y_1, \ldots, Y_k$* is defined as the set of conditional distributions

$$P(x_0 \mid y_1, \ldots, y_k) := \frac{\exp \sum_{i=1}^{k} \beta_{x_0|i}\, y_i}{\sum_{x_0'=1}^{n_0} \exp \sum_{i=1}^{k} \beta_{x_0'|i}\, y_i} \qquad (7)$$

where the $\beta_{x_0|_i}$ are allowed to take on all values in $\mathbb{R}$. This defines a conditional model parameterized in $\mathbb{R}^{n_0 \cdot k}$. Now, for $i \in \{0, \ldots, M\}$, $x_i \in \{1, \ldots, n_i\}$ and pa_i in the set of parent configurations of X_i, let

$$Y_{(x_i, pa_i)} := \begin{cases} 1 & \text{if } X_i = x_i \text{ and } Pa_i = pa_i \\ 0 & \text{otherwise.} \end{cases} \qquad (8)$$

The indicator random variables $Y_{(x_i,pa_i)}$ thus obtained can be lexicographically ordered and renamed $1, \ldots, k$, which shows that each L-model corresponding to a Bayesian network structure $\mathcal{B}$ as in (6) is formally identical to the logistic model (7) with dependent variable X_0 and covariates given by (8). So, for all network structures $\mathcal{B}$, the corresponding L-model $\mathcal{M}^L$ is the standard multiple logistic model, where the input variables for the logistic model are transformations of the input variables to the L-model, the transformation being determined by the network structure $\mathcal{B}$.

It turns out that the conditional log-likelihood in the L-parametrization is a concave function of the parameters:

Theorem 2. *The parameter set $\mathbf{\Theta^L}$ is convex, and the conditional log-likelihood $S^L(D; \Theta^L)$ is concave, though not strictly concave.*

Proof. The first part is obvious since each parameter can take values in $(-\infty, \infty)$. Concavity of $S^L(D; \Theta^L)$ is a direct consequence of the fact that multiple logistic regression models are exponential families; see, e.g., [McLachlan, 1992, p.260]. For an example showing that the conditional log-likelihood is not strictly concave, see [Wettig *et al.*, 2002]. ∎

Remark. Non-strictness of the proven concavity may pose a technical problem in optimization. This can be avoided by assigning a strictly concave prior on the model parameters and then maximizing the 'conditional posterior' [Grünwald *et al.*, 2002; Wettig *et al.*, 2002] instead of the likelihood. One may also prune the model of weakly supported parameters and/or add constraints to arrive at a strictly concave conditional likelihood surface. Our experiments [Wettig *et al.*, 2002] suggest that for small data samples this should be done in any case, in order to avoid over-fitting; see also Section 5. Any constraint added should of course leave the parameter space a convex set, e.g. a subspace of the full $\mathbf{\Theta^L}$.

Corollary 1. *There are no local, non-global, maxima in the likelihood surface of an L-model.*

The conditions under which a global maximum exists are discussed in, e.g., [McLachlan, 1992] and references therein. A possible solution in cases where no maximum exists is to introduce a strictly concave prior as discusssed above.

The global conditional maximum likelihood parameters obtained from training data can be used for prediction of future data. In addition, as discussed in [Heckerman and Meek, 1997a], they can be used to perform model selection among several competing model structures using, e.g., the BIC or (approximate) MDL criteria. In [Heckerman and Meek, 1997a] it is stated that for general conditional Bayesian network models $\mathcal{M}^{\mathcal{B}}$, "although it may be difficult to determine a global maximum, gradient-based methods [...] can be used to locate local maxima". Theorem 2 shows that if the network structure $\mathcal{B}$ is such that the two models are equivalent, $\mathcal{M}^{\mathcal{B}} = \mathcal{M}^L$, we can find even the *global* maximum of the conditional likelihood by reparametrizing $\mathcal{M}^{\mathcal{B}}$ to the L-model, and using some local optimization method. Thus, the question under which condition $\mathcal{M}^{\mathcal{B}} = \mathcal{M}^L$ becomes crucial. It is this question we address in the next section.

Remark. Because the log-transformation is continuous, it follows (with some calculus) that, if $\mathcal{M}^{\mathcal{B}} = \mathcal{M}^L$, then all maxima of the (concave) conditional likelihood in the L-parameterization are global (and connected) maxima also in the original parametrization. Nevertheless, the likelihood surface as a function of $\Theta^{\mathcal{B}} \in \mathbf{\Theta}^{\mathcal{B}}$ has some unpleasant properties (see [Wettig *et al.*, 2002]): it is *not* concave in general and, worse, it can have 'wrinkles': by these we mean convex subsets $\mathbf{\Theta}^{\mathcal{B}}_{\mathbf{0}}$ of $\mathbf{\Theta}^{\mathcal{B}}$, such that, under the constraint that $\Theta^{\mathcal{B}} \in \mathbf{\Theta}^{\mathcal{B}}_{\mathbf{0}}$, the likelihood surface does exhibit local, non-global maxima. This suggests that it is computationally preferrable to optimize over $\mathbf{\Theta^L}$ rather than $\mathbf{\Theta}^{\mathcal{B}}$. Empirical evidence for this is reported in [Greiner and Zhou, 2002].

4 Main Result

By setting $\theta^L_{x_i|pa_i} = \ln \theta^{\mathcal{B}}_{x_i|pa_i}$, it follows that each distribution in $\mathcal{M}^{\mathcal{B}}$ is also in $\mathcal{M}^L$ (Thm. 1). This suggests that, by doing the reverse transformation

$$\theta^{\mathcal{B}}_{x_i|pa_i} = \exp \theta^L_{x_i|pa_i}, \qquad (9)$$

we could also show that distributions in $\mathcal{M}^L$ are also in $\mathcal{M}^{\mathcal{B}}$. However, $\mathbf{\Theta^L}$ contains distributions that violate the 'sum-up-to-one constraint', i.e., for some $\Theta^L \in \mathbf{\Theta^L}$ we have $\sum_{x_i=1}^{n_i} \exp \theta^L_{x_i|pa_i} \neq 1$ for some $i \in \{0, \ldots, M'\}$ and pa_i. Then the corresponding $\Theta^{\mathcal{B}}$ is *not* in $\mathbf{\Theta}^{\mathcal{B}}$. But, since the L-parameterization is redundant (many different Θ^L index the same conditional distribution $P(\cdot \mid \cdot) \in \mathcal{M}^L$), it may still be the case that the *distribution* $P(\cdot \mid \cdot, \Theta^L)$ *indexed by* Θ^L is in $\mathcal{M}^{\mathcal{B}}$. Indeed, it turns out that for *some* network structures $\mathcal{B}$, the corresponding $\mathcal{M}^L$ is such that *each* distribution in $\mathcal{M}^L$ can be expressed by a parameter vector Θ^L such that $\sum_{x_i=1}^{n_i} \exp \theta^L_{x_i|pa_i} = 1$ for all $i \in \{0, \ldots, M'\}$ and pa_i. In that case, by (9), we *do* have $\mathcal{M}^{\mathcal{B}} = \mathcal{M}^L$. Our main result is that this is the case if $\mathcal{B}$ satisfies the following condition:

Condition 1. For all $j \in \{1, \ldots, M\}$, there exists $X_i \in Pa_j$ such that $Pa_j \subseteq Pa_i \cup \{X_i\}$.

Remark. Condition 1 implies that the class X_0 must be a 'moral node', i.e., it cannot have a common child with a node it is not directly connected with. But Condition 1 demands more than that; see Figures 1 and 2.

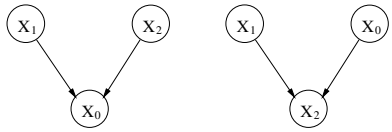

Figure 1: A simple Bayesian network (the class variable is denoted by X_0) satisfying Condition 1 (left); and a network that does not satisfy the condition (right).

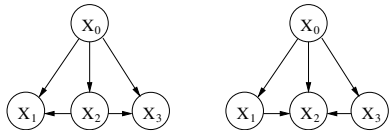

Figure 2: A tree-augmented Naive Bayes (TAN) model satisfying Condition 1 (left); and a network that is not TAN (right). Even though in both cases the class variable X_0 is a moral node, the network on the right does not satisfy Condition 1.

Example 1. Consider the Bayesian networks depicted in Figure 1. The leftmost network, $\mathcal{B}_1$, satisfies Condition 1, the rightmost network, $\mathcal{B}_2$, does not. Theorem 4 shows that the conditional likelihood surface of $\mathcal{M}^{\mathcal{B}_2}$ can have local maxima, implying that in this case $\mathcal{M}^{\mathcal{B}} \neq \mathcal{M}^{L}$. $\Diamond$

Examples of network structures that satisfy Condition 1 are the Naive Bayes (NB) and the tree-augmented Naive Bayes (TAN) models [Friedman *et al.*, 1997]. The latter is a generalization of the former in which the children of the class variable are allowed to form tree-structures; see Figure 2.

Proposition 1. *Condition 1 is satisfied by the Naive Bayes and the tree-augmented Naive Bayes structures.*

Proof. For Naive Bayes, we have $Pa_j \subseteq \{X_0\}$ for all $j \in \{1, \ldots, M\}$. For TAN models, all children of the class variable have either one or two parents. For children with only one parent (the class variable) we can use the same argument as in the NB case. For any child X_j with two parents, let X_i be the parent that is not the class variable. Because X_i is also a child of the class variable, we have $Pa_j \subseteq Pa_i \cup \{X_i\}$. ∎

Condition 1 is also automatically satisfied if X_0 only has incoming arcs[1] ('diagnostic' models, see [Kontkanen *et al.*, 2001]). For Bayesian network structures $\mathcal{B}$ for which the condition does not hold, we can always add some arcs to arrive at a structure $\mathcal{B}'$ for which the condition does hold (for instance, add an arc from X_1 to X_3 in the rightmost network in Figure 2). Therefore, $\mathcal{M}^{\mathcal{B}}$ is always a subset of a larger model $\mathcal{M}^{\mathcal{B}'}$ for which the condition holds. We are now ready to present our main result (for proof see Appendix A):

Theorem 3. *If $\mathcal{B}$ satisfies Condition 1, then $\mathcal{M}^{\mathcal{B}} = \mathcal{M}^{L}$.*

Together with Corollary 1, Theorem 3 shows that Condition 1 suffices to ensure that the conditional likelihood surface of $\mathcal{M}^{\mathcal{B}}$ has no local (non-global) maxima. Proposition 1 now implies that, for example, the conditional likelihood surface of TAN models has no local maxima. Therefore, a global maximum can be found by local optimization techniques.

But what about the case in which Condition 1 does not hold? Our second result, Theorem 4 (proven in Appendix A) says that in this case, there can be local maxima:

Theorem 4. *Let $\mathcal{B}_2 = X_1 \rightarrow X_2 \leftarrow X_0$ be the network structure depicted in Figure 1 (right). There exist data samples such that the conditional likelihood has local, non-global maxima over $\mathcal{M}^{\mathcal{B}_2}$.*

The theorem implies that $\mathcal{M}^{L} \neq \mathcal{M}^{\mathcal{B}_2}$. Thus, Condition 1 is not superfluous. We may now ask whether our condition is *necessary* for having $\mathcal{M}^{L} = \mathcal{M}^{\mathcal{B}}$; that is, whether $\mathcal{M}^{L} \neq \mathcal{M}^{\mathcal{B}}$ for *all* network structures that violate the condition. We plan to address this intriguing open question in future work.

5 Concluding Remarks

We showed that one can effectively find the parameters maximizing the conditional (supervised) likelihood of NB, TAN and many other Bayesian network models. We did this by showing that the network structure of these models satisfies our 'Condition 1', which ensures that the conditional distributions corresponding to such models are equivalent to a particular multiple logistic regression model with unconstrained parameters. An arbitrary network structure can always be made to satisfy Condition 1 by adding arcs. Thus, we can embed any Bayesian network model in a larger model (with less independence assumptions) that satisfies Condition 1.

Test runs for the Naive Bayes case in [Wettig *et al.*, 2002] have shown that maximizing the conditional likelihood in contrast to the usual practice of maximizing the joint (unsupervised) likelihood is feasible and yields greatly improved classification. Similar results are reported in [Greiner and Zhou, 2002]. Our conclusions are also supported by theoretical analysis in [Ng and Jordan, 2001]. Only on very small data sets we sometimes see that joint likelihood optimization outperforms conditional likelihood, the reason apparently being that the conditional method is more inclined to overfitting. We conjecture that in such cases, rather than resorting to maximizing the joint instead of the conditional likelihood, it may be preferable to use a simpler model or simplify (i.e. prune or restrict) the model at hand and still choose its parameters in a discriminative fashion. In our setting, this would amount to model selection using the L-parametrization. This is a subject of our future research.

References

[Friedman *et al.*, 1997] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[Greiner and Zhou, 2002] R. Greiner and W. Zhou. Structural extension to logistic regression: Discriminant parameter learning of belief net classifiers. In *Proceedings of the Eighteenth Annual National Conference on Artificial Intelligence (AAAI-02)*, pages 167–173, Edmonton, 2002.

[Grünwald *et al.*, 2002] P. Grünwald, P. Kontkanen, P. Myllymäki, T. Roos, H. Tirri, and H. Wettig. Supervised posterior distri-

[1] It is easy to see that in that case the maximum conditional likelihood parameters may even be determined analytically.

butions, 2002. Presented at the Seventh Valencia International Meeting on Bayesian Statistics, Tenerife, Spain.

[Heckerman and Meek, 1997a] D. Heckerman and C. Meek. Embedded bayesian network classifiers. Technical Report MSR-TR-97-06, Microsoft Research, 1997.

[Heckerman and Meek, 1997b] D. Heckerman and C. Meek. Models and selection criteria for regression and classification. In D. Geiger and P. Shenoy, editors, *Uncertainty in Arificial Intelligence 13*, pages 223–228. Morgan Kaufmann Publishers, San Mateo, CA, 1997.

[Kontkanen *et al.*, 2001] P. Kontkanen, P. Myllymäki, and H. Tirri. Classifier learning with supervised marginal likelihood. In J. Breese and D. Koller, editors, *Proceedings of the 17th International Conference on Uncertainty in Artificial Intelligence (UAI'01)*. Morgan Kaufmann Publishers, 2001.

[McLachlan, 1992] G.J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons, New York, 1992.

[Ng and Jordan, 2001] A.Y. Ng and M.I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in Neural Information Processing Systems*, 14:605–610, 2001.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[Wettig *et al.*, 2002] H. Wettig, P. Grünwald, T. Roos, P. Myllymäki, and H. Tirri. On supervised learning of Bayesian network parameters. Technical Report HIIT-2002–1, Helsinki Institute for Information Technology (HIIT), 2002. Available at http://cosco.hiit.fi/Articles/hiit2002-1.ps.

A Proofs

Proof of Theorem 3. We introduce some more notation. For $j \in \{1,\ldots,M\}$, let m_j be the maximum number in $\{0,\ldots,M\}$ such that $X_{m_j} \in Pa_j$, $Pa_j \subseteq Pa_{m_j} \cup \{X_{m_j}\}$. Such an m_j exists by Condition 1. To see this, note that the $X_i \in Pa_j$ mentioned in Condition 1 must lie in the set $\{X_0, X_1, \ldots, X_m\}$ (otherwise $X_0 \in Pa_j, X_0 \notin Pa_i$, so $Pa_j \not\subseteq Pa_i \cup \{X_i\}$, contradiction).

Condition 1 implies that pa_j is completely determined by the pair (x_{m_j}, pa_{m_j}). We can therefore introduce functions Q_j mapping (x_{m_j}, pa_{m_j}) to the corresponding pa_j. Hence, for all $x = (x_0,\ldots,x_{M'})$ and $j \in \{1,\ldots,M\}$ we have

$$pa_j = Q_j(x_{m_j}, pa_{m_j}). \tag{10}$$

We introduce, for all $i \in \{0,\ldots,M\}$ and for each configuration pa_i of Pa_i, a constant $c_{i|pa_i}$ and define, for any $\Theta^L \in \mathbf{\Theta^L}$,

$$\theta^{(c)}_{x_i|pa_i} := \theta^L_{x_i|pa_i} + c_{i|pa_i} - \sum_{j:m_j=i} c_{j|Q_j(x_i,pa_i)}. \tag{11}$$

The parameters $\theta^{(c)}_{x_i|pa_i}$ constructed this way are combined to a vector $\Theta^{(c)}$ which is clearly a member of $\mathbf{\Theta^L}$.

Having introduced this notation, we now show that no matter how we choose the constants $c_{i|pa_i}$, for all Θ^L and corresponding $\Theta^{(c)}$ we have $S^L(D;\Theta^{(c)}) = S^L(D;\Theta^L)$.

We first show that, for all possible vectors x and the corresponding parent configurations, no matter how the $c_{i|pa_i}$ are chosen, it holds that

$$\sum_{i=0}^{M} \theta^{(c)}_{x_i|pa_i} = \sum_{i=0}^{M} \theta^L_{x_i|pa_i} + c_{0|pa_0}. \tag{12}$$

To derive (12) we substitute all terms of $\sum_{i=0}^{M} \theta^{(c)}_{x_i|pa_i}$ by their definition (11). Clearly, for all $j \in \{1,\ldots,M\}$, there is exactly one term of the form $c_{j|pa_j}$ that appears in the sum with a positive sign. Since for each $j \in \{1,\ldots,M\}$ there exists exactly one $i \in \{0,\ldots,M\}$ with $m_j = i$, it must be the case that for all $j \in \{1,\ldots,M\}$, a term of the form $c_{j|Q_j(x_i,pa_i)}$ appears exactly once in the sum with a negative sign. By (10) we have $c_{j|Q_j(x_i,pa_i)} = c_{j|pa_j}$. Therefore all terms $c_{j|pa_j}$ that appear once with a positive sign also appear once with a negative sign. It follows that, except for $c_{0|pa_0}$, all terms $c_{j|pa_j}$ cancel. This establishes (12). By plugging in (12) into (6), it follows that $S^L(D;\Theta^{(c)}) = S^L(D;\Theta^L)$ for all D.

Now set, for all x_i and pa_i,

$$\theta^{\mathcal{B}}_{x_i|pa_i} = \exp \theta^{(c)}_{x_i|pa_i}. \tag{13}$$

We show that we can determine the constants $c_{i|pa_i}$ such that for all $i \in \{0,\ldots,M\}$ and pa_i, the 'sum up to one' constraint is satisfied, i.e., we have

$$\sum_{x_i=1}^{n_i} \theta^{\mathcal{B}}_{x_i|pa_i} = 1. \tag{14}$$

We achieve this by sequentially determining values for $c_{i|pa_i}$ in a particular order.

We need some additional terminology: we say 'c_i *is determined*' if for all configurations pa_i of Pa_i, we have already determined $c_{i|pa_i}$. We say 'c_i *is undetermined*' if we have determined $c_{i|pa_i}$ for *no* configuration pa_i of Pa_i. We say 'c_i is *ready to be determined*' if c_i is undetermined and at the same time all c_j with $m_j = i$ have been determined.

We note that as long as some c_i with $i \in \{0,\ldots,M\}$ are undetermined, there must exist $c_{i'}$ that are ready to be determined. To see this, first take any $i \in \{0,\ldots,M\}$ with c_i undetermined. Either c_i itself is ready to be determined (in which case we are done), or there exists $j \in \{1,\ldots M\}$ with $m_j = i$ (and hence $X_i \in Pa_j$) such that c_j is undetermined. If c_j is ready to be determined, we are done. Otherwise we repeat the argument, move forward in $\mathcal{B}$ restricted to $\{X_0,\ldots,X_M\}$ and (because $\mathcal{B}$ is acyclic) within M steps surely find a c_l that is ready to be determined.

We now describe an algorithm that sequentially assigns values to $c_{i|pa_i}$ such that (14) is satisfied. We start with all c_i undetermined and repeat the following steps:

WHILE there exists c_i, $i \in \{0,\ldots,M\}$, that is undetermined DO

1. Pick the largest i such that c_i is ready to be determined.
2. Set, for all configurations pa_i of Pa_i, $c_{i|pa_i}$ such that $\sum_{x_i=1}^{n_i} \theta^{\mathcal{B}}_{x_i|pa_i} = 1$ holds.

DONE

The algorithm loops $M+1$ times and then halts. Step 2 does not affect the values of $c_{j|pa_j}$ for any j, pa_j such that $c_{j|pa_j}$ has already been determined. Therefore, after the algorithm halts, (14) holds.

Let $\Theta^L \in \mathbf{\Theta^L}$. Each choice of constants $c_{i|pa_i}$ determines a corresponding vector $\Theta^{(c)}$ with components given by (11). This in turn determines a corresponding vector $\Theta^{\mathcal{B}}$ with components given by (13). In Stage 2 we showed that we can take the $c_{i|pa_i}$ such that (14) holds. This is the choice of $c_{i|pa_i}$ which we adopt. With this particular choice, $\Theta^{\mathcal{B}}$ indexes a distribution in $\mathcal{M}^{\mathcal{B}}$. By applying the log-transformation to the components of $\Theta^{\mathcal{B}}$ we find that for any D of any length, $S^{\mathcal{B}}(D;\Theta^{\mathcal{B}}) = S^L(D;\Theta^{(c)})$, where $S^{\mathcal{B}}(D;\Theta^{\mathcal{B}})$ denotes the conditional log-likelihood of $\Theta^{\mathcal{B}}$ as given by summing the logarithm of (3). The result of Stage 1 now implies that $\Theta^{\mathcal{B}}$ indexes the same conditional distribution as Θ^L. Since $\Theta^L \in \mathbf{\Theta^L}$ was chosen arbitrarily, this shows that $\mathcal{M}^L \subseteq \mathcal{M}^{\mathcal{B}}$. Together with Theorem 1 this concludes the proof. ∎

Proof (sketch) of Theorem 4. Use the rightmost network in Figure 1 with structure $X_0 \rightarrow X_2 \leftarrow X_1$. Let the data be $D = ((1,1,1),(1,1,2),(2,2,1),(2,2,2))$. Note that X_0 and X_1 always have the same value. We first show that with this data, there are four local, non-connected suprema of the conditional likelihood.

