May 11, 2020

Probabilistic Graphical Models for Sequential Data

Ivo Verhoeven
Supervisor: Dr. Ir. R. van den Doel

i.verhoeve@ucr.nl

University College R*oosevelt*
Lange Noordstraat 1
NL-4331 CB Middelburg
The Netherlands
http://www.ucr.nl

Abstract

This Bachelor's thesis discusses the rich theory of probabilistic graphical models and their relationship to structured data. Hidden Markov models are derived from the causal, directed and acyclic Bayesian networks. Conditional random fields are instead derived from the pseudo-probabilistic and undirected Markov random fields. The relationship between these models and more general structures are made clear. Inferential algorithms are discussed, tackling each of the fundamental HMM problems. Generalisation of these models to general topologies is achieved via the belief propagation algorithm. Parameter estimation and learning paradigms are discussed, with specific attention the Expectation-Maximisation principle; specifically, Baum-Welch learning. Finally, the use of PGMs to large vocabulary continuous speech recognition is discussed, specifically with regards to the TIMIT dataset. Limited experiments are preformed with a variety of graphical topologies. The results verify the validity of these models to automatic speech recognition, and indicate how PGMs might be used for other structured machine learning tasks.

Contents

Chapter 1

Introduction & Preliminaries

1.1 Probabilistic Graphical Models: A Brief Overview

While much of the notation used within the field can be attributed to the works of Gibbs [Gibbs2014] and Markov [Basharin et al.2004] (respectively scholars in statistical mechanics and probability theory), the first conscious application of a graphical method is typically ascribed to the geneticist Sewall Wright, a little over a century ago [Wright1920]. Interested in studying the causal effects of several confounding variables on another, Wright proceeded in generating a methodology that would later be expanded into contemporary path analysis/structural equation modelling/simultaneous equation modelling.

In the 1920 paper, Wright was interested in attributing the coat of offspring to hereditary and population stochastic effects. After establishing some basic tendencies of coat colouring, Wright continues by considering possible correlative relationships between the parents-offspring, both within and between groups. Using his knowledge on genetics, a set of causating variables were linked via the novel path coefficients; a single number denoting the importance between an independent cause and its effect. The resulting model is displayed in Fig.1.1, where the genetic backgrounds of the parents (H) and a dose of chance lead to the production of germ cells (G), which in combination with the other ancestor, environmental factors common (E) and unique (D) to litter mates, lead to eventual pattern of the offspring's coat. Note the use of arrows to denote *directionality* in the causal relationships, and nodes (black circles for abstract variables, and Guinea pig illustrations for their coats).

Unique to Wright's approach, beyond the summarisation of a high-dimensional probability space in a single succinct graphic, is the he used language; variables that were *correlated*, could under the right assumptions be used to generate *causal* claims. Virtually all undergraduate students will have heard the central tenet of statistics, "correlation does not imply causation"[1], and should at this point raise concerns regarding Wright's approach. Indeed, the statistics community, still in its infancy at this point, held on to this belief of Fisher and Pearson as if it were dogma [Pearl and Mackenzie2018]. As such, the method was considered highly controversial until social scientists stumbled upon it many decades later, rephrasing the theory of path coefficients into the formulation of structural equation modelling and its variants.

[1] It should at this point be noted that Wright did not imply this statement to be false. Rather, with sufficient understanding of a system of variables, *partial* causation between variables may be inferred. For a thorough response of Wright to his many intellectual adversaries, refer to [Wright1923].

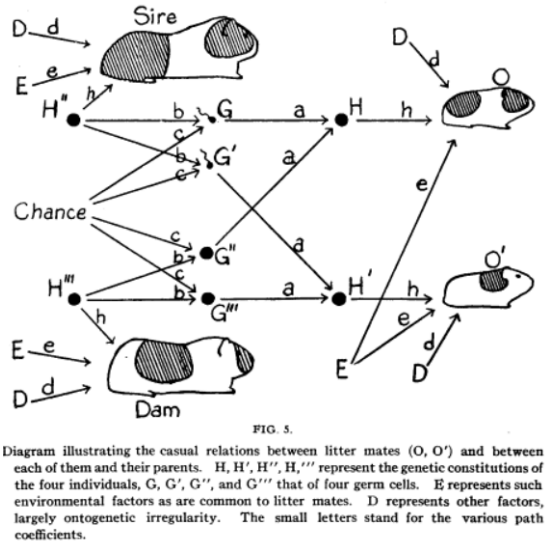

FIG. 5.

Diagram illustrating the casual relations between litter mates (O, O') and between each of them and their parents. H, H', H'', H,''' represent the genetic constitutions of the four individuals, G, G', G'', and G''' that of four germ cells. E represents such environmental factors as are common to litter mates. D represents other factors, largely ontogenetic irregularity. The small letters stand for the various path coefficients.

Fig. 1.1: Potentially the very first graphical model ever created, showing the confounding nature of parental genotypes and their expression in the phenotype of children (in guinea pigs). Taken from [Wright1920].

Of relevance to this thesis, however, is the work on probabilistic graphical models (PGMs) in the AI community in the 1980s. Koller and Friedman attribute the development of PGMs within AI to the efforts of Judea Pearl with the majorly important book *Probabilistic Reasoning in Intelligent Systems*, and the outlining of a framework for efficient calculations in causal networks by Lauritzen and Spiegelharter [Koller and Friedman2009,Pearl2014,Lauritzen and Spiegelhalter1988]. Beyond laying the foundation for Bayesian networks, many of the core concepts used today, like belief propagation and structure learning, find their origins in the works of Pearl. Much of his research has since tried to expand the definition of BNs to become general causal inference machines. For his efforts, Pearl was awarded the 2012 ACM Turing Award. At the time of his work, he too was required detailing why a probabilistic method was optimal for his purposes. Two suggested reasons for the continued controversy around graphical models are, i) the implication of causality in cases of observed correlation, ii) the adoption of a Bayesian philosophy to information updating [Pearl and Mackenzie2018].

While Pearl focused on a specific subset of PGMs, namely Bayesian networks, while the work of Lauritzen was more general, these now all fall under a central framework, perhaps best detailed in Koller and Friedman's book "*Probabilistic Graphical Models: Principles and Techniques*" [Koller and Friedman2009]. As it turns out, PGMs lie at the very heart of a vast domain of modern machine learning techniques, some of which will be discussed in this thesis. The next section will give a whirlwind tour of the motivation behind PGMs, summarising parts of the previously cited works.

1.1.1 PGM Concepts and Methods

Probabilistic graphical models is an umbrella term and theoretical framework for a variety of artificial intelligence techniques developed in the 20th century. While on one level, this collection of methodologies is merely a way of compactly and succinctly summarising high-dimensional probability spaces, it is generally the hope that these models will be able to separate reasoning and knowledge.

PGMs express probability spaces as a combination of a graph $\mathcal{G}$ with vertices and edges $(\mathcal{V}, \mathcal{E})$ and a (joint-)probability density function $\mathcal{P}$. In the graph, the vertices (or nodes) represent variables, and connecting edges (or arcs) represent dependencies. The nature of these dependencies are determined by the application, e.g. causal for Bayesian networks and correlative for Markov random fields. The necessity of graphs as method of representation is not merely aesthetic or psychological, it also carries over into inferential problems. Consider the simplest possible network of K binary variables, such that relationships between nodes are comprised entirely out of 2×2 contingency tables. Expressing the conditional probability function of $\mathcal{G}$ is exponential in K, such that updating one variable in the network results in $\mathcal{O}(2^{K-1})$ updates. Realistic applications will see variables considerably more complex, making these representations virtually intractable[2]. PGMs address this problem by considering the global probability function as a product of several local functions, dependent on subsets of variables [Sutton et al.2012].

Bayesian Networks The first major representational form of PGMs considered are the Bayesian networks (BNs). In these, each variable is represented by a node and the *directed* vertices indicate causal relationships between variables. For such a graph to represent a PDF, it is crucial that it is *acyclic*, such that Bayesian networks can also be referred to as directed acyclic graphs (DAG). Simple examples might include the naive Bayes classifier, Markov chains, while more complex examples might be those used for medical diagnoses. Important for Bayesian networks are the possibility of establishing natural parent-child relationships between variables.

Markov Random Fields The second major representational form of PGMs are the Markov Random Fields (MRFs). Unlike BNs, these can be *undirected*, and allow for cycloids within the graph. While the nodes still represent variables, the edges now represent a quantity closer to correlation. These models are no longer directly probabilistic, but instead considers normalised 'affinities' between variables. MRFs find their conception in the Ising model for ferromagnetism, a classic example of a natural phenomenon where cause is hard to attribute but correlations between atoms induce regions of magnetisation.

[2] Keeping in mind these models were developed in the early 1980s.

1.1.2 Structured Data

Consider a dataset consisting of independent variables ($\mathbf{X}$) that relate to a set of labels ($\mathbf{Y}$) through some function $f(\mathbf{X}) = y$. In the regular machine learning task, the set of labels fall on the real line, such that $y \in \mathbb{R}$. In classification these would be discrete classes, whereas for regression task this would any continuous value.

For the structure data learning task, this is not possible. Rather, the variables indicate the label to be of some complex structure itself. There is necessary information that cannot be found in the data itself, but rather in the structure it takes. This structure typically has to be determined *a priori*[3]. The following definition is taken from Nowozin and Lampert's review of tasks and solutions within structured data learning for computer vision [Nowozin et al.2011]:

> **Structured Data**: data that consists of several parts, and not only the parts themselves contain information, but also the way in which the parts belong together.

Structured data is everywhere, and practically ubiquitous common to human reasoning. For example, the linguistic analysis of natural language has shown tremendous amounts of phenomena which are dependent on context. The word "back" in and of itself has an extremely ambiguous grammatical meaning, being a noun, adjective, verb and adverb all at once. The word "back" in the sentence, "Janet will back the bill"[4], however, is clearly a verb. The part of speech of the desired word ("back") only became clear after knowledge of the entire sentence.

Another classic example is that of image segmentation, see for example Fig.1.2. Here two images are segmented into interesting regions, with the images coming from very differing contexts. The third and fourth columns give the predictions using two different machine learning paradigms: the third column as a standard classification task using only the information within the super-pixels, whereas the fourth column shows structured learning in action, where spatial relationships between super-pixels are also taken into account. It's clear that the structured learning method much better matches the complex objects human vision immediately spots and relates together.

PGMs for Structured Data Surprisingly, independent of the developments within PGMs, the 1960s saw the development of the hidden Markov model (HMM) specifically for speech recognition by the Baum, Welch and colleagues [Baum and Petrie1966]. In his very influential tutorials, Rabiner notes that these techniques came from the necessity to model both the static and dynamic processes of physical signals (i.e. human speech) [Rabiner and Juang1986, Rabiner1989]. While these methods drew inspiration

[3] Although the task of structure learning, specifically within the field of PGMs, is a burgeoning and promising research area.

[4] Example taken from [Martin and Jurafsky2018]

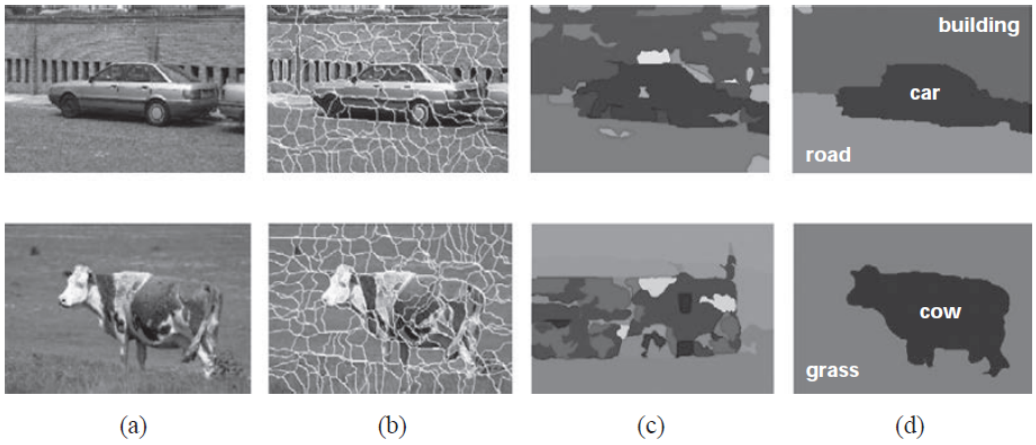

Fig. 1.2: Semantic labelling on two images within very different contexts. Column (a) gives the natural image, column (b) gives the image pre-processed in super-pixels (similar regions), column (c) gives prediction using only the information within the super-pixels, column (d) gives prediction using a CRF, clearly showing the separation of structures into accurate latent groups. Taken from [Koller and Friedman2009].

from information theory and generally from Markov processes, today these are recognised as the simplest form of dynamic BNs. An intense period of research ensued, but the widespread use of HMMs would have to wait until the late 1980s. The derivation of HMMs, the fundamental questions one can raise and solutions to those questions were generally introduced by Rabiner, and have since then seen little innovation beyond extensions to sequential analysis fields outside of automated speech recognition (e.g. protein alignment in DNA, spelling correction, various labelling tasks in natural processing) [Xing et al.2010].

It would not be until 2001 that the conditional random field (CRF) was derived as a generalisation of HMMs from a dynamic BN to an equivalent MRF form. The motivation Lafferty, McCallum and Pereira provide is the difficulty that arises in training generative HMM (aim to estimate the joint probability distribution $P(x, y)$) and the *label bias* problem in discriminative (aim to estimate the conditional probability distribution $P(y|x)$) extensions of HMMs [Lafferty et al.2001]. Since their introduction, McCallum has been a co-author on an influential tutorial similar to that of Rabiner [Sutton et al.2012]. Much like the original HMMs, a great deal of applications exist, with among others natural language processing, bio-informatics, robotics and computer vision [Nowozin et al.2011]. Specific areas in which CRFs are now employed, as identified by Truyen, include speech recognition, word segmentation (audio), parts of speech tagging and named entity recognition, information extraction, image segmentation, object recognition, stereo vision, activity recognition and sequential classification [Truyen2008].

1.2 Probability

Before delving into the largely abstract and vastly complex framework of probabilistic graphical models, a small but succinct review of the fundamental concept of probability will prove useful. While this section in no way aims to be a complete reference, it does review and collect some basic results that will provide the fundamental building blocks upon which all later theory may be placed.

While the choice for a probabilistic framework is today an obvious one, for the reader perhaps already obvious due to the title of this work, this was not always the case within the AI community. While already centuries old at the time of publication of Pearl's '*Probabilistic Reasoning in Intelligent Systems*', the author deemed it necessary to motivate the use of probabilistic reasoning as opposed to the many alternatives available at the time. Of special interest to researchers at the time, when confronted with uncertainty in proposed intelligent systems, were methods like Dempster-Schafer calculus and fuzzy logic. In a similar manner, the frequentist and Bayesian methodologies for probabilistic reasoning, peppered with some important realisations for a qualitative understanding of probability, as understood around the time of Laplace will be presented.

1.2.1 Defining Probability

Despite the aforementioned maturity of probability as a framework for dealing with uncertainty, there exist multiple, sometimes conflicting but often complimentary, philosophical and mathematical interpretations. The most common are typically referred to as either frequentist or Bayesian. The distinction is often subtle, but of great importance to many subject areas. Frequentist, the classical viewpoint of Pearson and Fisher, relates probability to long-run occurences of events, whereas Bayesian seeks to quantitatively express the information known of a phenomenon and update it by Bayes/Laplace's theorem.

- **Frequentist:** define the probability of an event occurring as the long term relative frequency of the event occurring. The frequency of events is dependent on the rate of occurrence in the sample space $\mathcal{S}$, such that the probability of an event $S_i = A$ is defined as,
$$P(A) = \frac{N_A}{N_{\mathcal{S}}}.$$
While this definition has served the field of statistics and experimental methodology very well, there are shortcomings with regard to queries that do not make use of large number of repetitions under similar circumstances. Natural questions like "What is the likelihood of this Bachelor's thesis being completed on time?" cannot be simply answered without a list of assumptions.
- **Bayesian:** define the probability of an event occurring as the degree of personal, and therefore per definition subjective, belief in an event occurring. The question above then becomes an amalgamation of the person's prior experience with similar events and the information currently available; dependent on the person being asked,

the answer will vary. It is assumed for rational persons, the rules of probability that flow naturally from the frequentist definition still hold when applying a subjective perspective.

Probabilistic graphical models were born out of the necessity for general reasoning patterns within intelligent systems. As such, like many modern machine learning techniques, it is placed firmly within the Bayesian interpretation. The implication of this is presented later in this section.

Regardless of philosophical inclination, mathematically there exists only one definition of probability. While a generalisation of probabilities to more general *factor* functions is provided later, the laws by which any probability measure must abide remain constant.

Definition 1 (Probability). *Let $P(X)$ be a real-valued function, defined over the variable X, that assigns to every event $x \in X$ in the sample space $\mathcal{S}$ a value that satisfy the following conditions:*

$$0 \leq P(X = x) \leq 1 \tag{1a}$$

$$\sum_{X \in \mathcal{S}} P(X_i) = 1 \tag{1b}$$

$$P(X_1 \cup \ldots \cup X_N) = P(X_1) + \ldots + P(X_N) = \sum_{\forall i} P(X_i) \tag{1c}$$

$$\textit{iff } X_i \cap X_j = \emptyset, \textit{ for all } i \neq j$$

One reads the statement $P(X = x)$ as '*the probability of the variable X taking the value x as $P(X = x)$*'. Generally $P(X = x)$ is written simply as $P(x)$. Recall that $P(X)$ is defined as a function, it simply assigns to an assignment of the variable over which it is defined a probability measure, and may be either continuous or discrete.

While the provided definition is sufficient for understanding other properties of probability, a number of crucially important attributes are further outlined in the subsections below; proofs may be found in any introductory text on probability theory.

Joint Probability Functions A probability function may be defined over one or more variables, where one defines a joint-probability over many variables as a function of many marginal (univariate) functions. An example of a two dimensional joint-probability function is given in Fig. 1.3. The rules for specifying the form of this joint-probability functions are defined by the concept of statistical dependence. Joint-probability functions specify the likelihood of an event of one variable co-occuring with the events of another,

$$P(X, Y) = P(X \cap Y).$$

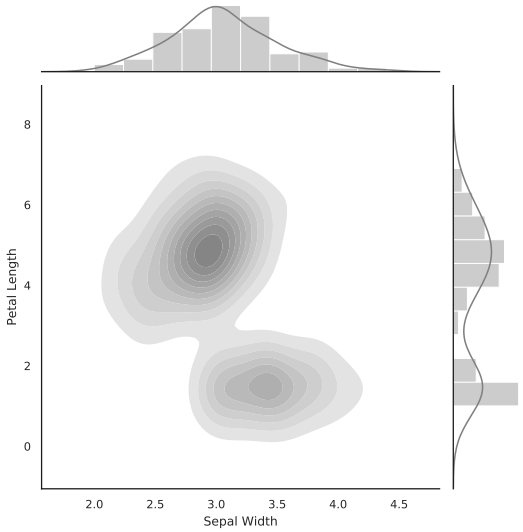

Fig. 1.3: A joint probability function using two variables from the Iris dataset (see the example on Naive Bayes classifiers). The sepal width is depicted on the horizontal, the petal length on the vertical.

One may consider $P(X, Y)$ as a (perhaps infinitely) long three-dimensional table. Every joint-event, $X = x_i \cap Y = y_i$ is associated to a single probability. One can always, once given a joint-probability space, revert back to the marginal distribution of which it is comprised. Consider a variable X, comprised of many possible events $X = x_i$, which are per definition disjoint ($X_i \cap X_j = \varnothing$, for all $i \neq j$). If one sums $P(X \cap Y)$ over every possible assignment to X, this is equivalent to taking the union of all $x \in X$ or

$$\sum_{\forall x_i \in X} P(x_i \cap Y) = P((x_1 \cup \ldots \cup x_N) \cap Y) = P((\cup_{\forall x_i \in X} x_i) \cap Y).$$

Note that it is given that the sum exhausts all possible assignments of X, and therefore simply returns the sample space $\mathcal{S}$. The union of the sample space with a variable is simply the variable itself. As such, it follows that the sum over one variable in a joint probability function simply returns the marginal distribution in terms of all other variables. This property will often be recalled in this thesis, generally under the name of a marginalisation operation, defined as,

$$\sum_{\mathbf{X}} P(\mathbf{X}, \mathbf{Y}) = P(\mathbf{Y}). \tag{2}$$

Eq. 2 reads as '*the sum over the probability function defined by the sets of variables* $\mathbf{X}$ *and* $\mathbf{Y}$*, for all variables in* $\mathbf{X}$ *returns the probability function defined in the set of variables* $\mathbf{Y}$*,* $P(\mathbf{Y})$'.

Conditional Probability If additional information is presented, will this effect the probability of an event occurring? Questions of this nature are captured by conditional probabilities, where an '|' operator denotes the observation of additional information.

For example, the probability of achieving an 'A' grade ($G = `A'$) is likely dependent on the difficulty of the course $D$; for easier courses $P(G = `A', D = \text{Easy})$ increases, and for more difficult courses $P(G = `A', D = \text{Hard})$ decreases. The relationship between the joint probability $P(G = `A', D)$ is the product of the probability of an 'A' given the difficulty of the course, and the probability of the course's difficulty.

$$P(\text{Grade='A', Dificulty='Hard'}) = P(G = \text{'A'}|D = \text{Hard})P(D = \text{Hard}) \tag{3}$$

A more general form of the above equation follows as,

$$P(X_i, X_j) = P(X_i|X_j)P(X_j) = P(X_j|X_i)P(X_i). \tag{4}$$

Not all conditional probabilities will be meaningful, however. The probability of an 'A' does not depend on whether homework, variable H, needs to be submitted using a pencil or a pen. In cases like these, referred to as *statistically independent* or disjoint events ($X_i \cap X_j = \emptyset$, for all $i \neq j$), the conditional probability collapses into just the probability of the event; the additional information has no influence on the considered variable.

$$\begin{aligned} &P(\text{Grade='A',Homework='Pencil'}) = P(G = \text{'A'})P(H = \text{Pencil}) \\ &\because P(G = \text{'A'}|H = \text{Pencil}) = P(G = \text{'A'}). \end{aligned} \tag{5}$$

Statistical independence and dependence will be denoted using a perpendicular symbol between relevant variables,

$$\text{Independent} \quad (X_i \perp X_j) \tag{6}$$

$$\text{Dependent} \quad (X_i \not\perp X_j). \tag{7}$$

An extension of Eq. 4 in the context of (large) joint probability functions is the *chain rule of probabilities*, or simply the *product rule*. Consider a set of variables $\mathbf{X} = \{X_1, \ldots, X_K\}$, which for clarity of argument is indexed according to topological ordering, between which exists different statistical (in)dependencies, for example $(X_1 \not\perp \emptyset)$, $(X_2 \not\perp X_1)$, $(X_3 \not\perp X_1, X_2)$, etc. The full joint probability distribution may be written as

$$P(\mathbf{X}) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)\ldots P(X_K|X_{K-1}, X_{K-2}, X_{K-4}).$$

More generally, this may be written as a product of dependent and independent factors.

$$P(\mathbf{X}) = \prod_{\forall X_i \in \mathbf{X}} P(X_i|\mathbf{Y} : (X_i \not\perp Y)). \tag{8}$$

In the same manner as above, Eq. 8 may be read as '*the probability function* $P(\mathbf{X})$ *is the product of the probability functions* $P(X)$ *conditioned on all variables of which* X *is statistically dependent*'. This notion of expanding a joint probability function into dependency components is one that will be formalised and exhaustively discussed in Ch. *Bayesian Networks*.