We are interested in predicting the value of X_0 given X_1, and X_2. The parameter defining the distribution of X_1 has no effect on conditional predictions and we can ignore it. For the remaining five parameters we use the following notation:

$$\begin{aligned}
\theta_2 &:= P(X_0 = 2),\\
\theta_{2|1,1} &:= P(X_2 = 2 \mid X_0 = 1, X_1 = 1),\\
\theta_{2|1,2} &:= P(X_2 = 2 \mid X_0 = 1, X_1 = 2),\\
\theta_{2|2,1} &:= P(X_2 = 2 \mid X_0 = 2, X_1 = 1),\\
\theta_{2|2,2} &:= P(X_2 = 2 \mid X_0 = 2, X_1 = 2).
\end{aligned} \tag{15}$$

The conditional log-likelihood can be written as

$$S^{\mathcal{B}}(D;\Theta^{\mathcal{B}}) = g(1-\theta_2, \theta_{2|1,1}, \theta_{2|2,1}) + g(\theta_2, \theta_{2|2,2}, \theta_{2|1,2}), \tag{16}$$

where

$$g(x,y,z) := f(x,y,z) + f(x,1-y,1-z), \tag{17}$$

and

$$f(x,y,z) := \ln \frac{xy}{xy + (1-x)z}. \tag{18}$$

Figure 3 illustrates functions $g(x,y,z)$ at $x = 0.5$. In (16) each parameter except θ_2 appears only once. Thus, for a fixed θ_2 we can maximize each term separately. ¿From Lemma 1 below it follows that the supremum of the log-likelihood with θ_2 fixed is $\ln(1-\theta_2) + \ln(\theta_2)$, which achieves its maximum value $-2\ln 2$ at $\theta_2 = 0.5$. Furthermore, the lemma shows that the log-likelihood approaches its supremum when $\theta_{2|2,1} \in \{0,1\}$, $\theta_{2|1,2} \in \{0,1\}$, $\theta_{2|1,1} \rightarrow \theta_{2|2,1}$, and $\theta_{2|2,2} \rightarrow \theta_{2|1,2}$.

Setting $y = 0.5$ results in

$$\sup_{0 \le z \le 1} g(x, 0.5, z) = \ln \frac{x}{2-x} < \ln x. \tag{19}$$

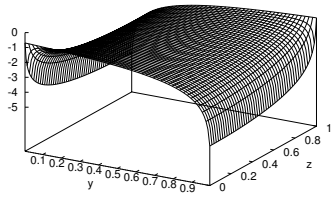

Figure 3: Function g(x,y,z) given by (17) with $x = 0.5$.

Therefore setting either $\theta_{2|1,1}$ or $\theta_{2|2,2}$ to 0.5 results in a smaller supremum of the log-likelihood than the above choices. Consequently, the four suprema are separated by areas where the log-likelihood is smaller, i.e., the suprema are local and not connected.

To conclude the proof we still need to address two issues: (a) the four local suprema give the same conditional log-likelihood $-2\ln 2$, and (b), they are suprema, not maxima (not achieved by any $\Theta^{\mathcal{B}} \in \mathbf{\Theta^{\mathcal{B}}}$). To deal with (a), consider data D' consisting of n_1 repetitions of $(1,1,1)$, n_2 repetitions of $(1,1,2)$, n_3 repetitions of $(2,2,1)$ and n_4 repetitions of $(2,2,2)$. By doing a slightly more involved analysis, one can show that, for some choices of n_1, n_2, n_3, n_4, the supervised log-likelihood still has four suprema, but they have different likelihood values. To deal with (b), let D'' be equal to D' but with four extra data vectors $(1,2,1),(2,1,1),(1,2,2),(2,1,2)$. If n_1, n_2, n_3 and n_4 are chosen large enough, the supervised likelihood for D'' has four maxima (rather than suprema), not all of which achieve the same supervised likelihood. We omit further details. ∎

Lemma 1. *With $0 < x < 1$ fixed and y and z both varying between 0 and 1, the supremum of $g(x,y,z)$ defined by* (17) *is given by*

$$\sup_{0 \le y,z \le 1} g(x,y,z) = \ln(x). \tag{20}$$

The function approaches its supremum when $z \in \{0,1\}$, and $y \rightarrow z$. That is, $\lim_{y \downarrow 0} g(x,y,0) = \lim_{y \uparrow 1} g(x,y,1) = \ln x$.

Proof. Differentiating twice wrt. z gives

$$\begin{aligned}
\frac{\partial^2}{\partial^2 z} g(x,y,z) &= \frac{(1-x)^2}{(xy + (1-x)z)^2}\\
&\quad + \frac{(1-x)^2}{(x(1-y) + (1-x)(1-z))^2},
\end{aligned} \tag{21}$$

which is always positive and the function achieves its maximum values at $z \in \{0,1\}$. At these two points derivating wrt. y yields

$$\begin{aligned}
\frac{\partial}{\partial y} g(x,y,0) &= \frac{x-1}{(1-y)(1-xy)},\\
\frac{\partial}{\partial y} g(x,y,1) &= \frac{1-x}{y(xy+1-x)}.
\end{aligned} \tag{22}$$

Since in the first case the derivative is always negative, and in the second case the derivative is always positive, $g(x,y,0)$ increases monotonically as $y \rightarrow 0$, and $g(x,y,1)$ increases monotonically as $y \rightarrow 1$. In both cases the limiting value is $\ln(x)$. ∎

Publication II

Hannes Wettig, Petri Kontkanen and Petri Myllymäki

Calculating the Normalized Maximum Likelihood Distribution for Bayesian Forests

IADIS International Journal on Computer Science and Information Systems 2 (2007) 2 (October)

In *Proceedings of the IADIS International Conference Intelligent Systems and Agents 2007.* Lisbon, Portugal, 2007.

Calculating the Normalized Maximum Likelihood Distribution for Bayesian Forests

Hannes Wettig Petri Kontkanen
Petri Myllymäki
Complex Systems Computation Group (CoSCo)
Helsinki Institute for Information Technology (HIIT)
University of Helsinki & Helsinki University of Technology
P.O.Box 68 (Department of Computer Science)
FIN-00014 University of Helsinki, Finland
{Firstname}.{Lastname}@hiit.fi

ABSTRACT

When learning Bayesian network structures from sample data, an important issue is how to evaluate the goodness of alternative network structures. Perhaps the most commonly used model (class) selection criterion is the marginal likelihood, which is obtained by integrating over a prior distribution for the model parameters. However, the problem of determining a reasonable prior for the parameters is a highly controversial issue, and no completely satisfying Bayesian solution has yet been presented in the non-informative setting. The normalized maximum likelihood (NML), based on Rissanen's information-theoretic MDL methodology, offers an alternative, theoretically solid criterion that is objective and non-informative, while no parameter prior is required. It has been previously shown that for discrete data, this criterion can be computed in linear time for Bayesian networks with no arcs, and in quadratic time for the so called Naive Bayes network structure. Here we extend the previous results by showing how to compute the NML criterion in polynomial time for tree-structured Bayesian networks. The order of the polynomial depends on the number of values of the variables, but neither on the number of variables itself, nor on the sample size.

KEYWORDS

Machine Learning, Bayesian Networks, Minimum Description Length, Normalized Maximum Likelihood.

1 INTRODUCTION

We consider the problem of learning a Bayesian network structure, based on a sample of data collected from the domain to be studied. We focus on the *score-based* approach, where first a model selection score is defined, yielding a goodness criterion that can be used for comparing different model structures, and any search method of choice can then be used for finding the structure with the highest score.

In this paper we study the problem of choosing and computing an appropriate model selection criterion. Naturally, any reasonable criterion must possess some desirable optimality properties. For a Bayesian, the most obvious choice is to use the model structure posterior, given the data and some model structure prior that has to be fixed in advance. Assuming a uniform prior over the possible structures, this leaves us with the *marginal likelihood*, which is the most commonly used criterion for learning Bayesian networks. Calculation of the marginal likelihood requires us to define a prior distribution over the parameters defined by the model structure under consideration. Under certain

assumptions, computing the marginal likelihood is then straightforward, see e.g. [1, 2]. Perhaps somewhat surprisingly, determining an adequate prior for the model parameters of a given class, in an objective manner has turned out to be a most difficult problem.

The uniform parameter prior sounds like the obvious candidate for a *non-informative* prior distribution, but it is not transformation-invariant, and produces different marginal likelihood scores for dependence-equivalent model structures [2]. This is due to the fact that there is no objective way of defining uniformity, but any prior can be uniform at most *with respect to a chosen representation*. The problem of transformation-invariance can be remedied by using the prior distribution suggested in [3], but this still leaves us with a single parameter, the *equivalent sample size*, the value of which is highly critical with respect to the result of the model structure search. Alternatively, one might resort to using the transformation-invariant Jeffreys prior, but although it can in the Bayesian network setting be formulated explicitly [4], computing it appears to be quite difficult in practice.

For the above reasons, in this paper we take the alternative approach of using the information-theoretic *normalized maximum likelihood* (NML) criterion [5, 6] as the model selection criterion. The NML score is – under certain conditions – asymptotically equivalent to the marginal likelihood with the Jeffreys prior [6], but it does not require us to define a prior distribution on the model parameters. Based on the data at hand only, it is fully objective, non-informative and transformation-invariant. What is more, the NML distribution can be shown to be the optimal distribution in a certain intuitively appealing sense. It may be used for selection of a model class among very different candidates. We need not assume a model family of nested model classes or the like, but we may compete against each other any types of model classes for which we can compute the NML distribution. Consequently, the NML score for Bayesian networks is of great importance both as a theoretically interesting problem and as a practically useful model selection criterion.

Although the NML criterion yields a theoretically very appealing model selection criterion, its usefulness in practice depends on the computational complexity of the method. In this paper we consider Bayesian network models for discrete data, where all the conditional distributions between the variables are assumed to be multinomial. For a single multinomial variable (or, an empty Bayesian network with no arcs), the value of the NML criterion can be computed in linear time [7], and for the Naive Bayes structure in quadratic time [8]. In this paper we consider more general forest-shaped network structures, and introduce an algorithm for computing the NML score in polynomial time – where the order of the polynomial depends on the number of possible values of the network variables. Although the problem of computing the NML for general Bayesian network structures remains unsolved, this work represents another step towards that goal.

The paper is structured as follows. In Section 2 we briefly review some basic properties of the NML distribution. Section 3 introduces the Bayesian Forest model family and some inevitable notation. The algorithm that calculates the NML distribution for Bayesian forests is developed in Section 4 and summarized in Section 5. We close with the concluding remarks of Section 6.

2 PROPERTIES OF THE NML DISTRIBUTION

The NML distribution, founding on the *Minimum Description Length* (MDL) principle, has several desirable properties. Firstly, it automatically protects against overfitting in the model class selection process. Secondly, there is no need to assume that there exists some underlying "true" model, while most other statistical methods do: in NML the model class is only used as a technical device to describe the data, not as a hypothesis. Consequently, the model classes amongst which to choose are allowed to be of utterly different types; any collection of model classes may be considered as long as the corresponding NML distributions can be computed. For this reason we find it important to push the boundaries of NML computability and develop algorithms that extend to more and more complex model families.

NML is closely related to Bayesian inference. There are, however, some fundamental differences dividing the two, the most important being that NML is not dependent on any prior distribution, it only uses the data at hand. For more discussion on the theoretical motivations behind NML and the MDL principle see, e.g., [6, 9, 10, 11, 12, 13].

In the following, we give the definition of the NML distribution and discuss some of its theoretical properties.

2.1 Definition of a Model Class and Family

Let $\mathbf{x}^n = (x_1, \ldots, x_n)$ be a data sample of n outcomes, where each outcome x_j is an element of some space of observations $\mathcal{X}$. The n-fold Cartesian product $\mathcal{X} \times \cdots \times \mathcal{X}$ is denoted by $\mathcal{X}^n$, so that $\mathbf{x}^n \in \mathcal{X}^n$. Consider a set $\Theta \subseteq \mathbb{R}^d$, where d is a positive integer. A class of parametric distributions indexed by the elements of Θ is called a *model class*. That is, a model class $\mathcal{M}$ is defined as

$$\mathcal{M} = \{P(\cdot \mid \boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta\}, \tag{1}$$

and the set Θ is called a *parameter space*.

Consider a set $\Phi \subseteq \mathbb{R}^e$, where e is a positive integer. Define a set $\mathcal{F}$ by

$$\mathcal{F} = \{\mathcal{M}(\boldsymbol{\phi}) : \boldsymbol{\phi} \in \Phi\}. \tag{2}$$

The set $\mathcal{F}$ is called a *model family*, and each of the elements $\mathcal{M}(\boldsymbol{\phi})$ is a model class. The associated parameter space is denoted by $\Theta_{\boldsymbol{\phi}}$. The model class selection problem can now be defined as a process of finding the parameter vector $\boldsymbol{\phi}$, which is optimal according to some pre-determined criteria.

2.2 The NML Distribution

One of the most theoretically and intuitively appealing model class selection criteria is the *Normalized Maximum Likelihood*. Denote the parameter vector that maximizes the likelihood of data $\mathbf{x}^n$ for a given model class $\mathcal{M}(\boldsymbol{\phi})$ by $\hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\phi}))$:

$$\hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\phi})) = \arg\max_{\boldsymbol{\theta} \in \Theta_{\boldsymbol{\phi}}} \{P(\mathbf{x}^n \mid \boldsymbol{\theta})\}. \tag{3}$$

The *normalized maximum likelihood* (NML) distribution [5] is now defined as

$$P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(\boldsymbol{\phi})) = \frac{P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\phi})))}{\mathcal{C}(\mathcal{M}(\boldsymbol{\phi}), n)}, \tag{4}$$

where the normalizing term $\mathcal{C}(\mathcal{M}(\boldsymbol{\phi}), n)$ in the case of discrete data is given by

$$\mathcal{C}(\mathcal{M}(\boldsymbol{\phi}), n) = \sum_{\mathbf{y}^n \in \mathcal{X}^n} P(\mathbf{y}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{y}^n, \mathcal{M}(\boldsymbol{\phi}))), \tag{5}$$

and the sum goes over the space of data samples of size n. If the data is continuous, the sum is replaced by the corresponding integral. From this definition, it is immediately evident that NML is invariant with respect to any kind of parameter transformation, since such transformation does not affect the maximum likelihood $P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\phi})))$.

In the MDL literature – which views the model class selection problem as a task of minimizing the resulting code length – the minus logarithm of (4) is referred to as the *stochastic complexity* of the data $\mathbf{x}^n$ given model class $\mathcal{M}(\boldsymbol{\phi})$ and the logarithm of the normalizing sum $\log \mathcal{C}(\mathcal{M}(\boldsymbol{\phi}), n)$ is referred to as the *parametric complexity* or *(minimax) regret* of $\mathcal{M}(\boldsymbol{\phi})$.

The NML distribution (4) has several important theoretical optimality properties. The first one is that NML provides a unique solution to the minimax problem posed in [5],

$$\min_{\hat{P}} \max_{\mathbf{x}^n} \log \frac{P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\phi})))}{\hat{P}(\mathbf{x}^n \mid \mathcal{M}(\boldsymbol{\phi}))} \tag{6}$$

i.e., the minimizing $\hat{P}$ is the NML distribution, and it assigns a probability to any data that differs from the highest achievable probability within the model class – the *maximum likelihood*

$P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\phi})))$ – by the constant factor $\mathcal{C}(\mathcal{M}(\boldsymbol{\phi}), n)$. In this sense, the NML distribution can be seen as a truly uniform prior, with respect to the data itself, not its representation by a model class $\mathcal{M}(\boldsymbol{\phi})$. In other words, the NML distribution is the *minimax optimal universal model*. The term universal model in this context means that the NML distribution represents (or mimics) the behaviour of all the distributions in the model class $\mathcal{M}(\boldsymbol{\phi})$. Note that the NML distribution itself does not have to belong to the model class, and typically it does not.

A related property of NML was proven in [11]. It states that NML also minimizes

$$\min_{\hat{P}} \max_{g} E_g \log \frac{P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\boldsymbol{\phi})))}{\hat{P}(\mathbf{x}^n \mid \mathcal{M}(\boldsymbol{\phi}))} \tag{7}$$

where the expectation is taken over $\mathbf{x}^n$ and g is the worst-case data generating distribution.

3 THE BAYESIAN FOREST MODEL FAMILY

We assume m variables $X_1, \ldots, X_m$ with given value cardinalities $K_1, \ldots, K_m$. We further assume a data matrix $\mathbf{x}^n = (x_{ji}) \in \mathcal{X}^n$, $1 \leq j \leq n$ and $1 \leq i \leq m$, given.

A Bayesian network structure $\mathcal{G}$ encodes independence assumptions so that if each variable X_i is represented as a node in the network, then the joint probability distribution factorizes into a product of local probability distributions, one for each node, conditioned on its parent set. We define a *Bayesian forest* (BF) to be a Bayesian network structure $\mathcal{G}$ on the node set $X_1, \ldots, X_m$ which assigns at most one parent $X_{pa(i)}$ to any node X_i. Consequently, a *Bayesian tree* is a connected Bayesian forest and a Bayesian forest breaks down into component trees, i.e. connected subgraphs. The root of each such component tree lacks a parent, in which case we write $pa(i) = \emptyset$.

The parent set of a node X_i thus reduces to a single value $pa(i) \in \{1, \ldots, i-1, i+1, \ldots, m, \emptyset\}$. Let further $ch(i)$ denote the set of children of node X_i in $\mathcal{G}$ and $ch(\emptyset)$ denote the "children of none", i.e. the roots of the component trees of $\mathcal{G}$.

The corresponding model family $\mathcal{F}_{\mathrm{BF}}$ can be indexed by the network structure $\mathcal{G} \in \Phi_{BF} \subset \mathbb{N} \subset \mathbb{R}$ according to some enumeration of all Bayesian forests on $(X_1, \ldots, X_m)$:

$$\mathcal{F}_{BF} = \{\mathcal{M}(\mathcal{G}) : \mathcal{G} \text{ is a forest}\}. \tag{8}$$

Given a forest model class $\mathcal{M}(\mathcal{G})$, we index each model by a parameter vector $\boldsymbol{\theta}$ in the corresponding parameter space $\Theta_{\mathcal{G}}$.

$$\Theta_{\mathcal{G}} = \{\boldsymbol{\theta} = (\theta_{ikl}) : \ \theta_{ikl} \geq 0, \ \sum_{l} \theta_{ikl} = 1, \ i = 1, \ldots, m, \ k = 1, \ldots, K_{pa(i)}, \ l = 1, \ldots, K_i\}, \tag{9}$$

where we define $K_\emptyset := 1$ in order to unify notation for root and non-root nodes. Each such θ_{ikl} defines a probabilty

$$\theta_{ikl} = P(X_i = l \mid X_{pa(i)} = k, \ \mathcal{M}(\mathcal{G}), \ \boldsymbol{\theta}) \tag{10}$$

where we interpret $X_\emptyset = 1$ as a null condition.

The joint probability distribution that such a model $M = (\mathcal{G}, \ \boldsymbol{\theta})$ assigns to a data vector $\mathbf{x} = (x_1, \ldots, x_m)$ becomes

$$P(\mathbf{x} \mid \mathcal{M}(\mathcal{G}), \boldsymbol{\theta}) = \prod_{i=1}^{m} P(X_i = x_i \mid X_{pa(i)} = x_{pa(i)}, \mathcal{M}(\mathcal{G}), \boldsymbol{\theta}) = \prod_{i=1}^{m} \theta_{i, x_{pa(i)}, x_i}. \tag{11}$$

For a sample $\mathbf{x}^n = (x_{ji})$ of n vectors $\mathbf{x}_j$ we define the corresponding frequencies

$$f_{ikl} := |\{j : x_{ji} = l \ \wedge \ x_{j,pa(i)} = k\}| \quad \text{and} \quad f_{il} := |\{j : x_{ji} = l\}| = \sum_{k=1}^{K_{pa(i)}} f_{ikl}. \tag{12}$$

By definition, for any component tree root X_i we have $f_{il} = f_{i1l}$. The probability assigned to an i.i.d. sample $\mathbf{x}^n$ can then be written as

$$P(\mathbf{x}^n \mid \mathcal{M}(\mathcal{G}), \boldsymbol{\theta}) = \prod_{i=1}^{m} \prod_{k=1}^{K_{pa(i)}} \prod_{l=1}^{K_i} \theta_{ikl}^{f_{ikl}}, \tag{13}$$

which is maximized at

$$\hat{\theta}_{ikl}(\mathbf{x}^n, \mathcal{M}(\mathcal{G})) = \frac{f_{ikl}}{f_{pa(i),k}}, \tag{14}$$

where we define $f_{\emptyset,1} := n$. The maximum data likelihood thereby is

$$P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n, \mathcal{M}(\mathcal{G}))) = \prod_{i=1}^{m} \prod_{k=1}^{K_{pa(i)}} \prod_{l=1}^{K_i} \left(\frac{f_{ikl}}{f_{pa(i),k}} \right)^{f_{ikl}}. \tag{15}$$

4 CALCULATING THE NML DISTRIBUTION

The goal is to calculate the NML distribution $P_{\mathrm{NML}}(\mathbf{x}^n \mid \mathcal{M}(\mathcal{G}))$ defined in (4). This consists of calculating the maximum data likelihood (15) and the normalizing term $\mathcal{C}(\mathcal{M}(\mathcal{G}), n)$ given in (5). The former involves frequency counting – one sweep through the data – and multiplication of the appropriate values. This can be done in time $\mathcal{O}(n + \sum_i K_i K_{pa(i)})$. The latter involves a sum exponential in n, which clearly makes it the computational bottleneck of the algorithm.

Our approach is to break up the normalizing sum in (5) into terms corresponding to subtrees with given frequencies in either their root or its parent. We then calculate the complete sum by sweeping through the graph once, bottom-up. The exact ordering will be irrelevant, as long as we deal with each node before its parent. Let us now introduce the needed notation.

Let $\mathcal{G}$ be a given Bayesian forest. In order to somewhat shorten our notation, from now on we do not write out the model class $\mathcal{M}(\mathcal{G})$ anymore, as it may be assumed fixed. We thus write e.g. $P(\mathbf{x}^n \mid \boldsymbol{\theta})$, meaning $P(\mathbf{x}^n \mid \boldsymbol{\theta}, \mathcal{M}(\mathcal{G}))$. When in the following we restrict to subsets of the attribute space, we implicitly restrict the model class accordingly, e.g. in (16) below, we write $P(\mathbf{x}^n_{sub(i)} \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{sub(i)}))$ as a short notation for $P(\mathbf{x}^n_{sub(i)} \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{sub(i)}), \mathcal{M}(\mathcal{G}_{sub(i)}))$.

For any node X_i denote the subtree rooting in X_i by $\mathcal{G}_{sub(i)}$ and the forest built up by all descendants of X_i by $\mathcal{G}_{dsc(i)}$. The corresponding data domains are $\mathcal{X}_{sub(i)}$ and $\mathcal{X}_{dsc(i)}$, respectively. Denote the partial normalizing sum over all n-instantiations of a subtree by

$$\mathcal{C}_i(n) := \sum_{\mathbf{x}^n_{sub(i)} \in \mathcal{X}^n_{sub(i)}} P(\mathbf{x}^n_{sub(i)} \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{sub(i)})) \tag{16}$$

and for any vector $\mathbf{x}_i^n \in X_i^n$ with frequencies $\mathbf{f}_i = (f_{i1}, \ldots, f_{iK_i})$ we define

$$\mathcal{C}_i(n \mid \mathbf{f}_i) := \sum_{\mathbf{x}^n_{dsc(i)} \in \mathcal{X}^n_{dsc(i)}} P(\mathbf{x}^n_{dsc(i)}, \mathbf{x}^n_i \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{dsc(i)}, \mathbf{x}^n_i)) \tag{17}$$

to be the corresponding sum with fixed root instantiation, summing only over the attribute space spanned by the descendants on X_i. Note, that we condition on $\mathbf{f}_i$ on the left-hand side, and on $\mathbf{x}_i^n$ on the right-hand side of the definition. This needs to be justified. Interestingly, while the terms in the sum depend on the ordering of $\mathbf{x}_i^n$, the sum itself depends on $\mathbf{x}_i^n$ only through its frequencies $\mathbf{f}_i$. To see this pick any two representatives $\mathbf{x}_i^n$ and $\bar{\mathbf{x}}_i^n$ of $\mathbf{f}_i$ and find, e.g. after lexicographical ordering of the elements, that

$$\{(\mathbf{x}^n_i, \mathbf{x}^n_{dsc(i)}) : \mathbf{x}^n_{dsc(i)} \in \mathcal{X}^n_{dsc(i)}\} = \{(\bar{\mathbf{x}}^n_i, \mathbf{x}^n_{dsc(i)}) : \mathbf{x}^n_{dsc(i)} \in \mathcal{X}^n_{dsc(i)}\} \tag{18}$$

Next, we need to define corresponding sums over $\mathcal{X}_{sub(i)}$ with the frequencies at the subtree root parent $X_{pa(i)}$ given. For any $\mathbf{f}_{pa(i)} \sim \mathbf{x}^n_{pa(i)} \in X^n_{pa(i)}$ define

$$\mathcal{L}_i(n \mid \mathbf{f}_{pa(i)}) := \sum_{\mathbf{x}^n_{sub(i)} \in \mathcal{X}^n_{sub(i)}} P(\mathbf{x}^n_{sub(i)} \mid \mathbf{x}^n_{pa(i)}, \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{sub(i)}, \mathbf{x}^n_{pa(i)})) \tag{19}$$

Again, this is well-defined since any other representative $\bar{\mathbf{x}}^n_{pa(i)}$ of $\mathbf{f}_{pa(i)}$ yields summing the same terms in different order.

After having introduced this notation, we now briefly outline the algorithm and – in the following subsections – give a more detailed description of the steps involved. As stated before, we go through $\mathcal{G}$ bottom-up. At each inner node X_i, we receive $\mathcal{L}_j(n \mid \mathbf{f}_i)$ from each child X_j, $j \in ch(i)$. Correspondingly, we are required to send $\mathcal{L}_i(n \mid \mathbf{f}_{pa(i)})$ up to the parent $X_{pa(i)}$. At each component tree root X_i we then calculate the sum $\mathcal{C}_i(n)$ for the whole connectivity component and then combine these sums to get the normalizing sum $\mathcal{C}(n)$ for the complete forest $\mathcal{G}$.

4.1 Leaves

It turns out, that for a leave node X_i we can calculate the terms $\mathcal{L}_i(n \mid \mathbf{f}_{pa(i)})$ without listing the frequencies $\mathbf{f}_i$ at X_i itself. The parent frequencies $\mathbf{f}_{pa(i)}$ split the n data vectors into $K_{pa(i)}$ subsets of sizes $f_{pa(i),1}, \ldots, f_{pa(i),K_{pa(i)}}$ and each of them can be modelled independently as a multinomial. We have

$$\mathcal{L}_i(n \mid \mathbf{f}_{pa(i)}) = \prod_{k=1}^{K_{pa(i)}} \mathcal{C}_{\mathrm{MN}}(K_i, f_{pa(i),k}). \tag{20}$$

where

$$\mathcal{C}_{\mathrm{MN}}(K_i, n') = \sum_{\mathbf{x}_i \in X_i} P(\mathbf{x}_i^{n'} \mid \hat{\boldsymbol{\theta}}(x_i^{n'}), \mathcal{M}_{\mathrm{MN}}(K_i)) = \sum_{\mathbf{x}_i \in X_i} \prod_{l=1}^{K_i} \left(\frac{f_{il}}{n'}\right)^{f_{il}} \tag{21}$$

is the normalizing sum (5) for the multinomial model class $\mathcal{M}_{\mathrm{MN}}(K_i)$ for a single discrete variable with K_i values, see e.g. [8, 14, 7] for details. [7] derives a simple recurrence for these terms, namely

$$\mathcal{C}_{\mathrm{MN}}(K+2, n') = \mathcal{C}_{\mathrm{MN}}(K+1, n') + \frac{n'}{K}\mathcal{C}_{\mathrm{MN}}(K, n'), \tag{22}$$

which we can use to precalculate all $\mathcal{C}_{\mathrm{MN}}(K_i, n')$ (for $n' = 0, \ldots, n$) in linear time each, i.e. in quadratic time altogether, for details see [7].

4.2 Inner Nodes

For inner nodes X_i we divide the task into two steps. First collect the messages $\mathcal{L}_j(n \mid \mathbf{f}_i)$ sent by each child $X_j \in ch(i)$ into partial sums $\mathcal{C}_i(n \mid \mathbf{f}_i)$ over $\mathcal{X}_{dsc(i)}$, then "lift" these to sums $\mathcal{L}_i(n \mid \mathbf{f}_{pa(i)})$ over $\mathcal{X}_{sub(i)}$ which are the messages to the parent.

The first step is simple. Given an instantiation $\mathbf{x}_i^n$ at X_i or, equivalently, the corresponding frequencies $\mathbf{f}_i$, the subtrees rooting in the children $ch(i)$ of X_i become independent of each other.

Thus we have

$$\mathcal{C}_i(n \mid \mathbf{f}_i) = \sum_{\mathbf{x}^n_{dsc(i)} \in \mathcal{X}^n_{dsc(i)}} P(\mathbf{x}^n_{dsc(i)}, \mathbf{x}^n_i \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{dsc(i)}, \mathbf{x}^n_i)) \tag{23}$$

$$= P(\mathbf{x}^n_i \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{dsc(i)}, \mathbf{x}^n_i)) \left(\sum_{\mathbf{x}^n_{dsc(i)} \in \mathcal{X}^n_{dsc(i)}} \prod_{j \in ch(i)} P(\mathbf{x}^n_{dsc(i)|sub(j)} \mid \mathbf{x}^n_i, \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{dsc(i)}, \mathbf{x}^n_i)) \right) \tag{24}$$

$$= P(\mathbf{x}^n_i \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{dsc(i)}, \mathbf{x}^n_i)) \prod_{j \in ch(i)} \left(\sum_{\mathbf{x}^n_{sub(j)} \in \mathcal{X}^n_{sub(j)}} P(\mathbf{x}^n_{sub(j)} \mid \mathbf{x}^n_i, \hat{\boldsymbol{\theta}}(\mathbf{x}^n_{dsc(i)}, \mathbf{x}^n_i)) \right) \tag{25}$$

$$= \prod_{l=1}^{K_i} \left(\frac{f_{il}}{n} \right)^{f_{il}} \prod_{j \in ch(i)} \mathcal{L}_j(n \mid \mathbf{f}_i) \tag{26}$$

where $\mathbf{x}^n_{dsc(i)|sub(j)}$ is the restriction of $\mathbf{x}_{dsc(i)}$ to columns corresponding to nodes in $\mathcal{G}_{sub(j)}$. We have used (17) for (23), (11) for (24) and (25) and finally (15) and (19) for (26).

Now we calculate the outgoing messages $\mathcal{L}_i(n \mid \mathbf{f}_{pa(i)})$ from the incoming messages we have just combined into $\mathcal{C}_i(n \mid \mathbf{f}_i)$. This is the most demanding part of the algorithm, as we need to list all possible conditional frequencies, of which there are $\mathcal{O}(n^{K_i K_{pa(i)} - 1})$ many, the -1 being due to the sum-to-n constraint. For fixed i, we arrange the conditional frequencies f_{ikl} into a matrix $\mathbf{F} = (f_{ikl})$ and define its marginals

$$\boldsymbol{\rho}(\mathbf{F}) := \left(\sum_k f_{ik1}, \ldots, \sum_k f_{ikK_i} \right) \quad \text{and} \quad \boldsymbol{\gamma}(\mathbf{F}) := \left(\sum_l f_{i1l}, \ldots, \sum_l f_{iK_{pa(i)}l} \right) \tag{27}$$

to be the vectors obtained by summing the rows of $\mathbf{F}$ and the columns of $\mathbf{F}$, respectively. Each such matrix then corresponds to a term $\mathcal{C}_i(n \mid \boldsymbol{\rho}(\mathbf{F}))$ and a term $\mathcal{L}_i(n \mid \boldsymbol{\gamma}(\mathbf{F}))$. Formally we have

$$\mathcal{L}_i(n \mid \mathbf{f}_{pa(i)}) = \sum_{\mathbf{F}: \boldsymbol{\gamma}(\mathbf{F}) = \mathbf{f}_{pa(i)}} \mathcal{C}_i(n \mid \boldsymbol{\rho}(\mathbf{F})). \tag{28}$$

4.3 Component Tree Roots

For a component tree root $X_i \in ch(\emptyset)$ we do not need to pass any message upward. All we need is the complete sum over the component tree

$$\mathcal{C}_i(n) = \sum_{\mathbf{f}_i} \frac{n!}{f_{i1}! \ldots f_{iK_i}!} \mathcal{C}_i(n \mid \mathbf{f}_i) \tag{29}$$

where the $\mathcal{C}_i(n \mid \mathbf{f}_i)$ are calculated using (26). The summation goes over all non-negative integer vectors $\mathbf{f}_i$ summing to n. The above is trivially true since we sum over all instantiations $\mathbf{x}^n_i$ of X^n_i and group like terms – corresponding to the same frequency vector $\mathbf{f}_i$ – keeping track of their respective count, namely $n!/(f_{i1}! \ldots f_{iK_i}!)$.

5 THE ALGORITHM

For the complete forest $\mathcal{G}$ we simply multiply the sums over its tree components. Since these are independent of each other, in analogy to (23)-(26) we have

$$\mathcal{C}(n) = \prod_{i \in ch(\emptyset)} \mathcal{C}_i(n). \tag{30}$$

Algorithm 1 collects all the above into pseudo-code.

Algorithm 1
Computing $P_{\mathrm{NML}}(\mathbf{x}^n)$ for a Bayesian Forest $\mathcal{G}$.

1: Count all frequencies f_{ikl} and f_{il} from the data $\mathbf{x}^n$
2: Compute $P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n))= \prod_{i=1}^{m} \prod_{k=1}^{K_{pa(i)}} \prod_{l=1}^{K_i} \left(\frac{f_{ikl}}{f_{pa(i),k}}\right)^{f_{ikl}}$
3: **for** $K' = 1, \ldots, K_{max} := \max_{i:X_i \text{ is a leaf}} \{K_i\}$ and $n' = 0, \ldots, n$ **do**
4: Compute $\mathcal{C}_{\mathrm{MN}}(K', n')$ using recurrence (22)
5: **end for**
6: **for** each node X_i in some bottom-up order **do**
7: **if** X_i is a leaf **then**
8: **for** each frequency vector $\mathbf{f}_{pa(i)}$ of $X_{pa(i)}$ **do**
9: Compute $\mathcal{L}_i(n \mid \mathbf{f}_{pa(i)}) = \prod_{k=1}^{K_{pa(i)}} \mathcal{C}_{\mathrm{MN}}(K_i, \mathbf{f}_{pa(i)k})$
10: **end for**
11: **else if** X_i is an inner node **then**
12: **for** each frequency vector $\mathbf{f}_i$ of X_i **do**
13: Compute $\mathcal{C}_i(n \mid \mathbf{f}_i) = \prod_{l=1}^{K_i} \left(\frac{f_{il}}{n}\right)^{f_{il}} \prod_{j \in ch(i)} \mathcal{L}_j(n \mid \mathbf{f}_i)$
14: **end for**
15: initialize $\mathcal{L}_i \equiv 0$
16: **for** each non-negative $K_i \times K_{pa(i)}$ integer matrix $\mathbf{F}$ with entries summing to n **do**
17: $\mathcal{L}_i(n \mid \boldsymbol{\gamma}(\mathbf{F})) \mathrel{+}= \mathcal{C}_i(n \mid \boldsymbol{\rho}(\mathbf{F}))$
18: **end for**
19: **else if** X_i is a component tree root **then**
20: Compute $\mathcal{C}_i(n) = \sum_{\mathbf{f}_i} \prod_{l=1}^{K_i} \left(\frac{f_{il}}{n}\right)^{f_{il}} \prod_{j \in ch(i)} \mathcal{L}_j(n \mid \mathbf{f}_i)$
21: **end if**
22: **end for**
23: Compute $\mathcal{C}(n) = \prod_{i \in ch(\emptyset)} \mathcal{C}_i(n)$
24: Output $P_{\mathrm{NML}}(\mathbf{x}^n) = \frac{P(\mathbf{x}^n \mid \hat{\boldsymbol{\theta}}(\mathbf{x}^n))}{\mathcal{C}(n)}$

The time complexity of this algorithm is $\mathcal{O}(n^{K_i K_{pa(i)}-1})$ for each inner node, $\mathcal{O}(n(n+K_i))$ for each leaf and $\mathcal{O}(n^{K_i-1})$ for a component tree root of $\mathcal{G}$. When all $m' < m$ inner nodes are binary it runs in $\mathcal{O}(m'n^3)$, independent of the number of values of the leaf nodes. This is polynomial wrt. the sample size n, while applying (5) directly for computing $\mathcal{C}(n)$ requires exponential time. The order of the polynomial depends on the attribute cardinalities: the algorithm is exponential wrt. the number of values a non-leaf variable can take.

Finally, note that we can speed up the algorithm when $\mathcal{G}$ contains multiple copies of some subtree. Also we have $\mathcal{C}_i/\mathcal{L}_i(n \mid \mathbf{f}_i) = \mathcal{C}_i/\mathcal{L}_i(n \mid \pi(\mathbf{f}_i))$ for any permutation π of the entries of $\mathbf{f}_i$. However, this does not lead to considerable gain, at least in $\mathcal{O}$-notation. Also, we can see that in line 16 of the algorithm we enumerate all frequency matrices $\mathbf{F}$, while in line 17 we sum the same terms whenever the marginals of $\mathbf{F}$ are the same. Unfortunately, computing the number of non-negative integer matrices with given marginals is a #P-hard problem already when one of the matrix dimensions is fixed to 2, as proven in [15]. This suggests that for this task there may not exist an algorithm that is polynomial in all input quantities. The algorithm presented here is polynomial in both the sample size n and the graph size m. For attributes with relatively few values, the polynomial is of tolerable degree.

6 CONCLUSION

The information-theoretic normalized maximum likelihood (NML) criterion offers an interesting, non-informative approach to Bayesian network structure learning. It has some links to the Bayesian marginal likelihood approach — NML converges asymptotically to the marginal likelihood with the

Jeffreys prior — but it avoids the technical problems related to parameter priors as no explicitly defined prior distributions are required. Unfortunately a straightforward implementation of the criterion requires exponential time. In this paper we presented a computationally feasible algorithm for computing the NML criterion for tree-structured Bayesian networks: Bayesian trees and forests (collections of trees).

The time complexity of the algorithm presented here is polynomial with respect to the sample size and the number of domain variables, but the order of the polynomial depends on the number of values of the inner nodes in the tree to be evaluated, which makes the algorithm impractical for some domains. However, we consider this result as an important extension of the earlier results which were able to handle only Naive Bayes structures, i.e., Bayesian trees of depth one with no inner nodes. In the future we plan to test the validity of the suggested NML approach in practical problem domains, and we also wish to extend this approach to more complex Bayesian network structures.

7 ACKNOWLEDGEMENTS

This work was supported in part by the Finnish Funding Agency for Technology and Innovation under projects PMMA, KUKOT and SIB, by the Academy of Finland under project CIVI, and by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

[1] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[2] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, September 1995.

[3] W. Buntine. Theory refinement on Bayesian networks. In B. D'Ambrosio, P. Smets, and P. Bonissone, editors, *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann Publishers, 1991.

[4] P. Kontkanen, P. Myllymäki, T. Silander, H. Tirri, and P. Grünwald. On predictive distributions and Bayesian networks. *Statistics and Computing*, 10:39–54, 2000.

[5] Yu M. Shtarkov. Universal sequential coding of single messages. *Problems of Information Transmission*, 23:3–17, 1987.

[6] J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, January 1996.

[7] P. Kontkanen and P. Myllymäki. A linear-time algorithm for computing the multinomial stochastic complexity. *Submitted to Information Processing Letters*, 2007.

[8] P. Kontkanen, P. Myllymäki, W. Buntine, J. Rissanen, and H. Tirri. An MDL framework for data clustering. In P. Grünwald, I.J. Myung, and M. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*. The MIT Press, 2006.

[9] A. Barron, J. Rissanen, and B. Yu. The minimum description principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, October 1998.

[10] Q. Xie and A.R. Barron. Asymptotic minimax regret for data compression, gambling, and prediction. *IEEE Transactions on Information Theory*, 46(2):431–445, March 2000.

[11] J. Rissanen. Strong optimality of the normalized ML models as universal codes and information in data. *IEEE Transactions on Information Theory*, 47(5):1712–1717, July 2001.

[12] P. Grünwald. Minimum description length tutorial. In P. Grünwald, I.J. Myung, and M. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*, pages 23–79. The MIT Press, 2006.

[13] J. Rissanen. Lectures on statistical modeling theory, August 2005. Available online at www.mdl-research.org.

[14] P. Kontkanen and P. Myllymäki. MDL histogram density estimation. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (to appear)*, San Juan, Puerto Rico, March 2007.

[15] M.E. Dyer, R. Kannan, and J. Mount. Sampling contingency tables. *Random Structures and Algorithms*, 10(4):487–506, 1997.

Publication III

Hannes Wettig and Roman Yangarber

Probabilistic Models for Aligning Etymological Data

Pp. 246–253 in Proceedings of the 18th Nordic Conference of Computational Linguistics NODALIDA 2011
Editors: Bolette Sandford Pedersen, Gunta Nešpore and Inguna Skadiņa
NEALT Proceedings Series, Vol. 11 (2011), viii-ix

Probabilistic Models for Alignment of Etymological Data

Hannes Wettig, Roman Yangarber
Department of Computer Science
University of Helsinki, Finland
First.Last@cs.helsinki.fi

Abstract

This paper introduces several models for aligning etymological data, or for finding the best alignment at the sound or symbol level, given a set of etymological data. This will provide us a means of measuring the quality of the etymological data sets in terms of their internal consistency. Since one of our main goals is to devise automatic methods for aligning the data that are as objective as possible, the models make no a priori assumptions—e.g., no preference for vowel-vowel or consonant-consonant alignments. We present a baseline model and successive improvements, using data from Uralic language family.

1 Introduction

We present work on induction of alignment rules for etymological data in a project that studies genetic relationships among the Uralic language family. Our interest is in methods that are as objective as possible, i.e., rely only on the data rather than on prior assumptions or "universal" principles about the data, possible rules and alignments. Another goal is to derive measures of quality of data sets in terms of their internal consistency—a data-set that is more consistent should receive a higher score. We seek methods that analyze the data automatically in an unsupervised fashion. The question is whether a complete description of the correspondences can be discovered automatically, directly from raw etymological data—sets of cognate words from languages within the language family. Another way of looking at this is: what alignment rules are "inherently encoded" in a data-set (the *corpus*) itself. Thus, at present, our aim is to analyze given etymological data-sets, rather than to construct one from scratch.

Several approaches to etymological alignment have emerged over the last decade, summarized in section 2. In prior work, it was observed that etymology induction may have potential applications, among them aiding machine translation systems for resource-poor languages. Our interest is somewhat more theoretical; we are at present less interested in applications than in building models that are principled and avoid building ad-hoc heuristics into the models from the outset.

We review related work in Section 2, present a statement of the etymology alignment problem in Section 3, our models in Section 3, results in Section 5, and the next steps in Section 6.

1.1 Computational Etymology

Etymology involves several problems, including: determination of genetic relations among groups of languages, from raw linguistic data; discovering *regular sound correspondences* across languages in a given language family; reconstruction of proto-forms for a hypothetical parent language, from which the word-forms found in the daughter languages derive.

Computational etymology is interesting from the point of view of computational linguistics and machine learning. Computational methods can provide valuable feedback to the etymological/linguistic community. The methods can be evaluated by whether they perform certain aspects of etymological analysis correctly, that is, whether automatic analysis—at least in some cases—is able to produce results that match the theories established by manual analysis.

Why is computational etymology useful—can results obtained by automatic analysis clarify or improve upon established theories?

First, even if computational methods yield no new insights from the linguistic point of view, and only validate previously established theories, that would still be a useful result. Because computational approaches differ in nature from traditional linguistic methods, a matching result would serve

Bolette Sandford Pedersen, Gunta Nešpore and Inguna Skadiņa (Eds.)
NODALIDA 2011 Conference Proceedings, pp. 246–253

as a non-trivial, independent confirmation of correctness of traditional methods.

Second, while some major language families have been studied extensively from the etymological perspective, many have not. Language families such as the Indo-European have received more attention than others and have been studied in greater detail, mainly because more relevant data has been collected and available to scholars for a longer time. For the less-studied language families, automatic analysis will allow linguists to bootstrap results quickly, to provide a foundation for further, more detailed investigation.

Third, the significant matter of uncertainty: Most etymological resources—dictionaries and handbooks—label certain relationships as "dubious," to a varying degree, usually due to violation of some expected regularity. Different (re)sources contain different decisions, which result in conflicts, because they are based on different theories. There is currently no way to objectively assess the relative likelihood of competing theories. Uncertainty is typically not quantified in a disciplined way, making it difficult for the linguist to know just how un/likely a particular relationship may be.

When etymology is approached by computational means, decisions are made within a rigorous framework, which makes it possible to state in probabilistic terms how likely any decision is to be correct given the data, and the relative likelihood of competing hypotheses.

Finally, a serious problem in manual etymological analysis is the potential bias of the human investigator. Bias may arise for many reasons; for example, at the time when a certain relationship is accepted as valid, some relevant data may be unknown or unavailable to the researcher, or may be available but ignored. Automatic analysis has the advantage of using all available data, without bias.