1.3 Bayes' Theorem

An immediate, but crucially important, extension of Eq. 4 is Bayes' theorem:

$$\begin{aligned} P(X_i|X_j)P(X_j) &= P(X_j|X_i)P(X_i) \\ &\implies P(X_i|X_j) = P(X_j|X_i)\frac{P(X_i)}{P(X_j)}, \quad \text{or,} \\ &\implies P(X_j|X_i) = P(X_i|X_j)\frac{P(X_j)}{P(X_i)} \end{aligned} \quad . \tag{9}$$

The importance of Bayes' theorem comes from the fact that it allows one to calculate the inverse conditional probability whenever given a conditional probability. When used within the context of subjective probability, this theorem becomes an exceptionally powerful tool for learning given prior beliefs and evidence in the form of data. Within the context of the Bayesian interpretation, the posterior probability of an unknown quantity Θ (otherwise, simply the conditional probability of an unknown when new evidence is provided), can be defined as,

$$\begin{aligned} \text{Posterior} &\propto \text{Likelihood} \cdot \text{Prior} \\ P(\theta|\mathbf{X}) &\propto P(\mathbf{X}|\theta) \cdot P(\theta), \end{aligned} \tag{10}$$

where all that is further required is normalising by the probability of the evidence $\mathbf{X}$. This allows one to update the posterior given a set of observations and the prior state of knowledge. The likelihood, $P(\mathbf{X}|\theta)$, captures the simplest available hypothesis, i.e. the form of Θ, that fits the available observations $\mathbf{X}$. The prior, $P(\theta)$, is what makes Bayesian subjective (hence, the interchangeability of subjective and Bayesian when referring to the philosophical framework) [Murphy2012]. This factor captures the probability of hypothesis given some earlier evidence, or if none is available, the subjective opinion of the interpreter.

Despite the application of Bayesian formalism to human reasoning, quite literally intuitive, the use of Bayes' theorem in updating information from new evidence to new posteriors can be confusing at first. The following is a simple example wherein the details are largely irrelevant for this thesis, but the method frequently used:

Example: How good is Messi anyways?
A new football player is about to take his first professional penalty kick. While the name Messi carries significant weight nowadays (i.e. his prior has seen significant updating), at the time one assumes nothing is known. The likelihood of him scoring the free-kick, otherwise, the likelihood of success in a Bernoulli trial, is denoted p. The player being totally unknown allows for constructing a prior distribution that expresses the uncertainty in the true value of Messi's p. One (albeit naive) choice would be the uniform distribution between 0 and 1.

Now imagine looking back after Messi's first season. Since his debut year was highly succesful, with 4 penalties netted and 1 missed, one would expect the posterior,

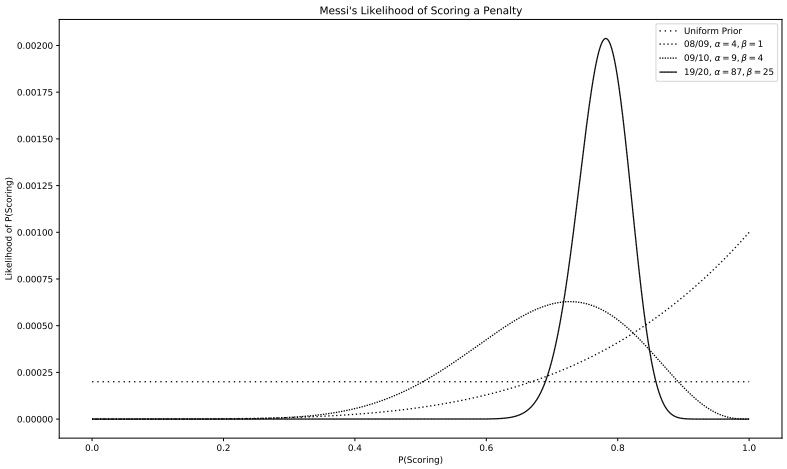

Fig. 1.4: The prior and several posteriors for Messi's ability to score penalty kicks. Note the uniform distribution when completely uncertain, and the ever narrower curve as information is added.

the likelihood of his scoring given the now available evidence, to be positively updated; the distribution should lie much closer to 1 than the uniform prior suggested. Indeed, Fig 1.4 shows the 08/09 posterior slanted strongly towards the right bound, with a mean of $p = 0.80$. Note however the large uncertainty; possible values lie anywhere between $p = 0.4$ and $p = 1$.

The second season, 09/10, was considerably worse, with another 5 scored along with 3 misses. Again, the posterior reflects change in the expected direction, with a mean now at $p = 0.70$. The confidence in the estimate has however increased considerably as estimates of the true parameter being more tightly distributed than the earlier posterior and prior. Finally, and skipping ahead a decade, the posterior has settled somewhere in between the posteriors of the earlier seasons. The mean lies at $p = 0.78$, with Fig. 1.4 showing feasible values to lie somewhere in the region $0.7 < p < 0.9$.

Thus, a posterior about Messi's ability is constructed by first establishing a subjective prior, the considering the likelihood of the incoming evidence. The knowledge about the problem updates gradually, with the model fitting this scenario gaining in confidence as concurring data is provided.

1.3.1 Naive Bayes: A Generative Classifier

Statistical classification is the task of assigning to unseen data the category to which it most likely belongs. Consider a dataset with a set of observed features $\mathbf{X}$ and a target label y. The feature vector contains K variables, and N samples. For the special case of

binary classification the labels are of the set $y \in \{0, 1\}$, otherwise the labels fall in range defined by the cardinality of the label set.

Armed with only Bayes' theorem, and some familiarity with basic statistical distributions, it is already possible to create a simply but surprisingly powerful classifier: naive Bayes (NBC). Let the result of the NBC be the label that maximises the posterior likelihood, given the input feature vector $\mathbf{X}$ and some model parameters θ, then,

$$y = \arg\max P(y \,|\, \mathbf{X}, \theta). \tag{11}$$

Using Bayes' theorem, this may be rewritten into a product of a likelihood and prior. Keeping with notation common to existing literature, let π denote the prior of the label, such that,

$$\begin{aligned} P(y \,|\, \mathbf{X}, \theta) &\propto P(y) P(\mathbf{X} \,|\, y, \theta) \\ &\propto \pi \prod_{\forall x \in \mathbf{X}} P(x \,|\, y, \theta). \end{aligned}$$

The last step in the above is why this classifier is typically called naive or idiot Bayes. The likelihood probability $P(\mathbf{X} \,|\, y, \theta)$ is broken into its constituent parts and multiplied out via a product. The implication is clear; it is assumed that there exists no relationship whatsoever between the features, or in the notation of statistical independence,

$$(x_i \perp x_j) \forall x \in \mathbf{X}, i \neq j.$$

Natural language processing is a field where the NBC is commonly employed as a baseline classifier. The number, or presence of words are typically used as features. Linguistically, the naive Bayes assumption of independence between features is clearly erroneous, as little thought is required to convince oneself that certain words hint at the inclusion of other words. Regardless, the NBC generally models textual data remarkable well.

Depending on the datatype, a number of different distributions may be used for modelling $P(x \,|\, y, \theta)$. For example, one may take the Gaussian distribution, with $\theta = \{\mu, \sigma\}$, for continuous variables, or a multinoulli (generalised Bernoulli) distribution for general categorical variables. Fitting the NBC is outside the scope of this section (see chapter 'Learning'), however the MLE parameters estimates are provided in the example below.

Example: Gaussian Naive Bayes for Iris Classification
An NBC is trained on the classical Iris dataset. It contains 150 measurements of iris petals and sepals (as a non-phytomorphologist, the coloured and green leaves of the flower respectively), distributed over 3 species' classes, each with 50 observations. A 1:1 train-test split is made, such that there are equal numbers of training and evaluating data instance.

No specific prior is provided, and as such a simply prior based on the relative frequencies of the classes is constructed. The training set contains 23 Setosa, 25 Versicolor

Table 1: A shuffled subset of the famous Iris dataset introduced by Fisher [Fisher1936]. The measurements are all in cm.

Sepal length	Sepal width	Petal length	Petal width	Species
6.9	3.1	4.9	1.5	Versicolor
5.1	3.8	1.5	0.3	Setosa
6.7	3.3	5.7	2.1	Virginica
6.3	2.5	5.0	1.9	Virginica
5.1	3.5	1.4	0.3	Setosa

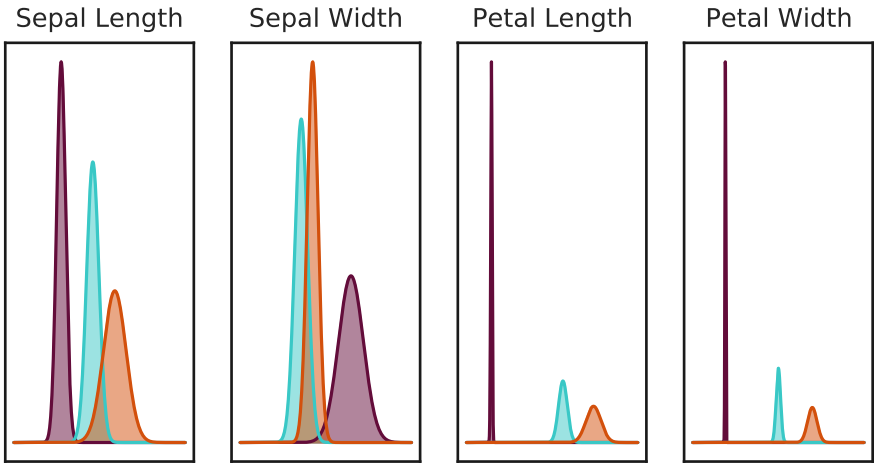

Fig. 1.5: The fitted likelihood functions for each variable. Colours denote species; Setosa in purple, Versicolor in orange and Virginica in blue.

> and 27 Virginica flowers, and as such, $\pi = \{0.31, 0.33, 0.36\}$. A Gaussian likelihood function is specified (given the continuous measurements and the qualitative shape of distributions), such that $P(x \mid y, \mu, \sigma) = \mathcal{N}(\mu_{x,y}, \sigma_{x,y})$. The likelihood function for the sepal width of Setosa is $\mathcal{N}(3.38, 0.17)$ whereas that of Virginica is $\mathcal{N}(2.93, 0.10)$. At evaluation, this model achieves 93% accuracy.

The NBC, as compared to logistic regression introduced in the next section, is a generative classifier. From the name, once having trained the classification model, it is always possible to construct synthetic data that (hopefully) closely approximates the real provided training data. It is indicative of generative models that these are generally probabilistic in nature, and have some Bayes' theorem somewhere in its definition. As such, the classifier is trained to be able to recreate both the observed data and the class to which the data would belong. This provides the analyst with very intuitive results and allow for natural interpretation, and it can do so from relatively little data. However, generative training often specifies strict assumptions (the naive Bayes assumption really is naive in virtually all application domains), and deviation from those assumptions

Table 2: Sequences of coin flips, their relative frequencies and their related entropy.

Sequence	Observation	P(H)	P(T)	H
1	H T H H H H H H H H	0.9	0.1	0.47
2	T T T H T T H T H T	0.3	0.7	0.88
3	H T H T H T H T H T	0.5	0.5	1.00

can quickly invalidate the model. For a more thorough discussion on generative models, consult Murphy [Murphy2012] or in the context of PGMs, consult Koller Friedman [Koller and Friedman2009].

1.4 Information Theory

The field of information theory provides a framework for situations where only probability theory proves insufficient. The concept of informational entropy is fundamental to many machine learning tasks, and provides a theoretical basis for many discriminative classifiers. Furthermore, related concepts like relative entropy and the Kullback-Leibler divergence give natural definitions to statistical distance. Consider a function, say $S(x)$, that captures the amount of surprise of of an event occurrence. Given that $P(x)$ captures the likelihood of occurrence of an event, the surprise should be proportional to the inverse of the probability: very unlikely events are far more surprising than likely events. More formally, $S(x) \propto P(x)^{-1}$.

Very informally, one considers the amount of information provided by a data generating process the average amount of surprise within each observation (or message). In Shannon‘s original work, he was concerned with the encoding of messages into bits, with his measure of entropy providing an efficiency bound to possible encodings. While the specific procedure for doing so is outside of the scope of this thesis, a simple example to introduce the concept of information entropy in the same spirit as Shannon's work is possible.

Example: Entropy of Bernoulli Trial
Consider a sequence of results from a coin flip, such as in Table 2. In the first sequence, the observer experiences incredible surprise at the occurrence of a tails observation, providing much information for the encoding of the data generating process into a discrete probability distribution. However, the occurrence of such surprising events are very rare, and as such the average information received by the observer (denoted by H) is relatively low. In contrast, the second sequence provides much more information per event, with the likelihood of either coin face lying much closer together. Lastly, a sequence where either observation occurs at the same relative frequency as the other observation has the observer sitting on the edge of their seat, with each new observation being surprising and as such providing valuable information. As a result, the average information content is highest.

The choice of the symbol H and the related term, information entropy, is done to highlight the relation between Shannon's for information theory and Boltzmann's definition within statistical mechanics. The specific form of H follows from the largely theoretical requirements for a suitable definition for entropy. Let $h(x)$ be the information generated by the occurrence of an event:

1. The information function $h(x)$ is continuous in $P(x)$.
2. If $P(x_i)$ is the uniform distribution, such that $P(x_i) = \frac{1}{N}$, $h(x)$ should be a monotonic function in terms of N, where higher N implies greater uncertainty through greater choice.
3. The information provided by an event is a function of the surprise of that event; otherwise, it is a function of the inverse probability of an event.
4. The information provided by two disjoint events, such that $P(x^{(1)}, x^{(2)}) = P(x^{(1)})P(x^{(2)})$, is the sum of the information of each individual event, such that $h(x^{(1)}, x^{(2)}) = h(x^{(1)}) + h(x^{(2)})$.

A logical candidate for a suitable function is the logarithm of the surprise function, or the negative logarithm of the probability, as,

$$h(x) = \log S(x) = -\log P(x).$$

See Shannon's Appendix 2 for proof that it is the *only* possible function that meets the above criteria. The average information provided by an event is denoted as information entropy; the probability weighted sum of the above defined information function.

Definition 2 (Information Entropy). *For a random variable X that is distributed according to a discrete, categorical distribution, the Shannon or information entropy is defined as*

$$H(x) = E(-\log P(x)) = -\sum P(x) \log P(x)$$

The entropy function is maximised for if $P(x)$ is most evenly distributed over all x, and as such takes the uniform distribution, and minimised at $H(x) = 0$ for a random variable that is constant.

While in itself a fundamental result for information and probability theory in and of itself, the concept of entropy is immensely useful for another reason: the maximum entropy principle. While in the framework of machine learning this method is relatively new, often referred to as MaxEnt, its origin coincides with the birth of probability theory, being reinvented multiple and going by either the Principle of Insufficient Reason (Bernoulli), or the Indifference Principle (Laplace), and has drawn the attention of thinkers of the calibre of Keynes and Poincare [Weissteinb]. While an adamant opponent, Keynes provides an exceptionally clear formulation of the principle:

> "The Principle of Indifference asserts that if there is no known reason for predicating of our subject one rather than another of several alternatives, then relatively

> to such knowledge the assertions of each of these alternatives have an equal probability. Thus equal probabilities must be assigned to each of several arguments, if there is an absence of positive ground for assigning unequal ones." [Keynes2013]

A possible reason for Keynes' protest are the many possible interpretations that indifference can take, allowing one to easily use the principle to stumble into a paradox. Otherwise, the choice for an as uniform distribution as possible appears to be as arbitrary as any other. This and more is noted by Jaynes [Jaynes1957] in their attempt to define the method of maximum-entropy estimates. However, given that Shannon's entropy measure assigns a single, unique number to the preference of a 'flat' to a 'peaked', while simultaneously accounting for all possible events, Jaynes considers entropy to be the missing link in the earlier formulated indifference principle. From Jaynes' paper,

> "The maximum-entropy distribution may be asserted for the positive reason that it is uniquely determined as which is maximally noncommittal with regard to missing information, instead of the negative one that there was no reason to think otherwise." [Jaynes1957]

Firmly affirming oneself in the Bayesian philosophy, the interpretation reads closely similar to the notion of Occam's razor. With no additional information available beyond the prior, the most probable posterior is the one that most evenly spreads the probability over the possibles states. The principle of maximising entropy is especially useful since it can be done independently of the probability distributions' momenta. If a mean and variance are known, the maximum-entropy posterior can adhere to the summary statistics while assigning the least amount of bias to any individual state.

1.4.1 Logistic Regression: A Discriminative Classifier

Often called the maximum entropy classifier, logistic regression is the discriminative and information theoretic equivalent to naive Bayes. A concise derivation is provided here, within the framework of information theory. For a more in depth discussion on frequentist and information theoretical derivation of MaxEnt classifiers see Mount's paper [Mount2011], or consult any standard machine learning textbook. A discussion on logistic regression, log-linear tables and some basic PGMs, see Christensen's '*Log-linear Models*' [Christensen2006].

Consider a function, $f(x)$, that serves as the ideal discriminative classifier. Ideal here implies the following three properties, i) it assigns believable probabilities of an observation to lie in a class, ii) it closely and preferably perfectly simulates the indicator function of an observation belonging to a class, and lastly iii) it maximises the entropy of the found classifier. The first two qualities ensures good behaviour of the classifier, and the third ensure the minimal bias to any particular configuration of the classifier (i.e. maximum entropy ensures a reduction of overfit).

As such, finding the function $f(x)$ is defined as a constrained optimisation problem. Without considering too many details, a common way for solving problems of this nature is by defining a Lagrangian whose behaviour near critical values is similar to that of the true optimised function. The Lagrangian takes the form of,

$$L = \sum_{\forall i \in y} \sum_{\forall x \in \mathbf{X}} H(f(x)) + \sum_{\forall i \in y} \sum_{\forall x \in \mathbf{X}} \theta_x (f(x) - 1) + \sum_{\forall i \in y} \sum_{\forall j \in y} \sum_{\forall x \in \mathbf{X}} \beta \text{Loss}(f(x), j).$$

This function looks horrendous, but has as redeeming factor that it's derivative is far simpler. The first component expresses the desire to maximise the entropy of the function, summed over all possible class assignments for every data point. The second is the probability constraint ($0 \geq f(x) \geq 1$), with a Lagrangian multiplier θ. The last component is the loss function, expressing the need to most closely approximate the indicator class function, summed over every possible assignment of true and estimated label for each data point, again with a Lagrangian multiplier β.

The derivative of interest, or the actual function one wishes to optimise, is in terms of $f(x)$ for a specific class, thus $f(x)_i$. Again, without detailed derivation,

$$\frac{\partial L}{\partial f(x)_i} = 0 \implies f(x)_i = \exp \theta_i x + \beta_i - 1.$$

From the problem definition, it is already known that the sum over all possible values is 1 (from the constraint of behaving like a probability). This allows for rewriting of the β Lagrangian multiplier as,

$$e^{\beta} = \frac{1}{\sum_{\forall y \in Y} \exp \theta_i x}.$$

Finally the form of $f(x)_i$ is known, and to little surprise to anyone who has familiarity with logistic regression, it takes the sigmoid function of the input. From a frequentist derivation of logistic regression, the input represents the linear regression of the data $\mathbf{X}$ against the odds ratio of belonging to class $i \in y$:

$$P(Y \mid \mathbf{X}, \theta) = \frac{\exp \theta_i \mathbf{X}}{\sum_{\forall y \in Y} \exp \theta_i \mathbf{X}}. \tag{12}$$

A slight change in notation is introduced to match with earlier results. Given the prior enforced constraints, the function $f(x)$ is entirely equivalent to the posterior likelihood of the NBC. The large difference lies in the fact that the NBC employs Bayes' theorem to express the posterior in term of likelihoods and priors, whereas the MaxEnt classifier directly optimises the posterior to most accurately discriminate the data. Thus, where the NBC was generative, MaxEnt is a discriminative classifier. In general, discriminative classifiers achieve higher accuracy, potentially due to their task being 'simpler' or because they do not impose strict assumptions necessary for generative models, and can take as input any form of the processed feature vector $\mathbf{X}$. However, they typically require more data, are slower to train and cannot handle missing data very well. Thus, while achieving better evaluative results, this comes at the price of less interpretable results.

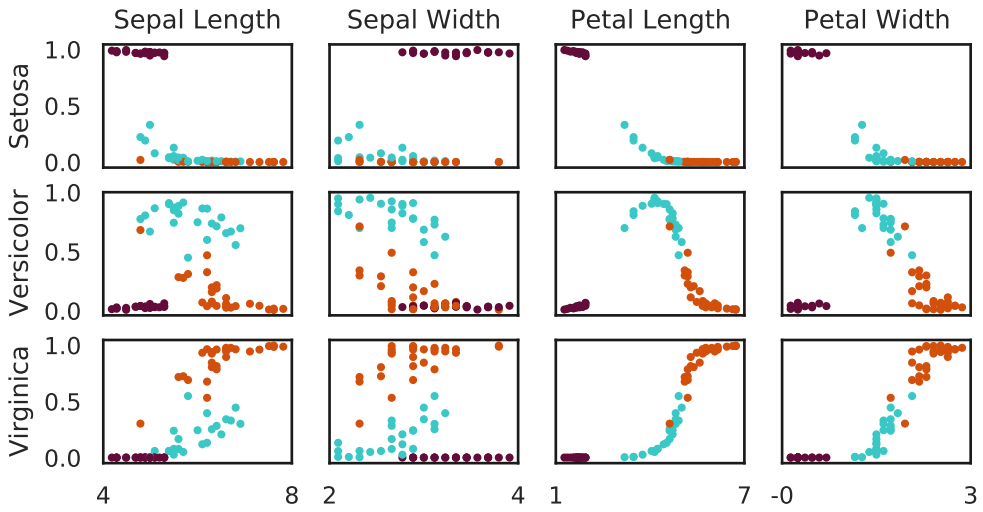

Fig. 1.6: Logistic regression predictions for each species using one of the four variables available in the Iris dataset.

Example: MaxEnt for Iris Classification
Similar to the previous example, an MaxEnt classifier is trained on the Iris dataset. The same train-test split is used for the training of this classifier, allowing for direct comparison. An accuracy of 97% was achieved, a 4% gain compared to the NBC. While odds ratios can be requested, no immediately interpretable output is generally provided. Fig.1.6 provides the classification prediction of species per every variable; note the sigmoid shape apparent in some of the plots.

1.5 Thesis Structure

At this point, the nature of probabilistic graphical models and structure data have been described, and the necessary preliminaries for connecting PGMs to structured data has been provided in some detail. This thesis aims to provide a theoretical backing for Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) within the context of PGMs and discuss some techniques for inference and parameter inference. Lastly, as a proof of concept, a limited application is provided isolated phoneme recognition using the TIMIT speech dataset.

Ch. 2 is dedicated entirely to Bayesian networks, and laying the foundation for HMMs as their dynamic, Markovian variant. Key concepts are Pearl‘s directed separation and the equivalence of a graphoid to a set of local probability tables. Ch. 3 has the same task, but instead focuses on the undirected variant, Markov random fields, again laying the foundation for CRFs. Here many concepts crucial to Bayesian networks fall away, but using factors instead of probability functions and Lauritzen‘s Markovian

Table 3: Notation used in this thesis

$\mathcal{X}$	Calligraphic uppercase for sets and data structures
$\mathbf{X}$	Bold uppercase for sets of variables and matrices
X	Uppercase for variables and vectors
x, y, z	Lowercase for assignments of variables
$x_{i,j}$	for specified values of variables/sets and elements of matrices/vectors

blanket, a similar probabilistic model can be constructed. Ch. 4 provides a brief introduction to Rabiner's seminal papers on HMMs and their influence on later literature within the field of PGMs for structured data. The basic problems are discussed and algorithmic solutions for the first 2 are presented. Since applications often require multiple latent layers, beyond just forward-backward inference, belief propagation is also briefly discussed. A distinction between MPM and MAP inference for sequences is discussed, and Viterbi decoding is provided as a good inferential method. Ch. 5 covers the hardest of the basic problems, namely parameter estimation. While many techniques are applicable, with ongoing research and innumerable available methods, Baum-Welch learning was an EM-technique specifically designed for HMMs and as such remains within the scope of this thesis. Some likelihood functions are provided for CRFs, and strategies for optimising these are discussed, but without delving into the underlying motivation. Ch. 6, as mentioned, provides an application of structured PGMs on large vocabulary phone classification. The focus there is utilising the architectures, not necessarily state-of-the-art performance. As such, the models discussed and utilised are only the simplest form of a modern classifier pipeline, but clearly show the core arguments. Finally, Ch. 7 provides a summary of discussed topics, and goes into topics not discussed within this thesis, but relevant for future research.