2 Related Work

We use two digital Uralic etymological resources, *SSA—Suomen Sanojen Alkuperä* ("The Origin of Finnish Words"), (Itkonen and Kulonen, 2000), and StarLing, (Starostin, 2005). StarLing was originally based on (Rédei, 1988 1991), and differs in several respects from SSA. StarLing has under 2000 Uralic cognate sets, compared with over 5000 in SSA, and does not explicitly indicate dubious etymologies. However, Uralic data in StarLing is more evenly distributed, because it is not Finnish-centric like SSA is—cognate sets in StarLing are not required to contain a member from Finnish. StarLing also gives a *reconstructed* form for each cogset, which may be useful for testing algorithms that perform reconstruction.

We are experimenting with the Uralic data by implementing algorithms modeling various etymological processes. A method due to Kondrak, (Kondrak, 2002) learns one-to-one regular sound correspondences between pairs of related languages in the data. The method in (Kondrak, 2003) finds attested complex (one-to-many) correspondences. These models are somewhat simplistic in that they operate only on one language pair at a time, and do not model the *contexts* of the sound changes, while we know that most etymological changes are conditioned on context. Our implementation of (Bouchard-Côté et al., 2007) found correspondence rules with correct contexts, using more than two languages. However, we found that this model's running time did not scale if the number of languages is above three.

In validating our experiments we use rules found in, e.g., (Lytkin, 1973; Sinor, 1997).

The Uralic language family has not been studied by computational means previously.

3 Aligning Pairs of Words

We start with pairwise alignment: aligning two languages means aligning a list of pairs of words in the two languages, which our data set claims are related. The task is *alignment*, i.e., for each pair of words, finding which symbols correspond to each other. We expect that some symbols will align with themselves, while others have gone through changes over the time that the two related languages have been evolving separately. The simplest form of such alignment at the symbol level is a pair $(s, t) \in \Sigma \times T$, a single symbol s from the *source alphabet* Σ with a symbol t from the *target alphabet* T. We denote the sizes of the alphabets by $|\Sigma|$ and $|T|$, respectively.

Clearly, this type of atomic alignment alone does not enable us to align a source word $\mathbf{s}$ of length $|\mathbf{s}|$ with a target word $\mathbf{t}$ of length $|\mathbf{t}| \neq |\mathbf{s}|$.[1] We also need to allow *insertions* and *deletions*. We augment both alphabets with the empty symbol, denoted by a dot, and write $\Sigma_{\cdot}$ and $T_{\cdot}$ to refer to the augmented alphabets. We can now align word pairs such as *kaikki—kõik* (meaning "all" in

[1] We use boldface to denote words, as vectors of symbols.

Finnish and Estonian), for example, as either of:

k	a	i	k	k	i
|	|	|	|	|	|
k	$\tilde{o}$	i	k	.	.

k	a	.	i	k	k	i
|	|	|	|	|	|	|
k	.	$\tilde{o}$	i	k	.	.

The alignment on the right consists of the pairs of symbols: (k:k), (a:.), (.:õ), (i:i), (k:k), (k:.), (i:.).

Note that we speak of "*source*" and "*target*" language for convenience only—our models are completely symmetric, as will become apparent.

3.1 The Baseline Model

We wish to encode these aligned pairs as compactly as possible, following the Minimum Description Length Principle (MDL), see e.g. (Grünwald, 2007). Given a data corpus $D = (\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_N, \mathbf{t}_N)$ of N word pairs, we first choose an alignment of each word pair $(\mathbf{s}_i, \mathbf{t}_i)$, which we then use to "transmit" the data, by simply listing the sequence of the atomic pairwise symbol alignments.[2] In order for the code to be uniquely decodable, we also need to encode the word boundaries. This can be done by transmitting a special symbol $\#$ that we do not use in any other context, only at the end of a word.

Thus, we transmit objects, or *events* e, from the event space E, in this case:

$$E = \Sigma_{.} \times T_{.} \cup \big\{(\# : \#)\big\}$$

We do this by means of Bayesian Marginal Likelihood (Kontkanen et al., 1996), or *prequential* coding, giving the total code length as:

$$\begin{aligned} L_{base}(D) = &- \sum_{e \in E} \log \Gamma\big(c(e) + \alpha(e)\big) \\ &+ \sum_{e \in E} \log \Gamma\big(\alpha(e)\big) \\ &+ \log \Gamma \left[\sum_{e \in E} \big(c(e) + \alpha(e)\big) \right] \\ &- \log \Gamma \left[\sum_{e \in E} \alpha(e) \right] \end{aligned} \qquad (1)$$

The *count* $c(e)$ is the number of times event e occurs in a complete alignment of the corpus; in particular, $c(\# : \#) = N$ occurs as many times as there are word pairs. The alignment counts are maintained in a corpus-global *alignment matrix* M, where $M(i, j) = c(i : j)$. The $\alpha(e)$ are the (Dirichlet) priors on the events. In the baseline algorithm, we set $\alpha(e) = 1$ for all e, the so-called uniform prior, which does not favour any distribution over E, *a priori*. Note that this choice nulls the second line of equation 1.

Our baseline algorithm is simple: we first randomly align the entire corpus, then re-align one word pair at a time, greedily minimizing the total cost in Eq. 1, using dynamic programming.

In the Viterbi-like matrix below in Figure 1, each cell corresponds to a partial alignment: reaching cell (i, j) means having read off i symbols of the source and j symbols of the target word. We iterate this process, *re-aligning* the word pairs; i.e., for a given word pair, we subtract the contribution of its current alignment from the global count matrix, then re-align the word pair, then add the newly aligned events back to the global count matrix. Re-alignment continues until convergence.

Re-alignment Step: align a source word σ consisting of symbols $\sigma = [\sigma_1 ... \sigma_n] \in \Sigma^*$ with a target word $\tau = [\tau_1 ... \tau_m]$. We fill in the matrix via dynamic programming, e.g., top-to-bottom, left-to-right:[3]

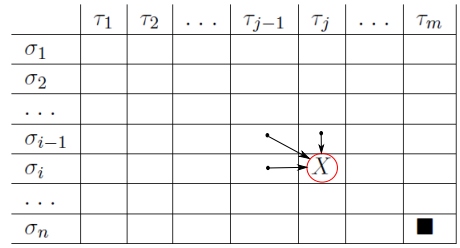

Figure 1: Dynamic programming matrix to search for most probable alignment.

Any alignment of σ and τ must correspond in a 1-1 fashion to some path through the matrix starting from top-left cell and terminating in bottom-right cell, moving only downward or rightward.

Each cell stores the probability of the *most probable* path to that point: the most probable way to have scanned the source word σ up to symbol σ_i and the target word up to τ_j, marked X in Figure 1.

[2] By *atomic* we mean that the symbols are not analyzed—in terms of their phonetic features—and treated by the baseline algorithm as atoms. In particular,

[3] NB: Figure 1 uses an extra column on the left and an extra row at the top, to store the costs for deleting symbols from the source *at the beginning of the word*, and from the target, respectively.

$$V(\sigma_i, \tau_j) = \min \begin{cases} V(\sigma_i, \tau_{j-1}) & +L(. : \tau_j) \\ V(\sigma_{i-1}, \tau_j) & +L(\sigma_i : .) \\ V(\sigma_{i-1}, \tau_{j-1}) & +L(\sigma_i : \tau_j) \end{cases} \quad (2)$$

In each case, the term $V(.)$ has been computed earlier by the dynamic programming; the term $L(.)$—the cost of aligning the two symbols—is a parameter of the model, computed in equation (3).

The parameters $L(e)$ or $P(e)$, for every observed event e, are computed from the *change* in the total code-length—the change that corresponds to the cost of adjoining the new event e to the set of previously observed events E:

$$L(e) = \Delta_e L = L\big(E \cup \{e\}\big) - L(E)$$

$$P(e) = 2^{-\Delta_e L} = \frac{2^{-L\left(E \cup \{e\}\right)}}{2^{-L(E)}} \quad (3)$$

Combining eqs. 1 and 3 gives the probability:

$$P(e) = \frac{c(e) + 1}{\sum_{e'} c(e') + |E|} \quad (4)$$

In particular, the cost of the most probable *complete* alignment of the two words will be stored in the bottom-right cell, $V(\sigma_n, \tau_m)$, marked ■.

3.2 The Two-Part Code

The baseline algorithm has revealed two problems. First, the algorithm seems to get stuck in local optima, and second, it produces many events with very low counts (occurring only once or twice).

To address the first problem we use simulated annealing with a sufficiently slow cooling schedule. This yields a reduction in the cost, and a better—more sparse—alignment count matrix.

The second problem is more substantial. Events that occur only once clearly have not got much support in the data. In theory, starting out from a common ancestor language, the number of changes that occurred in either language should be small. This does not necessarily mean many self-alignments of a symbol with itself, since a change may apply to many occurrences, e.g., all occurrences of the sound h at the end of a word have disappeared in Finnish. However, we still expect *sparse* data: we expect a relatively small portion of all *possible* events in E^+ to actually ever occur.

We incorporate this notion into our model by means of a two-part code. We first encode which events have occurred/have been observed: we first send the number of non-zero-count events—this costs $\log(|E| + 1)$ bits—and then transmit which subset E^+ of the events have non-zero counts—this costs $\log \binom{|E|}{|E^+|}$ bits. This first part of the code is called the *codebook*. Given the codebook, we transmit the complete data using Bayesian marginal likelihood. The code length becomes:

$$\begin{aligned} L_{tpc}(D) = &\log(|E| + 1) + \log \binom{|E|}{|E^+|} \\ &- \sum_{e \in E^+} \log \Gamma\big(c(e) + 1\big) \\ &+ \log \Gamma \left[\sum_{e \in E^+} \big(c(e) + 1\big) \right] - \log \Gamma(|E^+|) \end{aligned} \quad (5)$$

where E^+ denotes the set of events with non-zero counts, and we have set all $\alpha(e)$'s to one. Optimizing the above function with Simulated Annealing yields very good quality alignments.

3.3 Aligning Multiple Symbols

Multiple symbols are aligned in (Bouchard-Côté et al., 2007; Kondrak, 2003). In Estonian and Finnish appear frequent geminated consonants, which correspond to single symbols/sounds in other languages; diphthongs may align with single vowels. We allow correspondences of at most two symbols on both the source and the target side. Thus, the set of admissible kinds of events is:

$$K = \left\{ \begin{array}{lll} (\# : \#), & (\sigma : .), & (\sigma\sigma' : .), \\ (. : \tau), & (\sigma : \tau), & (\sigma\sigma' : t), \\ (. : \tau\tau'), & (\sigma : \tau\tau'), & (\sigma\sigma' : \tau\tau') \end{array} \right\} \quad (6)$$

We do not expect correspondences of the different types to behave similarly, so we encode the occurrences of all event kinds separately in the codebook part of the two-part code:

$$L_{mult}(D) = L(CB) + L(D|CB) \quad (7)$$

$$L(CB) = \sum_{k \in K} \left[\log(N_k + 1) + \log \binom{N_k}{M_k} \right] \quad (8)$$

$$\begin{aligned} L(D|CB) = &- \sum_{e \in E} \log \Gamma\big(c(e) + 1\big) \\ &+ \log \Gamma \left[\sum_{e \in E} \big(c(e) + 1\big) \right] - \log \Gamma(|E|) \end{aligned} \quad (9)$$

where N_k is the number of possible events of kind k and M_k the corresponding number of such events actually present in the alignment; by definition $\sum_k M_k \equiv |E|$.

Then, the parameters $P(e)$, for every observed event e, are again computed from the *change* in the code-length, eq. 3. But e may be of a kind that has been already observed previously, or it maybe of a new kind. Eq. 4 gives the formula for probability when $c(e) > 0$—that is, if $e \in E$—whereas

$$P(e) = \frac{1}{\sum_{e'} c(e') + |E|} \cdot \frac{|E|}{\sum_{e'} c(e') + |E| + 1} \cdot \frac{M_k + 1}{N_k - M_k} \tag{10}$$

when $e \notin E$, and e is of kind k. If the event e has been already observed, the value of $P(e)$ is computed by plugging equation (9) into eq. (3)—yielding eq. (4); if this is the first time e is observed, $P(e)$ is computed by plugging *both* eq. (9) and eq. (8) into eq. (3), since then the codebook also changes—yielding eq. (10).

Again we optimize this cost function by means of Simulated Annealing.

4 3-Dimensional Alignment

The baseline models section we restricted ourselves to aligning two languages. The alignment models allow us to learn 1-1 patterns of correspondence in the language family. The model is easily extensible to *any* number of languages. Other methods for aligning more than two languages were presented in (Bouchard-Côté et al., 2007).

We extend the 2-D model to three-dimensions as follows. We seek an alignment where symbols correspond to each other in a 1-1 fashion, as in the 2-D baseline. A three-dimensional alignment is a triplet of symbols $(\sigma : \tau : \xi) \in \Sigma \times \mathrm{T} \times \Xi$. For example, (*yhdeksän* : *üheksa* : *veχksa*)—meaning "9" in Finnish, Estonian and Mordva, can be aligned *simultaneously* as:

y	.	h	d	e	k	s	$ä$	n
|	|	|	|	|	|	|	|	|
$ü$	.	h	.	e	k	s	a	.
|	|	|	|	|	|	|	|	|
v	e	χ	.	.	k	s	a	.

In 3-D alignment, the input data contains all examples where words *in at least two* languages are present[4]—i.e., a word may be *missing* from one of the languages, (which allows us to utilize more of the data). Thus we have two types of examples: *complete* examples, those that have all three words present (as "9" above), and *incomplete* examples—containing words in only two languages. For example, the alignment of (*haamu*:—:*čama*)—meaning "ghost" in Finnish and Mordva—is an example where the cognate Estonian word is missing.

We must extend the 2-D alignment matrix and the 2-D Viterbi matrices to 3-D. The 3-D Viterbi matrix is directly analogous to the 2-D version. For the alignment counts in 3-D, we handle complete and incomplete examples separately.

4.1 Marginal 3-D Model

The "marginal" or "pairwise" 3-D alignment model aligns three languages simultaneously, using *only* the marginal 2-D matrices, each storing pairwise 2-D alignments. The marginal matrices for three languages are denoted $M_{\Sigma T}$, $M_{\Sigma\Xi}$ and $M_{T\Xi}$. The algorithm optimizes the total cost of the complete data, which is defined as the *sum* of the three 2-D costs obtained from applying prequential coding to the marginal alignment matrices.

When computing the cost for event $e = (\sigma, \tau, \xi)$, we consider complete and incomplete examples separately. In "incomplete" examples, we use the counts from the corresponding marginal matrix directly. E.g., for event count $c(e)$, where $e = (\sigma, -, \xi)$, and $-$ denotes the missing language, the event count is given by: $M_{\Sigma\Xi}(\sigma, \xi)$, and the cost of each alignment is computed as in the baseline model, directly in 2 dimensions.

In case when the data triplet is complete—fully observed—the alignment cost is computed as the *sum of the pairwise 2-D costs*, given by three marginal alignment count matrices:

$$\begin{aligned} L(\sigma : \tau : \xi) &= L_{\Sigma T}(\sigma : \tau) \\ &+ L_{\Sigma\Xi}(\sigma : \xi) \\ &+ L_{T\Xi}(\tau : \xi) \end{aligned} \tag{11}$$

The cost of each pairwise alignment is computed using prequential two-part coding, as in sec. 3.2.

Note that when we register a complete alignment (σ, τ, ξ), we register it in *each* of the base

[4] In the baseline 2-D algorithm, this requirement was also satisfied trivially, because in 2-D each example contains a word from both the source and the target language.

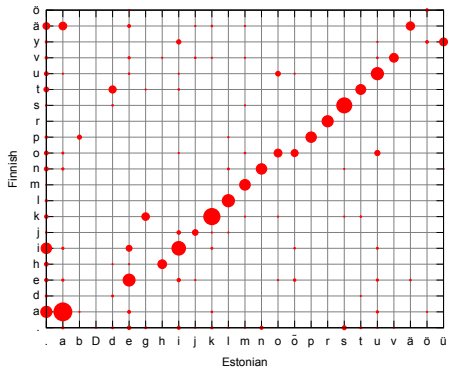

Figure 2: Alignment count matrix for Estonian-Finnish, using the two-part code.

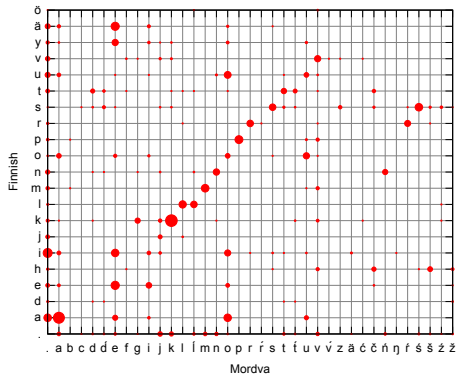

Figure 3: Mordva-Finnish 2-part code alignment.

matrices—we increment each of the marginal counts: $M_{\Sigma T}(\sigma, \tau)$, $M_{\Sigma \Xi}(\sigma, \xi)$, and $M_{T \Xi}(\tau, \xi)$. To deregister, we decrement all three counts.

To calculate the transition costs in the Viterbi algorithm, we also have two cases, complete and incomplete. For incomplete examples, we perform Viterbi in 2-D, using the costs directly from the corresponding marginal matrix, equation (5).

Note that in 3-D a non-empty symbol in one language may align to the deletion symbol "." in two languages, e.g., (.:.:d) in the 3-D example above. This means that the alignment (.:.) can now have non-zero count and marginal probability, as any other 1-1 alignment.[5]

Re-alignment: the re-alignment phase for the *complete* examples in 3-D is analogous to the re-alignment in 2-D, equation (2). The cell in the re-alignment matrix $V(\sigma_i, \tau_j, \xi_k)$—the cumulative cost of the cheapest path leading to the cell (i, j, k)—is calculated via dynamic programming, from the symbol-alignment costs $L(\sigma : \tau : \xi)$:

$$V(\sigma_i, \tau_j, \xi_k) = \min \begin{cases} V(\sigma_{i-1}, \tau_j, \xi_k) & +L(\sigma_i : . : .) \\ V(\sigma_i, \tau_{j-1}, \xi_k) & +L(. : \tau_j : .) \\ V(\sigma_i, \tau_j, \xi_{k-1}) & +L(. : . : \xi_k) \\ V(\sigma_{i-1}, \tau_{j-1}, \xi_k) & +L(\sigma_i : \tau_j : .) \\ V(\sigma_i, \tau_{j-1}, \xi_{k-1}) & +L(. : \tau_j : \xi_k) \\ V(\sigma_{i-1}, \tau_j, \xi_{k-1}) & +L(\sigma_i : . : \xi_k) \\ V(\sigma_{i-1}, \tau_{j-1}, \xi_{k-1}) & +L(\sigma_i : \tau_j : \xi_k) \end{cases}$$

[5]NB: this count is always zero in 2-D alignments, and remains impossible when aligning incomplete examples in 3-D.

5 Results

Evaluation of the results of the alignment algorithms is not a simple matter. One way to evaluate thoroughly would require a *gold-standard* aligned corpus; the algorithms produce alignments, which should be compared to the alignments that we would expect to find. We currently have linguists working on a gold-standard alignment for the Uralic data. Given a gold-standard alignment, we can measure performance quantitatively, e.g., in terms of accuracy.

Alignment: We can still perform qualitative evaluation, by checking how many correct sound correspondences the algorithm finds, by inspecting the final alignment of the corpus and the alignment matrix. Sample matrices for 2-D alignments of Finnish-Estonian and Finnish-Mordva (Erzä di-

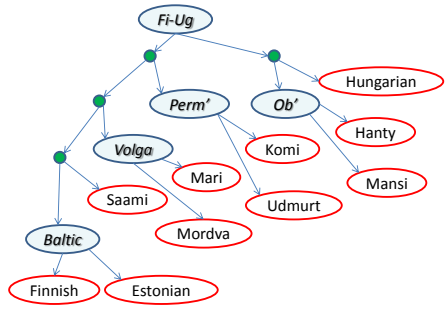

Figure 4: The Finno-Ugric sub-family of Uralic.

	est	*fin*	*khn*	*kom*	*man*	*mar*	*mrd*	*saa*	*udm*	*ugr*
est		**.372**	.702	.704	.716	.703	.665	.588	.733	.778
fin	.372		.731	.695	.754	.695	.635	.589	.699	.777
khn	.702	.719		.672	**.633**	.701	.718	.668	.712	.761
kom	.698	.703	.659		.675	.656	.678	.700	**.417**	.704
man	.702	.711	.633	.649		.676	.718	.779	.688	.752
mar	.715	.694	.731	.671	.746		**.648**	.671	.674	.738
mrd	.664	.624	.658	.678	.713	.648		.646	.709	.722
saa	.643	.589	.733	.706	.733	.621	.660		.686	.760
udm	.684	.712	.697	.417	.644	.694	.623	.677		.759
ugr	.780	.778	.761	.714	.755	.721	.743	.766	.741	

Table 1: Pairwise normalized compression costs for Finno-Ugric sub-family of Uralic, in StarLing data.

alect) are in figures 2 and 3. The size of each ball in the grid is proportional to the number of alignments in the corpus of the corresponding symbols.

Finnish and Estonian are the nearest languages in StarLing, and we observe that the alignment shows a close correspondence—the algorithm finds the *diagonal*, i.e., most sounds correspond to *"themselves"*. It must be noted that the algorithm has no *a priori* knowledge about the nature of the symbols, e.g., that Finnish *a* has any relation to Estonian *a*. The languages could be written, e.g., with different alphabets—as they are in general (we use transcribed data). This is evident in the Finnish-Estonian correspondence *y*∼*ü*, which is the same sound written using different symbols. The fact that the model finds a large number of "self" correspondences is due to the algorithm.

The model finds many Finnish-Estonian correspondences—according to rules we find in handbooks, e.g., (Lytkin, 1973; Sinor, 1997). For example, *ä*∼*a* or *ä*∼*ä* about evenly: this reflects the rule that original front vowels (as *ä*) become back in non-first syllables in Estonian. Plosives *t, k* become voiced *d, g* in certain contexts in non-initial positions. Word-final vowels *a, i, ä* are often deleted. These can be observed directly in the alignment matrix, and in the aligned corpus.

In the Finnish-Mordva alignment, the diagonal is not as pronounced, since the languages are further apart and sound correspondences more complex. Many more sounds are deleted, there is more entropy than in Finnish-Estonian; for example, many Finnish vowels map correctly to Erzä *e*, especially the front and high vowels; the back vowels do so much less often. Finnish *h* is mapped correctly to *č* or *š*. There is a (correct) preference to align *o* to *u*, and vice versa.

Compression: We can evaluate the quality of the alignment indirectly, through distances between languages. We align all languages in StarLing pairwise, using the two-part code model. We can then measure the *Normalized Compression Distance* (Cilibrasi and Vitanyi, 2005):

$$\delta(\mathbf{a},\mathbf{b}) = \frac{C(\mathbf{a},\mathbf{b}) - \min(C(\mathbf{a},\mathbf{a}), C(\mathbf{b},\mathbf{b}))}{\max(C(\mathbf{a},\mathbf{a}), C(\mathbf{b},\mathbf{b}))}$$

where $0 < \delta < 1$, and $C(\mathbf{a},\mathbf{b})$ is the compression cost—i.e., the cost of the complete aligned data for languages A and B.[6] The pairwise compression distances are shown in Table 1. Even with the simple 1x1 baseline model we see emerging patterns that mirror relationships within the Uralic family tree, shown in Fig. 4, e.g., one adapted from (Anttila, 1989). For example, scanning the row corresponding to Finnish, the compression distances *grow* as: Estonian .372, Saami .589, Mordva .635, Mari .695, Komi .695, Udmurt .699, Hanty .731, Mansi .754, and Hungarian .777, as the corresponding distance within the family tree also grows. The same holds true for Estonian.

In bold figures are sister languages, identified as being closest within their rows, (top to bottom): the Baltic, Ob', Permic, and Volgaic sub-branches.

Although the distances are not perfect (for some languages, the estimates are not 100% accurate) this confirms that the model is able to compress better—i.e., find *more regularity*—between languages that are are more closely related.

[6] $C(\mathbf{a},\mathbf{a})$ is a monolingual "alignment" of a language with itself—which is very primitive, since the 1x1 model is then able to model only the symbol frequencies.

6 Current Work and Conclusions

We have presented several models of increasing complexity for alignment of etymological data-sets. The baseline 1x1 model is improved upon by introducing a two-part coding scheme and simulated annealing—this helps reduce the cost and improves the alignment. Introducing 2x2 alignment helps to reduce the cost further, but produces many spurious symbol pairs, because certain combinations of sounds appear frequently within a single language. We conclude that the proper way to handle this is by modeling context explicitly, as described above. The powerful extension of the baseline to multiple languages performs well in terms of costs and resulting alignments—these will be tested against a gold-standard in future work. An interesting consequence of the MDL-based alignment procedure, is the ability to use the alignment costs as a measure of language relation, as shown in Table 1.[7]

Although the simulated annealing heuristic already yields useful results, the algorithm still tends to end up in different final alignment states—even with a slow cooling schedule—which differ in quality in terms of the cost function, eq. 7.

We are currently extending the alignment model in two ways: by modeling context—assigning different probabilities to the same event in different environments, and by using the phonetic feature representation of the alphabet symbols.

The presented methods are not intended to replace traditional methods for etymological analysis. We are addressing only a narrow slice of the problem of etymological analysis. However, we believe these models provide an initial basis for building more interesting and complex models in the future. In particular, we can use them to approach the question of comparison of "competing" etymological data-sets or theories. The cost of an optimal alignment obtained over a given data set gives an indication of the internal regularity within the set, which can be used as an indication of consistency and quality.

We have not begun to address many important questions in etymology, including borrowing and semantics, etc. We initially focus on phonological phenomena only. Earlier work, (Kondrak, 2004) has shown that even semantics can begin to be approached in a rigorous way by computational means. Borrowing will require building models that can span across language families, which will require more mature models in the future.

[7] To save space, we focus on the Finno-Ugric sub-family of Uralic, and leave out the Samoyedic branch.

Acknowledgements

Research supported by the Uralink Project of the Academy of Finland, Grant 129185. We thank Arto Vihavainen and Suvi Hiltunen for their contribution to the implementation and testing of the algorithms. We are grateful to the anonymous reviewers for their comments and suggestions.

References

R. Anttila. 1989. *Historical and comparative linguistics*. John Benjamins.

A. Bouchard-Côté, P. Liang, T.Griffiths, and D. Klein. 2007. A probabilistic approach to diachronic phonology. In *Proc. EMNLP-CoNLL*, Prague.

R. Cilibrasi and P.M.B. Vitanyi. 2005. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4).

P. Grünwald. 2007. *The Minimum Description Length Principle*. MIT Press.

E. Itkonen and U.-M. Kulonen. 2000. *Suomen Sanojen Alkuperä (The Origin of Finnish Words)*. Suomalaisen Kirjallisuuden Seura, Helsinki, Finland.

G. Kondrak. 2002. Determining recurrent sound correspondences by inducing translation models. In *Proceedings of COLING 2002*, Taipei.

G. Kondrak. 2003. Identifying complex sound correspondences in bilingual wordlists. In *A. Gelbukh (Ed.) CICLing*, Mexico. Springer LNCS, No. 2588.

G. Kondrak. 2004. Combining evidence in cognate identification. In *Proceedings of Canadian-AI 2004*, London, ON. Springer-Verlag LNCS, No. 3060.

P. Kontkanen, P. Myllymäki, and H. Tirri. 1996. Constructing Bayesian finite mixture models by the EM algorithm. Technical Report NC-TR-97-003, ESPRIT Working Group on NeuroCOLT.

V. I. Lytkin. 1973. *Voprosy Finno-Ugorskogo Jazykoznanija (Issues in Finno-Ugric Linguistics)*, volume 1–3. Nauka, Moscow.

K. Rédei. 1988–1991. *Uralisches etymologisches Wörterbuch*. Harrassowitz, Wiesbaden.

Denis Sinor, editor. 1997. *The Uralic Languages: Description, History and Foreign Influences (Handbook of Uralic Studies)*. Brill Academic Publishers.

S. A. Starostin. 2005. Tower of babel: Etymological databases. http://newstar.rinet.ru/.

ISSN 1736-6305 Vol. 11
http://hdl.handle.net/10062/16955

Publication IV

Hannes Wettig, Suvi Hiltunen and Roman Yangarber

MDL-based Models for Aligning Etymological Data

Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP-2011)
Hissar, Bulgaria

MDL-based Models for Alignment of Etymological Data

Hannes Wettig, Suvi Hiltunen, Roman Yangarber
Department of Computer Science
University of Helsinki, Finland
First.Last@cs.helsinki.fi

Abstract

We introduce several models for alignment of etymological data, that is, for finding the best alignment, given a set of etymological data, at the sound or symbol level. This is intended to obtain a means of measuring the quality of the etymological data sets, in terms of their internal consistency. One of our main goals is to devise automatic methods for aligning the data that are as objective as possible, the models make no a priori assumptions—e.g., no preference for vowel-vowel or consonant-consonant alignments. We present a baseline model and several successive improvements, using data from the Uralic language family.

1 Introduction

We present work on induction of alignment rules for etymological data, in a project that studies genetic relationships among the Uralic language family. This is a continuation of previous work, reported in (Wettig and Yangarber, 2011), where the methods were introduced. In this paper, we extend the models reported earlier and give a more comprehensive evaluation of results. In addition to the attempt to induce alignment rules, we aim to derive measures of quality of data sets in terms of their internal consistency. More consistent dataset should receive a higher score in the evaluations. Currently our goal is to analyze *given*, existing etymological datasets, rather than to construct cognate sets from raw linguistic data. The question to be answered is whether a complete description of the correspondence rules can be discovered automatically. Can they be found directly from raw etymological data—sets of cognate words from languages within the language family? Are the alignment rules are "inherently encoded" in a dataset (the *corpus*) itself? We aim to develop methods that are as objective as possible, that rely only on the data, rather than on any prior assumptions about the data, the possible rules and alignments.

Computational etymology encompasses several problem areas, including: discovery of sets of genetically related words—*cognates*; determination of genetic relations among groups of languages, from raw or organized linguistic data; discovering *regular sound correspondences* across languages in a given language family; and reconstruction, either diachronic—i.e., reconstruction of proto-forms for a hypothetical parent language, from which the word-forms found in the daughter languages derive, or synchronic—i.e., of word forms that are missing from existing languages.

Several approaches to etymological alignment have emerged over the last decade. The problem of discovering cognates is addressed, e.g., in, e.g., (Bouchard-Côté et al., 2007; Kondrak, 2004; Kessler, 2001). In our work, we do not attempt to find cognate sets, but begin with given sets of etymological data for a language family, possibly different or even conflicting. We use the principle of *recurrent sound correspondence*, as in much of the literature, including the mentioned work, (Kondrak, 2002; Kondrak, 2003) and others. Modeling relationships within the language family arises in the process of evaluation of our alignment models. Phylogenetic reconstruction is studied extensively by, e.g.,(Nakhleh et al., 2005; Ringe et al., 2002; Barbancon et al., 2009); these work differ from ours in that they operate on pre-compiled sets of "characters", capturing divergent features of entire languages within the family, whereas we operate at the level of words or cognate sets. Other related work is further mentioned in the body of the paper.

We describe our datasets in the next section, present a statement of the etymology alignment problem in Section 3, cover our models in detail in Sections 4– 6, and discuss results and next steps in Section 7.

2 Data

We use two digital Uralic etymological resources, *SSA—Suomen Sanojen Alkuperä*, "The Origin of Finnish Words", (Itkonen and Kulonen, 2000), and the StarLing database, (Starostin, 2005). StarLing, originally based on (Rédei, 1988 1991), differs from SSA in several respects. StarLing has about 2000 Uralic cognate sets, compared with over 5000 in SSA, and does

Proceedings of Recent Advances in Natural Language Processing, pages 111–117,
Hissar, Bulgaria, 12-14 September 2011.

not explicitly indicate dubious etymologies. However, Uralic data in StarLing is more evenly distributed, because it is not Finnish-centric like SSA is—cognate sets in StarLing are not required to contain a member from Finnish. The Uralic language family has not been studied by computational means previously.

3 Aligning Pairs of Words

We begin with pairwise alignment: aligning a set of pairs of words from two related languages in our data set. The task of alignment means, for each word pair, finding which symbols correspond. We expect that some symbols will align with themselves, while others have undergone changes over the time when the two related languages have been evolving separately. The simplest form of such alignment at the symbol level is a pair $(\sigma : \tau) \in \Sigma \times T$, a single symbol σ from the *source alphabet* Σ with a symbol τ from the *target alphabet* T. We denote the sizes of the alphabets by $|\Sigma|$ and $|T|$, respectively.[1]

Clearly, with this type of 1x1 alignment alone we cannot align a source word $\boldsymbol{\sigma}$ of length $|\boldsymbol{\sigma}|$ with a target word $\boldsymbol{\tau}$ of length $|\boldsymbol{\tau}| \neq |\boldsymbol{\sigma}|$.[2] To model also *insertions* and *deletions*, we augment both alphabets with the empty symbol, denoted by a dot, and use $\Sigma_.$ and $T_.$ as augmented alphabets. We can then align word pairs such as *ien—ige*, meaning "gum" in Finnish and Established, for example, as:

$$\begin{array}{ccc} i & e & n \\ | & | & | \\ i & g & e \end{array} \qquad \begin{array}{cccc} i & . & e & n \\ | & | & | & | \\ i & g & e & . \end{array}$$

etc. The (historically correct) alignment on the right consists, e.g., of symbol pairs: (i:i), (.:g), (e:e), (n:.).

4 The Baseline Model

We wish to encode these aligned pairs as compactly as possible, following the Minimum Description Length Principle (MDL), see e.g. (Grünwald, 2007; Rissanen, 1978). Given a data corpus $D = (\boldsymbol{\sigma}_1, \boldsymbol{\tau}_1), \ldots, (\boldsymbol{\sigma}_N, \boldsymbol{\tau}_N)$ of N word pairs, we first choose an alignment of each word pair $(\boldsymbol{\sigma}_i, \boldsymbol{\tau}_i)$, which we then use to "transmit" the data, by simply listing the sequence of the atomic pairwise symbol alignments.[3] In order for the code to be uniquely decodable, we also need to encode the word boundaries. This can be done by transmitting a special symbol # that we use only at the end of a word.

Thus, we transmit objects, or *events*, e, in the event space E—which is in this case:

$$E = \Sigma_. \times T_. \cup \{(\# : \#)\}$$

We do this by means of Bayesian marginal likelihood, or *prequential* coding, see e.g., (Kontkanen et al., 1996), giving the total code length as:

$$\begin{aligned} &L_{base}(D) = \qquad (1) \\ &- \sum_{e \in E} \log \Gamma\big(c(e) + \alpha(e)\big) + \sum_{e \in E} \log \Gamma\big(\alpha(e)\big) \\ &+ \log \Gamma \left[\sum_{e \in E} \big(c(e) + \alpha(e)\big) \right] - \log \Gamma \left[\sum_{e \in E} \alpha(e) \right] \end{aligned}$$

The *count* $c(e)$ is the number of times event e occurs in a complete alignment of the corpus; in particular, $c(\# : \#) = N$ occurs as many times as there are word pairs. The alignment counts are maintained in a corpus-global *count matrix* M, where $M(i, j) = c(i : j)$. The $\alpha(e)$ are the (Dirichlet) priors on the events. In the baseline algorithm, we set $\alpha(e) = 1$ for all e, the so-called uniform prior, which does not favor any distribution over E, *a priori*. Note that this choice nulls the second summation in equation 1.

Our baseline algorithm is simple: we first randomly align the entire corpus, then re-align one word pair at a time, greedily minimizing the total cost in Eq. 1, using dynamic programming.

In the matrix in Fig. 1, each cell corresponds to a partial alignment: reaching cell (i, j) means having read off i symbols of the source and j symbols of the target word. We iterate this process, *re-aligning* the word pairs, i.e., for the given word pair, we subtract the contribution of its current alignment from the global count matrix, then re-align the word pair, then add the newly aligned events back to the global count matrix. Re-alignment continues until convergence.

Re-alignment Step: align source word $\boldsymbol{\sigma}$ consisting of symbols $\boldsymbol{\sigma} = [\sigma_1 ... \sigma_n] \in \Sigma^*$ with target word $\boldsymbol{\tau} = [\tau_1 ... \tau_m]$. We use dynamic programming to fill in the matrix, e.g., top-to-bottom, left-to-right:[4]

Alignments of $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ correspond in a 1-1 fashion to paths through the matrix, starting with cost equal to 0 in top-left cell and terminating in bottom-right cell, moving only downward or rightward.

Each cell stores the cost of the *most probable* path so far: the most probable way to have scanned $\boldsymbol{\sigma}$ up to symbol σ_i and $\boldsymbol{\tau}$ up to τ_j, marked X in the Figure:

$$V(\sigma_i, \tau_j) = \min \begin{cases} V(\sigma_i, \tau_{j-1}) & +L(. : \tau_j) \\ V(\sigma_{i-1}, \tau_j) & +L(\sigma_i : .) \\ V(\sigma_{i-1}, \tau_{j-1}) & +L(\sigma_i : \tau_j) \end{cases} \qquad (2)$$

Each term $V(.,.)$ has been computed earlier by the dynamic programming; the term $L(.)$—the cost of align-

[1] We refer to "*source*" and "*target*" language for convenience only—our models are symmetric, as will become apparent.

[2] We use boldface to denote words, as vectors of symbols.

[3] By *atomic* we mean that the symbols are not analyzed—in terms of their phonetic features—and treated by the baseline algorithm as atoms. In particular, the model has no notion of identity of symbols *across* the languages!

[4] NB: in Fig. 1, the left column and the top row store the costs for symbol deletions *at the beginning* of the source and the target word, respectively.

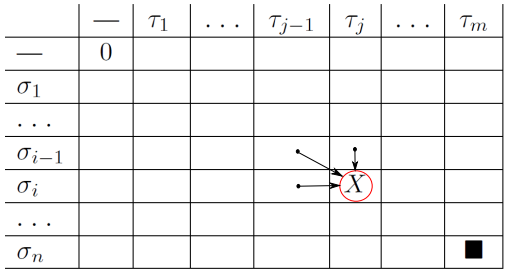

Figure 1: Re-alignment matrix: computes Dynamic Programming search for the most probable alignment.

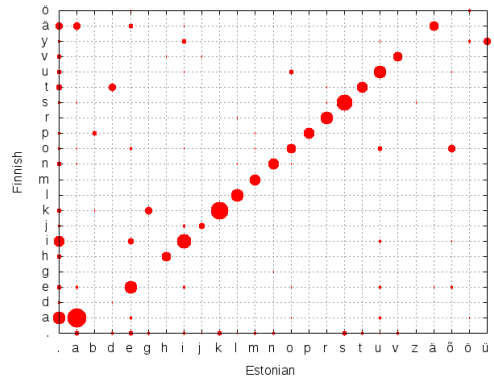

Figure 2: Global count matrix, using two-part model

ing the two symbols—is a parameter of the model, computed in equation (3).

The parameters $L(e)$, or $P(e)$, for every observed event e, are computed from the *change* in the total code-length—the change that corresponds to the cost of adjoining the new event e to the set of previously observed events E:

$$L(e) \quad = \quad \Delta_e L = L\big(E \cup \{e\}\big) - L(E)$$

$$P(e) \quad = \quad 2^{-\Delta_e L} = \frac{2^{-L\left(E \cup \{e\}\right)}}{2^{-L(E)}} \tag{3}$$

Combining eqs. 1 and 3 gives the probability:

$$P(e) = \frac{c(e)+1}{\sum_{e'} c(e') + |E|} \tag{4}$$

In particular, the cost of the most probable *complete* alignment of the two words will be stored in the bottom-right cell, $V(\sigma_n, \tau_m)$, marked ■. An example alignment count matrix is shown in Fig. 2.

4.1 The Two-Part Code

The baseline model revealed two problems. First, it seems to get stuck in local optima, and second, it produces many events with very low counts (occurring only once or twice).

To address the first problem we use simulated annealing with a sufficiently slow cooling schedule. This yields a reduction in the cost, and a better—more sparse—alignment count matrix.

The second problem is more substantial. Starting from a common ancestor language, the number of changes that occurred in either language should be small. We expect *sparse* data—that only a small proportion of all *possible* events in E will actually ever occur.

We incorporate this notion into the model by means of a two-part code. First we encode which events have occurred/have been observed: we send **a.** the number of events with non-zero counts—this costs $\log(|E|+1)$ bits, and **b.** specifically which subset $E^+ \subset E$ of the events have non-zero counts—this costs $\log \binom{|E|}{|E^+|}$ bits. This first part of the code is called the *codebook*. Given the codebook, we transmit the complete data, E^+, using Bayesian marginal likelihood. The code length becomes:

$$\begin{aligned} L_{tpc}(D) = &\log(|E|+1) + \log \binom{|E|}{|E^+|} \\ &- \sum_{e \in E^+} \log \Gamma\big(c(e)+1\big) \\ &+ \log \Gamma\left(\sum_{e \in E^+} \big(c(e)+1\big) \right) - \log \Gamma(|E^+|) \end{aligned} \tag{5}$$

where E^+ denotes the set of events with non-zero counts, and we have set all $\alpha(e)$'s to one. Optimizing the above function with simulated annealing yields much better alignments.

4.2 Aligning Multiple Symbols

Multiple symbols are aligned in (Bouchard-Côté et al., 2007; Kondrak, 2003). For example, Estonian and Finnish have frequent geminated consonants, which correspond to single symbols/sounds in other languages; diphthongs may align with single vowels; etc. We extend the baseline model to a 2x2 model, to allow correspondences of up to two symbols on both the source and the target side. The set of admissible *kinds* of events is then extended to include:

$$K = \left\{ \begin{array}{lll} (\# : \#), & (\sigma : .), & (\sigma\sigma' : .), \\ (. : \tau), & (\sigma : \tau), & (\sigma\sigma' : t), \\ (. : \tau\tau'), & (\sigma : \tau\tau'), & (\sigma\sigma' : \tau\tau') \end{array} \right\} \tag{6}$$

We expect correspondences of the different types to behave differently, so we encode the occurrences of different event kinds separately in the codebook:

$$L_{mult} \quad = \quad L(CB) + L(Data|CB) \tag{7}$$

$$L(CB) \quad = \quad \sum_{k \in K} \left[\log(N_k+1) + \log \binom{N_k}{M_k} \right] \tag{8}$$

$$L(D|CB) = -\sum_{e \in E} \log \Gamma\big(c(e)+1\big) \qquad (9)$$
$$+ \log \Gamma \left[\sum_{e \in E} \big(c(e)+1\big) \right] - \log \Gamma(|E|)$$

where N_k is the number of possible events of kind k and M_k the corresponding number of such events actually observed in the alignment; $\sum_k M_k \equiv |E|$.

5 Three-Dimensional Alignment

The baseline models align languages pairwise. The alignment models allow us to learn 1-1 patterns of correspondence in the language family. This model is easily extended to *any* number of languages. The model in (Bouchard-Côté et al., 2007) also aligns more than two languages at a time. We extend the 2-D model to three dimensions as follows. We seek an alignment where symbols correspond to each other in a 1-1 fashion, as in the 2-D baseline. A three-dimensional alignment is a triplet of symbols $(\sigma : \tau : \xi) \in \Sigma_{\cdot} \times T_{\cdot} \times \Xi_{\cdot}$. For example, the words meaning "9" in Finnish, Estonian and Mordva, can be aligned simultaneously as:

y	.	*h*	*d*	*e*	*k*	*s*	*ä*	*n*
|	|	|	|	|	|	|	|	|
ü	.	*h*	.	*e*	*k*	*s*	*a*	.
|	|	|	|	|	|	|	|	|
v	*e*	*χ*	.	.	*k*	*s*	*a*	.

In 3-D alignment, the input data contains all examples where words *in at least two* languages are present[5]—i.e., a word may be missing from one of the languages, (which allows us to utilize more of the data). Thus we have two types of examples: *complete*—where all three words present (as "9" above), and *incomplete*—containing words in only two languages. For example, for (*haamu*:—:*čama*)—"ghost" in Finnish and Mordva—the cognate Estonian word is missing.

We next extend the 2-D count matrix and the 2-D re-alignment algorithm to three dimensions. The 3-D re-alignment matrix is directly analogous to the 2-D version. For the alignment counts in 3-D, we handle complete and incomplete examples separately.

Our "marginal" 3-D alignment model aligns three languages simultaneously, using three marginal 2-D matrices, each storing a pairwise 2-D alignment. The marginal matrices for three languages are denoted $M_{\Sigma T}$, $M_{\Sigma \Xi}$ and $M_{T \Xi}$. The algorithm optimizes the total cost of the complete data, which is defined as the *sum* of the three 2-D costs obtained from applying prequential coding to the marginal alignment matrices.

When computing the cost for event $e = (\sigma, \tau, \xi)$, we consider complete and incomplete examples separately. In "incomplete" examples, we use the counts from the corresponding marginal matrix directly. E.g., for event count $c(e)$, where $e = (\sigma, -, \xi)$, and "$-$" denotes the missing word, the event count is given by: $M_{\Sigma \Xi}(\sigma, \xi)$,

[5]This was true by definition in the baseline 2-D algorithm.