There exist major notational differences between the seminal sources discussed in this thesis. As this only obfuscates the meaning and derivation of necessary results, an attempt is made to retain a uniform set of symbols and nomenclature. Table 3 provides a rough taxonomy of used notation.

Chapter 2

Directed Graphical Models

2.1 Bayesian Networks

Bayesian Networks, like all PGMs, attempt to express high dimensional probability spaces in a sparsely parameterised data structure. While useful for intuition, the reasoning behind this is not entirely aesthetic or psychological. Consider a joint PDF in terms of K Bernoulli random variables (binary in values). If it is, naively, assumed that every variable is dependent on every other variable, the memory required follows $\mathcal{O}(2^N)$; this problem, the easiest of all problems dealing with discrete random variables, will become very quickly intangible.

To combat this, a BN introduces a paired data-structure to capture information inherent to these large probability spaces, but not immediately expressed in either data-structure. Let $\mathcal{B}$ denote a BN, and let $\mathcal{P}$ denote the probability function it models, and $\mathcal{G}$ denote the graph that adheres to $\mathcal{P}$, such that

$$\mathcal{B} = \{\mathcal{P}, \mathcal{G}\}. \tag{13}$$

The probability component, $\mathcal{P}$, is simply a joint probability density function as described in Ch. 1, with K variables/vertices.

Definition 1 (Bayesian Network Probability Function). *Let $\mathcal{P}$ denote a joint probability density function, comprised of a set of variables $\mathcal{V}^{\mathcal{P}}$, such that $\mathcal{P} = P(\mathcal{V}^{\mathcal{P}})$. The support of $\mathcal{P}$ matches the nodes of $\mathcal{G}$.*

Typically, the number of variables included in $\mathcal{P}$ is denoted as $|\mathcal{V}^{\mathcal{P}}| = K$. The use of Bayes' theorem, along with *a priori* knowledge of the modelled phenomenon, within the $\mathcal{P}$ component is described in Sec. 2.1.1.

The graphical component, $\mathcal{G}$, consists of a set of nodes ($\mathcal{V}$) and edges ($\mathcal{E}$), where the nodes represent the variables in $\mathcal{P}$ and the edges the causal relationships in $\mathcal{P}$. Variables that cause other variables are typically referred to as *parents*, while variables caused by others are *children*; parents cause children which may cause grand-children.

For BNs it is required that $\mathcal{G}$ is *directed* and *acyclic*. These conditions follow from the specification of $\mathcal{E}$ as *causal* relationships between the variables. Directedness follows from the assumption that no variable can simultaneously be caused by another variable while also causing the other variable. Otherwise, no child is allowed to simultaneously

be its own direct ancestor. Acyclicity follows from the assumption that no variable may cause its causator, extending the previous condition to include non-directed relationships. Otherwise, no child is allowed to be its own grandparent.

Definition 2 (Bayesian Network Graph). *Let $\mathcal{G}$ denote a Bayesian Network graph, comprised of a set of vertices and edges, such that $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Two restrictions are placed on $\mathcal{G}$:*

1. *the vertex set $\mathcal{V}$ correspond to the variable set of $\mathcal{P}$ and the edge set $\mathcal{E}$ correspond to the causal relationships in the same variable set.*
2. *no variable may be an ancestor of itself. As an implication of this restriction, all $\mathcal{G}$ must be directed and acyclic, or DAGs.*

The strict dependence relationship between $\mathcal{G}$ and $\mathcal{P}$ is discussed in Sec.2.1.2. Perhaps prematurely, the only edges present in $\mathcal{G}$ will be ones denoting parent-child relationships, corresponding at least to statistical dependence in $\mathcal{P}$. The decomposition of $\mathcal{P}$ into the sparser $\mathcal{G}$ thus follows the factorisation,

$$P(\mathbf{X}_1, \ldots, \mathbf{X}_K) = \prod_{i=1}^{K} P(\mathbf{X}_i \,|\, \mathrm{PA}(\mathbf{X}_i)). \tag{14}$$

where $\mathrm{PA}(\mathbf{X}_i)$ denotes the parents of the variables in the set $\mathbf{X}_i$.

Assuming, without loss of generality [Koller and Friedman2009], that the Bayesian network graph has a topological ordering[5] of the K variables present, implying that the parents of variable $i-1$ are in i, the probability function may be factorised as,

$$\begin{aligned} \mathcal{P} = P(\mathbf{X}_1, \ldots, \mathbf{X}_K) = & P(\mathbf{X}_n|\mathbf{X}_1, \ldots, \mathbf{X}_{K-1}) \cdot \\ & P(\mathbf{X}_{K-1}|\mathbf{X}_1, \ldots, \mathbf{X}_{n-2}) \cdot \\ & \vdots \\ & P(\mathbf{X}_2|\mathbf{X}_1) \cdot \\ & P(\mathbf{X}_1). \end{aligned} \tag{15}$$

Example: A Student's Application Letter

This example comes from Koller & Friedman [Koller and Friedman2009], and has been slightly altered to reflect the experience of a prototypical student at UCR.

Consider a senior student in the process of applying to master programs. Due to the competitive nature of the potential master programs, a glowing letter of recommendation from the professor of the relevant course will prove crucial. While participation

[5] Generally this is the way BN graphs are drawn, such that causality flows from the top of the page to the bottom

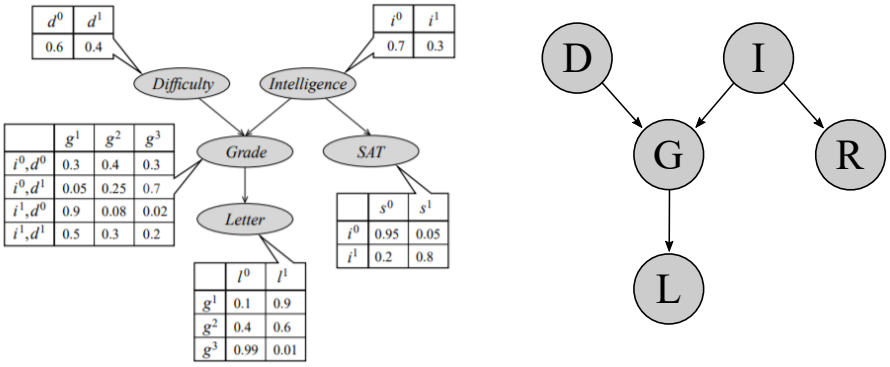

Fig. 2.1: (l) The student example BN with parameter tables attached to nodes. Taken from [Koller and Friedman2009]. (r) The same BN in the notation common to this thesis.

and motivation are undoubtedly rewarded, it is assumed that professors only look to the grade received by the student asking a for letters of recommendation; a high grade means high likelihood of letters. The received grade is in turn dependent on two factors: the difficulty of the course, and the intelligence of the student. To allow for fluke grades, the student's intelligence is further dependent on the performance in requirement courses (all of which are assumed to be of similar difficulty). A joint probability function describing such a phenomenon is depicted as

$$P(D, I, G, R, L),$$

with as support the variables course difficulty (D), student's intelligence (I), the received course grade (G), performance on requirements (R), and finally receiving the letter (L).

Using *a priori* information available in the example description -the dependencies of variables- this function may be decomposed as,

$$P(D, I, G, R, L) = P(L \mid G)P(G \mid D, I)P(R \mid I)P(D)P(I).$$

An equivalent graphical representation is provided in Fig. 2.1.

2.1.1 Reasoning Patterns

The reference to Bayes' theorem when BNs were coined by Judea Pearl, was purposeful [Pearl and Russel2011]. The probabilistic backbone of BNs is Bayes' theorem, and generally the treatment of information and probabilities follows a Bayesian philosophy. While the directed nature of the edges in $\mathcal{G}$ denotes the flow of causation, using Bayes' theorem one can easily traverse backwards along causal patterns. Again turning to Example 2.1, a student receiving a letter of recommendation suggests a high grade, which in turns suggests both an easy course and high intelligence, which suggests high per-

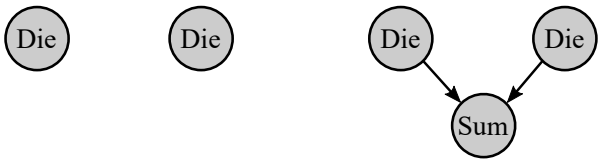

Fig. 2.2: Two fair dice are per definition independent. When including a sum of the faces as a third variable, the conditional independence of the two becomes a more tentative issue. When controlling for (conditioning on) the sum, a correlation is bound to exist between the dice.

formance on required courses. As such, from the definition of Bayes' theorem, the two reasoning patterns within Bayesian networks become

$$\begin{aligned} \textbf{Bayes Theorem} \quad & P(\text{Cause} \,|\, \text{Evidence}) = P(\text{Evidence} \,|\, \text{Cause}) \frac{P(\text{Cause})}{P(\text{Evidence})}, \\ & \text{Causal Reasoning} \quad X_{\text{Cause}} \longrightarrow X_{\text{Evidence}} \\ & \text{Evidential Reasoning} \quad X_{\text{Cause}} \longleftarrow X_{\text{Evidence}}. \end{aligned} \tag{16}$$

2.1.2 Independencies

Recall the definition of statistical independence, allowing one to decompose the joint probability function of (sets of) variables into the product of the marginal probability functions:

$$(\mathbf{X} \perp \mathbf{Y}) \iff P(\mathbf{X}, \mathbf{Y}) = P(\mathbf{X}) P(\mathbf{Y}). \tag{17}$$

The concept of conditional independence, used in the decomposition in Example 2.1, extends this definition by including a set of mediating variables (denoted by $\mathbf{Z}$). Conditioning on the mediators between two variables might alter the independence relationship between variable sets.

$$(\mathbf{X} \perp \mathbf{Y} \,|\, \mathbf{Z}) \iff P(\mathbf{X}, \mathbf{Y} \,|\, \mathbf{Z}) = P(\mathbf{X} \,|\, \mathbf{Z}) P(\mathbf{Y} \,|\, \mathbf{Z}). \tag{18}$$

The relationship between conditional and true independence follows as

$$(\mathbf{X} \perp \mathbf{Y}) = (\mathbf{X} \perp \mathbf{Y} \,|\, \varnothing). \tag{19}$$

In graphical depictions of Bayesian networks, independence is typically easy to spot; the first set of variables must be totally separated (i.e. no arrows exist between *any* of the

variables) from the second set. For most applications, this is trivial and totally uninteresting. The inclusion of totally independent variables provides no additional information for the model and only provide clutter. To relate it to more applied statistical methods, estimating the relationship of two totally unrelated variables is totally uninteresting unless the aim is to prove statistical independence.

Far more interesting will be the conditional independencies present within the model. In Example 2.1, the difficulty of the course indirectly influences the letter through lowering the expected grade of the student, and are therefore not independent. However, if the grade of the student is known and held constant (conditioned upon), the decision of the professor to provide a letter of recommendation is *no longer* influenced by the difficulty of the course; the information that could be provided by the difficulty of the course is already contained in the observation of the grade. This allows one to decompose the the full joint probability function into a far more sparse representation as in Fig. 2.1. Another example of the difference between true and conditional independence is provided in Fig. 2.2.

Let the $\mathcal{I}$ set represent the complete set of independencies present in the joint probability function $\mathcal{P}$, such that all $(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}) \subseteq \mathcal{I}(\mathcal{P})$ for all variants of $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$. The set $\mathcal{I}$ will be referred to as the independence set.

Definition 3 (Independence Set). *Let $\mathcal{P}$ be a distribution over a set of random variables $\mathbf{X}$. The set of independence relationships of the form $(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$ that hold in $\mathcal{P}$ is defined as $\mathcal{I}(\mathcal{P})$.*

Generally, one may decompose the joint probability according to the independencies present in the independence set $\mathcal{I}(\mathcal{P})$, extending the result defining conditional independence as Eq. 19. Assuming all conditionally dependent variables are present in the first i variables when conditioned on the variable set $\mathbf{Z}$, the full decomposition follows as,

$$P(X_1, \ldots, X_K) = \prod P(X_1, \ldots, X_i \mid \mathbf{Z}) : \mathcal{I}(\mathcal{P}). \tag{20}$$

I-Maps Not all DAG whose nodes reflect the variables in $\mathcal{P}$ are necessarily Bayesian networks. For the definition of $\mathcal{B}$ to hold as in Eq. 13, it is crucial that the vertices in $\mathcal{G}$ also reflect the causal relationships present in $\mathcal{P}$. In terms of independencies, ideally a Bayesian network graph $\mathcal{G}$ adheres to all the independence assertions present in $\mathcal{I}(\mathcal{P})$. Given the initial goal of a sparse parameterisation of high-dimensional probability spaces, requiring $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{P})$ may be counterproductive. Instead, all that is required of graphical models is for any visual independencies present in $\mathcal{G}$ to also be present in $\mathcal{P}$. Formally, this is referred to as an *I-map*.

Definition 4 (I-map). *If the graph structure $\mathcal{G}$ is a subset of the independence set of $\mathcal{P}$, such that $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(\mathcal{P})$, then $\mathcal{G}$ is an I-map of $\mathcal{P}$. Otherwise, the independence relations contained within $\mathcal{G}$ are satisfied by $\mathcal{P}$.*

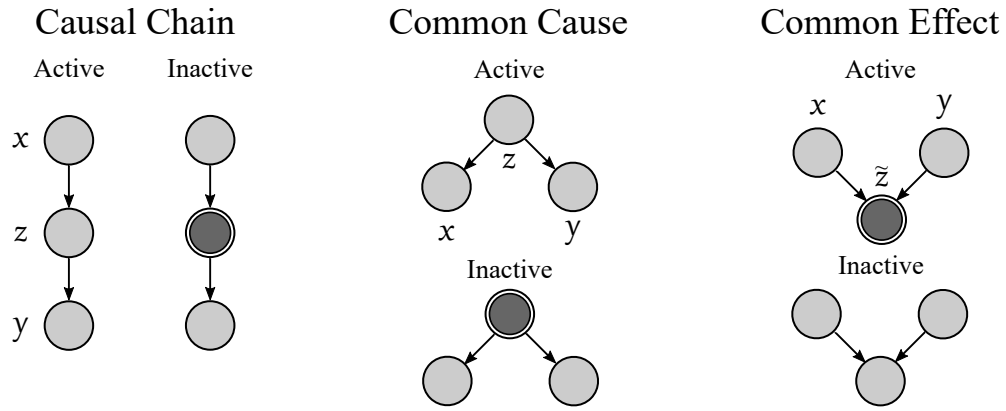

Fig. 2.3: Given two sets of variables **X** and **Y**, along with a set of mediators **Z**, three possible structures can exist within DAGs. The double lined nodes represent conditioning (i.e. $P(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z})$). The 'Causal Chain' may also be inverted via Bayes' theorem, and the 'Common Effect' is typically referred to as v-structures.

While close, the relationship between a Bayesian network probability function and its graphical representation needs to be stricter. To make certain the graphical representation does not mislead the audience as to the independencies present $\mathcal{G}$ must at least be a minimal I-map of $\mathcal{P}$; no vertex may be deleted from $\mathcal{G}$ without violating the $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(\mathcal{P})$. Otherwise, $\mathcal{G}$ may not introduce independencies not present in $\mathcal{I}(\mathcal{P})$.

D-Separation While the importance of faithfully depicting independencies within $\mathcal{P}$ via $\mathcal{G}$ is hopefully clear, the question remains as to how it is guaranteed that this is the case. When does it become clear that the nodes and vertices in $\mathcal{G}$ are an accurate representation of $\mathcal{P}$?

The criterion coined by Pearl [Pearl2014] to guarantee conditional independence in $\mathcal{G}$ is that of *directed separation*, typically just *d-sep*. Rather than being concerned with strictly causal relationships, d-sep focuses on the influence of additional evidence as per Bayes' theorem over chains of causally related variables. A directly related topic would be that of correlation with control for a set of variables outside of the investigation of the effect of X on Y.

Over graphs with two interacting variable sets (cause and effect) and mediating variables that provide evidence, three non-trivial and distinct scenarios exist, displayed in Fig. 2.3. Mediating variables are defined as any variables that lie on the path $X \to Y$, here Z. If influence between variables can exist, this is labelled an *active trail*, with an inactive trail denoting the opposite. Conditioning on, controlling for and holding X constant all denote the $(X \perp Y \mid Z)$ operation; full information of Z is provided such that it can no longer be considered a random variable (it is equivalent to stating $Z = z$

with certainty). The following structures have also been graphically depicted in Fig. 2.3. Note that the following uses variables, but the same reasoning and validity of results exist when using sets of variables instead.

Causal Chain Here the relationship $X \to Z \to Y$ or $X \leftarrow Z \leftarrow Y$ exists. Note that these are equivalent when applying Bayes' theorem to enable evidential rather than causal reasoning. The edge case where $Z = \varnothing$ is trivial; no matter the information provided by any other variable in the Bayesian network, there will always exists an influence between X and Y.

The cases where $Z \neq \varnothing$, where no direct relationship exists between X and Y[6] have active trails unless Z is conditioned on. If the variable Z is held constant, no probabilistic link remains between variables X and Y, and as such no additional information in either will exert influence over ones understanding in the other.

Common Cause Here the relationship $X \leftarrow Z \to Y$ exists. The edge case where $Z = \varnothing$ is again trivial; since no ancestral relationship exists, these variables may be assumed to be statistically independent, assuming no pedigree exists. Furthermore, so long as the variable Z is not conditioned on, an active trail exists. If the common cause is held constant, no probabilistic link exists between the cause and effect, and as such the trail is made inactive.

Common Effect Here the relationship $X \to Z \leftarrow Y$, commonly referred to as a v-structure. Here the pattern of the previous graphical relationships is ruined; an active trail can only exist between X and Y if the mediator Z, or any of its children Z, is conditioned on. Note that variable X and Y are totally statistically independent, save for their relationship through Z. If the value of Z is unknown, or left as a random variable, no additional information in X would lead to Y. An attempt at an intuitive explanation is made in the following example.

Example: Directed Separation in Family Trees
The genetic information within a family, depicted graphically using a family tree, is a great example for understanding directed separation and Bayesian networks in general. Not only were the first causal diagrams employed by Sewall Wright, as suggested by Judea Pearl [Pearl and Mackenzie2018], but the language of kinship naturally extends to concepts in conditional independence (ancestors, parents, children). Fig. 2.4 serves as a guide for this example.

First, the generations are layered such that the graph is a topological, or causal, ordering of the variables. Since no causal relationships exist between siblings and cou-

[6] This would simply lead back to the case where $Z = \varnothing$, as influence between X and Y is guaranteed regardless of Z.

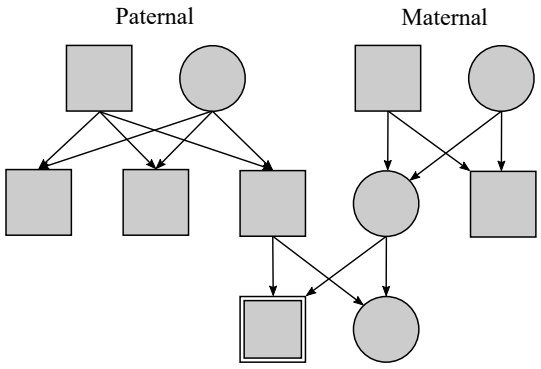

Fig. 2.4: The family tree used in the family tree example. The males are represented by squares, while the females are represented by circles. The example works from the perspective of the male in the 3rd generation, in this case the double lined square.

ples, these are displayed on the same level. Ancestors are always placed above their pedigree.

Second, between parents and children, there always exists a causal chain relationship. Between grandparents and (grand)children, this relationship is mediated by the parents. Using the notion of directed separation, no amount of additional information from the grandparents will influence ones understanding of the children if the parents are held constant. Keeping with the genetic information story, checking the grandparents DNA will be useless for increasing information about the child when the DNA of the parents is already known; there exists no method whereby the grandparents may introduce additional genetic information outside through their child.

Third, between siblings there exists a common cause relationship. Additional information in a sibling will influence ones understanding of the child through the shared parents. Again, if the genetic information of the parents is already known, the sibling cannot exert additional influence over their sibling.

Fourth, between the paternal and maternal families, no (reasonable) genetic relationship exists until genetic information of the children, that serve as the union of the families, is taken into account. This is an example of the common effect structure. Note that if information of a fourth generation (stemming from the children in Fig. 2.4) is controlled for, this will create a causal chain, and will still allow for influence between families.

General Case For the general case, one can establish an active trail between X and Y, given a set of observed variables O and a set of mediators Z, if:

1. any and all v-structures have either Z or any of its descendants, $\widetilde{Z}$, within the set of observed variables O.

2. no other variable in Z is also in O.

Definition 5 (Directed Separation). *For the variable sets* $\mathbf{X}, \mathbf{Y}$ *and the observed variable set* $\mathbf{Z}$ *in* $\mathcal{G}$*, we declare* $\mathbf{X}$ *and* $\mathbf{Y}$ *d-separated if there are no active trails between any node* $X \in \mathbf{X}$ *and* $Y \in \mathbf{Y}$ *given* $\mathbf{Z}$*.*
This is denoted by $d\text{-}sep_{\mathcal{G}}(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$*. The independencies set of* G *can be rewritten to take into account the d-separated variables:*

$$\mathcal{I}(\mathcal{G}) = \{(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}) : d\text{-}sep_{\mathcal{G}}(\mathbf{X}, \mathbf{Y}|\mathbf{Z})\}. \tag{21}$$

From the definition of directed separation, Pearl comes with the following theorem, crucial for the relationship between $\mathcal{G}$ and $\mathcal{P}$. A particularly important implication of this theorem is the ability to replace conditional independence in probability function by directed separation in graphs.

Theorem 1 (Probabilistic Implications of Direct Separation). *If sets* $\mathbf{X}$ *and* $\mathbf{Y}$ *are d-sep by* $\mathbf{Z}$ *in a DAG* $\mathcal{G}$*, then* $\mathbf{X}$ *is independent of* $\mathbf{Y}$ *conditional on* $\mathbf{Z}$ *in every distribution compatible with* $\mathcal{G}$*. Conversely, if* $\mathbf{X}$ *and* $\mathbf{Y}$ *are not d-sep by* $\mathbf{Z}$ *in a DAG* $\mathcal{G}$*, then* $\mathcal{X}$ *and* $\mathcal{Y}$ *are dependent conditional on* $\mathcal{Z}$ *in at least one distribution compatible with* $\mathcal{G}$*.*

Definition 6 (Factorisation). *If the graph structure* $\mathcal{G}$ *is an I-map for the distribution* $\mathcal{P}$ *over the variables* $X_1, \ldots, X_n$*, such that* $\mathcal{P}$ *can be expressed as the product*

$$P(X) = \prod_{i=1}^{n} P(X_i|PA(X_i)) \tag{22}$$

it is said that $\mathcal{P}$ *factorises according to* $\mathcal{G}$*. Here* $PA(X_i)$ *denote the parents, or the direct causal predecessors, of the variable* X_i*. The individual terms* $PA(X_i)$ *are considered the conditional or local probability functions that parameterise the variable* X_i*.*

Definition 7 (Bayesian Network). *A Bayesian network is the pair* $\mathcal{B} = (\mathcal{G}, \mathcal{P})$ *where* $\mathcal{P}$ *factorises over* $\mathcal{G}$*, and where* $\mathcal{P}$ *is specified as the joint probability function, generated by the set of local probability functions associated with the nodes in* $\mathcal{G}$*.*

2.2 Dynamic Bayesian Networks

Before developing temporal or dynamic Bayesian networks, the highly useful plate-model notation is introduced. Generally when constructing Bayesian networks, the variables included in the full specification represent attributes of a common identifying (or indexing) variable. For each value of the set of variables included in the plate model are instantiated to take values logical for the indexed variables. Not only does this allow

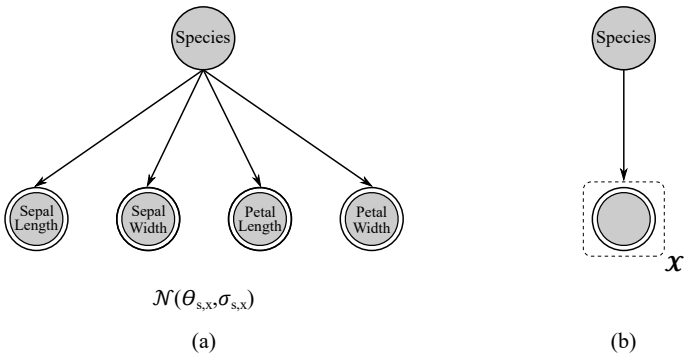

Fig. 2.5: A Naive Bayes classifier as described in Ch. 1, now in the notation of Bayesian networks. Sub-figure (a) presents it exactly as described in the previous example (a multinomial 'Species' variable causing 4 continuous measurement variables), sub-figure (b) as a plate model.

for significant notational clarity when working with complex or temporal BNs, it also allows for creating structures of objects with shared parameters. Each instantiation of an object is indexed, and produces a single *ground Bayesian network*, which in turn can be influenced by variables outside of the plate.

Example: Naive Bayes Revisited
Recall the example of a generative classifier, the Naive Bayes model, applied to iris species classification. Unknown to us at the time, the NBC happens to be a very simple example of a Bayesian network. A single class variable has a causal relationship between the 4 measurement variables. It had previously been stipulated that the Naive Bayes assumption assumes statistical independence between the observations, $(x_i \perp x_j)$. However, within the scope of Bayesian networks, this is no longer relevant. Instead it may be specified that the observations are *conditionally* independent given the class variable, allowing one to invoke the directed separation criterion and construct a graph as in Fig. 2.5,

$$(x_i \perp x_j \,|\, y) \implies \text{d-sep}(x_i, x_j \,|\, y).$$

More than just a BN, the NBC is also a simple example of a plate model. Each iris recording (all 150) instantiates a specific set of measurement variables. Where Fig. 2.5(a) shows the general network for 1 such iris and Fig. 2.5(b) explicitly records the plate nature of the model.

2.2.1 Markov Chains

A model remarkably similar to Bayesian networks, beyond just the naming conventions, is the Markov chain. In a sense, the Markov chain is a form of PGM, consisting of a

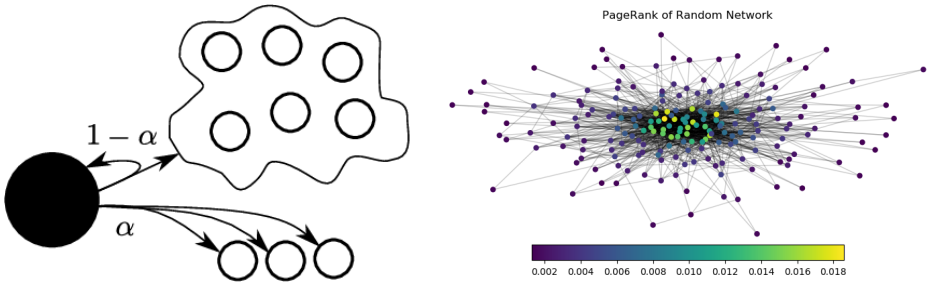

Fig. 2.6: Left: A graphical depiction of the PageRank algorithm. The user is in the large node with a certain probability, and moves to a linked page with probability α and a random node in the web with probability $1 - \alpha$. Taken from [Gleich and Saunders2009]. **Right**: a PageRank solution for a random graph. The colour provides the likelihood of being in page i at time $t \to \infty$.

graph (either directed or undirected) and a multinomial probability distribution encoding transitions across nodes. However, rather than using probability distributions, typically transitions are recorded using stochastic matrices. The fundamental underlying equation of a Markov chain is simply,

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t$$

Here the vectors $\mathbf{x}$ denote state vectors, such that every entry x_i gives the probability of being in state i. Note the t indexing variable. The Markov chain is separated by discrete time-steps, incrementing t by 1. As usual, the symbol π is reserved for probabilities for $t = 1$. The matrix $\mathbf{A}$ gives the transition matrix, such that each entry $a_{i,j}$ gives the probability of moving from state i to state j, $P(j_{t+1} \mid i_t)$. It is required that $\mathbf{A}$ is a stochastic matrix; every column may be seen as an independent multinomial distribution, abiding to the basic properties of probabilities as described in definition 1:

$$\mathbf{A} = \begin{pmatrix} P(Q_1 \mid Q_1) & P(Q_1 \mid Q_2) & \dots & P(Q_1 \mid Q_{K-1}) & P(Q_1 \mid Q_K) \\ P(Q_2 \mid Q_1) & P(Q_2 \mid Q_2) & \dots & P(Q_2 \mid Q_{K-1}) & P(Q_2 \mid Q_K) \\ \vdots & \ddots & & & \\ P(Q_{K-1} \mid Q_1) & P(Q_{K-1} \mid Q_2) & \dots & P(Q_{K-1} \mid Q_{K-1}) & P(Q_{K-1} \mid Q_K) \\ P(Q_K \mid Q_1) & P(Q_K \mid Q_2) & \dots & P(Q_K \mid Q_{K-1}) & P(Q_K \mid Q_K) \end{pmatrix}. \tag{23}$$

In probabilistic notation, the equivalent would be,

$$P(j_{t+1}) = P(j_{t+1} \mid i_t)P(i_t).$$

Much like Bayesian networks generating sparse depictions of high-dimensional probability spaces due to leveraging directed separation, Markov chains need not be fully connected; many graph topologies are possible. In fact, the majority of applications for Markov chains see very sparse transition matrices. Ergodic transition matrices are generally the exception, as these require all states to be reachable by all others, given infinite time duration. Left-to-right transitions instead require another index, such that a transition can only occur into the current state or the 'next'.

Example: PageRank
On the worldwideweb, drawn as an incredibly dense and intricate graph with hyperlinks being the edges between the vertices that represent web pages, which sites are important for querying? As it turns out, this was the trillion-dollar question that Page, Brin, Motwani and Winograd answered [Page et al.1999]. PageRank, the algorithm behind Google's search, approaches the problem by composing a Markov Chain over the graph, such that the columns represent the sum normalised outgoing links. To avoid dangling nodes, a small correction is made to allow for 'teleportation' of a user to another random node (effectively making the model both sparse and ergodic). The importance vector $\mathbf{g}$ is defined by the steady-state matrix,

$$\mathbf{g} = \lim_{t\to\infty} \mathbf{A}^t \pi.$$

A graphical example of PageRank is provided in Fig. 2.6.

Regardless of topology or application, the power of a Markov chain to probabilistically model a series of discrete steps is hopefully clear. In Markov's work on the namesake chains, two assumptions were made. The first[7] assumption simplifies the temporal relationships between states, such that each is dependent only on the directly influencing state prior to it. The second assumption simplifies models over long stretches of time, requiring a constant parameterisation of the transition matrix. More formally, these assumptions are given by the definitions below,

Definition 8 (Markovian Assumption). *Given a Bayesian network, such that a joint probability distribution can be decomposed using inferred conditional independence relationships, and assuming discrete time steps, all necessary information for future prediction can be modelled using only dependencies between the present and the immediate past. The PDF decomposes as*

$$P(\mathbf{X}_T) = P(\mathbf{X}_1)\prod_{t=2}^{T} P(\mathbf{X}_t|\mathbf{X}_{t-1}) \tag{24}$$

The Markovian assumption in conditional independence relationship notation yields,

$$\mathcal{P} : (\mathbf{X}_{t+1} \perp \mathbf{X}_{t-1}|\mathbf{X}_t); \text{d-sep}_{\mathcal{G}}(\mathbf{X}_{t+1}, \mathbf{X}_{t-1}|\mathbf{X}_t). \tag{25}$$

The Markovian assumption can extend to incorporate greater time dependencies. The above definition is only the first-order, higher-order Markovian assumptions follow as,

$$P(\mathbf{X}) = P(\mathbf{X}_1)\prod_{t=2}^{T} P(\mathbf{X}_t|\mathbf{X}_{t-1,\ldots,}\mathbf{X}_{t-n}). \tag{26}$$

[7] Also named after A.A. Markov, which in turn lead to many models within this thesis *also* being named after him.

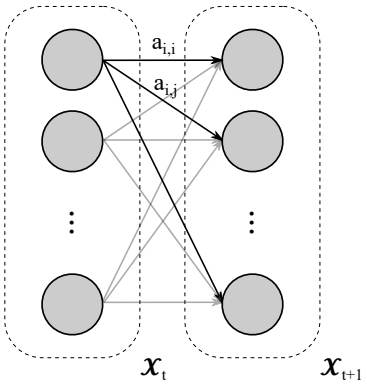

Fig. 2.7: A 2 time-slice dynamic Bayesian network. Self persistency is denoted by $a_{i,i}$, time-persistency by $a_{i,j}$.

Definition 9 (Time-Invariance). *Given a Markov model with state transition matrix A, there is no dependency of the transition probabilities on time. Processes with constant transition probabilities are referred to as time-invariant,homogeneous or stationary.*

Discussions regarding the Markovian models limits itself entirely to time-invariant processes, and will explicitly state when deviating from the first-order Markovian assumption.

The above definitions explicitly link a Markov chain to a Bayesian network, further implying that the matrix and probabilistic notation of either is equivalent. A Markov chain may thus be defined as,

$$\begin{aligned} P(\mathbf{X}_T) &= P(\mathbf{X}_1)\prod_{t=2}^{T} P(\mathbf{X}_t|\mathbf{X}_{t-1}) \\ P(\mathbf{X}_T) &= \pi\prod_{t=2}^{T} a_{i,j} \end{aligned} \tag{27}$$

However, the graphical component of BNs were explicitly defined to be DAG structures, Markov chains were not. For practical uses, Markov chains are typically already directed (their transition matrix is not symmetric). To ensure acyclicity, one may construct the Markov chain as a plated 2-time slice dynamic Bayesian network. While the name is wordy, the graphical structure is fairly simple. The plates are time-indexed, such that each plate contains the state vector for a single time period, $\mathbf{x}_t$. The edges between plates are time-persistencies. Fig. 2.7 gives a graphical representation of such a model.

Latent Variables in Markov Models Sequential data typically requires separating the observation from the state of variables that produce the observation. Typically, it is not the observation that is of interest, but rather the generating process. Unlike the observations, naturally produced and easily recorded, the state variables are typically hidden or latent and require inferring from the sequence of observations. Many, many

natural phenomena are modelled using latent variable models. The difference between a target variable kept hidden during evaluation, like in the iris classification examples, and a latent variable, is that the latent variable will never be visible.

Example: Parts of Speech Tagging
Markov models incorporating latent variables are very common in the field of Natural Language Processing. A particularly well known problem is that of parts-speech-tagging; given a sentence, what is the syntactical function of each word within the sentence? While the word, or observation, is known, the syntatic part-of-speech (POS) is not. If it is assumed the POS is the generating variable for the observed word, POS-tagging may require latent variable model.

Taking into account only the word and its dictionary definition will produce ambiguous dependent variables. For example, the word 'back' might be a noun (someone's back), and adverb ('back there') or part of a verb phrase ('back-up'). However, by assuming a first-order Markov assumption, such that each POS is dependent on its preceding POS, models already achieves much greater predictive power. Thus, this serves as a good example of a Markovian process with latent variables.

The Bayesian network that satisfies the process described above can be generally captured by a template graphical model as in Fig. 2.7, however the nodes are now hidden to the developer. Each observation, now labelled O to emphasize that these are fully observed, is dependent only on the latent variables, or states, now Q to emphasize that these are hidden. Another variable is needed to capture the relationship between the hidden states and the observation. Two key independencies are noted,

$$(Q_{t+1} \perp Q_{t-1}|Q_t) \quad \& \quad (O_t \perp O_{t+1}|Q_t). \tag{28}$$

As such, the transition matrix now applies to the movements between $Q_{i,t} \rightarrow Q_{j,t+1}$, but no longer captures the information provided by an observation. This requires incorporation of another term: the emission matrix. Similar to the transition matrix A, the emission matrix B captures the movement between $Q_{i,t} \rightarrow O_{j,t}$. Otherwise, each element of the emission matrix captures the conditional probability of an observation given the hidden state value. The $N \times K$ emission matrix follows as,

$$\mathbf{B} = \begin{pmatrix} P(o_1 \,|\, Q_1) & P(o_1 \,|\, Q_2) & \dots & P(o_1 \,|\, o_{K-1}) & P(o_1 \,|\, Q_K) \\ P(o_2 \,|\, Q_1) & P(o_2 \,|\, Q_2) & \dots & P(o_2 \,|\, Q_{K-1}) & P(o_2 \,|\, Q_K) \\ \vdots & \ddots & & & \\ P(o_{N-1} \,|\, Q_1) & P(o_{N-1} \,|\, Q_2) & \dots & P(o_{N-1} \,|\, Q_{K-1}) & P(o_{N-1} \,|\, Q_K) \\ P(o_N \,|\, Q_1) & P(o_N \,|\, Q_2) & \dots & P(o_N \,|\, o_{K-1}) & P(o_N \,|\, Q_K) \end{pmatrix}. \tag{29}$$

Markovian models with a latent states generating observations, especially those defined by a transition and emission matrix, are called Hidden Markov Models.

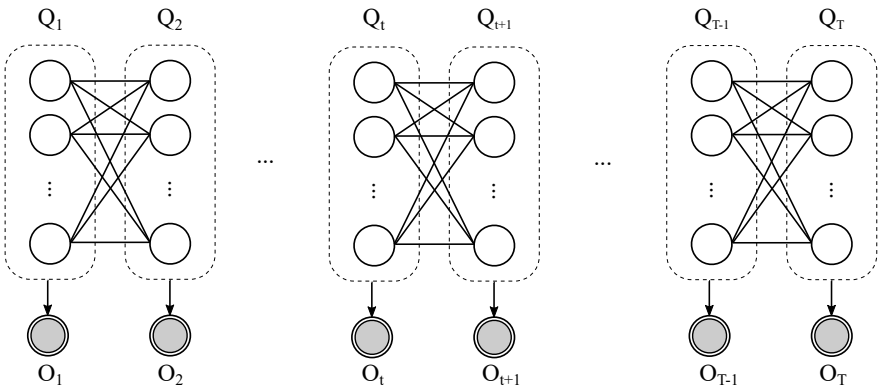

Fig. 2.8: A Hidden Markov Model depicted as a plate Bayesian network, where the uncoloured nodes are hidden and the grey nodes are observations. Directions of edges have been omitted as these clutter the graph considerably, however, transitions are limited entirely to $t \to t+1$.

2.2.2 Hidden Markov Models

The simplest of all possible state-observation models [Koller and Friedman2009], the first-order HMM already manages to accurately model a large domain of natural phenomena. Due to sparse transitions being typical, HMMs are typically depicted as Markov chains (i.e. directed, but cyclic graphs), but as discussed before this can easily be converted to a dynamic BN, see for example Fig. 2.8.

Definition 10 (Hidden Markov Model). *Let λ denote an instantiation of a hidden Markov model, A an $K \times K$ transition matrix, B a $N \times K$ emission matrix and π a $K \times 1$ column vector, such that $\lambda = \{A, B, \pi\}$. The relationship between hidden state transition and observation emission is captured as,*

$$\begin{aligned} P(\mathbf{O}, \mathbf{Q}) &= P(\mathbf{Q}_{Init}) \prod_{\forall t} P(\mathbf{Q}_t \,|\, \mathbf{Q}_{t-1}) P(o_t \,|\, \mathbf{Q}_t) \\ &= \pi \prod_t A_{i,j} B_{o,i} \end{aligned} \tag{30}$$

Every instant in time has K possible states being active and one observation out of the sequence $\mathcal{O}$.

It is given that a HMM is also a dynamic Bayesian network, where the independencies captured are

$$(\mathbf{Q}_{t+1} \perp \mathbf{Q}_{t-1} \,|\, \mathbf{Q}_t), (O_t \perp \mathbf{Q}_{t-1} \,|\, \mathbf{Q}_t)$$

While the above presented independencies are most important, the graphs that describe HMMs implicitly allow for many more to exist. For a more complete list, see [Bishop2006].

While the time steps are necessarily discrete, the emissions need not be. Like the Ch. 1 example of a Naive Bayes classifier with normal distributions for each feature, HMMs may also include general distributions as the entries for the emission matrix. For many applications, like speech recognition, modelling requires continuous distributions like the Gaussian mixture of models (without proof, these can be used to model any arbitrary distribution as a mixture of many Gaussians). Regardless, a common introductory example (and one that will be used often within the context of this thesis), is the 'Balls in Urns'. While most likely popularised for HMMs by Rabiner [Rabiner1989] and Juang [Rabiner and Juang1986], these kinds of problems have been around for much longer.

Example: Balls in Containers

Consider a room, totally hidden to observers, containing K urns with N distinct colours of balls. The urns contain different proportions of colours, but these proportions have been recorded when the room was originally built and stored in an $N \times K$ emissions matrix $\mathbf{B} = \{b_1, \ldots, b_N\}; b_i = \{P(colour_1), \ldots, P(colour_K)\}$.

A robotic arm selects an urn, draws a ball from that urn and places it on a conveyor belt for a group of outsiders recording each emission from the room in an $1 \times T$ observation vector $O = \{o_1, \ldots, o_T\}$. The urn selection process is not uniformly random, with the choice being largely dependent on the previous urn, however, the preference for the urn-to-urn transition has been recorded in a $K \times K$ transition matrix $\mathbf{A} = \{a_1, \ldots, a_N\}; a_i = \{a_{i\to 1}, \ldots, a_{i\to N}\}$.

The HMM model λ that models this phenomenon may be characterised as,

$$\lambda = (\underset{[K\times K]}{\mathbf{A}}, \underset{[N\times K]}{\mathbf{B}}, \underset{[K\times 1]}{\pi})$$

Where the individual components of the HMM have the form of,

$$\mathbf{A} = \begin{matrix} \\ \text{Urn} \\ \text{Barrel} \\ \text{Vase} \end{matrix} \begin{matrix} \begin{matrix} \text{Urn} & \text{Barrel} & \text{Vase} \end{matrix} \\ \begin{pmatrix} 0.7 & 0.1 & 0.2 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.1 & 0.6 \end{pmatrix} \end{matrix}, \quad \mathbf{B}^{T\,a} = \begin{matrix} \\ \text{Red} \\ \text{Green} \\ \text{Blue} \\ \text{Yellow} \\ \text{Orange} \end{matrix} \begin{matrix} \begin{matrix} \text{Urn} & \text{Barrel} & \text{Vase} \end{matrix} \\ \begin{pmatrix} 0.7 & 0.1 & 0.2 \\ 0.1 & 0.5 & 0.0 \\ 0.2 & 0.1 & 0.4 \\ 0.3 & 0.1 & 0.2 \\ 0.2 & 0.15 & 0.0 \\ 0.2 & 0.15 & 0.4 \end{pmatrix} \end{matrix} \quad \pi = \begin{matrix} \\ \text{Urn} \\ \text{Barrel} \\ \text{Vase} \end{matrix} \begin{pmatrix} 0.25 \\ 0.5 \\ 0.25 \end{pmatrix}$$

[a] Transposed here for display purposes.

[a] Transposed here for display purposes.

Chapter Summary

This chapter, the first dealing explicitly with the content around which this thesis is centred, provides the basics for describing Bayesian networks. The key results by Pearl are introduced and further concepts are explained using a framework common to many standard texts on PGMs. Dynamic Bayesian networks are briefly discussed, although their purpose was explicitly to derive the form and underlying equations for developing Hidden Markov models. The next chapter discusses the undirected PGM equivalent, Markov random fields and conditional random fields. At this point it is already possible to skip ahead to the inference chapter, as the problems and solutions presented there are applicable to either form of PGM.

Chapter 3

Undirected Graphical Models

3.1 Markov Random Fields

So far the discussion of PGMs has limited itself to strictly DAGs with probability distributions factorising according to it, not entirely by accident. Bayesian networks are capable of intuitively capturing the manner in which one thinks about real world phenomena, relating variables or states to causal patterns that allow for forward and backward reasoning. Within the development of PGMs for AI purposes, these also represent the first to receive widespread attention, potentially for Pearl's promise of being able to model human-like reasoning that artificial neural networks might not be [Pearl and Mackenzie2018].

Historically, however, and when considering all fields that make use of PGMs without directly referencing to them, it was the undirected variant that came first. Generally, there are many more situations where it is hard or impossible to assign a causal pattern of influence between any two occurrences. A very early ancestor of modern techniques is the Ising model, where atoms are modelled with a spin and able to interact with other atoms in some neighbourhood to produce macroscopic magnetisation. It would be silly to consider a single atom as the root cause for magnetising regions around it when it is in turn also influenced by its neighbourhood. Within the social sciences, where the sentence 'Correlation does not imply causation' might as well be considered dogma, certain forms of non-BN PGMs are prominent for complex interdependent phenomena. As an example, consider structural equation modelling, a technique that found its invention in the work of Sewall Wright, see Fig. 1.1.

Thus, while such phenomena do exist, it remains a challenge to model them with the machinery currently set out within this thesis. The conditions of directed acyclicity of the graphs and the capturing of independencies via Pearl's directed separation property were necessary to assume factorisation of a graph according to a probability distribution. It takes little consideration to note that none of these properties hold for an undirected graphical model. An example of a phenomenon modelled by an undirected graphical model, and its comparison to directed variants follows.

Example: A Student's Application Letter, Undirected
Consider again the senior student in their quest for letters of recommendations. The undirected graphical model that is equivalent to the probability distribution and the I-map defined in the example is provided in Fig. 3.1. Note that compared to the directed

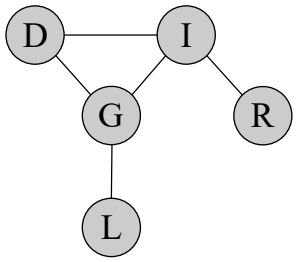

Fig. 3.1: The student example, but in undirected graphical notation.

variant, besides the loss of arrowheads, an additional edge is generated between the course difficulty and intelligence.

The edges now no longer hold causal meaning, rather these depict something akin to correlation. While there is no causal reason for a student's intelligence and the difficulty of the course to be related, given a student's grades a correlation between the variables will exist (recall the small example of correlated dice and their sum in the discussion of v-structures). The graph is definitely no longer DAG and it is hard to make a case for directed separation remaining intact either.

Without causality it becomes hard to assign quantitative values to the occurrence of variable values. For example, given that it becomes known the student received a letter of recommendation L, what can one say about their intelligence I? Bayes' theorem no longer applies with a cycle between the variables D, I and G.

The example sketches the problem in modelling with undirected networks. The answer to these concerns are Markov Random Fields (MRFs), described using normalised Gibbs functions rather than probability distributions. These functions are a composite of two products; a set of factors ϕ reminiscent of the local conditional density functions use with BNs, and a normalisation constant $Z(\mathbf{X})$ (historically known as the partition function). While these are an abstracted version of probability theory as discussed in Ch. 1, these can be leveraged to allow for pseudo-probabilistic inference and learning. The remnants of this sections lays out the definitions for factor and Gibbs functions before revisiting the topic of encoding independencies in graphs. This discussion follows, in broad strokes, the one presented in *Probabilistic Graphical Models* [Koller and Friedman2009].

3.1.1 Factors

Factors, in name especially, come from the field of applied statistics. Globally, these are defined as a discrete variable whose levels are used to discriminate between experimental groups. These are not to be confused with *factor nodes*, used in Ch. 4. The values of factors are non-probabilistic, for example count data as used extensively in log-linear

A	B	$\psi(A,B)$
0	0	57
1	0	82
0	1	34
1	1	74

$\times$

B	C	$\psi(B,C)$
0	0	2
1	0	13
0	1	58
1	1	40

$\Rightarrow$

A	B	C	$\psi(A,B,C)$	P(A,B,C)
0	0	0	$57{\cdot}2 = 114$	0.008
1	0	0	$82{\cdot}2 = 166$	0.012
0	1	0	$34{\cdot}13 = 442$	0.031
1	1	0	$74{\cdot}13 = 962$	0.068
0	0	1	$57{\cdot}58 = 3306$	0.234
1	0	1	$82{\cdot}58 = 4814$	0.341
0	1	1	$34{\cdot}40 = 1360$	0.096
1	1	1	$74{\cdot}40 = 2960$	0.210
		$\sum$	14124	1.0

Fig. 3.2: A factor product between binary valued variables, showing both the output factor function and the Gibbs distribution it forms.

models and logistic regression. In the context of undirected graphical models, values of factor functions are referred to as affinities, potentials, energies or a variety of other names [Koller and Friedman2009]. Regardless of name or context, the value of a factor denotes propensity for occurrence, like probabilities, by assigning an arbitrary but typically non-negative number to a combination of variable assignments, unlike probabilities.