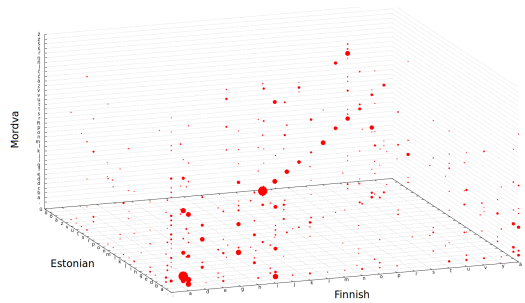

Figure 3: 3-dimensional alignment matrix.

and the cost of each alignment is computed as in the baseline model, directly in two dimensions.

In case when the data triplet is complete—fully observed—the alignment cost is computed as the *sum of the pairwise 2-D costs*, given by three marginal alignment count matrices:[6]

$$\begin{aligned} L(\sigma : \tau : \xi) &= L_{\Sigma T}(\sigma : \tau) \\ &+ L_{\Sigma \Xi}(\sigma : \xi) \\ &+ L_{T \Xi}(\tau : \xi) \end{aligned} \qquad (10)$$

The cost of each pairwise alignment is computed using prequential two-part coding, as in sec. 4.1. Note that when we register a complete alignment (σ, τ, ξ), we register it in *each* of the base matrices—we increment each of the marginal counts: $M_{\Sigma T}(\sigma, \tau)$, $M_{\Sigma \Xi}(\sigma, \xi)$, and $M_{T \Xi}(\tau, \xi)$.

To calculate the transition costs in the Viterbi algorithm, we also have two cases, complete and incomplete. For incomplete examples, we perform Viterbi in 2-D, using the costs directly from the corresponding marginal matrix, equation (5).

3-D re-alignment phase: for complete examples in 3-D, is a direct analogue of the 2-D re-alignment—in the (i, j) plane—in eq. (2), extended to the third dimension, k. The cell $V(\sigma_i, \tau_j, \xi_k)$—the cost of the most probable path leading to the cell (i, j, k)—is calculated by Dynamic Programming, using the symbol-alignment costs $L(\sigma : \tau : \xi)$. In addition to the three source cells as in eq. (2), in plane k, there are four additional source cells from the previous plane, $k - 1$.

Visualization: We wish to visualize the distribution of counts in the final 3-D alignment, except that now we must deal with *expected counts*, rather than observed counts, because some of the examples are incomplete. We can form a 3-D *visualization matrix* M^* as follows:

- Compute $|D|$, the total number of alignments in the complete data (including the end-of-word alignments)

[6]Note that this results in an incomplete code, since every symbol is coded twice, but that does not affect the learning.

- For each cell (i, j, k) in M^*, the weight in that cell is given by $P(i : j : k) \cdot |D|$, where $P(i : j : k)$ is the probability of the alignment.

- The matrix of expected counts will have no zero-weight cells, since there are no zero-probability events—except $(. : . : .)$. To suppress visualizing events with very low expected counts, we don't show cells with counts below a threshold, say, 0.5.

A distribution of the expected counts in 3-D alignment is shown in figure 3. The three languages are Finnish, Estonian and Mordva. The area of each point in this figure is proportional to the expected count of the corresponding 3-way alignment.

6 Nuisance Suffixes

The existing etymological datasets are not always perfectly suited to the alignment task as we have defined it here. For example, the SSA contains mostly complete word-forms from all the languages, as they would appear in a dictionary. As a consequence, this frequently includes morphological material that is not relevant from the point of view of etymology or alignment. To illustrate this (in the Indo-European family), consider aligning English *maid* and German *mädchen*—in German, the word-form without the suffix has disappeared. Many instances with such suffixes are found in the SSA; StarLing presents *stemmed* data to a larger extent, though assuring that every form in the dataset is perfectly stemmed is a very difficult task. From the point of view of computational alignment, such "nuisance" suffixes present a problem, by confusing the model.

We extend the model to handle, or discover, the nuisance suffixes automatically, as follows. Consider, in the realignment matrix in Fig. 1, the cells (i, j) (marked X,) (i, m), and (j, n). We always end by transitioning from cell marked ■, to the terminal cell, via the special end-of-word alignment event $(\# : \#)$, whose cost is computed from N, the number of word pairs in the data (this final transition is not shown in the figure).

While previously, we could only reach the terminal cell from cell ■ via event $(\# : \#)$, we now also permit a *hyper-jump* from any cell in the matrix to the terminal cell, which is equivalent to treating the remainder of source and/or target word as a nuisance suffix. Thus, hyper-jump from cell marked X means that we code the remaining symbols $[\sigma_{i+1}...\sigma_n]$ in $\boldsymbol{\sigma}$ and $[\tau_{j+1}...\tau_m]$ in $\boldsymbol{\tau}$ separately, *not* using the global count matrix.

That is, to align $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$, we first code the symbols up to X jointly, prequentially, using the global count matrix. After X, we code a special event $(- : -)$, meaning an aligned *morpheme boundary*, similar to $(\# : \#)$ which says we have aligned the *word* boundaries. Then we code the rest of $[\sigma_{i+1}...\sigma_n]$, and the rest of $[\tau_{j+1}...\tau_m]$, both followed by $\#$.

If we hyper-jump from cell (i, m), rather than from X, then we code the event $(- : \#)$—empty suffix on target side, and then code the rest of $[\sigma_{i+1}...\sigma_n]$ in $\boldsymbol{\sigma}$ and $\#$. Symmetrically for the hyper-jump from (j, m).

	Two-part model	*Suffix model*
Fin-Est	21748.29	21445.01
Fin-Ugr	10987.98	10794.87

Table 1: Nuisance suffix models.

The cost of each symbol in the suffix can be coded, for example, according to: a uniform language model: each source symbol costs $-\log 1/(|\Sigma| + 1)$; a unigram model: for each source symbol σ (including $\#$), compute its frequency $p(\sigma)$ from the raw source data, and let $cost(\sigma) = -\log p(\sigma)$; a bigram model; etc.

Table 1 compares the code length between the original 1x1 two-part code model and a nuisance suffix model (for two language pairs). The code length is always lower in the nuisance suffix model.

Although it finds instances of true nuisance suffixes, the model may be fooled by certain phenomena. For example, when aligning Finnish and Estonian, the model decides that final vowels in Finnish which have disappeared in Estonian are suffixes, whereas that is historically not the case. To avoid such misinterpretation, the suffix detection feature should be used in conjunction with other model variants, including alignment of more than a pair of languages.

7 Results

One way to evaluate the presented models thoroughly would require a *gold-standard* aligned corpus; the models produce alignments, which would be compared to expected alignments. Given a gold-standard, we could measure performance quantitatively, e.g., in terms of accuracy. However, no gold-standard alignment for the Uralic data currently exists, and building one is very costly and slow.

Alignment: We can perform a qualitative evaluation, by checking how many correct sound correspondences a model finds—by inspecting the final alignment of the corpus and the alignment matrix. A matrix for a 2-D, 1x1 two-part model alignment of Finnish-Estonian is shown in figure 2. The size of each ball is proportional to the number of alignments in the corpus of the corresponding symbols.

Finnish and Estonian are closely related, and the alignment shows a close correspondence—the model finds the "diagonal," i.e., most sounds correspond to "themselves." We must note that this model has no *a priori* knowledge about the nature of the symbols, e.g., that Finnish *a* is identical to or has any relation to Estonian *a*. The languages are coded separately, and they may have different alphabets—as they do in general (we use transcribed data).

Rules of correspondence: One of our main goals is to model complex rules of correspondence among languages. We can evaluate the models based on how

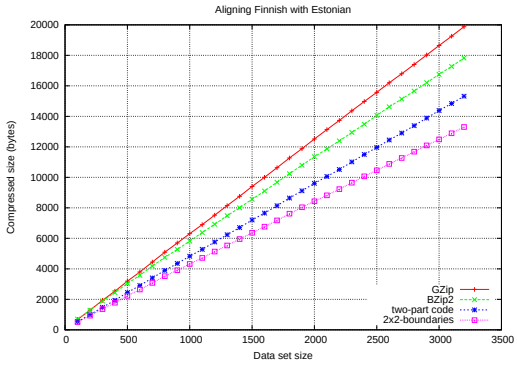

Figure 4: Comparison of compression power. Two-part code model refers to the 1x1 model that is described in section 4.1 and 2x2-boundaries model multiple symbol alignment model that is discussed in section 4.2.

well they discover rules, and how complex the rules are. In Fig. 2, the baseline model finds that Fin. *u* ∼ Est. *u*, but sometimes to *o*—this entropy is left unexplained by this model. However, the more complex 2x2 model identifies the cause exactly—by discovering that Finnish diphthongs *uo, yö, ie* correspond to Estonian long vowels *oo, öö, ee*, which covers (i.e., explains!) all instances of (*u*:*o*).

The plot shows many Finnish-Estonian correspondences, which can be found in handbooks, e.g., (Lytkin, 1973; Sinor, 1997). For example, *ä*∼*ä* vs. *ä*∼*a* about evenly—reflecting the rule that original front vowels (*ä*) became back (*a*) in non-first syllables in Estonian; word-final vowels *a*, *i*, *ä*, preserved in Finnish are often deleted in Estonian; etc. These can be observed directly in the alignment matrix, and in the aligned corpus.

Compression: In figure 4, we compare the models against standard compressors, gzip and bzip, tested on over 3200 Finnish-Estonian word pairs from SSA. The data given to our models is processed by the compressors, one word per line. Of course, our models know that they should align pairs of consecutive lines. This shows that learning about the "vertical" correspondences achieves much better compression rates—extract regularity from the data.

Language distance: We can use alignment to measure inter-language distances. We align all languages in StarLing pairwise, e.g., using a two-part 1x1 model. We can then measure the *Normalized Compression Distance* (Cilibrasi and Vitanyi, 2005):

$$NCD(\mathbf{a},\mathbf{b}) = \frac{C(\mathbf{a},\mathbf{b}) - \min(C(\mathbf{a},\mathbf{a}), C(\mathbf{b},\mathbf{b}))}{\max(C(\mathbf{a},\mathbf{a}), C(\mathbf{b},\mathbf{b}))}$$

where $0 < NCD < 1$, and $C(\mathbf{a},\mathbf{b})$ is the compression cost—i.e., the cost of the complete aligned data for languages $\mathbf{a}$ and $\mathbf{b}$. The pairwise compression distances

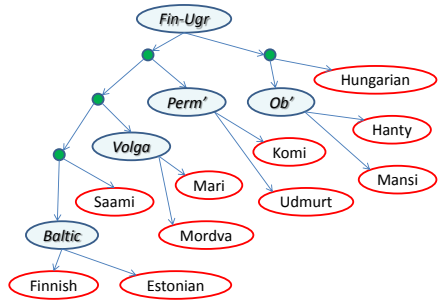

Figure 5: Finno-Ugric branch of the Uralic family

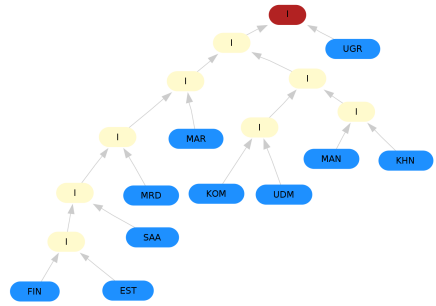

Figure 6: Finno-Ugric tree induced by NCD

are shown in table 2. We can then use these distances to draw phylogenetic trees, using hierarchical clustering methods. We used the UPGMA algorithm, (Murtagh, 1984), the resulting tree shown in Fig. 6. More sophisticated methods, such as the Fast Quartet method, CompLearn, (Cilibrasi and Vitanyi, 2011) may produce even more accurate trees. Even such a simple model as the 1x1 baseline shows emerging patterns that mirror the relationships in the Uralic family tree, shown in Fig. 5, adapted from (Anttila, 1989). For example, scanning the entries in the table corresponding to Finnish, the compression distances *grow* as the corresponding distance within the family tree grows. Sister languages (in bold) should be closest among all their relations. This confirms that the model is able to compress better—find *more regularity* in the data—between languages that are are more closely related.

8 Conclusions and Future Work

We have presented a baseline model for alignment, and several extensions. We have evaluated the models qualitatively, by examining the alignments and the rules of correspondence that they discover, and quantitatively by measuring compression cost and language distances. We trust that the methods presented here provide a good basis for further research.

We are developing methods that take *context*, or en-

	fin	*khn*	*kom*	*man*	*mar*	*mrd*	*saa*	*udm*	*ugr*
est	**.37**	.70	.70	.71	.70	.66	.58	.73	.77
fin		.73	.69	.75	.69	.63	.58	.69	.77
khn			.67	**.63**	.70	.71	.66	.71	.76
kom				.67	.65	.67	.70	**.41**	.70
man					.67	.71	.77	.68	.75
mar						**.64**	.67	.67	.73
mrd							.64	.70	.72
saa								.68	.76
udm									.75

Table 2: Pairwise normalized compression costs for Finno-Ugric sub-family of Uralic, in StarLing data.

vironment into account in modeling. The idea is to code sounds and environments as vectors of phonetic features and instead of aligning symbols, to align individual features of the symbols. The gain from introducing the context enables us to discover more complex rules of correspondence. We also plan to extend our models to diachronic reconstruction, which allows reconstruction of proto forms.

Acknowledgments

This research was supported by the Uralink Project of the Academy of Finland, grant 129185. We thank Teemu Roos for his suggestions, and Arto Vihavainen for his work on the implementation of the algorithms.

References

R. Anttila. 1989. *Historical and comparative linguistics*. John Benjamins.

F. Barbancon, T. Warnow, D. Ringe, S. Evans, and L. Nakhleh. 2009. An experimental study comparing linguistic phylogenetic reconstruction methods. In *Proc. Conf. on Languages and Genes*, UC Santa Barbara. Cambridge University Press.

A. Bouchard-Côté, P. Liang, T.Griffiths, and D. Klein. 2007. A probabilistic approach to diachronic phonology. In *Proc. EMNLP-CoNLL*, Prague.

R. Cilibrasi and P.M.B. Vitanyi. 2005. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4).

R.L. Cilibrasi and P.M.B. Vitanyi. 2011. A fast quartet tree heuristic for hierarchical clustering. *Pattern Recognition*, 44(3):662–677.

P. Grünwald. 2007. *The Minimum Description Length Principle*. MIT Press.

E. Itkonen and U.-M. Kulonen. 2000. *Suomen Sanojen Alkuperä (The Origin of Finnish Words)*. Suomalaisen Kirjallisuuden Seura, Helsinki, Finland.

B. Kessler. 2001. *The Significance of Word Lists: Statistical Tests for Investigating Historical Connections Between Languages*. The University of Chicago Press, Stanford, CA.

G. Kondrak. 2002. Determining recurrent sound correspondences by inducing translation models. In *Proceedings of COLING 2002*, Taipei.

G. Kondrak. 2003. Identifying complex sound correspondences in bilingual wordlists. In *A. Gelbukh (Ed.) CICLing*, Mexico. Springer LNCS, No. 2588.

G. Kondrak. 2004. Combining evidence in cognate identification. In *Proceedings of Canadian-AI 2004*, London, ON. Springer-Verlag LNCS, No. 3060.

P. Kontkanen, P. Myllymäki, and H. Tirri. 1996. Constructing Bayesian finite mixture models by the EM algorithm. Technical Report NC-TR-97-003, ESPRIT Working Group on NeuroCOLT.

V. I. Lytkin. 1973. *Voprosy Finno-Ugorskogo Jazykoznanija (Issues in Finno-Ugric Linguistics)*, volume 1–3. Nauka, Moscow.

F. Murtagh. 1984. Complexities of hierarchic clustering algorithms: the state of the art. *Computational Statistics Quarterly*, 1.

L. Nakhleh, D. Ringe, and T. Warnow. 2005. Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. *Language*, 81(2).

K. Rédei. 1988–1991. *Uralisches etymologisches Wörterbuch*. Harrassowitz, Wiesbaden.

D. Ringe, T. Warnow, and A. Taylor. 2002. Indo-european and computational cladistics. *Transact. Philological Society*, 100(1).

J. Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14(5).

Denis Sinor, editor. 1997. *The Uralic Languages: Description, History and Foreign Influences (Handbook of Uralic Studies)*. Brill Academic Publishers.

S. A. Starostin. 2005. Tower of babel: Etymological databases. http://newstar.rinet.ru/.

H. Wettig and R. Yangarber. 2011. Probabilistic models for alignment of etymological data. In *Proc. NODALIDA*, Riga, Latvia.

Publication V

Hannes Wettig, Kirill Reshetnikov and Roman Yangarber

Using context and phonetic features in models of etymological sound change

EACL Joint Workshop of LINGVIS & UNCLH 2012
Avignon, France

Using context and phonetic features in models of etymological sound change

Hannes Wettig[1], **Kirill Reshetnikov**[2] and **Roman Yangarber**[1]
[1]Department of Computer Science
University of Helsinki, Finland
First.Last@cs.helsinki.fi

[2]Institute of Linguistics
Academy of Sciences
Moscow, Russia

Abstract

This paper presents a novel method for aligning etymological data, which models context-sensitive rules governing sound change, and utilizes phonetic features of the sounds. The goal is, for a given corpus of cognate sets, to find the best alignment at the sound level. We introduce an imputation procedure to compare the goodness of the resulting models, as well as the goodness of the data sets. We present evaluations to demonstrate that the new model yields improvements in performance, compared to previously reported models.

1 Introduction

This paper introduces a context-sensitive model for alignment and analysis of etymological data. Given a raw collection of etymological data (the *corpus*)—we first aim to find the "best" alignment at the sound or symbol level. We take the corpus (or possibly several different corpora) for a language family as *given*; different data sets are typically conflicting, which creates the need to determine which is more correct. Etymological data sets are found in digital etymological databases, such as ones we use for the Uralic language family. A database is typically organized into *cognate sets*; all elements within a cognate set are posited (by the database creators) to be derived from a common origin, which is a word-form in the ancestral proto-language.

Etymology encompasses several problems, including: discovery of sets of cognates—genetically related words; determination of genetic relations among groups of languages, based on linguistic data; discovering *regular sound correspondences* across languages in a given language family; and reconstruction of forms in the proto-languages.

Computational methods can provide valuable tools for the etymological community. The methods can be judged by how well they model certain aspects of etymology, and by whether the automatic analysis produces results that match theories established by manual analysis.

In this work, we allow *all* the data—and only the data—to determine what rules underly it, rather than relying on external (and possibly biased) rules that try to explain the data. This approach will provide a means of measuring the quality of the etymological data sets in terms of their internal consistency—a dataset that is more consistent should receive a higher score. We seek methods that analyze the data automatically, in an unsupervised fashion, to determine whether a complete description of the correspondences can be discovered automatically, directly from raw etymological data—cognate sets within the language family. Another way to state the question is: what alignment rules are "inherently encoded" in the given corpus itself.

At present, our aim is to analyze given etymological datasets, rather than to construct new ones from scratch. Because our main goal is to develop methods that are as objective as possible, the models make no *a priori* assumptions or "universal" principles—e.g., no preference to align vowel with vowels, or a symbol with itself. The models are not aware of the *identity* of a symbol across languages, and do not try to preserve identity, of symbols, or even of features—rather they try to find maximally regular correspondences.

In Section 2 we describe the data used in our experiments, and review approaches to etymological alignment over the last decade. We formalize the problem of alignment in Section 3, give the

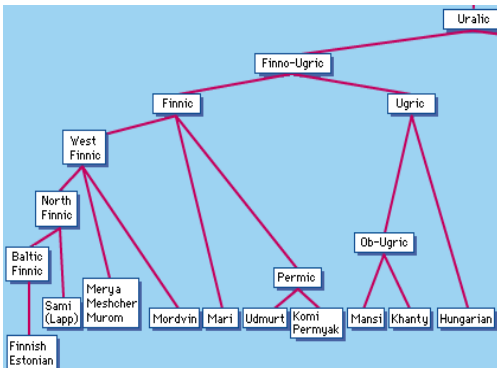

Figure 1: Finno-Ugric branch of Uralic language family (the data used in the experiments in this paper)

technical details of our models in Section 4. We present results and discussion in Sections 5 and 6.

2 Data and Related Work

We use two large Uralic etymological resources. The StarLing database of Uralic, (Starostin, 2005), based on (Rédei, 1988 1991), contains over 2500 cognate sets. *Suomen Sanojen Alkuperä* (SSA), "The Origin of Finnish Words", a Finnish etymological dictionary, (Itkonen and Kulonen, 2000), has over 5000 cognate sets, (about half of which are only in languages from the Balto-Finnic branch, closest to Finnish). Most importantly, for our models, SSA gives "dictionary" word-forms, which may contain extraneous morphological material, whereas StarLing data is mostly stemmed.

One traditional arrangement of the Uralic languages[1] is shown in Figure 1. We model etymological processes using these Uralic datasets.

The methods in (Kondrak, 2002) learn regular one-to-one sound correspondences between pairs of related languages in the data. The methods in (Kondrak, 2003; Wettig et al., 2011) find more complex (one-to-many) correspondences. These models operate on one language pair at a time; also, they do not model the *context* of the sound changes, while most etymological changes are conditioned on context. The MCMC-based model proposed in (Bouchard-Côté et al., 2007) explicitly aims to model the context of changes, and operates on more than a pair of languages.[2]

We should note that our models at present operate at the phonetic level only, they leave semantic judgements of the database creators unquestioned. While other work, e.g. (Kondrak, 2004), has attempted to approach semantics by computational means as well, our model uses the given cognate set as the fundamental unit. In our work, we do not attempt the problem of discovering cognates, addressed, e.g., in, (Bouchard-Côté et al., 2007; Kondrak, 2004; Kessler, 2001). We begin instead with a set of etymological data (or more than one set) for a language family *as given*. We focus on the principle of *recurrent sound correspondence*, as in much of the literature, including (Kondrak, 2002; Kondrak, 2003), and others.

As we develop our alignment models at the sound or symbol level, in the process of evaluation of these models, we also arrive at modeling the relationships among groups of languages within the family. Construction of phylogenies is studied extensively, e.g., by (Nakhleh et al., 2005; Ringe et al., 2002; Barbançon et al., 2009). This work differs from ours in that it operates on manually pre-selected sets of *characters*, which capture divergent features of languages within the family, whereas we operate on the raw, *complete* data.

There is extensive work on alignment in the machine-translation (MT) community, and it has been observed that methods from MT alignment may be projected onto alignment in etymology. The intuition is that translation sentences in MT correspond to cognate words in etymology, while words in MT correspond to sounds in etymology. The notion of *regularity* of sound change in etymology, which is what our models try to capture, is loosely similar to contextually conditioned correspondence of translation words across languages. For example, (Kondrak, 2002) employs MT alignment from (Melamed, 1997; Melamed, 2000); one might employ the IBM models for MT alignment, (Brown et al., 1993), or the HMM model, (Vogel et al., 1996). Of the MT-related models, (Bodrumlu et al., 2009) is similar to ours in that it is based on MDL (the Minimum Description Length Principle, introduced below).