More formally, factors are functions that map all distinct combinations of the variable assignments within its variables scope to the positive reals. In this framework, probability functions are also factors, only differing by an added stipulation of bounding values between the range [0,1] and properties for their addition and multiplication (see Definition 1).

Definition 1 (Factor). *Let* $\mathbf{D}$ *be a set of random variables. The factor* Φ *is a function that maps every combination of values of the set* $\mathbf{D}$ *to the positive reals,*

$$\Phi : val(\mathbf{D}) \to \mathbb{R}^+.$$

The set of variables included in $\mathbf{D}$ *is the scope of the factor,*

$$scope(\Phi(\mathbf{D})) = \mathbf{D}.$$

A common way of depicting factors is in tabular form, perhaps stemming from the overlap with log-linear models, and these will also be used here to elucidate a variety of factor operations. A number of operations (the product chain of probability, marginalisation and conditioning) where discussed in the introductory chapter of this thesis, and have proved crucial for developing Bayesian networks within a consistent framework. Factor functions have an equivalent for each of these as well, merely brought to a higher level of abstraction.

– **Factor Products**: join two factors in terms of disjoint variables together in a common function. Consider three disjoint variables A, B, C and two factors defined in

$\psi(A, B = 1, C) =$

A	B	C	$\psi(A,B,C)$	P(A,B,C)
0	0	0	114	0.008
1	0	0	166	0.012
0	1	0	442	0.031
1	1	0	962	0.068
0	0	1	3306	0.234
1	0	1	4814	0.341
0	1	1	1360	0.096
1	1	1	2960	0.210
			$\sum$ 14124	1.0

Fig. 3.3: A factor product between binary valued variables, showing both the output factor function and the Gibbs distribution it forms.

terms of the combination of two of these variables, $\psi(A, B)$ and $\psi(B, C)$. The multiplication of these factors, or generally their product, joins the factors at the common element and takes as value the product of the individual values. In a sense, the factor product closely resembles the Cartesian product between sets. The resulting factor $\psi(A, B, C) = \psi(A, B) \times \psi(B, C)$ has as scope all variables related to the parameterisation of both factor functions. As this is a general operation, no notion of statistical independence is considered. Generally, the factor product follows the structure of,

$$\psi(\mathbf{X}) = \prod \psi(X_i, X_j), \quad \forall i \neq j. \tag{31}$$

The specific factorisation according to the independencies present within the MRF follows later in this chapter. Figure 3.2 depicts a factor product in tabular form.

– **Factor Marginalisation**: follows the exact same method as for probability distributions. To remove a variable's influence from the overall joint distribution, sum over all possible assignments the variable could take. The values of specific assignments become the sum of all matching assignments, essentially leaving the overall joint distribution intact except for the unspecified variable. Factor marginalisation follows the form of,

$$\psi(\mathbf{X}) = \sum_{\mathbf{Y}} \psi(\mathbf{X}, \mathbf{Y}). \tag{32}$$

– **Factor Reduction**: is the factor equivalent to conditioning on a variable. In essence, when considering the conditioning an addition of evidence not present during the factor definition, it is simple keeping a single variable constant and reducing the space of possible variable assignments to those that match the conditioned evidence. Notationally, factor reduction is hard to describe, but a tabular example is provide in Fig. 3.3. Note that unlike the marginalisation operation, the reduction operation does not alter the factor values.

3.1.2 Normalised Gibbs Distributions

The examples presented in Figs. 3.2 and 3.3 included an additional column of sum-normalised (whose value is in the botton row) factor, labelled as one would label a PDF. In the introduction of this chapter it was mentioned that a pseudo-probabilistic framework is desired. Using such normalisation constants (i.e. the sum of all possible assignments) is one way of achieving this. Those familiar with statistical mechanics would term such a pseudo-probabilistic function a (normalised) Gibbs Distribution. These distributions are a compound, requiring a set of factors combined via a factor product and a sum over the joint factor to normalise.

Definition 2 (Gibbs Distribution). *A distribution P_Φ is a Gibbs distribution if parameterised by a set of factors $\psi = \{\psi_1(D_1), \ldots, \psi_K(D_K)\}$, and derived as*

$$P(X_1, \ldots, X_N) = \frac{1}{Z} \cdot \tilde{P}(X_1, \ldots, X_N), \tag{33}$$

where the product of the unnormalised factors are denoted by $\tilde{P}(X_1, , X_K)$ and the partition function that normalises them as $\frac{1}{Z}$. These in turn are defined as,

$$\begin{aligned} \tilde{P}(X_1, , X_K) &= \prod_{i=1}^{K} \psi_i(D_i) \\ Z &= \sum_{x_i=1}^{K} \tilde{P}(X_1, , X_K). \end{aligned} \tag{34}$$

An important extension of the Gibbs distribution is the Boltzmann distribution, or within the field of statistics, the log-linear model. Generally, Boltzmann distributions may be defined as Gibbs distributions with factors in exponential form,

$$\psi(D) = e^{-\epsilon(D)}.$$

Here $\epsilon(D)$ is referred to as the energy function of D. Not only does this form ensure non-negativity (a crucial condition introduced later in this chapter), it also comes incredibly close to the definition of the maximum entropy classifier defined in Ch. 1. Within the applied statistics community, the Boltzmann distribution is part of a general class of log-linear models (see for example on log-linear models and its relationship to PGMs [Christensen2006]).

The definition of the Gibbs distribution introduces an important trade-off. While using factor products and partition functions provides one with values consistent with the definition of probabilities in the introductory chapter, it flies in the face of the goal of PGMs; the normalising sum requires the factors to be fully specified and taken into consideration. With Gibbs distributions, it is no longer possible to break the high dimensional probability space and break it into a set of local probability relationships, whose form is determined by a set of independencies common to both components of a

PGM. The use of a PGM data structure remains as memory intensive as before. Furthermore, while considering the local factor functions, it becomes impossible to predict the behaviour of related factors in a probabilistic manner.

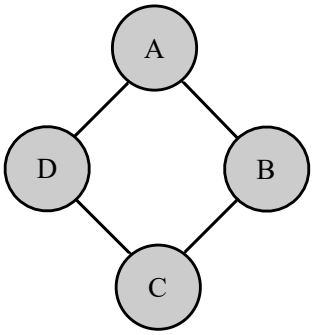

Fig. 3.4: The graphical model corresponding to the dinner table discussion example.

Example: Dinner Table Discussion
This example corresponds to the 'Misconception' example in *Probabilistic Graphical Models* [Koller and Friedman2009].

A group of friends, whose initials happen to start with A, B, C and D, are having dinner at a circle table with an annoyingly large vase in the centre. As such, it is hard for the friends opposite each other to communicate. Instead, during their dinner discussion, each friend talks to those on their side before coming to a global consensus. As this is their favourite restaurant, recordings of their previous discussions and agreement exist.

Friends C and D are quite close and agree very often. During previous discussions between them, agreement or disagreement with the eventual conclusion happened as,

C	D	Disagree	Agree
Disagree		6	2
Agree		1	10

With this information, how likely is it for friend D to agree with the group's consensus?

This is a trick question; for the given description of the distributions and the graphical model as in Fig. 3.4, it's simply impossible to determine this likelihood. When considering more data, however, the problem does become solvable.

A	B	Disagree	Agree
Disagree		7	3
Agree		10	9

,

B	C	Disagree	Agree
Disagree		4	3
Agree		3	5

,

A	D	Disagree	Agree
Disagree		9	6
Agree		10	2

What one can infer from the first table is that C and D prefer agreeing as opposed to disagreeing. However, from the additional data it becomes clear that D loves disagreeing

in conversation where A is also involved, while A and B or B and C aren't inclined to agreement or disagreement with each other.

From the joint table, the most likely configuration appears to be A=Agree, B=Disagree, C=Disagree and D=Disagree, whereas the least likely appear to be A=Agree/Disagree, B=Agree, C=Disagree and D=Agree. Summing over the conditional tables, D=Agree and D=Disagree, the probability for D is found,

$$P(D = \text{Agree}) = \sum_{A,B,C} P(A, B, C, D = \text{Agree}) = 0.381$$
$$P(D = \text{Disagree}) = \sum_{A,B,C} P(A, B, C, D = \text{Disagree}) = 0.619$$

While the example is trivial, it does depict the penalty for using Gibbs distributions. The final computation required fully defining the joint table which, given all variables were binary, is already $2^4 = 16$ entries long. It takes little thought to convince oneself that this becomes intractable for realistic sizes of networks. An MRF equivalent of directed separation is sorely needed.

3.1.3 Conditional Independence Reformulated

Markov random fields did not get its name for nothing. At some level, a Markovian assumption is being made. While the edges in the MRF graph represent relationships between variables, much like the Bayesian network, for computation the relationships that *are not* present are just as important. Thus while it is again possible to construct independence sets, $\mathcal{I}$, it still remains to be proven whether these also translate to equivalent probabilistic functions, such that $(X \perp Y \mid \mathbf{Z})$, holds for both the probabilistic function and the graph by which it factorises. This is, within the context of undirected graphical models, referred to as the global Markov property. While it is the ideal, allowing one to construct functions factorising according to the graphical structure according to any triple of variable sets, it is also the hardest to prove.

Far easier to spot are the local Markov dependencies. The first of these is the *pairwise* Markov property. The reasoning is somewhat as follows: two variables directly connected are per definition always statistically dependent, and conversely, two variables not directly connected can always be rendered statistically independent. Verging on the trivial, symbolically this would be,

$$\mathcal{I}_P = (X \perp Y \mid \mathcal{V} - X, Y), \tag{35}$$

or as the statement, "*two variables are statistically independent if conditioning on all other variables*".

A third Markovian structure exists, more general than the pairwise condition, but ultimately still local. This independency set is closest to the definition of the directed

separation of in Bayesian networks. This *local* definition makes use of the Markov blanket, a generalised notion of parent-child ancestry for undirected graphical models.

Definition 3 (Markov Blanket). *Let* $\mathbf{X}$ *be a set of variables, and* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ *a graph. A neighbour of the set* $\mathbf{X}$*, denoted* $N(\mathbf{X})$*, is any node* Y *such that* $\{Y, \mathbf{X}\} \in \mathcal{E}$*. The Markov blanket, denoted* $MB(\mathbf{X})$ *is the set of all neighbours of* $\mathbf{X}$*.*

Thus in summary, three possible independency structures exist for any given graph:

1. **Pairwise**: target sets of nodes are independent if conditioning on all nodes not included in the target sets
2. **Local**: target sets of nodes are independent if conditioning on the *Markov blanket* of the target nodes
3. **Global**: target sets of nodes are independent if conditioning on mediating sets of variables.

Again, of these three it is the global definition that is desired, but it is the local and pairwise structures most easily found (read, tractable to check if this holds between factorisation and graph structure). However, generally it can be found that the pairwise independency set is strictly weaker than the local independency set, which in turn is strictly weaker than the global set,

$$\mathcal{I}_g(\mathcal{G}) \implies \mathcal{I}_\ell(\mathcal{G}) \implies \mathcal{I}_p(\mathcal{G}). \tag{36}$$

Without proof or motivation, the converse can also be proven, when restricting all possible factorisation of $\mathcal{G}$ to strictly positive functions contained in $\mathcal{P}$. This theorem, and its proof, is generally attributed to Hammersley-Clifford [Hammersley and Clifford1971], although discussions, proofs and extensions can be found in standard texts by Lauritzen [Lauritzen1996, Maathuis et al.2018] or Koller and Friedman [Koller and Friedman2009].

Theorem 1 (Hammersley-Clifford). *Let* $\mathcal{P}$ *be a distribution with positive density. If the independencies encoded in* $\mathcal{P}$ *satisfy* $\mathcal{I}_p(\mathcal{G})$*, then they also satisfy* $\mathcal{I}_g(\mathcal{G})$*. As such, this also implies the local Markov independency structure is satisfied. Thus, the pairwise, local and global Markovian structures are all satisfied by the probability function* $\mathcal{P}$*,*

1. $\mathcal{P} \vdash \mathcal{I}_p(\mathcal{G})$

2. $\mathcal{P} \vdash \mathcal{I}_\ell(\mathcal{G})$

3. $\mathcal{P} \vdash \mathcal{I}_g(\mathcal{G})$

Thus, while the partition function Z must still take into account the whole MRF network to normalise the distributions included, the pseudo-probabilistic distribution $\widetilde{P}$

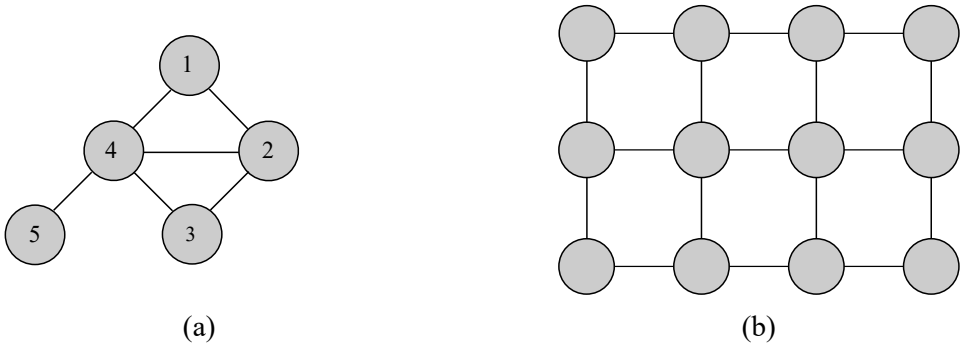

Fig. 3.5: A few simple MRF structures. Subfigure (a) corresponds to Murphy's fig. 10.1(b), and depicts an arbitrary but simple graph with maximal cliques $\{1, 2, 4\}$, $\{2, 3, 4\}$ and $\{4, 5\}$. Subfigure (b) is a standard pairwise model, common to statistical mechanics and image recognition.

may be factorised in a variety of manners. This is the primary motivation for the use of the Gibbs distribution, as stated earlier, as this is guaranteed to be positive for all energy functions input. A direct implication of the Hammersley-Clifford theorem are the two standard factorisations that ensure the independency set guaranteed by the graph is adhered to.

Corollary 1 (Maximal Clique Parameterisation). *If, and only if, a distribution is strictly positive and satisfies the independency set $\mathcal{I}$ encoded in the undirected graph $\mathcal{G}$, the distribution may be represented as a product of local factors, one per maximal clique c[8] from the set of all maximal cliques C,*

$$\widetilde{P}(X_1, \ldots, X_K) = \prod_{c \in \mathcal{C}} \psi_c(D_c). \tag{37}$$

Example: Factorisation of Simple MRF
Fig. 3.5 depicts some very simple MRF structures. Using the terminology discussed in this section, a corresponding factorisation may be defined. For Fig. 3.5(a), two possible standard factorisations exist (more likely exist). The first follows from the Hammersley-Clifford theorem, such that a a maximal clique parameterisation may be employed,

$$\widetilde{P}(X_1, X_2, X_3, X_4, X_5) = \prod_{c \in \mathcal{C}} \psi_c(D_c) = \psi_{1,2,4}(X_1, X_2, X_4)\psi_{2,3,4}(X_2, X_3, X_4)\psi_{4,5}(X_4, X_5),$$

[8] Loosely, maximal cliques are sets of nodes where all are connected to all other nodes. For a set of 2 nodes, this means an edge exists, for 3 a triangle is formed, etc.

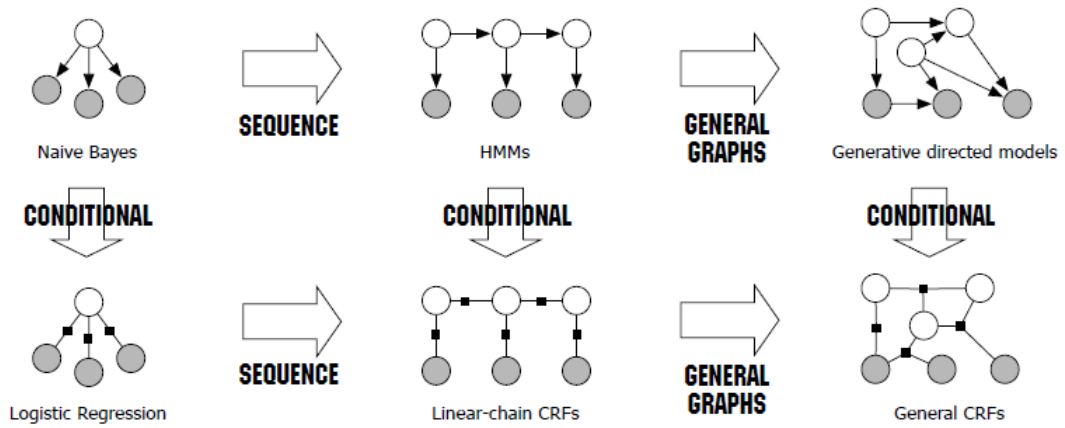

Fig. 3.6: The relationship between conditional random fields and other models discussed in this thesis. Taken from [Sutton et al.2012], with CRFs depicted using factor graphs (see Sec. 4.3).

whereas the second is the pairwise parameterisation that always applies, using one factor per edge in the graph,

$$\widetilde{P}(X_1, X_2, X_3, X_4, X_5) = \prod_{e \in \mathcal{E}} \psi_e(D_i, D_j) \quad = \psi_{1,2}(X_1, X_2)\psi_{1,4}(X_1, X_4)\psi_{2,3}(X_2, X_3) \cdot \psi_{3,4}(X_3, X_4)\psi_{4,5}(X_4, X_5).$$

The other graph, Fig. 3.5(b), depicts the simplest model within a variety of PGM utilising fields: the pairwise model. The only independency relevant here is the pairwise condition, as the only relevant statistical dependence are the nodes directly neighbouring. Models represented in this manner include those for image recognition. The parameterisation follows simply as,

$$\widetilde{P}(\mathbf{X}) = \prod_{e \in \mathcal{E}} \psi_e(D_i, D_j)$$

While this discussion on independencies does not greatly elucidate the practical use of MRFs, these will be tremendously useful for defining inferential algorithms in the next chapters, and will see some application in more specific forms of MRFs like the next section.

3.2 Conditional Random Fields

Where HMMs saw their development due to advances in speech recognition circa 1970, (see the Ch. 6 of this thesis), CRFs came out of the high-dimensional and tremendously

complex structures native to natural language processing [Lafferty et al.2001]. The reason for stepping away from HMMs and similar models for sequential data analysis is their generative nature. The difference between generative and discriminative models has already been discussed in Ch. 1. Briefly, due to generative models estimating the *joint* probability distribution $P(\mathbf{X}, Y)$ as opposed to the conditional distribution $P(Y \mid \mathbf{X})$, the dependencies between the variables in $\mathbf{X}$ need to be accounted for. This can be a costly process, virtually intractable for many applications, and mistaken assumptions can lead to solutions that do not remotely begin to reflect the reality intended to be modelled. Furthermore, in fields such as natural language processing and computer vision, the input space of $\mathbf{X}$ can be very high (order of magnitude 10^5 and higher input variables), making the joint probability function very complex, while the target or label variable Y is typically very simple. Conditional models deal with this by removing $\mathbf{X}$ from the output distribution entirely, leaving a typically simpler and equally accurate result. Furthermore, the choice in variables can be greatly extended. For example, the inclusion of parametric continuous variables or domain relevant features are common and greatly extend the model architecture [Koller and Friedman2009].

This discussion, between a HMM and a CRF perfectly mirrors that of the discussion between the Naive Bayes classifier and the MaxEnt model. In fact, the NBC may be seen as the simplest form of HMM, while the MaxEnt model is the simplest form of CRF. A classic overview from Sutton and McCallum's tutorial is provided in Fig. 3.6.

The definition of a CRF provided by Lafferty, McCallum and Pereira [Lafferty et al.2001], follows as,

Definition 4 (Conditional Random Fields). *Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a graph such that $\mathbf{Y} = (Y_i)_{i \in \mathcal{V}}$ and is indexed by the vertices of $\mathcal{G}$. Then $(\mathbf{X}, \mathbf{Y})$ is a conditional random field in case, when conditioned on $\mathbf{X}$, the random variables $\mathbf{Y}_i$ obey the Markov property with respect to the graph: $P(\mathbf{Y}_i \mid \mathbf{X}, \mathbf{Y}_j, i \neq j) = P(\mathbf{Y}_i \mid \mathbf{X}, \mathbf{Y}_j, Y_j \in MB(Y_i))$.*

Equivalently, $(\mathbf{X}, \mathbf{Y})$ is a conditional random field if for any value x of $\mathbf{X}$, the distribution $P(Y \mid X)$ factorises according to $\mathcal{G}$.

Note that the notation here does away with hidden states and observations (Q and O) common to the first chapter of this thesis. CRFs are described in dealing with input and output, a set of independent and dependent variables, with the vanilla case not including latent variables. More complex variants of CRFs do manage to include latent variables, however these are marginalised out using inferential methods. Regardless, the derivation of the simplest CRF, a linear chain, follows from using the HMM as a special case.

3.2.1 Linear Chain Conditional Random Fields

Recall the definition of a Hidden Markov Model as the product of a regular Markov model, with only state transitions, and an emission model along with an initial state

probability model. Disregarding this last component, the HMM is defined as in Eq. 30, repeated below in the notation of this chapter,

$$\begin{aligned} P(Y,X) &= \prod_{t=1}^{T} P(Y_t \mid Y_{t-1}) P(X_t \mid Y_t) \\ &= \pi \prod_{t=2}^{T} A_{Y_{t-1},Y_t} B_{Y_t,X_t}. \end{aligned}$$

The HMM equation can be written in exponential form when considering the natural logarithm and exponent as each other's inverse function, such that for every base $b^{\log_b(x)} = x$.

$$\begin{aligned} P(Y,X) &= \prod_{t} \exp\{\ln P(Y_t \mid Y_{t-1}) P(X_t \mid Y_t)\} \\ &= \prod_{t} \exp\{\ln P(Y_t \mid Y_{t-1}) + \ln P(X_t \mid Y_t)\} \end{aligned}$$

A notational trick introduced by Lafferty, McCallum and Pereira [Lafferty et al.2001][9] is to include a feature function, composed of two indicator functions that can be summed out. Let $\mathbf{1}_i$ be a function that is 0 for all $j \neq i$ and 1 only for i, such that,

$$f(x) \cdot \mathbf{1}_i = \begin{cases} f(x) & x = x_i \\ 0 & x \neq x_i \end{cases},$$

implying the sum of the product of the function with the indicator, over all possible values of the arguments of the indicator, simply returns the function:

$$\sum_i f(x) \cdot \mathbf{1}_i = f(x).$$

Without altering the definition of a HMM, two indicator functions may be introduced for each log-term; two for the transition component and two for the emission component. These represent being in state Y_j at $t-1$, Y_i at t and emitting X_i. For clarity, these are combined in a single feature function denoted by $f_k(y_t, y_{t-1}, x_t)$. Lastly, to avoid overly cumbersome exponents, the probability terms are contained in a single term θ and the feature functions range over K, such that

$$\begin{aligned} P(\mathbf{Y},\mathbf{X}) &= \prod_t \exp\{\sum_{i\in\mathcal{Y}}\sum_{j\in\mathcal{Y}} \ln P(Y_t \mid Y_{t-1}) \mathbf{1}_{Y_t=i} \mathbf{1}_{Y_{t-1}=j} + \sum_{i\in\mathcal{Y}}\sum_{o\in\mathcal{X}} \ln P(X_t \mid Y_t) \mathbf{1}_{X_t=o} \mathbf{1}_{Y_t=i}\} \\ &= \prod_t \exp\{\sum_{i\in\mathcal{Y}}\sum_{j\in\mathcal{Y}} \ln P(Y_t \mid Y_{t-1}) f_{ij}(Y_t, Y_{t-1}, X_t) + \sum_{i\in\mathcal{Y}}\sum_{o\in\mathcal{X}} \ln P(X_t \mid Y_t) f_{io}(Y_t, Y_{t-1}, X_t)\} \\ &= \prod_t \exp\{\sum_{K\in K} \theta_K f_K(Y_t, Y_{t-1}, X_t)\}. \end{aligned}$$

[9] In truth this is much more than just a notation trick, providing one of the main strengths of CRFs over HMMs. These indicator functions, and its combination as a feature function, can provide domain knowledge not inherent to the models.