[1] Adapted from Encyclopedia Britannica and (Anttila, 1989)

[2] Using this method, we found that the running time did not scale well for more than three languages.

3 Aligning Pairs of Words

We begin with pairwise alignment: aligning pairs of words, from two related languages in our corpus of cognates. For each word pair, the task of alignment means finding exactly which symbols correspond. Some symbols may align with "themselves" (i.e., with similar or identical sounds), while others may have undergone changes during the time when the two related languages have been evolving separately. The simplest form of such alignment at the symbol level is a pair $(\sigma : \tau) \in \Sigma \times T$, a single symbol σ from the *source alphabet* Σ with a symbol τ from the *target alphabet* T. We denote the sizes of the alphabets by $|\Sigma|$ and $|T|$.

To model *insertions* and *deletions*, we augment both alphabets with a special empty symbol—denoted by a dot—and write the augmented alphabets as $\Sigma_{\cdot}$ and $T_{\cdot}$. We can then align word pairs such as *vuosi—al* (meaning "year" in Finnish and Xanty) , for example as any of:

v	u	o	s	i		v	u	o	s	i	
|	|	|	|	|		|	|	|	|	|	etc...
a	l	.	.	.		.	a	.	l	.	

The alignment on the right then consists of the symbol pairs: (v:.), (u:a), (o:.), (s:l), (i:.).

4 Context Model with Phonetic Features

The context-aware alignment method we present here is built upon baseline models published previously, (Wettig et al., 2011), where we presented several models that do not use phonetic features or context. Similarly to the earlier ones, the current method is based on the *Minimum Description Length* (MDL) Principle, (Grünwald, 2007).

We begin with a raw set of (observed) data—the not-yet-aligned word pairs. We would like to find an alignment for the data—which we will call the *complete* data—complete with alignments, that make the most sense globally, in terms of embodying regular correspondences. We are after the regularity, and the more regularity we can find, the "better" our alignment will be (its goodness will be defined formally later). MDL tells us that the more regularity we can find in the data, the fewer bits we will need to encode it (or compress it). More regularity means lower entropy in the distribution that describes the data, and lower entropy allows us to construct a more economical code. That is, if we have no knowledge about any regularly of correspondence between symbols, the joint distribution over all possible pairs of symbols will be very flat (high entropy). If we know that certain symbol pairs align frequently, the joint distribution will have spikes, and lower entropy. In (Wettig et al., 2011) we showed how starting with a random alignment a good joint distribution can be learned using MDL. However the "rules" those baseline models were able to learn were very rudimentary, since they could not use any information in the context, and we know that many regular correspondences are conditioned by context.

We now introduce models that leverage information from the context to try to reduce the uncertainty in the distributions further, lowering the coding cost. To do that, we will code sounds in terms of their phonetic features: rather than coding the symbols (sounds) as atomic, we code them as vectors of phonetic features. Rather than aligning symbol pairs, we align the corresponding features of the symbols. While coding each feature, the model can make use of features of other sounds in its context (environment), through a special decision tree built for that feature.

4.1 Features

We will code each symbol, to be aligned in the complete data, as a feature vector. First we code the **Type** feature, with values: K (consonant), V (vowel), dot, and *word boundary*, which we denote as $\#$. Consonants and vowels have their own sets of features, with 2–8 values per feature:

Consonant articulation		
M	**Manner**	plosive, fricative, glide, ...
P	**Place**	labial, dental, ..., velar
X	**Voiced**	– , +
S	**Secondary**	– , affricate, aspirate, ...
Vowel articulation		
V	**Vertical**	high–low
H	**Horizontal**	front–back
R	**Rounding**	– , +
L	**Length**	1–5

4.2 Contexts

While coding any symbol, the model will be allowed to query a fixed, finite set of *candidate contexts*. A context is a triplet (L, P, F), where L is the level—either source or target,—and P is

one of the positions that the model may query—relative to the position currently being coded; for example, we may allow positions as in Fig. 2. F is one of the possible features found at that position. Therefore, we will have about 2 levels * 8 positions * 2–6 features $\approx$ 80 candidate contexts that can be queried by the model, as explained below.

I	itself,
–P	previous position
–S	previous non-dot symbol
–K	previous consonant
–V	previous vowel
+S	previous or self non-dot symbol
+K	previous or self consonant
+V	previous or self vowel

Figure 2: An example of a set of possible positions in the context—relative to the position currently being coded—that can be queried by the context model.

4.3 The Two-Part Code

We code the complete (i.e., aligned) data using a *two-part code*, following the MDL Principle. We first code which particular model instance we select from our *class* of models, and then code the data, given the defined model. Our model class is defined as: a set of decision trees (forest), with one tree to predict each feature on each level. The model instance will define the particular structures for each of the trees.

The forest consists of 18 decision trees, one for each feature on the source and the target level: the type feature, 4 vowel and 4 consonant features, times 2 levels. Each node in such tree will either be a leaf, or will be split by querying one of the candidate contexts defined above. The cost of coding the structure of the tree is one bit for every node—to encode whether this node was split (is an internal node) or is a leaf—plus $\approx \log 80$ times the number of internal nodes—to encode *which* particular context was chosen to split that node. We will explain how the best context to split on is chosen in Sec. 4.6.

Each feature and level define a tree, e.g., the "voiced" (**X**) feature of the source symbols corresponds to the source-**X** tree. A node N in this tree holds a distribution over the values of **X** of only those symbol instances in the complete data that have reached in N by following the context queries, starting from the root. The tree structure tells us precisely which path to follow—completely determined by the context. For example, when coding a symbol α based on another symbol found in the context of α—at some level (say, target), some position (say, –K), and one of its features (say, **M**)—the next edge down the tree is determined by that feature's value; and so on, down to a leaf. For an example of an actual decision tree learned by the model, see Fig. 5.

To compute the code length of the complete data, we only need to take into account the distributions at *the leaves*. We could choose from a variety of coding methods; the crucial point is that the chosen code will assign a particular number—the *cost*—to every possible alignment of the data. This code-length, or cost, will then serve as the *objective function*—i.e., it will be the value that the algorithm will try to optimize. Each reduction in cost will correspond directly to reduction in the entropy of the probability distribution of the symbols, which in turn corresponds to more certainty (i.e., regularity) in the correspondences among the symbols, and to improvement in the alignment. This is the link to our goal, and the reason for introducing code lengths—it gives us a single number that describes the quality of an alignment.

We use *Normalized Maximum Likelihood* (NML), (Rissanen, 1996) as our coding scheme. We choose NML because it has certain optimality properties. Using NML, we code the distribution at each leaf node separately, and summing the costs of all leaves gives the total cost of the aligned data—the value of our objective function.

Suppose n instances end up in a leaf node N, of the λ-level tree, for feature F having k values (e.g., consonants satisfying N's context constraints in the source-**X** tree, with $k = 2$ values: $-$ and $+$), and the values are distributed so that n_i instances have value i (with $i \in \{1, \ldots, k\}$). Then this requires an NML code-length of

$$\begin{aligned} L_{NML}(\lambda; F; N) &= -\log P_{NML}(\lambda; F; N) \\ &= -\log \frac{\prod_i \left(\frac{n_i}{n}\right)^{n_i}}{C(n,k)} \end{aligned} \quad (1)$$

Here $\prod_i \left(\frac{n_i}{n}\right)^{n_i}$ is the maximum likelihood of the multinomial data at node N, and the term

$$C(n,k) = \sum_{n'_1+\ldots+n'_k=n} \prod_i \left(\frac{n'_i}{n}\right)^{n'_i} \quad (2)$$

is a normalizing constant to make P_{NML} a probability distribution.

In the MDL literature, e.g., (Grünwald, 2007), the term $-\log C(n,k)$ is called the *stochastic complexity* or the *(minimax) regret* of the model, (in this case, the multinomial model). The NML distribution provides the unique solution to the minimax problem posed in (Shtarkov, 1987),

$$\min_{\hat{P}} \max_{\mathbf{x^n}} \log \frac{P(\mathbf{x^n}|\hat{\mathbf{\Theta}}(\mathbf{x^n}))}{\hat{P}(\mathbf{x^n})} \qquad (3)$$

where $\hat{\Theta}(\mathbf{x^n}) = \arg\max_{\mathbf{\Theta}} \mathbf{P}(\mathbf{x^n})$ are the *maximum likelihood parameters* for the data $\mathbf{x^n}$. Thus, P_{NML} minimizes the worst-case regret, i.e., the number of excess bits in the code as compared to the best model in the model class, with hind-sight. For details on the computation of this code length see (Kontkanen and Myllymäki, 2007).

Learning the model from the observed data now means aligning the word pairs and building the decision trees in such a way as to minimize the two-part code length: the sum of the model's code length—to encode the structure of the trees,—and the data's code length—to encode the aligned word pairs, using these trees.

4.4 Summary of the Algorithm

The full learning algorithm runs as follows:

We start with an initial *random* alignment for each pair of words in the corpus, i.e., for each word pair choose some random path through the matrix depicted in Figure 3.

From then on we alternate between two steps: **A.** re-build the decision trees for all features on source and target levels, and **B.** re-align all word pairs in the corpus. Both of these operations monotonically decrease the two-part cost function and thus compress the data.

We continue until we reach convergence.

4.5 Re-alignment Procedure

To align source word $\vec{\sigma}$ consisting of symbols $\vec{\sigma} = [\sigma_1 ... \sigma_n]$, $\vec{\sigma} \in \Sigma^*$ with target word $\vec{\tau} = [\tau_1 ... \tau_m]$ we use dynamic programming. The tree structures are considered fixed, as are the alignments of all word pairs, except the one currently being aligned—which is subtracted from the counts stored at the leaf nodes.

We now fill the matrix V, left-to-right, top-to-bottom. Every possible alignment of $\vec{\sigma}$ and $\vec{\tau}$ corresponds to exactly one path through this matrix: starting with cost equal to 0 in the top-left cell, moving only downward or rightward, and terminating in the bottom-right cell. In this Viterbi-like matrix, every cell corresponds to a partially completed alignment: reaching cell (i,j) means having read off i symbols of the source word and j symbols of the target. Each cell $V(i,j)$—marked X in the Figure—stores the cost of the *most probable* path so far: the most probable way to have scanned $\vec{\sigma}$ through symbol σ_i and $\vec{\tau}$ through τ_j:

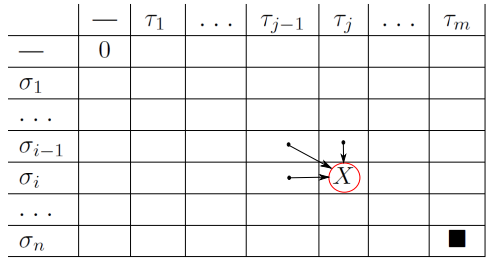

Figure 3: Dynamic programming matrix V, to search for the most probable alignment

$$V(i,j) = \min \begin{cases} V(i,j-1) & +L(.:\tau_j) \\ V(i-1,j) & +L(\sigma_i:.) \\ V(i-1,j-1) & +L(\sigma_i:\tau_j) \end{cases}$$

Each term $V(\cdot,\cdot)$ has been computed earlier by the dynamic programming; the term $L(\cdot)$—the cost of aligning the two symbols, inserting or deleting—is determined by the change in *data* code length it induces to add this event to the corresponding leaf in all the feature trees it concerns.

In particular, the cost of the most probable *complete* alignment of the two words will be stored in the bottom-right cell, $V(n,m)$, marked ■.

4.6 Building Decision Trees

Given a complete alignment of the data, we need to build a decision tree, for each feature on both levels, yielding the lowest two-part cost. The term "decision tree" is meant in a probabilistic sense here: instead of a single value, at each node we store a *distribution* of the corresponding feature values, over all instances that reach this node. The distribution at a leaf is then used to code an instance when it reaches the leaf in question. We code the features in some fixed, pre-set order, and source level before target level.

We now describe in detail the process of building the tree for feature **X**, for the source level, (we will need do the same for all other features, on both levels, as well). We build this tree as follows. First, we collect all instances of consonants on the source level, and gather the the counts for feature **X**; and build an initial count vector; suppose it is:

value of **X:**	+	–
	1001	1002

This vector is stored at the *root* of the tree; the cost of this node is computed using NML, eq. 1.

Next, we try to split this node, by finding such a context that if we query the values of the feature in that context, it will help us reduce the entropy in this count vector. We check in turn all possible candidate contexts, (L, P, F), and choose the best one. Each candidate refers to some symbol found on the source (σ) or the target (τ) level, at some relative position P, and to one of that symbol's features F. We will condition the split on the possible values of F. For each candidate, we try to split on its feature's values, and collect the resulting alignment counts.

Suppose one such candidate is (σ, –V, **H**), i.e., (source-level, previous vowel, Horizontal feature), and suppose that the **H**-feature has two values: *front/back*. The vector at the root node (recall, this tree is for the **X**-feature) would then split into two vectors, e.g.:

value of **X**:	+	–
X | **H**=*front*	1000	1
X | **H**=*back*	1	1001

This would likely be a very good split, since it reduces the entropy of the distribution in each row almost to zero. The criterion that guides the choice of the best candidate to use for splitting a node is the sum of the *code lengths* of the resulting split vectors, and the code length is proportional to the entropy.

We go through all candidates exhaustively, and greedily choose the one that yields the greatest reduction in entropy, and drop in cost. We proceed recursively down the tree, trying to split nodes, and stop when the total tree cost stops decreasing.

This completes the tree for feature **X** on level σ. We build trees for all features and levels similarly, from the current alignment of the complete data.

We augment the set of possible values at every node with two additional special branches: $\neq$, meaning the symbol at the queried position is of the wrong type and does not have the queried feature, and $\#$, meaning the query ran past the beginning of the word.

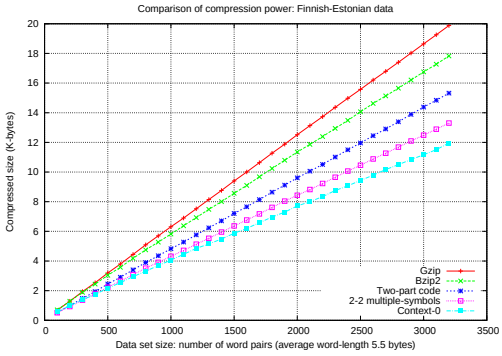

Figure 4: Comparison of compression power: Finnish-Estonian data from SSA, using the context model vs. the baseline models and standard compressors.

5 Evaluation and Results

One way to evaluate the presented models would require a *gold-standard* aligned corpus; the models produce alignments which could be compared to the gold-standard alignments, and we could measure performance quantitatively, e.g., in terms of accuracy. However, building a gold-standard aligned corpus for the Uralic data proved to be extremely difficult. In fact, it quickly becomes clear that this problem is at least as difficult as building a full reconstruction for all internal nodes in the family tree (and probably harder), since it requires full knowledge of all sound correspondences within the family. It is also compounded by the problem that the word-forms in the corpus may contain morphological material that is etymologically unrelated: some databases give "dictionary" forms, which contain extraneous affixes, and thereby obscure which parts of a given word form stand in etymological relationship with other members in the cognates set, and which do not. We therefore introduce other methods to evaluate the models.

Compression: In figure 4, we compare the context model, and use as baselines the standard data compressors, Gzip and Bzip, as well as the more basic models presented in (Wettig et al., 2011), (labeled "1x1 and "2x2"). We test the compression of up to 3200 Finnish-Estonian word pairs, from SSA. Gzip and Bzip compress data

	fin	*khn*	*kom*	*man*	*mar*	*mrd*	*saa*	*udm*	*ugr*
est	**0.26**	0.66	0.64	0.65	0.61	0.57	0.57	0.62	0.62
fin		0.63	0.64	0.65	0.59	0.56	0.50	0.62	0.63
khn			0.65	**0.58**	0.69	0.64	0.67	0.66	0.66
kom				0.63	0.68	0.66	0.70	**0.39**	0.66
man					0.68	0.65	0.72	0.62	0.62
mar						0.65	0.69	0.65	0.66
mrd							0.58	0.66	0.63
saa								0.67	0.70
udm									0.65

Table 1: Pairwise normalized edit distances for Finno-Ugric languages, on StarLing data (symmetrized by averaging over the two directions of imputation).

by finding regularities in it (i.e., frequent substrings). The comparison with Gzip is a "sanity check": we would like to confirm whether our models find more regularity in the data than would an off-the-shelf data compressor, that has no knowledge that the words in the data are etymologically related. Of course, our models know that they should align pairs of consecutive lines. This test shows that learning about the "vertical" correspondences achieves much better compression rates—allows the models to extract greater regularity from the data.

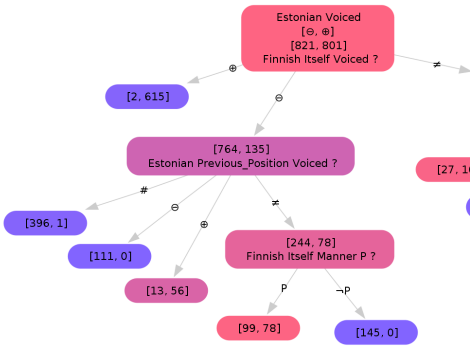

Figure 5: *Part of a tree, showing the rule for voicing of medial plosives in Estonian, conditioned on Finnish.*

Rules of correspondence: One our main goals is to model rules of correspondence among languages. We can evaluate the models based on how good they are at discovering rules. (Wettig et al., 2011) showed that aligning multiple symbols captures some of the context and thereby finds more complex rules than their 1-1 alignment model.

However, certain alignments, such as *t∼t/d*, *p∼p/b*, and *k∼k/g* between Finnish and Estonian, cannot be explained by the multiple-symbol model. This is due to the rule of *voicing of word-medial plosives* in Estonian. This rule could be expressed in terms of Two-level Morphology, (Koskenniemi, 1983) as: a voiceless plosive in Finnish, *may correspond* to voiced in Estonian, if not word-initial.[3] The context model finds this rule, shown in Fig. 5. This tree codes the *Target-level* (i.e., Estonian) *Voiced* consonant feature. In each node, the counts of corresponding feature values are shown in brackets. In the root node—prior to knowing anything about the environment—there is almost complete uncertainty (i.e., high entropy) about the value of *Voiced* feature of an Estonian consonant: 821 voiceless to 801 voiced in our data. Redder nodes indicate higher entropy, bluer nodes—lower entropy. The query in the root node tells us to check the context *Finnish Itself Voiced* for the most informative clue about whether the current Estonian consonant is voiced or not. Tracing the options down left to right from the root, we obtain the rules. The leftmost branch says, if the Finnish is voiced ($\oplus$), then the Estonian is almost certainly voiced as well—615 voiced to 2 voiceless in this case. If the Finnish is voiceless (Finnish Itself Voiced $= \ominus$), it says voicing *may occur*, but only in the red nodes—i.e., only if preceded by a voiced consonant on Estonian level (the branch marked by $\oplus$, 56 cases), or—if previous position is *not a consonant* (the $\neq$ branch indicates that the candidate's query does not apply: i.e., the sound found in that position is not a consonant)—it can be voiced only if the corresponding Finnish is a plosive (P, 78 cases). The blue nodes in this branch say that otherwise, the Estonian consonant almost certainly remains voiceless.

The context models discover numerous complex rules for different language pairs. For example, they learn a rule that initial Finnish *k* "changes" (corresponds) to *h* in Hungarian, if it is followed by a back vowel; the correspondence between Komi trills and Udmurt sibilants; etc.

Imputation: We introduce a novel test of the quality of the models, by using them to *impute* unseen data, as follows. For a given model, and a language pair (L_1, L_2)—e.g., (Finnish, Estonian)—hold out one word pair, and train the model on the remaining data. Then show the model the hidden Finnish word and let it guess

[3] In fact, phonetically, in modern spoken Estonian, the consonants that are written using the symbols *b,d,g* are not technically voiced, but that is a finer point, we use this rule for illustration of the principle.

the corresponding Estonian. Imputation can be done for all models with a simple dynamic programming algorithm, similar to the Viterbi-like search used during training. Formally, given the hidden Finnish string, the imputation procedure selects from all possible Estonian strings the most probable Estonian string, given the model. We then compute an edit distance between the imputed sting and the true withheld Estonian word (e.g., using the Levenshtein distance). We repeat this procedure for all word pairs in the (L_1, L_2) data set, sum the edit distances and normalize by the total size of the (true) L_2 data—this yields the Normalized Edit Distance $NED(L_2|L_1, M)$ between L_1 and L_2, under model M.

Imputation is a more intuitive measure of the model's quality than code length, with a clear practical interpretation. NED is also the ultimate test of the model's quality. If model M imputes better than M'—i.e., $NED(L_2|L_1, M) < NED(L_2|L_1, M')$—then it is difficult to argue that M could be in any sense "worse" than M'—it has learned more about the regularities between L_1 and L_2, and it knows more about L_2 given L_1. The context model, which has much lower cost than the baseline, almost always has lower NED. This also yields an important insight: it is an encouraging indication that optimizing the code length is a good approach—the algorithm does *not* optimize NED directly, and yet the cost correlates strongly with NED, which is a simple and intuitive measure of the model's quality.

6 Discussion

We have presented a novel feature-based context-aware MDL model, and a comparison of its performance against prior models for the task of alignment of etymological data. We have evaluated the models by examining the the rules of correspondence that they discovers, by comparing compression cost, imputation power and language distances induced by the imputation. The models take only the etymological data set as input, and require no further linguistic assumptions. In this regard, they is as objective as possible, given the data. The data set itself, of course, may be highly subjective and questionable.

The objectivity of models given the data now opens new possibilities for comparing entire data sets. For example, we can begin to compare the Finnish and Estonian datasets in SSA vs. StarLing, although the data sets have quite different characteristics, e.g., different size—3200 vs. 800 word pairs, respectively—and the comparison is done impartially, relying solely on the data provided. Another direct consequence of the presented methods is that they enable us to quantify uncertainty of entries in the corpus of etymological data. For example, for a given entry x in language L_1, we can compute exactly the probability that x would be imputed by any of the models, trained on all the remaining data from L_1 plus any other set of languages in the family. This can be applied equally to any entry, in particular to entries marked dubious by the database creators.

We can use this method to approach the question of comparison of "competing" etymological datasets. The cost of an optimal alignment obtained over a given data set serves as a measure of its internal consistency.

We are currently working to combine the context model with 3- and higher-dimensional models, and to extend these models to perform diachronic imputation, i.e., reconstruction of protoforms. We also intend to test the models on databases of other language families.