All that remains now is transforming the above result to the conditional probability distribution. By the definition of conditional probability as Eq. 4, this follows quite simply as,

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{P(\mathbf{Y}, \mathbf{X})}{P(\mathbf{X})} = \frac{\prod_t \exp\{\sum_k \theta_k f_k(Y_t, Y_{t-1}, X_t)\}}{\sum_{\forall y \in \mathcal{Y}} \prod_t \exp\{\sum_k \theta_k f_k(Y_t, Y_{t-1}, X_t)\}}. \tag{38}$$

Essentially, what has been derived here is the MaxEnt model, see Eq. 12, keeping in mind the transition term. Again, but finally formally, the Naive Bayes classifier is to a HMM what the MaxEnt model is to a CRF.

Two extensions may be introduced to Eq. 38 in order to bring it out of the BN and into the MRF framework. Firstly, the parameters θ_k originally representing log-likelihoods, are made to be general functions. It has been explicitly stated that factors are generalisations of probabilities, and as such this leaves the definition of the LC-CRF intact: given the exponential, this emulates a Gibbs distribution/loglinear model. Secondly, but building on the first extension, the feature functions $f_k(Y_t, Y_{t-1}, X_t)$ are also allowed to take non-binary features. Sutton and McCallum [Sutton et al.2012] provide an example of a named-entity recognition model that incorporates features like capitalisation, matching with capital cities, having the structure of initials, etc. all as binary variables. While naturally used by humans in recognising named entities in text, these features are not obvious from the transitions alone, and as such provide valuable additional information for the CRF.

Thus, the generalised linear-chain conditional random field may be defined as,

Definition 5 (Linear-Chain Conditional Random Field). *Let $\mathbf{Y}, \mathbf{X}$ be random vectors, with only $\mathbf{X}$ being observed during evaluation, and $\theta = \{\theta_k\} \in \mathbb{R}^K$ be a parameter vector, and $\mathbb{F} = \{f_k(y_t, y_{t-1}, x_t\}$ be a set of real valued feature functions. A linear-chain conditional random field is defined as a distribution $P(Y \mid \mathbf{X})$ that takes the form:*

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_t \exp\{\sum_k \theta_k f_k(Y_t, Y_{t-1}, X_t)\}, \tag{39}$$

where Z is the standard partition function or normalising constant, defined as,

$$Z(\mathbf{X}) = \sum_{\forall y \in Y} \prod_t \exp\{\sum_k \theta_k f_k(Y_t, Y_{t-1}, X_t)\}. \tag{40}$$

An equivalent definition, and one much more similar to that of the HMM, is as the product of two factors, one defining the potential of transition, and one defining the potential of emission,

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_t \psi(Y_t, Y_{t-1}) \psi(Y_t, X_t), \tag{41}$$

with the factors $\psi(.)$ being Boltzmann distributions. Here the initial emission factor, the CRF equivalent of π, has been dropped.

3.2.2 General Conditional Random Fields

Note the stark similarity between an MRF, defined using a product of generalised Gibbs functions, and a LC-CRF, defined using a product of log-linear functions. The only real difference -given that the log-linear functions of a LC-CRF may be defined as a Boltzmann function with the product of the parameters with the feature function as its energy- is that the normalisation in the CRF is only in terms of the label variables, $\mathbf{Y}$.

Given this similarity, there is little representational argument against generalising the linear-chain architecture to, for example, a pairwise model like Fig. 3.5(b). Many different CRF topologies exist, with many defined for specific applications. For most applications, however, the expressive power of the simple LC-CRF is sufficient, especially considering the added complexity of training and inference for general structures. Thus, while perhaps intractable, CRFs can easily extend to a variety of forms. Thus, where a MRF is defined by an undirected graph and a generalised Gibbs distribution that factorises according to it, a CRF is defined by an undirected graph and a *conditional* distribution to it.

Definition 6 (General CRF). *Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be an undirected graph over the variables $\mathbf{X}$ and $\mathbf{Y}$. The graph is a conditional random field if the distribution $P(\mathbf{Y} \mid \mathbf{X})$ factorises according to $\mathcal{G}$.*

A field that makes uses of general CRF topologies extensively, is that of computer vision[10]. As mentioned earlier, given the grid-like nature of images, the pairwise-CRF is to computer vision what the LC-CRF is to natural language processing. A toy example using 4×4 generated images is used to illustrate the appeal of general CRFs for modelling image data.

Example: CRFs for Computer Vision

This toy example comes from the introductory examples in the PyStruct documentation on Latent Variable Hierarchical CRF, created by Andreas Muller [Müller and Behnke2014].

Consider a series of 4×4 grids of white-space with 1 corner, a 2×2 square attached to edges of the grid, filled in. Two models are trained to predict from the data which corner is filled (again, a toy example, given that the input data perfectly reflects the output data). The first is a standard pairwise grid CRF, where each image pixel influences only the pixels directly neighbouring. The second model introduces a hidden node layer, connected to each corner. Prior to training iterations, the hidden nodes are 'filled' using message passing inference (see sec. 4.3) on the current training iteration. This topology is depicted in Fig. 3.7(a). The probability function that corresponds to

[10] This is not true in its entirety. The models that are popular within computer vision are a variant of CRFs, using max-margin methods for training and are thus closer akin to structural SVMs than the CRFs described in this thesis. Regardless, the same principle of tying observation to sequences with transitions, is used.

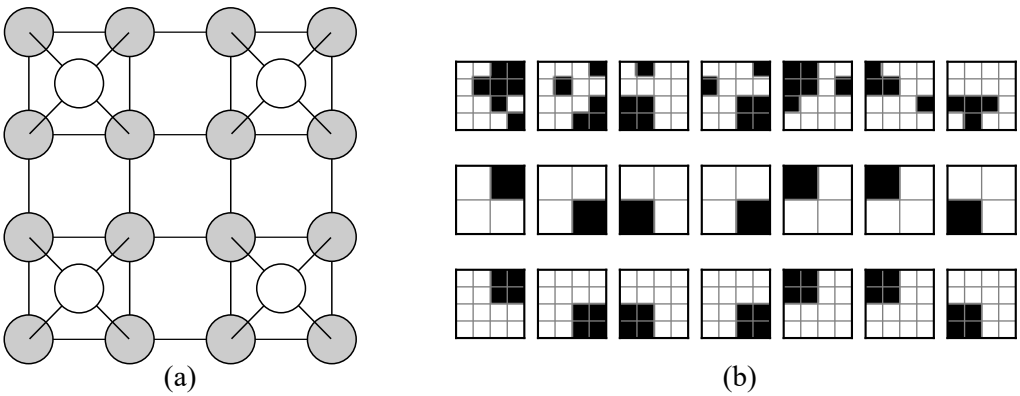

Fig. 3.7: The boxes dataset examples. Sub-figure (a) shows the hierarchical hidden CRF architecture used on the data with great success. Subfigure (b) shows various performances: top corresponding to a standard pairwise CRF, middle to the latent states predicted by the hierarchical model, and bottom the pixel states predicted by the hierarchical model.

this model is of the form,

$$P(\mathbf{Y} \mid \mathbf{X}) = \sum_{\mathbf{Q}} \frac{P(\mathbf{Y}, \mathbf{X}, \mathbf{Q})}{P(\mathbf{X})} = \frac{\sum_{\mathbf{Q}} \prod_t \exp\{\sum_k \theta_k f_k(Y_t, Y_{t-1}, X_t, Q_t)\}}{\sum_{\mathbf{Y}} \sum_{\mathbf{Q}} \prod_t \exp\{\sum_k \theta_k f_k(Y_t, Y_{t-1}, X_t, Q_t)\}},$$

where the latent states are inferred and then marginalised out prior to prediction of the label variables.

Not surprisingly, the first model has considerable trouble making accurate predictions. All it manages to model is whether to activate pixels given probabilities of neighbouring pixels and has no concept of corners. After 1000 iterations of 1-slack max-margin accuracy is only 90.03%. Results for a subset of the predictions is given in the top row of Fig. 3.7(b). Visually, it does appear that it comes close to finding the filled in corners, but also sees some additional noise in non-corner cells.

The second model sees much better performance, with the hidden layer smoothing the predictions of the grid CRF below to account for corners. The state of the hidden layers, again only found using inferential algorithms, is depicted in the middle row of Fig. 3.7(b). After the same amount of training, predictive accuracy is exactly 100.00%. The bottom row of fig. Fig. 3.7(b) depicts the pixel predictions of the second model.

Chapter Summary

The focus of this chapter has been on extending the BN notation, which is both incredibly intuitive and easy to generate using simple probability and linear algebra, to their undirected counterpart. These models lack an inherent probabilistic framework, and such the global properties can no longer be expressed using strictly local connections. However, by introducing extensions of probability spaces and revisiting independencies, a pseudo-probabilistic framework can still be considered. From this framework, and the definition of the HMM, the LC-CRF has been derived. The link between the NBC-MaxEnt and the HMM-CRF dichotomies has been made explicit, with the latter being the sequential counterparts. The expressive power of the CRF in comparison to the HMM has been made clear by considering the feature function and parameter product as an energy of a Boltzmann distribution. Lastly, and very briefly, the general CRF was introduced. Where HMMs can be extended to more general topologies, this can be difficult given the graph needs to remain a DAG. The notation of CRFs extends much more easily, at the cost of significantly more difficult inference and parameter estimation.

Chapter 4

Inference

Table 4: The fundamental problems of Hidden Markov models, as proposed by Rabiner and Juang [Rabiner and Juang1986] [Rabiner1989], with the addition of the 'Smoothing' task.

1	**Evaluation:**	given a model, $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$, and a particular observation sequence $\mathbf{O} = \{o_1, \ldots, o_T\}$, what is the probability of the observation sequence occurring, $P(\mathbf{O}\|\lambda)$?
$1'$	**Smoothing:**	given a model, $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$, and a particular observation sequence $\mathbf{O} = \{o_1, \ldots, o_T\}$, what is the probability of being in a latent variable at time t, $P(q_{i,t}\|\mathbf{O}, \lambda)$?
2	**Decoding:**	given a sequence of observations $\mathbf{O} = \{o_1, \ldots, o_T\}$, and the model λ, which sequence of hidden states $\mathbf{Q} = \{q_1, \ldots, q_T\}$ is most probable?
3	**Learning:**	what model parameters of $\lambda = \{\mathbf{A}, \mathbf{B}, \pi\}$ will maximise $P(\mathbf{O}\|\lambda)$ given that states $\mathbf{Q} = \{q_1, \ldots, q_t\}$ are hidden?

In the series of papers published by Rabiner on the Hidden Markov Model, immediately after discussing modelling and representation, the three basic problems are sketched out, whose formulation is generally accredited to Jack Ferguson [Rabiner1989]. The first two, 'Evaluation' and 'Decoding', are now generally considered as part of the more general statistical inference problem; reaching conclusions about the hidden states using provided evidence. The last basic problem, 'Learning', is most aptly summarised as parameter estimation, a considerably more difficult task than similar tasks within the category given the distinct structure of HMMs and CRFs. An overview of Rabiner's problems, with the addition of a related 'Smoothing' task, is provided in Table 4.

This chapter will provide a discussion on the first two proposed problems, and provide a variety of solutions applicable to both the linear-chain HMMs and CRFs proposed in Chs. 2 & 3, while also setting the framework for more general graph structures. The next chapter will focus entirely on parameter estimation, given the entirely distinct nature of this field.

While this chapter primarily follows Rabiner's tutorials, and as such use probabilities and generally refer to Bayesian network notation, the results achieved for HMMs are also directly relevant and implementable for linear-chain CRFs. Where inferential methods imply generalisation (either from probabilities to non-negative factors, or from linear-chains to general graphs) this will be explicitly stated. Each (sub-)section will be

concluded with a brief example, similar to the '*Balls in Urns*' example used for introducing the concept of a HMM.

4.1 Evaluation

Evaluation: given a model, $\lambda = (A, B, \pi)$, and a particular observation sequence $O = \{o_1, \ldots, o_T\}$, what is the probability of that sequence occurring, $P(O|\lambda)$?

The most straightforward, and naive, approach is a brute-force calculation of the probability of the observations for every conceivable sequence. The method for calculating the likelihood of an observation sequence for individual state sequences is relatively straightforward. First, consider the emission probability for a fixed state sequence, $P(\mathbf{O}|\mathbf{Q}, \lambda)$, as a product of the likelihood for each observation given the state, or,

$$\begin{aligned} P(\mathbf{O}|\mathbf{Q}, \lambda) &= \prod_{t=1}^{T} P(o_t|q_t, \lambda) \\ &= \prod_{t=1}^{T} b_{o_t, q_i}. \end{aligned} \tag{42}$$

This result is half of the solution already. The second part comes when considering the likelihood of the sequence occurring given the model. Since HMMs invoke the Markovian property, one may safely assume that the likelihood of a state occurring is dependent entirely on the state immediately before[11]. Assuming the initial state, $q_{i,t=1}$, of the sequence to be captured by the π parameter, the sequence of states following is the product of the transition likelihood for each state from $t = 2$ till $t = T$, or,

$$P(\mathbf{O}|\lambda) = \pi \cdot \prod_{t=2}^{T} P(q_t \,|\, q_{t-1}). \tag{43}$$

The joint probability of the observation and sequence occurring is simply the product of Eqs. 42 and 43:

$$P(\mathbf{O}, \mathbf{Q}|\lambda) = P(\mathbf{O}|\mathbf{Q}, \lambda) \cdot P(Q|\lambda) = \pi b_{o_1, t=1} \cdot \prod_{t=2}^{T} a_{q_t, q_{t-1}} b_{o_t, q_t}. \tag{44}$$

This would be the result if the state sequence was visible, however, this is per definition not the case. Let $\mathcal{Q}$ denote the set of all possible state sequences, such that $\mathbf{Q} \in \mathcal{Q}$, that could occur between $t = [0, T]$. Assuming a fully ergodic transition matrix $\mathbf{A}$, this would be $2TK^T$ potential sequences. The probability of the observation sequence occurring may be calculated by marginalising out all state sequences,

$$P(\mathbf{O}|\lambda) = \sum_{\mathcal{Q}} P(\mathbf{O}, \mathbf{Q}|\lambda). \tag{45}$$

[11] For semi-Marjov models, this factor simply includes longer dependencies.

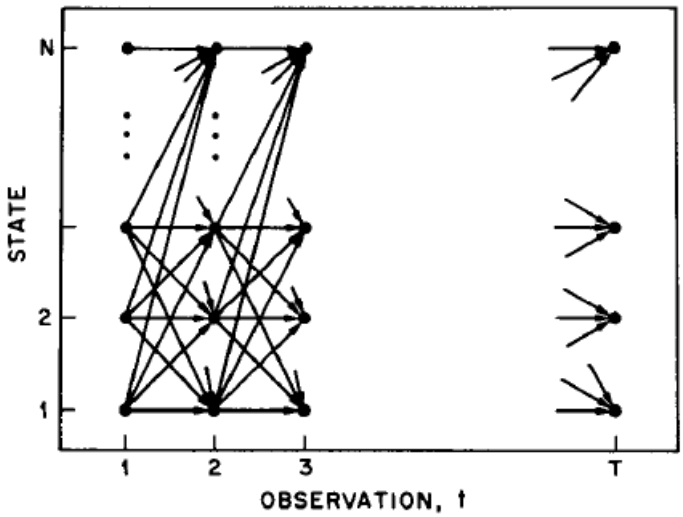

Fig. 4.1: A trellis depiction of a HMM as presented by Rabiner. Taken from [Rabiner1989].

Due to the simplicity of calculation and interpretation, the brute force method might be an attractive one at first. However, it takes little convincing that considering K^T potential sequences is prohibitive beyond unrealistically small values for T. Given the exponential explosion in the number of state sequences, the brute-force method has big-Oh complexity of $\mathcal{O}(K^T)$.

4.1.1 Forwards and Backwards

As such, a faster calculation method is necessary. The standard method in use is the *Forward Algorithm*, invented specifically for HMMs, but generally applicable as a shortest path search on a trellis. For a trellis depiction of an HMM see Fig. 4.1. Trellises have already been implicitly used within this thesis to describe HMMs, as these follow naturally as the graphical depiction of a dynamic Bayesian network with ergodic transitions. The name 'forward' comes from the explicit requirement that one moves from the past to the present, a distinction that becomes clear when considering the state posterior $\gamma(q_t)$.

Let $\gamma(q_t)$ be the probability of a state in the present given the observation sequence before and after, such that invoking Bayes' theorem allows constructing $\gamma(q_t)$ as the product of a likelihood and a prior as,

$$\gamma(q_t) \equiv P(q_t \,|\, \mathbf{O}) = P(q_t \,|\, o_1, \ldots, o_T) \tag{46}$$

$$= \frac{P(o_1, \ldots, o_T \,|\, q_t) P(q_t)}{P(\mathbf{O})}. \tag{47}$$

Note that for a linear chain PGM, the likelihood term can be further decomposed into observations until the present, $o_{t+1}, \ldots, o_t$, and future observations $o_{t+1}, \ldots, o_T$; one term

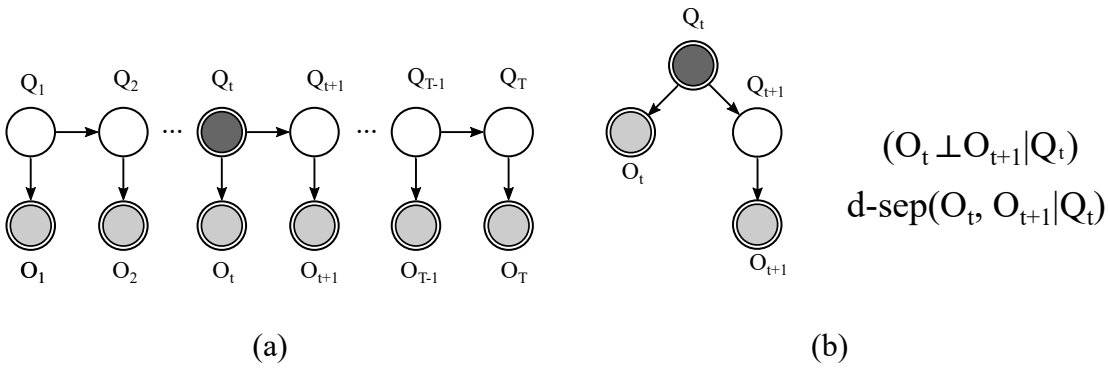

Fig. 4.2: Proof of conditional independencies in a Hidden Markov model. In this figure, the empty circles represent the hidden variables, the double lined light gray circles the observations and the dark gray double lined circles the hidden variable *of interest*. Figure (a) shows the HMM in standard notation, as a linear chain dynamic Bayesian network, while figure (b) shows the HMM in notation used earlier for directed separation, immediately showing the symmetry to the common cause inactive trail.

denotes past evidence, one term denotes future evidence. Staying within the framework of Bayesian networks, this may be proven by noting d-sep$(o_t, o_{t+1}|q_t)$ in the graphical depiction of the relevant HMM, thereby implying the existence of the conditional independence of $(o_t \perp o_{t+1}|q_t)$. See for example Fig. 4.2. As such, using the rules for statistical dependence as per equations 4,

$$\begin{aligned} P(o_1, \ldots, o_T|q_t)P(q_t) &= P(o_1, \ldots, o_t|q_t)P(o_{t+1}, \ldots, o_T|q_t)P(q_t) \\ &= P(o_1, \ldots, o_t, q_t)P(o_{t+1}, \ldots, o_T|q_t). \end{aligned} \tag{48}$$

Using α to denote the joint probability of the prior observations and the current state, and β the backwards likelihood of the future observations given the present state, Eq. 46 may be decomposed into a two pronged problem:

$$\gamma(q_t) = \frac{\alpha(q_t)\beta(q_t)}{P(O)} \tag{49}$$

Thus, the information for a single state in the present, given by $\gamma(q_t)$, is found by combining the evidence of past observations, given by $\alpha(q_t)$, and future observations given that the current state is q_t, given by $\alpha(q_t)$. From now, the forwards and backwards variables are defined as,

$$\text{Forwards} \quad \alpha(q_t) \equiv P(o_1, \ldots, o_t, q_t). \tag{50}$$

$$\text{Backwards} \quad \beta(q_t) \equiv P(o_{t+1}, \ldots, o_T|q_t). \tag{51}$$

The Forward Variable While crucially important, the forward-backward decomposition does not yet provide the necessary fast solution to evaluation problems. The forward component, however, can be leveraged in an elegant solution. For a given chain of observations, marginalising the forward variable at $t = T$ simply returns $P(\mathbf{O}|\lambda)$. For every $t \neq T$, treat the past observations *as if* these did complete an entire chain of evidence, and calculate α dependent on the observations immediately prior. Given that a HMM is a linear chain, this introduces a recursive solution, where each forward variable is dependent on the forward variable before it, treating each sub-chain as a complete chain.

The logic of this approach is not immediately clear. However, using the directed separation property of Bayesian networks, along with the basic properties of probabilities, allows one to derive a recursive relationship of α_t depending on α_{t-1}. From Eq. 50, it follows that

$$\begin{aligned}\alpha(q_t) &= P(o_1, \ldots, o_t, q_t) \\ &= P(o_1, \ldots, o_t \,|\, q_t) P(q_t) \\ &= P(o_t \,|\, q_t) P(o_1, \ldots, o_{t-1} \,|\, q_t) P(q_t) \\ &= P(o_t \,|\, q_t) P(o_1, \ldots, o_{t-1}, q_t). \end{aligned} \tag{52}$$

So far only use of the d-sep property given by Fig. 4.2 and the law of conditional independence, given by Eq. 4, have been used. Note that the conditional probability function $P(o_t|q_t)$ is simply the emission probability, and may be denoted as b_{o_t,q_t}. Further note that the conditional independence statement $(o_{t-1} \perp q_t)$ holds, most easily proven using d-separation and Fig. 4.2 (subtracting 1 from t) for reference. So far, the derivation has provided a method for accelerating inference considerably. Using few more algebraic manipulations, however, does:

$$\begin{aligned}\alpha(q_t) &= b_{q_t}(o_t) P(o_1, \ldots, o_{t-1}, q_t) \\ &= b_{q_t}(o_t) \sum_{q_{t-1} \in \mathbf{Q}_{t-1}} P(o_1, \ldots, o_{t-1}, q_{t-1}, q_t) \\ &= b_{q_t}(o_t) \sum_{q_{t-1} \in \mathbf{Q}_{t-1}} P(o_1, \ldots, o_{t-1}, q_t | q_{t-1}) P(q_{t-1}) \\ &= b_{q_t}(o_t) \sum_{q_{t-1} \in \mathbf{Q}_{t-1}} P(o_1, \ldots, o_{t-1} | q_{t-1}) P(q_t | q_{t-1}) P(q_{t-1}) \\ &= b_{q_t}(o_t) \sum_{q_{t-1} \in \mathbf{Q}_{t-1}} P(o_1, \ldots, o_{t-1}, q_{t-1}) P(q_t | q_{t-1}) \end{aligned} \tag{53}$$

Again, a conditional probability function included above has already been defined specifically for HMMs; note that $P(q_t|q_{t-1}$ is just the transition likelihood given by $a_{q_{i,t-1},q_{j,t}}$. Especially important here is the sum over the previous states, $\sum_{q_{t-1} \in \mathbf{Q}_{t-1}} P(o_1, \ldots, o_{t-1}, q_{t-1})$. There exists a stark similarity between it and the initial definition of the forward variable α.

$$\alpha_t \propto \sum_{q_{t-1} \in \mathbf{Q}_{t-1}} \alpha_{t-1} \tag{54}$$

The calculation of the forward variable is provided using two notations. The first follows immediately from Eq. 53 and is particularly useful for dynamic programming

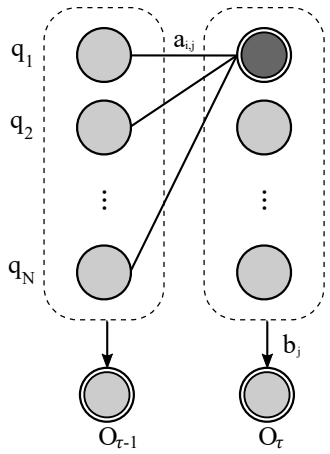

Fig. 4.3: A graphical representation of the forwards algorithm. Rather than representing states, the gray circles now represent the joint probability of the past observations and the state occupying the dark gray circle; the forwards variable.

assignments. The second places the forward variable in the more familiar context of the HMM as a Markov chain, and employs standard matrix notation. The $\circ$ operator denotes the Hadamard product, or element wise vector multiplication [Murphy2012].

$$\begin{aligned} \alpha(q_t) &= b_{q_t}(o_t) \sum_{q_{t-1} \in \mathbf{Q}_{t-1}} \alpha_{t-1} \cdot a(q_{i,t-1}, q_{j,t}) \\ \alpha_{q_j,t} &= b_j \circ (A_j^T \alpha_{t-1}) \end{aligned} \tag{55}$$

Forward Algorithm for Likelihood Evaluation Using the recurrence relation for α, defined in Eq. 55, the likelihood may be evaluated using a dynamic programming assignment. This implies calculating the forward variable using past evidence up until the present t, storing the result as an intermediate value in a a $[K \times T]$ array, increment t and repeating the forward variable computation using the intermediate results from the previous time step. This is repeated until $t = T$, when the forward variables represent the probability $P(\mathbf{O}, q_T)$, leaving the desired result a simple summation away. The algorithm is provided as a textual recipe below, and as pseudo code in Algorithm 1.

1. **Initialisation:** for $t = 1$, the first hidden state is given by π, such that the set of forward variables can be calculated as the product of the initial state and the emission probability of the first observation, dependent on the state α captures:

$$\alpha_{i,1} = \pi b_i(o_1). \tag{56}$$

Algorithm 1: Likelihood Evaluation using Forwards Variable

Input : K states, T observations, HMM: $\lambda = \{A, B, \pi\}$
Output: $P(\mathbf{O} \mid \lambda)$
$\alpha =$ `new array`$[K, T]$
for *t in 1:T* **do**
  if *t=1* **then**
    Initialisation: $\alpha[i, 1] = \pi B[i, 1]$
  else
    Recurrence: $\alpha[i, t] = B[i, t]$ `Sum` $\alpha[j, t-1] \cdot A[j, i]$
  end
end
return: $P(\mathbf{O}|\lambda) = \sum_{i=1}^{K} \alpha[i, T]$

2. **Induction:** the forwards variable at any point in time $1 < t < T$ can be calculated using the recurrence formula derived in Eq. 53:

$$\alpha_{j,t+1} = b_j(o_t) \cdot \sum_{i=1}^{N} \alpha_{i,t} a_{i,j}. \tag{57}$$

3. **Termination:** the last column of the forward variable matrix, corresponding to time $t = T$ contains the set of forward variables that take into account the full sequence $O = \{o_1, \ldots, o_T\}$. Each forward variable only captures the likelihood of the observation sequence and the final state it represents. For finding $P(o_1, \ldots, o_T)$, all that is necessary is marginalising out the $Q_T = q_{i,T}$ term by summing over all q:

$$P(o_1, \ldots, o_t) = \sum_{\forall q} \alpha_{i,T}. \tag{58}$$

The computational complexity loss of the forward algorithm compared to the brute-force method is considerable. For the induction step, one needs to iterate once over all K states, for each summing over all previous K states, until $t = T$, implying $\mathcal{O}(K^2T)$. While still quadratic in K, for many case the number of time persistencies are sparse, or at the very least not ergodic, allowing for approaching $O(KT)$ time complexity [Murphy2012].

Example: Balls in Containers: Likelihood of a Sequence of Balls
Consider the HMM as described in the previous example. After starting up, 10 balls come out of the room on the conveyor belt, described in order as,

$$\mathbf{O} = \{\text{Orange}, \text{Orange}, \text{Orange}, \text{Blue}, \text{Orange}, \text{Green}, \text{Red}, \text{Red}, \text{Red}, \text{Red}\}$$

The forwards algorithm, evaluates as $P(\mathbf{O} \mid \lambda) \approx 1.60\text{E}^{-6}$. Sampling 1 million chains from the HMM gives a relative frequency of occurrence as $\frac{N}{S} = 1.00\text{E}^{-6}$, differing from the calculated probability by 0.6E^{-6}, virtually negligible given the inherent variance of the empirical samples (SE $= \sqrt{\frac{p(1-p)}{N}} \approx 1.00\text{E}^{-6}$).

Algorithm 2: Finding Smoothed Posterior of Observations

Input : K states, T observations from dictionary of size N, HMM: $\lambda = \{A, B, \pi\}$
Output: $P(q_t|O)$
Forward Pass
for *t in 1:T* **do**
  if *t=1* **then**
    Initialisation: $\alpha[state_i, 1] = \pi B[i, 1]$
  else
    Recurrence: $\alpha(q_t) = B[state_i, t]$ `Sum` $\alpha[state_i, t] \cdot A[state_i, state_j]$
  end
end
Backward Pass
for *t in T:1* **do**
  if *t=T* **then**
    Initialisation: $\beta[state_j, T] = 1$
  else
    Recurrence: $\beta[state_j, t] = \texttt{Sum}\beta[state_i, t+1]A[state_i, state_j]B[state_i, t+1]$
  end
end
for *t in T:1* **do**
  $\gamma[t] = \alpha[state_j, t]\beta[state_j, t]/\texttt{Sum}(\alpha[state_j, t]\beta[state_j, t])$
end
return $\gamma[t]$

4.2 Smoothing

Smoothing: given a model, $\lambda = (A, B, \pi)$, and a particular observation sequence $\mathbf{O} = \{o_1, \ldots, o_T\}$, what is the probability of being in a latent variable at time t, $P(q_{i,t}|\mathbf{O}, \lambda)$?

The smoothing task is a generalisation of the evaluation task, where rather than bothering with probabilities of sequences, the only quantity expressed is the marginal distribution over a node in the chain graph[12]. Where the forward algorithm greedily made use of only needing to compute successive probabilities reliant on past observations, the smoothing task will require a little more finesse. Both the prior and future observations will be relevant for expressing the belief in the present states. Luckily, $\gamma(q_t)$ already expresses exactly that.

Recall the definition of the posterior distribution over the states as the normalised product of the forwards and backwards variable,

$$\gamma(q_t) = \frac{\alpha(q_t)\beta(q_t)}{P(O)}$$

Now that the forward variable α has been derived, the posterior is a small step away. The backward variable, β, may be derived in a similar manner to the method applied to the

[12] Or in the case of a dynamic BN, the marginal distribution over the plate of nodes.

α quantity (i.e. again making use of the basic properties of probability and a few independencies that may be derived from Fig. **??**. In a similar set of algebraic manipulations as before, the backwards probability follows as,

$$\begin{aligned}
\beta(q_t) &= P(o_{t+1}, \ldots, o_T | q_t) \\
&= \sum_{q_{t-1} \in \mathbf{Q}_{t-1}} P(o_{t+1}, \ldots, o_T, q_{t-1} | q_t) \\
&= \sum_{q_{t-1} \in \mathbf{Q}_{t-1}} P(o_{t+1}, \ldots, o_T \,|\, q_{t+1}, q_t) P(q_{t+1}, q_t) \\
&= \sum_{q_{t+1}} P(o_{t+1}, \ldots, o_T \,|\, q_{t+1}) P(q_{t+1}, q_t) \\
&= \sum_{q_{t+1}} P(o_{t+2}, \ldots, o_T \,|\, q_{t+1}) P(o_{t+1} \,|\, q_{t+1}) P(q_{t+1}, q_t)
\end{aligned} \tag{59}$$

Note that the first factor is the definition of the backwards variable at one time step prior (set $t+2 = t+1$, and $t+1 = t$ in Eq. 51). Furthermore, in similar fashion to the derivation of the forward variable, the other terms have already been specifically defined in the context of PGMs as the emission and transition probabilities. As such, the backward variable may also be written in a recurrence relation as,

$$\begin{aligned}
\beta(q_t) &= \sum_{q_{t+1}} b_{q_{t+1}}(o_{t+1}) a(q_{i,t}, q_{j,t+1}) \beta_{t+1} \\
\beta j, t &= A_i(B \circ \beta_t)
\end{aligned} \tag{60}$$

Rather than provide a textual or pseudo-code description of the backward algorithm, which is essentially the same as the forward algorithm, but calculates it in reverse, the algorithm for $\gamma(q_t)$ is immediately presented. Not only is *alpha* preferred for the evaluation problem, it is also preferred for calculating first in the forward-backward algorithm. Beyond merely matching one's natural tendency to follow causality, the forward algorithm is considerably more practical in dealing with numerical underflow (a common problem in calculations with probabilities). See the appendix for a discussion on avoiding numerical underflow in PGMs.

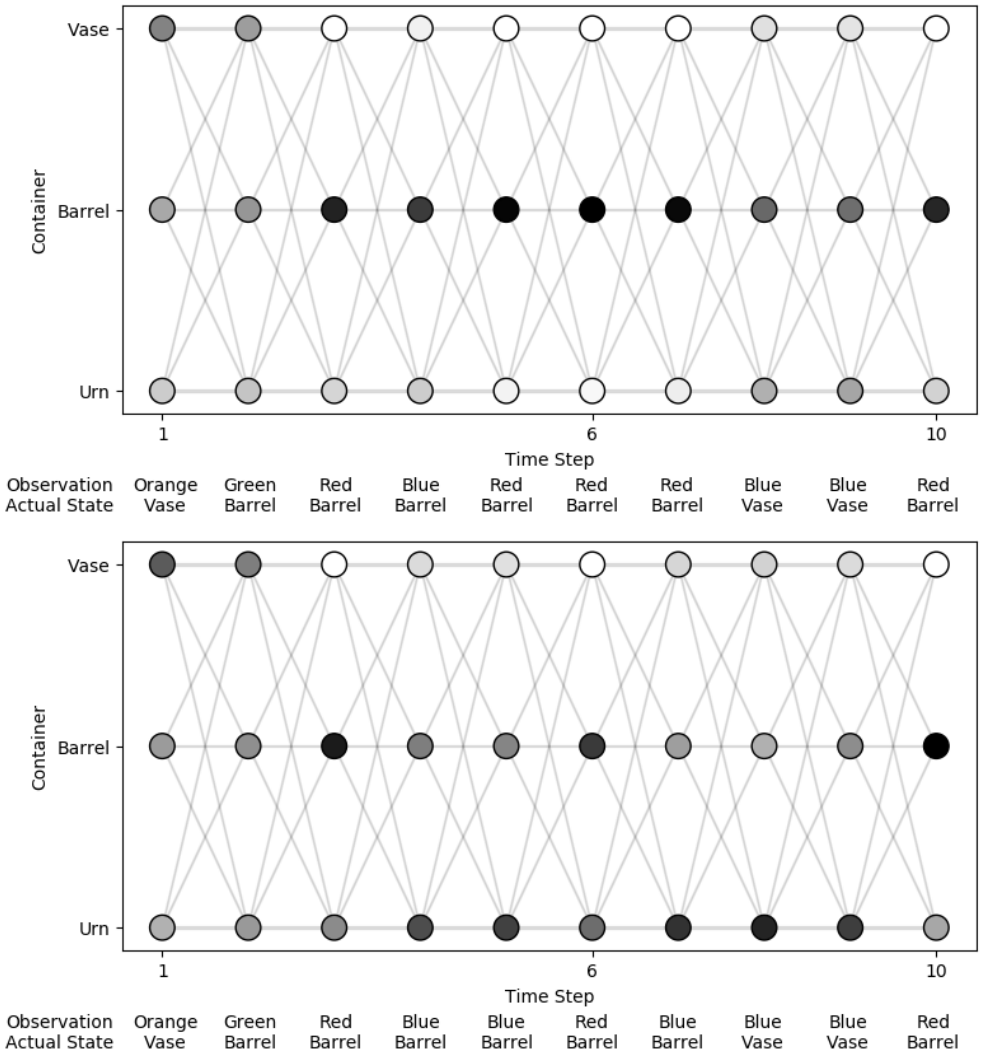

Fig. 4.4: The first and second sequences discussed in the example with relevant posteriors denoted by shades of black. The darker a node, the greater the likelihood of being in that state.

Example: Balls in Containers: Likelihood of a State
Consider a sequence of coloured balls similar to the previous example. Rather than considering the likelihood of the whole chain, consider the likelihood of having the ball in the 6th time step come from the 'Barrel' sized urn.

$$\mathbf{O}_1 = \{\text{Orange}, \text{Green}, \text{Red}, \text{Blue}, \text{Red}, \text{Red}, \text{Red}, \text{Blue}, \text{Blue}, \text{Red}\}$$

Using the parameter set defined earlier and the observation sequence as above, the forward-backward algorithm evaluates the likelihood of the 6th observation coming from the 'Barrel' sized urn as 97.2%, not surprising given the sequence of red and blue balls (which could very well come from that container).

If instead the balls neighbouring the 6th ball are changed to a drastically different possibility (a colour that is much more likely to come from a different container), one would expect the likelihood of the 6th observation to originate from the 'Barrel' to have declined considerably.

$$\mathbf{O}_2 = \{\text{Orange}, \text{Green}, \text{Red}, \text{Blue}, \text{Blue}, \text{Red}, \text{Blue}, \text{Blue}, \text{Blue}, \text{Red}\}$$

Considering the sequence above, showing more dominance towards the blue colour, the forwards-backwards algorithm assigns the 'Barrel' state in the 6th time point a 57% likelihood, preferring the 'Urn' sized container almost as much at 47% likelihood. Fig. 4.4 gives the trellis depiction of the two sequences. Note the ambiguity that is present in the second sequence (between the 'Barrel' and 'Urn') that is not present in the first sequence.

4.3 Evaluation & Smoothing for General Graphs: Belief Propagation

So far the discussion on inference has limited itself to linear-chains and exact results. When moving to more general graphical structures, exact solutions may not be tractable or even possible (e.g. cyclisms in graphs). However, specific graph structures do allow for leveraging topological simplicity into efficient and exact inferential algorithms.

Belief propagation is one of the most common of such methods, most likely due to its coinage for BNs by Judea Pearl [Pearl1982]. It has, however, seen independent derivations in fields tangential to PGMs: the 'Bethe Peierls approximation' in statistical mechanics or the Sum-Product algorithm in coding theory [Mezard and Montanari2009]. Another reason might be that while only exact for specific graphical structures, it is a good approximation method for general or loopy structures. While Pearl initially defined BP for tree-structured Bayesian networks, with a few extensions it is equally useful for MRFs. As such, belief propagation is but one of many variants within the family of message-passing algorithms.

Common modern derivations start with tree-structures, and discuss the loopy variant for extending to more complex topologies, and make use of notation common to MRFs.

4.3.1 Trees and Factor Graphs

Forests and trees are common network topologies within the fields of mathematics and computer science. Chains are simply a subset of trees, specifically ones with right-ward dominance.

Definition 1 (Trees). *Consider a graph, composed as usual by a set of vertices and connecting edges $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. The graph is a tree, if and only if,*

1. *it has K nodes and $K - 1$ edges,*
2. *it has no cycles,*
3. *there exists only one unique path from v_i to v_j,*

The tree root is defined as the node which has no incoming edges, or if undirected, the node with the lowest rank in the trees topological ordering. The tree's leaves are all nodes which have no outgoing edges, or if undirected, has the highest rank in the trees topological ordering.

Within the language of PGMs, trees have the property of having a maximal clique size of 2, such that a node is dependent only on its incoming node. Otherwise, for Bayesian

networks, the independence relationship $(x_i \perp x_j \,|\, \mathrm{PA}(x_j))$, although this is not particularly informative as all Bayesian networks must abide by this property. The pairwise Markov random field defined by such a graph is simply the standard factor product

$$P(\mathbf{X}) = \frac{1}{Z} \prod_{(i,j)\in\mathcal{E}} \psi(i,j).$$

For example, consider the graph depicted in Fig. 4.5. The corresponding factorisation would be

$$P(\mathbf{X}) = \psi(1,4)\psi(2,4)\psi(3,4)\psi(4,5)\psi(5,6)\psi(5,7).$$

Here node 4 is set to function as the target of the messages, or rather $P(x_4)$ serves as the desired quantity from inference. As such, x_4 is made to function as the root of the tree graph.

Rather than using PGMs, whether expressed as a BN or MRF, message passing algorithms make use of a closely related graphoid: factor graphs. Bishop notes the ability of factor graphs to express both directed and undirected graphs [Bishop2006]. The factors of factor graphs are not to be confused with the factors as generalisations of probability spaces, used in Ch. 3. Thus, the MRF graph in Fig. 4.5 may instead be written as the product of a set of factors (denoted graphically as black squares), such that it factorises as

$$\widetilde{P}(\mathbf{X}) = \prod_{s\in\mathrm{MB}(X)} F(X, X_s), \tag{61}$$

where $\widetilde{P}(\mathbf{X})$ is again the unnormalised factor product.

The factor nodes in a factor graph serve as 'summarisors', containing all necessary information for updating the desired node, given their connection to the rest of the network. Thus, rather than expressing probability spaces as the product of many local functions it expresses it as the product of many summaries of local functions. Factors may be defined for every node in the graph, or equivalently for every subgraph. For example, Fig. 4.5 depicts a factor graph with a factor for every node in the graph, 6 in total (excluding leaves). This could have been done with equivalently with only 3, one for the left sub-graph (nodes 1,2,3 to 4), one *inside* the right sub-graph (6 and 7 to 5) and one between the right subgraph and node 4.

4.3.2 Message Passing

Ultimately, the goal of belief propagation is a general algorithm that allows updating the belief of a node, given new evidence within the whole network (i.e. the smoothing task). Using the assumption of independence from all nodes outside of the Markovian blanket, this is entirely equivalent to calculating the updated belief in a node dependent on changes in neighbouring nodes. Keeping with the example depicted in Fig. 4.5, symbolically this

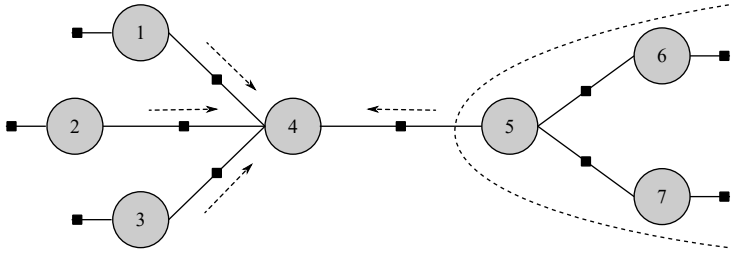

Fig. 4.5: A graphical representation of message passing in a small factor tree. The nodes and edges are as usual, the black squares denote factors. Dashed arrows denote the direction of the message, the dashed conical section denotes a sub-graph, displayed in full. Factors at the leaves of the factor tree represent sum-product of the variable with itself.

gives

$$\widetilde{P}(X_4) = \prod_{s \in \mathrm{MB}(X_4)} \sum_{X_s} F(X_4, \mathcal{X}_s) = \prod_{s \in \mathrm{MB}(X_4)} \mu_{F_s \to X_4}$$
$$= \mu_{F_1 \to X_4} \mu_{F_2 \to X_4} \mu_{F_3 \to X_4} \mu_{F_5 \to X_4}.$$

Thus, the marginal $P(x_4)$ is the product of incoming factors, marginalised such these are expressed entirely in terms of x_4. The quantity $\mu_{F_s \to x}(x)$ is called the message from F_s to x, defined as

$$\mu_{F \to X} = \sum_{s \in \mathrm{MB}(X)} F(X, X_s). \tag{62}$$

Of course, moving the computation of a marginal distribution of a node to its neighbouring factors merely shifts the responsibility rather than providing a tractable solution. After all, each of the neighbouring factors has connecting nodes influencing it, and the new evidence that forces x to update might do so indirectly through a path connected to one of the factors in $\mathrm{MB}(X)$. When considering the message of the mediating factors of x, one need also consider the *incoming* message for the factor itself.

This is suspiciously similar to a recurrence problem, especially like the one in the motivation for the forward-backward algorithm. Let $G(X_s, X_m)$ denote the subgraph with as node a variable from $\mathrm{MB}(X)$, and as connecting factors a set of neighbouring nodes $X_m \in \mathrm{MB}(X_s)$. Since the message passes from X_m to X, implying no influence from X on X_m, this is a self-contained and symmetrical problem to the one being resolved. Such a subgraph is depicted in Fig. 4.5 as a dotted cone about the X_5 node. Note that this subgraph is also a tree. The subgraph may be factorised in exactly the same manner as the whole network has been, dependent entirely on its neighbouring factors:

$$G(X_s, \mathbf{X}_m) = \prod_{m \in \mathrm{MB}(X_s)} F(X_s, X_m). \tag{63}$$

The incoming message for the factor $F_s(X, X_s)$ may thus be defined as the contribution of the connecting node, which in turn is dependent on its neighbouring factors. Symbolically, this implies

$$\mu_{X\to F} = \sum_{m\in \mathrm{MB}(X_s)} G(X_s, X_m) = \sum_{m\in \mathrm{MB}(X_s)} \prod_{m\in \mathrm{MB}(X_s)} F(X_s, X_m). \tag{64}$$

Solving for X thus becomes a recurrence problem, where the computation of marginal distributions is pushed out to the leaves of the tree, and influence is slowly traced to the root via a connected network of subgraphs. For the leaves, denoted in Fig. 4.5 by nodes with a 'dangling' factor, this is equivalent to simply marginalising itself and passing this as a message to its respective factor, which per the definition of probabilities is simply $\mu_{\mathrm{leaf}, f(X_{\neg leaf})} = 1$. The message of of the factor to variable message, Eq. 62, is appended with the product of its incoming messages, accounting for influence of the factor's neighbouring nodes.

$$\mu_{F\to X} = \sum_{s\in \mathrm{MB}(X)} (F(X, X_s) \prod_{m\in \mathrm{MB}(s)} \mu_{X\to F}) \tag{65}$$

Here the subscripts have been left out for readability. The above result may be read as '*the message from a factor to a target variable is the product of the incoming messages for the factor and the factor itself, with all non-target variables marginalised out*'.

Example: Belief Propagation on a Simple Graph

Consider, one last time, the graph as in Fig. 4.5. Let the probability function that factorises according to the graph be a MRF. The desired quantity is the marginal distribution $P(x_4)$. Using belief propagation, this can be written as the product of the messages from it's neighbouring factors.

$$\widetilde{P}(X_4) = \mu_{F(X_1,X_4)\to X_4}\mu_{F(X_2,X_4)\to X_4}\mu_{F(X_3,X_4)\to X_4}\mu_{F(X_4,X_5)\to X_4}$$

From the definition of the factor to variable message, this may also be written as a product of sums,

$$\begin{aligned}\widetilde{P}(X_4) &= \sum_{X_1} F(X_1, X_4) \sum_{X_2} F(X_2, X_4) \sum_{X_3} F(X_3, X_4)\mu_{X_5\to F(X_4,X_5)} \\ &= \sum_{X_1,X_2,X_3} F(X_1, X_4)F(X_2, X_4)F(X_3, X_4)\mu_{X_5\to F(X_4,X_5)}\end{aligned}$$

Nodes 1,2 and 3 are all leaves, and therefore their influence on X_4 is just the influence they exert. No form of recursion is needed there. For the node X_5, however, the message must be reconstructed by considering the nodes with the subgraph with X_5 as root

node. This is easily done using Eq. 65.

$$\begin{aligned}\widetilde{P}(X_4) &= \sum_{s\in \mathrm{MB}(X)} F(X_4, X_s) \\ &= \sum_{X_1,X_2,X_3} F(X_1,X_4)F(X_2,X_4)F(X_3,X_4)(\sum_{X_5}(F(X_4,X_5)\cdot \\ &\quad \mu_{X_5\to F(X_5,X_6)}\mu_{X_5\to F(X_5,X_7)})\end{aligned}$$

Again, from the definition of the variable to factor message, the last two messages may be rewritten as a sum product. Given that these nodes are leaves, no further recursion is required.

$$\begin{aligned}\widetilde{P}(X_4) &= \sum_{X_1,X_2,X_3} F(X_1,X_4)F(X_2,X_4)F(X_3,X_4)(\sum_{X_5} F(X_4,X_5)\cdot \\ &\quad \sum_{X_6} F(X_5,X_6)\sum_{X_7} F(X_5,X_7) \\ &= \sum_{X_1,X_2,X_3,X_5,X_6,X_7} F(X_1,X_4)F(X_2,X_4)F(X_3,X_4)F(X_4,X_5)F(X_5,X_6)F(X_5,X_7) \\ &= \prod_{i,j}\sum_{\mathbf{X}-X_4} F(X_i,X_j) \\ &= \sum_{\mathbf{X}-X_4} P(\mathbf{X})\end{aligned}$$

Using a few more definitions provided in the text, it is not hard to see that the BP algorithm collapses into the definition of the a factor graph, Eq. 61, marginalised over all variables not of interest.