Acknowledgments

We are very grateful to the anonymous reviewers for their thoughtful and helpful comments. We thank Suvi Hiltunen for the implementation of the models, and Arto Vihavainen for implementing some of the earlier models. This research was supported by the Uralink Project, funded by the Academy of Finland and by the Russian Fund for the Humanities.

References

Raimo Anttila. 1989. *Historical and comparative linguistics*. John Benjamins.

François G. Barbançon, Tandy Warnow, Don Ringe, Steven N. Evans, and Luay Nakhleh. 2009. An experimental study comparing linguistic phylogenetic reconstruction methods. In *Proceedings of the Conference on Languages and Genes*, UC Santa Barbara. Cambridge University Press.

Tugba Bodrumlu, Kevin Knight, and Sujith Ravi. 2009. A new objective function for word alignment. In *Proc. NAACL Workshop on Integer Linear Programming for NLP*.

Alexandre Bouchard-Côté, Percy Liang, Thomas Griffiths, and Dan Klein. 2007. A probabilistic ap-

proach to diachronic phonology. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 887–896, Prague, June.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Peter Grünwald. 2007. *The Minimum Description Length Principle*. MIT Press.

Erkki Itkonen and Ulla-Maija Kulonen. 2000. *Suomen Sanojen Alkuperä (The Origin of Finnish Words)*. Suomalaisen Kirjallisuuden Seura, Helsinki, Finland.

Brett Kessler. 2001. *The Significance of Word Lists: Statistical Tests for Investigating Historical Connections Between Languages*. The University of Chicago Press, Stanford, CA.

Grzegorz Kondrak. 2002. Determining recurrent sound correspondences by inducing translation models. In *Proceedings of COLING 2002: 19th International Conference on Computational Linguistics*, pages 488–494, Taipei, August.

Grzegorz Kondrak. 2003. Identifying complex sound correspondences in bilingual wordlists. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing (CICLing-2003)*, pages 432–443, Mexico City, February. Springer-Verlag Lecture Notes in Computer Science, No. 2588.

Grzegorz Kondrak. 2004. Combining evidence in cognate identification. In *Proceedings of the Seventeenth Canadian Conference on Artificial Intelligence (Canadian AI 2004)*, pages 44–59, London, Ontario, May. Lecture Notes in Computer Science 3060, Springer-Verlag.

Petri Kontkanen and Petri Myllymäki. 2007. A linear-time algorithm for computing the multinomial stochastic complexity. *Information Processing Letters*, 103(6):227–233.

Kimmo Koskenniemi. 1983. *Two-level morphology: A general computational model for word-form recognition and production*. Ph.D. thesis, University of Helsinki, Finland.

I. Dan Melamed. 1997. Automatic discovery of non-compositional compounds in parallel data. In *The Second Conference on Empirical Methods in Natural Language Processing*, pages 97–108, Hissar, Bulgaria.

I. Dan Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

Luay Nakhleh, Don Ringe, and Tandy Warnow. 2005. Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. *Language (Journal of the Linguistic Society of America)*, 81(2):382–420.

Károly Rédei. 1988–1991. *Uralisches etymologisches Wörterbuch*. Harrassowitz, Wiesbaden.

Don Ringe, Tandy Warnow, and A. Taylor. 2002. Indo-European and computational cladistics. *Transactions of the Philological Society*, 100(1):59–129.

Jorma Rissanen. 1996. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, January.

Yuri M. Shtarkov. 1987. Universal sequential coding of single messages. *Problems of Information Transmission*, 23:3–17.

Sergei A. Starostin. 2005. Tower of babel: Etymological databases. http://newstar.rinet.ru/.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of 16th Conference on Computational Linguistics (COLING 96)*, Copenhagen, Denmark, August.

Hannes Wettig, Suvi Hiltunen, and Roman Yangarber. 2011. MDL-based Models for Alignment of Etymological Data. In *Proceedings of RANLP: the 8th Conference on Recent Advances in Natural Language Processing*, Hissar, Bulgaria.

Publication VI

Hannes Wettig, Javad Nouri, Kirill Reshetnikov and Roman Yangarber

**Information-Theoretic Methods
for Analysis and Inference in Etymology**

Pp. 52–55 in Proceedings of the Fifth Workshop on Information Theoretic Methods in Science and Engineering (WITMSE 2012)
Edited by Steven de Rooij, Wojciech Kotłowski, Jorma Rissanen, Petri Myllymäki, Teemu Roos and Kenji Yamanishi.

INFORMATION-THEORETIC METHODS FOR ANALYSIS AND INFERENCE IN ETYMOLOGY

Hannes Wettig[1], *Javad Nouri*[1], *Kirill Reshetnikov*[2] *and Roman Yangarber*[1]

[1]Department of Computer Science, University of Helsinki, Finland, First.Last@cs.helsinki.fi
[2]Academy of Sciences, Institute of Linguistics, Moscow, Russia.

ABSTRACT

We introduce a family of minimum description length models which explicitly utilizes phonetic features and captures long-range contextual rules that condition recurrent correspondences of sounds within a language family. We also provide an algorithm to learn a model from this family given a corpus of cognates, sets of genetically related words. Finally, we present an *imputation* procedure which allows us compare the quality of alignment models, as well as the goodness of the data sets. Our evaluations demonstrate that the new model yields improvements in performance, as compared to those previously reported in the literature.

1. INTRODUCTION

This paper introduces a family of context-aware models for alignment and analysis of etymological data on the level of phonetic features. We focus on discovering the rules of regular (or recurrent) phonetic correspondence across languages and determining genetic relations among a group of languages, based on linguistic data. In this work, we use the StarLing database of Uralic, [1], based on [2], restricted to the Finno-Ugric sub-family, consisting of 1898 *cognate sets*, as well as *Suomen Sanojen Alkuperä* (SSA), "The Origin of Finnish Words," a Finnish etymological dictionary, [3], which contains over 5000 cognate sets. Elements within a given cognate set are words posited by the database creators to be derived from a common origin, a word-form in the ancestral *proto-language*.

One traditional arrangement of the Uralic languages—adapted from Encyclopedia Britannica—is shown in Figure 1; alternative arrangements found in the literature include moving Mari into a separate branch, or grouping it with Mordva into a branch, called "Volgaic".

We aim to find the best *alignment* at the level of single sounds. The database itself only contains unaligned sets of corresponding words, with no notion of which sounds correspond, i.e., how the sounds align. We learn rules of phonetic correspondence allowing only the data to determine what rules underly it, using no externally supplied (and possibly biased) prior assumptions or "universal" principles—e.g., no preference to align vowel with vowels, a symbol with itself, etc. Therefore, all rules we find are *inherently encoded* in the corpus itself.

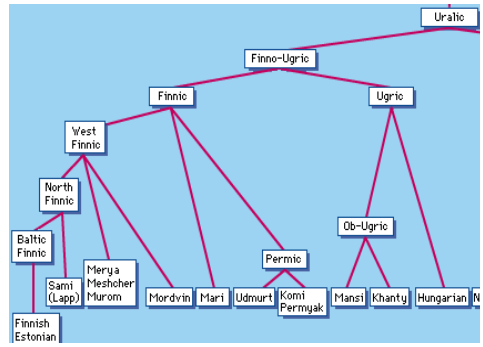

Figure 1. Finno-Ugric branch of Uralic language family

The criterion we use to choose a model (class) from the family we define is the code-length needed to communicate the complete (aligned) data. The learned minimum description length (MDL) models provide the desired alignments on the sound level, but also the underlying rules of correspondence, which enable us to *compress* the data. Apart from looking at the code-length, we also evaluate our models using an imputation (reconstruction of held-out data) procedure and by building phylogenies (family trees). We release the suite of etymological software for public use.

Most closely related to this work is our own previous work, e.g., [4], and work conducted at Berkeley, e.g., [5, 6]. The main improvement over these lies in awareness of a broader phonetic context of our models. We build decision trees to capture this context, where irrelevant context does not increase model complexity.

2. ALIGNING PAIRS OF WORDS

We begin with pairwise alignment: aligning pairs of words, from two related languages in our corpus of cognates. For each word pair, the task of alignment means finding exactly which symbols correspond. The simplest form of such alignment at the symbol level is a pair $(\sigma : \tau) \in \Sigma \times T$, a single symbol σ from the source alphabet Σ with a symbol τ from the target alphabet T. We denote the sizes of the alphabets by $|\Sigma|$ and $|T|$.

To model *insertions* and *deletions*, we augment both

alphabets with a special empty symbol—denoted by a dot—and write the augmented alphabets as $\Sigma_{\cdot}$ and $T_{\cdot}$. We can then align word pairs such as *vuosi—al* (meaning "year" in Finnish and Xanty), for example as any of:

v	u	o	s	i		v	u	o	s	i	etc...
|	|	|	|	|		|	|	|	|	|	
a	l	.	.	.		.	a	.	l	.	

The alignment on the right then consists of the symbol pairs: (v:.), (u:a), (o:.), (s:l), (i:.).

3. FEATURE-WISE CONTEXT MODELS

Rather than encoding symbols (sounds) as atomic, we code them in terms of their phonetic features. To this end, the corpus has been transcribed into feature vectors, where each sound is represented as a vector of five multinomials, taking on two to eight values, where the first entry is its type (consonant or vowel) and the remaining four entries are as listed in Figure 2. We also encode word boundaries (denoted by #) and dots (deletions/insertions) as extra types, with no additional features.

Consonant articulation		
M	**Manner**	plosive, fricative, glide, ...
P	**Place**	labial, dental, ..., velar, uvular
X	**Voiced**	– , +
S	**Secondary**	– , affricate, aspirate, ...
Vowel articulation		
V	**Vertical**	high—mid—low
H	**Horizontal**	front—center—back
R	**Rounding**	– , +
L	**Length**	1—5

Figure 2. Phonetic features for consonants and vowels.

We employ the MDL Principle [7] for model class selection and the MDL cost consists of two parts. First, we encode the model class $\mathcal{C}$, which is determined by a set of 18 decision trees, one for each feature (type plus four consonant and four vowel features) on both levels—source and target language. These trees query some *context* at each inner node, and their leaves provide the distribution to be used to encode the corresponding feature of a sound. More precisely the model (class) is allowed to query a fixed, finite a set of *candidate* contexts. A context is a triplet (L, P, F), where L is the level (source or target), P is a *position* relative to what we are currently encoding, and F is one of the possible features found at that position. An example of allowed candidate positions is given in Figure 3. In this setup, we have 2 levels $\times$ 8 positions $\times$ 2–6 features ≈ 80 candidate contexts, one of which defines an inner node of a feature tree. We can therefore encode each tree using one bit per node to indicate whether it is a leaf or not, plus about $\log 80$ bits for each inner node to specify the context on which it splits. For a model class $\mathcal{C}$, we need to encode all of its 18 trees in this way, the resulting total code-length we denote $L(\mathcal{C})$.

Context Positions	
I	itself, possibly dot
-P	previous position, possibly dot
–S	previous non-dot symbol
–K	previous consonant
–V	previous vowel
+S	previous or self non-dot symbol
+K	previous or self consonant
+V	previous or self vowel
...	*(other contexts possible)*

Figure 3. Context positions that a feature tree may query.

The second part of the code-length comes from encoding the aligned data using model class $\mathcal{C}$. We encode the feature in some fixed order, type first for it determines which other features need to be encoded. For each sound and each feature, we take a path from the root of the corresponding tree of $\mathcal{C}$ to a leaf, following at each inner node the branch that corresponds to the current context which is being queried. For example, when encoding feature **X** (voicedness) of a symbol σ in the source language we may arrive at a node given by $(L, P, F) = (target, -K, \mathbf{M})$ querying the manner of articulation of the previous consonant on the target level. This value (any manner of articulation or 'n/a' if there is no consonant on the target level between the current position and the beginning of the word) determines the edge we follow down the tree.

Each path from the root of a tree to a low-entropy leaf can be interpreted as as rule of phonetic correspondence. The path describes a contextual condition, the leaf gives the correspondence itself. High-entropy leaves represent variation that the model cannot explain.

In this way, all features of all symbols arrive at some node in the corresponding tree. We encode this data at each leaf independent of all other leaves using the normalized maximum likelihood (NML) distribution [8]. As the data at each leaf is multinomial, with cardinality $|F|$—the number of values feature F can take on—the corresponding code-length can be computed in linear time [9].

When $\mathcal{C} = \{\mathcal{T}_F^L\}$ consists of trees $\mathcal{T}_F^L$ for level L and feature F, and $\mathcal{D}$ is the aligned corpus such that $\mathcal{D}_{|L,F,\ell}$ is the portion arriving at a leaf $\ell \in \mathcal{T}_F^L$, then the overall code-length for $\mathcal{D}$ using $\mathcal{C}$ is

$$L(\mathcal{D}, \mathcal{C}) = L(\mathcal{C}) + \sum_L \sum_F \sum_\ell L_{NML}(\mathcal{D}_{|L,F,\ell}). \quad (1)$$

As implied, $L_{NML}(\mathcal{D}_{|L,F,\ell})$ is the multinomial stochastic complexity of the restricted data $\mathcal{D}_{|L,F,\ell}$. This code-length is the criterion to be minimized by the learning algorithm.

4. LEARNING

We start with an initial *random* alignment for each pair of words in the corpus. We then alternatively re-build the decision trees for all features on source and target levels as described below, and re-align all word pairs in the corpus using standard dynamic-programming, an analog procedure to the one described in [4]. Both of these operations decrease code-length. We continue until we reach convergence.

Given a complete alignment of the data, for each level L and feature F we need to build a decision tree. We

want to minimize the MDL criterion (1), the overall code-length. We do so in a greedy fashion by iteratively splitting the level-feature restricted data $\mathcal{D}_{|L,F}$ according to the cost-optimal decision (context to split upon). We start out by storing $\mathcal{D}_{|L,F}$ at the root node of the tree, e.g., for the voicedness feature **X** in Estonian (aligned to Finnish) we store data with counts:

+	801
-	821

In this example, there are 1622 occurrences of Estonian consonants in the data, 801 of which are voiced. The best split the algorithm found was on (Source, I, **X**), resulting in three new children. The data now splits according to this context into three subsets with counts:

+	
+	615
-	2

-	
+	135
-	764

n/a	
+	51
-	55

For each of these new nodes we split further, until no further drop in total code-length can be achieved. A split costs about $\log 80$ plus the number of decision branches in bits, the achieved gain is the drop in the sum of stochastic complexities at the leaves obtained by splitting the data.

5. EVALUATION

We present two views on evaluation: a *strict* view and an *intuitive* view. From a strictly information-theoretic point of view, a sufficient condition to claim that model (class) M_1 is better than M_2, is that M_1 yields better compression of the data. Figure 4 shows the absolute costs (in bits) for

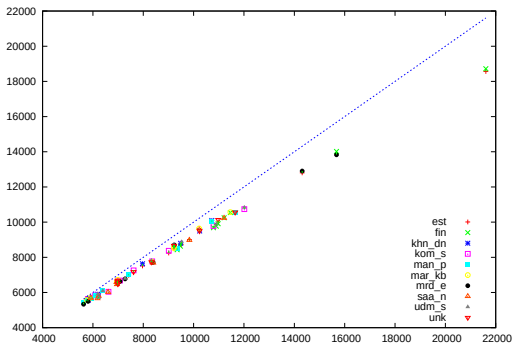

Figure 4. Comparison of code-lengths achieved by context model (Y-axis) and 1-1 baseline model (X-axis).

all language pairs[1]. The context model always has lower cost than the 1-1 baseline presented in [4]. In figure 5, we compare the context model against standard data compressors, Gzip and Bzip, as well as models from [4], tested on over 3200 Finnish-Estonian word pairs from SSA [3]. Gzip and Bzip need not encode any alignment, but neither can they exploit correspondence of sounds. These comparisons confirm that the new model finds more regularity in the data than the baseline model does, or an off-the-shelf data compressor, which has no knowledge that the words in the data are etymologically related.

[1]The labels appearing in the figures for the 10 Uralic languages used in the experiments are: est=Estonian, fin=Finnish, khn=Khanty, kom=Komi, man=Mansi, mar=Mari, mrd=Mordva, saa=Saami, udm=Udmurt, unk/ugr=Hungarian.

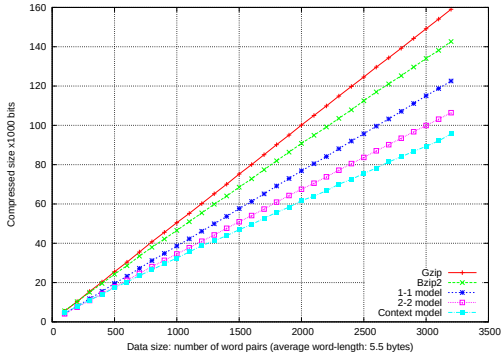

Figure 5. Comparison of compression power

For a more intuitive evaluation of the improvement in the model quality, we can compare the models by using them to *impute* unseen data. For a given model, and a language pair (L_1, L_2)—e.g., (Finnish, Estonian)—hold out one word pair, and train the model on the remaining data. Then show the model the hidden Finnish word and let it impute (i.e., guess) the corresponding Estonian. Imputation can be done for all models with a simple dynamic programming algorithm, very similar to the one used in the learning phase. Formally, given the hidden Finnish string, the imputation procedure selects from all possible Estonian strings the most probable Estonian string, given the model. Finally, we compute an edit distance (e.g., the Levenshtein distance) between the imputed string and the correct withheld Estonian word. We repeat this procedure for all word pairs in the (L_1, L_2) data set, sum the edit distances, and normalize by the total size (number of sounds) of the correct L_2 data—giving the *Normalized Edit Distance*: $NED(L_2|L_1, M)$ from L_1 to L_2, under model M.

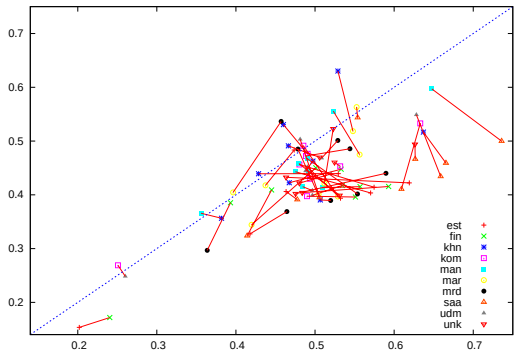

Figure 6. *Comparison of NED of context model (Y-axis) and "two-part 1-1" model (X-axis).*

The NED indicates how much regularity the model has captured. We use NED to compare models across all languages, Figure 6 compares the context model to the "two-part 1-1" model from [4]. Each of the $10 \cdot 9$ points is a directed comparison of the two models: the source language is indicated in the legend, and the target language is identified by the other endpoint of the segment on which the point lies. The further away a point is from the diagonal, the greater the advantage of one model over the other.

The context model always has lower cost than the baseline, and lower NED in 88% of the language pairs. This is an encouraging indication that optimizing the code length is a good approach—the models do *not* optimize NED directly, and yet the cost correlates with NED, which is a simple and intuitive measure of model quality.

A similar use of imputation was presented in [5] as a kind of cross-validation. However, the novel, normalized NED measure we introduce here provides yet another inter-language distance measure (similarly to how NCD was used in [4]). The NED (distances) can be used to make inferences about how far the languages are from each other, via algorithms for drawing phylogenetic trees. The pairwise NED scores were fed into the NeighborJoin algorithm, to produce the phylogeny shown in Fig. 7.

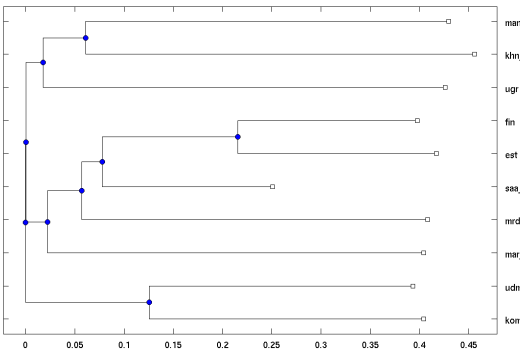

Figure 7. Finno-Ugric tree induced by imputation and normalized edit distances (via NeighborJoin)

To compare how far this is from a "gold-standard", we can use, for example, a distance measure for unrooted, leaf-labeled (URLL) trees found in [10]. The URLL distance between this tree and the tree shown in Fig. 1 is 0.12, which is quite small. Comparison with a tree in which Mari is not coupled with either Mordva or Permic—which is currently favored in the literature on Uralic linguistics—makes it a perfect match.

6. DISCUSSION AND FUTURE WORK

We have presented a feature-based context-aware MDL alignment method and compared it against earlier models, both in terms of compression cost and imputation power. Language distances induced by imputation allow building of phylogenies. The algorithm takes only an etymological data set as input, and requires no further assumptions. In this regard, it is as objective as possible, given the data (the data set itself, of course, may be highly subjective).

To our knowledge, this work represents a first attempt to capture *longer-range* context in etymological modeling, where prior work admitted minimum surrounding context for conditioning the edit rules or correspondences.

Acknowledgments

This research was supported by the Uralink Project of the Academy of Finland, and by the National Centre of Excellence "Algorithmic Data Analysis (ALGODAN)" of the Academy of Finland. Suvi Hiltunen implemented earlier versions of the models.

7. REFERENCES

[1] Sergei A. Starostin, "Tower of Babel: Etymological databases," http://newstar.rinet.ru/, 2005.

[2] Károly Rédei, *Uralisches etymologisches Wörterbuch*, Harrassowitz, Wiesbaden, 1988–1991.

[3] Erkki Itkonen and Ulla-Maija Kulonen, *Suomen Sanojen Alkuperä (The Origin of Finnish Words)*, Suomalaisen Kirjallisuuden Seura, Helsinki, Finland, 2000.

[4] Hannes Wettig, Suvi Hiltunen, and Roman Yangarber, "MDL-based Models for Alignment of Etymological Data," in *Proceedings of RANLP: the 8th Conference on Recent Advances in Natural Language Processing*, Hissar, Bulgaria, 2011.

[5] Alexandre Bouchard-Côté, Percy Liang, Thomas Griffiths, and Dan Klein, "A probabilistic approach to diachronic phonology," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, June 2007, pp. 887–896.

[6] David Hall and Dan Klein, "Large-scale cognate recovery," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[7] Peter Grünwald, *The Minimum Description Length Principle*, MIT Press, 2007.

[8] Jorma Rissanen, "Fisher information and stochastic complexity," *IEEE Transactions on Information Theory*, vol. 42, no. 1, pp. 40–47, January 1996.

[9] Petri Kontkanen and Petri Myllymäki, "A linear-time algorithm for computing the multinomial stochastic complexity," *Information Processing Letters*, vol. 103, no. 6, pp. 227–233, 2007.

[10] D.F. Robinson and L.R. Foulds, "Comparison of phylogenetic trees," *Math. Biosci.*, vol. 53, pp. 131–147, 1981.