While the example shows the equivalence of belief propagation and probabilistic marginalisation when trying to find marginal distributions, a necessary condition for any solution, it does not highlight the loss in computational complexity. Naively marginalising, a sum over all N possible values of all K variables, would take $\mathcal{O}(K^N)$ time. Belief propagation, which smartly orders the variables and recurses, can do the same in $\mathcal{O}(K^2)$ time [Maathuis et al.2018]. A very small numerical example is provided next, to elucidate how belief propagation can find the posterior of MRF/CRF nodes as an equivalent of the forward-backward algorithm.

Example: Belief Propagation for a Linear-Chain MRF
Consider the middle node, something akin to the current hidden node for an HMM, to be the root node. As such, its factor is constructed from three separate messages along the linear chain, or

$$\widetilde{P}(X_{\mathrm{Root}}) = \mu_{F(\mathrm{Left},\ \mathrm{Root})\to X_{\mathrm{Root}}}\mu_{F(\mathrm{Right},\ \mathrm{Root})\to X_{\mathrm{Root}}}\mu_{F(\mathrm{Bottom},\ \mathrm{Root})\to X_{\mathrm{Root}}}$$

.

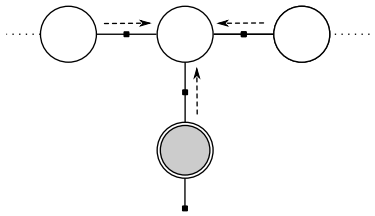

Fig. 4.6: The equivalence of belief propagation on a linear chain and general trees.

The binary node functions (conveniently normalised prior to use) are given as three tabular functions as

X_{Left}	X_{Root}	$\psi(\text{Left}, \text{Root})$
0	0	0.2
1	0	0.6
0	1	0.1
1	1	0.1

,

X_{Root}	X_{Right}	$\psi(\text{Root}, \text{Right})$
0	0	0.125
1	0	0.375
0	1	0.25
1	1	0.25

,

X_{Root}	X_{Botom}	$\psi(\text{Root}, \text{Bottom})$
0	0	0.125
1	0	0.375
0	1	0.25
1	1	0.25

.

Given that the left, right and bottom nodes are all root nodes, or at the very least made root nodes by recursion of BP on the sub-graphs before it, the variable to factor messages $\mu_{X \to F(X,\text{Root}}$, are all simply 1 (to see this, use Eq. 64 with X_s, without X_m and with no incoming messages of itself). As such, by Eq. 65, the messages follow simply as the marginalisation over the non-root variables, or

$$\mu_{F(\text{X, Root}) \to X_{\text{Root}}} = \sum_{X} F(X, X_{\text{Root}}), \forall X \neq X_{\text{Root}}.$$

These messages thus simply follow as,

$\mu(\text{Left}, \text{Root})$	$\psi(X_{\text{Root}})$
0	0.800
1	0.200

,

$\mu(\text{Right}, \text{Root})$	$\psi(X_{\text{Root}})$
0	0.375
1	0.625

,

$\mu(\text{Bottom}, \text{Root})$	$\psi(X_{\text{Root}})$
0	0.500
1	0.500

.

By Eq. 65, the final unnormalised posterior density function of the root node follows as the product of the messages. Normalisation follows by simply taking into consideration the final table (note that this is a *very* fast calculation of the partition function $Z(\mathbf{X})$, as it eliminates all unnecessary information already. Finally then, the posterior density is found to be

$$\widetilde{P}(X_{\text{Root}}) = \{0 : 0.15, 1 : 0.06\}.$$

The results aren't surprising when taking into account the affinities of the connected nodes: the left and right nodes both prefer a value of $X_{\text{Root}} = 0$, while the bottom node is indifferent, such that their combination will also see preference for $X_{\text{Root}} = 0$.

While it might not be immediately apparent from Fig. 4.6, belief propagation and the forward-backward algorithm are totally equivalent for linear-chain graphs, whether these graphs are BNs or MRFs. The observation serves as a leaf node, returning as message the conditional probability of a the parent state given the observation, and the left and right neighbours are are summed out, exactly like necessary for the α and β quantities in the forward-backward algorithm. As such, belief-propagation may be seen as an extension of the forward-backward algorithm for general trees (of which the linear chain is the simplest form).

4.3.3 Loopy Belief Propagation

It has already been explicitly stated that message passing schemes are only exact for trees or tree-like structures. However, due to its intuitive appeal and efficient computation, it remains a good baseline choice for inferential algorithm for general graphs (those including cycles) as well. Pearl was aware of the possibility of using BP on looped cycles, dedicating a chapter to it in *Probabilistic Reasoning*, but kept strong reservations. An oft quoted section summarises the principle behind loopy BP and its limitations [Pearl2014],

> When loops are present, the network is no longer singly connected and local propagation schemes will invariably run into trouble [...]. If we ignore the existence of loops and permit the nodes to continue communicating with each other as if the network were singly connected, messages may circulate indefinitely around the loops and the process may not converge to a stable equilibrium [...]. Such oscillations do not normally occur in probabilistic networks [...] which tend to bring all messages to some stable equilibrium as time goes on. However, this asymptotic equilibrium is not coherent, in the sense that it does not represent the posterior probabilities of all nodes of the networks.

In essence, the loopy version of BP simply iterates sum-product recurrences according to a message passing schedule. It is expected that eventually the posteriors converge to a value. Whether this value is remotely correct remains unknown, although evidence does exist for the accuracy of these iterative message passing schemes [Murphy2012]. Koller and Friedman dedicate a larger section to the merits and surprising revival of propagation based approximate inference methods [Koller and Friedman2009].

4.4 Decoding

Decoding: given a sequence of observations $O = \{O_1, \ldots, O_T\}$, and the model λ, which sequence of hidden states $Q = \{Q_1, \ldots, Q_T\}$ is most probable.

The last inferential problem described by Rabiner is also the most interesting one, for most applications. While understanding likelihoods of observations sequences and

Algorithm 3: Viterbi Decoding

Input : K states, T observations from dictionary of size N, HMM: $\lambda = \{A, B, \pi\}$
Output: $\arg\max_{\mathbf{Q}} P(\mathbf{Q}|\mathbf{O})$
for *t in 1:T* **do**
  if *t=1* **then**
    Initialisation: $\delta[state_i, 1] = \pi B[i, 1]$
  else
    Recurrence: $\delta[state_j, t] = \max \delta[state_i, t-1]A[i, j]B[j, x_t]$
    $\psi[state_j, t] = \arg\max_{state_i} \delta[state_i, t-1]A[i, j]B[j, x_t]$
  end
end
Termination $P^* = \max \delta[state_i, T]$ $q_T = \arg\max_{state_i} \delta[state_i, T]$
Back-Trace **for** *t in T-1:1* **do**
  $q_t = \psi[q_{t+1}, t+1]$
end
return $\{q_1, \ldots, q_T\}$

probabilities of states at individual time points is highly relevant, ideally one can also relate an observation sequence to an identical but ultimately hidden state sequence.

Two methods for estimating state sequences using likelihood functions are *Maximum a Posteriori* (MAP), which aims to find the maximum probability of the whole sequence, whereas *Maximizer of the Posterior Marginals* (MPM) aims to the maximum number of correct states in the sequence. While their goal is the same, the reasoning behind reaching the optimal latent state sequence is quite different. The MAP estimates the joint, or global, posterior probability distribution over the possible state variables, and selects as estimate the mode. The MPM instead takes as estimate the mode for individual time-steps. Thus, symbollically the difference between the methods comes to

$$\text{(MAP)} \ \arg\max P(q_{1:T}|o_{1:T}) \neq \{\arg\max P(q_1|o_1), \ldots, \arg\max P(q_T|o_T)\} \ \text{(MPM)}. \quad (66)$$

The method of MPM estimation easy to perform with the tools provided in the previous sections and typically fairly robust. However, the MAP estimates have the advantage of being globally consistent: rather than looking for a the most probably state at each step, it takes into account the most probable sequence over all steps. Further note that the MPM decoder does not take into account the transitions. Given two likely states neighbouring each other, however with an impossible transition between them, the MPM decoder would be oblivious to this fact, whereas the MAP decoder will filter this solution out.

Using the results from the previous sections, the MPM inference method for hidden state sequence could simply be:

$$\hat{\mathbf{Q}} = \{t \in T : \arg\max \gamma(t)\}. \quad (67)$$

4.4.1 MAP using Viterbi Decoding

The MAP requirement brings back needing to evaluate every possible chain of states; the forward-backward algorithm will not help in speeding up evaluation. The consideration of an linear-chain PGM as a trellis might however. Note the similarity of the MAP decoding problem and the shortest-path problem, where the distances are given by the complement of the probability of the transition. Common to shortest path problems is the realisation that the shortest path from any node is dependent on the previous node on the shortest path.

The Viterbi algorithm builds on top of the forwards-backwards algorithm discussion in the previous sections. Let $\delta_t(q_i)$ be the probability of reaching state q_i at time $t = t$, given that the previous states belong to the most probable sequence[13]:

$$\delta_t(q_j) \equiv \max P(q_1, \ldots, q_{t-1}, q_t = j | o_1, \ldots, o_t). \tag{68}$$

Note that the only way to reach the state on the most probable path, is by moving from the state on the most probable path one step backwards, such that $\delta_t(q = i)$ decomposes as

$$\delta_t(q_j) = \max \delta_{t-1}(q_i) a(i,j) b_j(o_t). \tag{69}$$

where a represents the specific transition probability from state i to state j, and b the emission probability of the observation at $t = t$ given state j. While the above equations give the probability of moving from across the sequence, the final desired result ought to be the actual sequences. A link is necessary between current observations and past observations. Therefore, let ψ be the previous state that maximises Eq. 68

$$\psi_t = \arg\max \delta_{t-1}(q_i) a(i,j) b_j(o_t). \tag{70}$$

Note the stark similarity between the forwards variables defined in Eq. 50 and Eq. 53 and the above defined variables. The only difference is the substitution of the max operator for the sum operator, and the tracking of the most probable paths. The interpretation also follows as such; the termination step finds the most probable final state for the observation sequence.

Let P^* be the probability of the most likely state sequence and q_T^* be the most likely terminal state of the sequence, such that

$$P^* = \max \delta_T(q_j) \tag{71}$$

$$q_T^* = \arg\max \delta_T(q_j). \tag{72}$$

The real difference between the forwards algorithm and the Viterbi decoding algorithm comes in using the links described by ψ_t to construct a chain of hidden states from the terminal state to the first; from the most likely state, the previous most likely state is that which maximises the transition from i at $t-1$ to j at t This process is typically referred to as back-tracing. Fig. 4.7 provides a graphical representation of the Viterbi decoding algorithm. A pseudo-code depiction is provided in Algorithm 3.

[13] The iterator j is used here to avoid confusion when employing a recurrence relation, with i being the iterator for the previous states

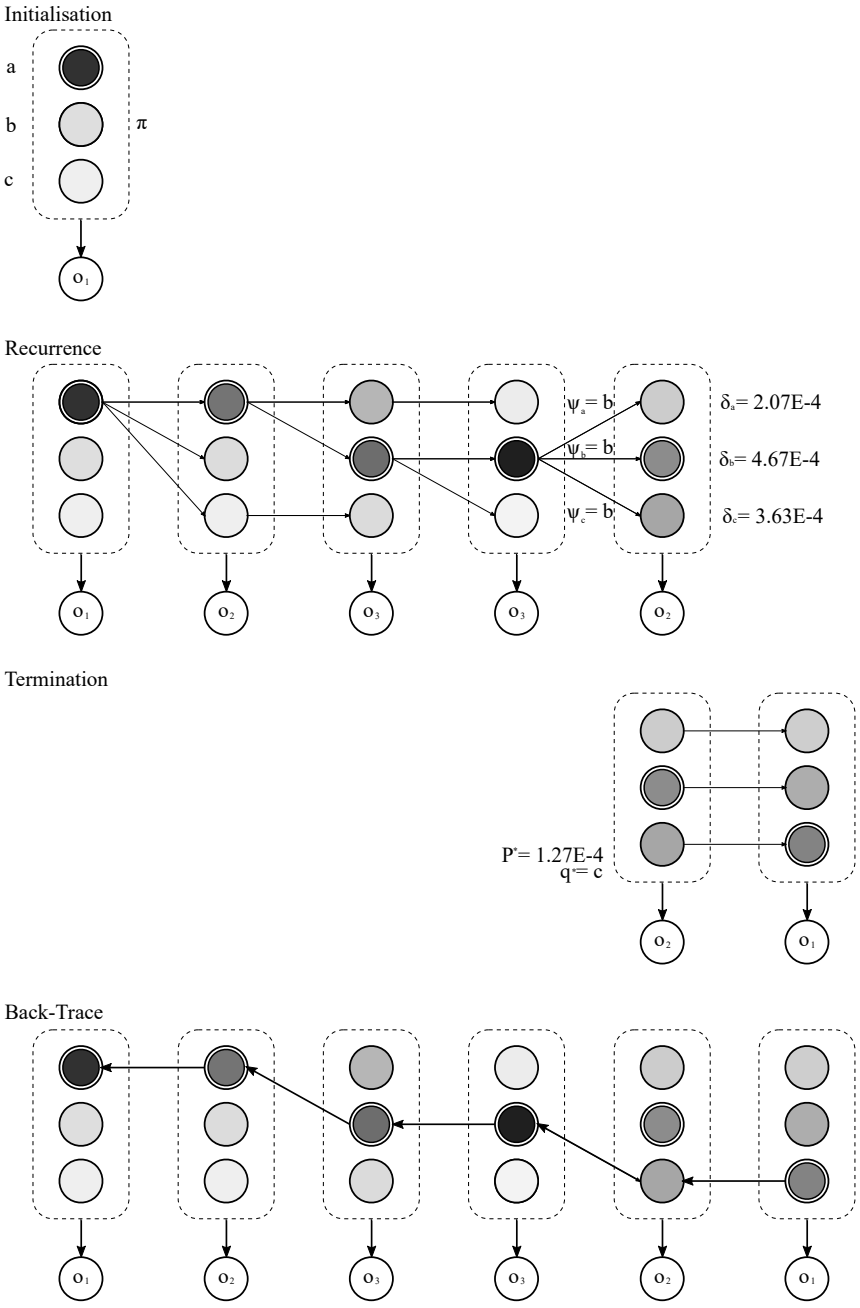

Fig. 4.7: A Viterbi decoding example, with 3 hidden states and 3 observation types across 6 time steps. The hidden variables show their relative Viterbi probabilities by their hue. The local maxima are denoted by a double lined circle, and the paths denote the most likely transition from $q_{i,t-1}$ to $q_{j,t}$.

Max-Product Where the Viterbi and Forwards algorithm share a stark similarity, the BP algorithm and its MAP decoded could also. Note that the main innovation of the Viterbi algorithm is exchanging the max operator for a sum at every instant of time. Without elaborating, much like the forwards-backwards algorithm could be generalised to belief propagation, the Viterbi decoding algorithm can see the same generalisation when exchanging the sum in the message passing equations, Eqs. 64 and 65, for a max operator. Much like a Viterbi/MAP path could be constructed through the HMM, a MAP path will be constructed through the factor graph that BP is being used on.

Chapter Summary

Where the previous chapters have been laden with definitions and requirements imposed by the phenomena being modelled, this chapter finally presents methods for generating answers. The fundamental problems of HMMs, originally presented by Rabiner and Juang and later used as a framework for similar works (for example Lafferty, McCallum and Pereira). The first two of these problems, along with a slight extension somewhere between the two, have been answered. While the focus has been primarily on Bayesian networks and linear chains, extensions to general graphs were also discussed.

Chapter 5

Learning

Learning: what model parameters of $\lambda = \{A, B, \pi\}$ will maximise $P(O|\lambda)$ while the states $Q = \{q_1, \ldots, q_t\}$ remain hidden?

5.1 Learning for HMMs

A very common method for computing the parameters of a model, although by no means the only method, is computing the maximum likelihood assignment of the parameters to the data. A potential possibility is the lack of adherence to auxiliary functions that complicate learning and introduce subjectivity to the model, for example prior distributions for Bayesian parameter estimation or loss functions for empirical risk minimisation. Instead, under the assumption that the data $\mathbf{X}$ is independently and identically, MLE produces estimates by considering the likelihood function. This section provides a very brief introduction to MLE estimation, before discussing an approximate method for latent variable models.

5.1.1 Maximum Likelihood Estimation

As always, starting with Bayes' theorem as Eq. 9, relating the desired CDF of the parameters given the observations to the likelihood of the observations given the parameters,

$$P(\theta|X) = \frac{P(X|\theta)}{P(\mathbf{X})}P(\theta). \tag{73}$$

Maximum likelihood estimation (MLE) makes use of the $P(X|\theta)$ term, denoting it as the likelihood function, or typically the log-likelihood function which follow as,

$$\mathcal{L}(\theta|\mathbf{X}) = P(\mathbf{X}|\theta) = \prod_{i=1}^{N} P(x_i|\theta), \tag{74}$$

$$\ell(\theta|\mathbf{X}) = \log(\mathcal{L}) = \sum_{i=1}^{N} \log(P(x_i|\theta)). \tag{75}$$

The parameter, or argument, that maximises the above functions, is aptly labelled the maximum likelihood estimator of $P(\theta|\mathbf{X})$:

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta|\mathbf{X}). \tag{76}$$

While MLE is a general framework for parameter estimation, it requires complete and observable data, rendering it useless for HMM learning tasks. To extend the MLE framework to models incorporating hidden states, an iterative convergence method called Expectation Maximisation (EM) can be applied.

5.1.2 Expectation Maximisation

Consider the multivariate MLE case $\mathcal{L}(\theta \mid \mathbf{X}, \mathbf{Y})$, where $\mathbf{X}$ represents a set of observations while $\mathbf{Y}$ represents the set of latent variables corresponding to the observations. Using Bayes' rule, this can be split into the product of an observable and latent conditional probability function

$$\mathcal{L}(\theta \mid \mathbf{X}, \mathbf{Y}) = P(\mathbf{X}, \mathbf{Y} \mid \theta) = P(\mathbf{Y} \mid \mathbf{X}, \theta) P(\mathbf{X} \mid \theta). \tag{77}$$

For the sake of argument, assume the set $\mathbf{X}$ and the parameter set θ to be constants, such that one is left with a single random variable Y, distributed as $f(Y \mid \mathbf{X}, \theta)$. Intuitively, the best possible guess for the value of Y is the mean, or expected value, of $f(y \mid \mathbf{X}, \theta)$. The next iterative estimate of θ should therefore take into account the distribution of the latent variables as well as the (log) likelihood function, or

$$\begin{aligned} Q(\theta, \theta_t) &= E(\ell(\theta \mid \mathbf{X}, \mathbf{Y}) \mid \mathbf{X}, \theta_t) && (78) \\ &= \int_{\mathbf{Y}} \log(\mathbf{X}, Y \mid \theta) f(Y \mid \mathbf{X}, \theta_t)\, dy\,. && (79) \end{aligned}$$

The iterative estimate $Q(\theta, \theta_t)$ is now a deterministic function easily maximised to find the EM equivalent of the maximum likelihood estimator. This estimator is used as the initial guess for the next iteration of Eq. 78. The iteration rules are thus,

$$\begin{aligned} \theta_{t+1} &= \arg\max_{\theta} Q(\theta, \theta_t), \\ \theta_t &\leftarrow \theta_{t+1}. \end{aligned} \tag{80}$$

Note that a direct implication of Eq. 80 is that the choice for θ_{t+1} must increase Q, such that the next estimate always improves the likelihood over the first current, or

$$Q(\theta_{t+1}, \theta_t) \geq Q(\theta, \theta_t). \tag{81}$$

Monotonicity Ideally, the path of convergence of the EM method follows is monotonically increasing. The implication of this would be that every next estimate increases the likelihood function of the parameters, intuitively providing one with a better estimate at every step. Otherwise,

$$\mathcal{L}(\theta_{t+1}) \geq \mathcal{L}(\theta_t). \tag{82}$$

Luckily, this is in fact the case. starting with the law of conditional probabilities, the likelihood function may be rewritten as

$$\begin{aligned} P(x, y \mid \theta) &= P(y \mid x, \theta) P(x \mid \theta) \\ P(x \mid \theta) &= \frac{P(x, y \mid \theta)}{P(y \mid x, \theta)} \\ \mathcal{L}(\theta \mid x) &= \frac{\mathcal{L}(\theta \mid x, y)}{P(y \mid x, \theta)} \\ \ell(\theta \mid x) &= \ell(\theta \mid x, y) - \log P(y \mid x, \theta). \end{aligned} \tag{83}$$

Similar to Eq. 78, take the expected value of the latent variable, noting that the left-hand side is already constant in terms of y. The first term is simply Eq. 78, denoted by $Q(\theta, \theta_t)$, and the second term is typically denoted by $H(\theta, \theta_t)$[14]. The difference that needs to be optimised then, is

$$\begin{aligned} \ell(\theta \mid x) &= E(\ell(\theta \mid x, y) \mid X, \theta_t) - E(\log P(y \mid x, \theta) \mid X, \theta_t) \\ &= Q(\theta, \theta_t) - H(\theta, \theta_t). \end{aligned} \tag{84}$$

Taking the first difference in the likelihood function, such that Eq. 84 resembles the original definition as in Eq. 82, albeit in log-space,

$$\begin{aligned} \ell(\theta_{t+1} \mid x) - \ell(\theta_t \mid x) = &[Q(\theta_{t+1}, \theta_t) - Q(\theta_t, \theta_t)] \quad (1) \\ &- [H(\theta_{t+1}, \theta_t) - H(\theta_t, \theta_t)] \quad (2). \end{aligned} \tag{85}$$

For Eq. 82 to hold, term (1) must be positive and term (2) must be non-positive. Note that the M step explicitly sets the difference in the expectation function to be positive, according to Eq. 81. Making use of Jensen's inequality for concave function, $E(\phi(x)) \leq \phi(E(x))$, and noting the strict concavity of the logarithm function [Weisstein a], monotonicity can finally be proved,

$$\begin{aligned} H(\theta, \theta_t) - H(\theta_t, \theta_t) &= E(\log P(y \mid x, \theta) - \log P(y \mid x, \theta_t)) \\ &= E(\log \frac{P(y \mid x, \theta)}{P(y \mid x, \theta_t)} \mid x, \theta_t) \\ &\leq \log E(\frac{P(y \mid x, \theta)}{P(y \mid x, \theta_t)} \mid x, \theta_t) \\ &= \log \int_Y \frac{P(y \mid x, \theta)}{P(y \mid x, \theta_t)} P(y \mid x, \theta_t)\, dy \\ &= \log \int_Y P(y \mid x, \theta)\, dy \\ &= \log 1 \\ &= 0 \\ H(\theta, \theta_t) - H(\theta_t, \theta_t) &\leq 0. \end{aligned} \tag{86}$$

[14] Recall that the entropy function as also labelled as $H(.)$. Without elaboration, there is a link between these quantities.

Therefore, it may be assumed that Eq. 82 holds, via Eq. 85, and that the log-likelihood increase by introduced by the EM method is monotonically increasing.

Convergence For any estimator, a basic criterion would be that of convergence to the truth value of θ as the number of iterations of approximations tends towards infinity. If monotonicity is the answer to whether EM iterations move upwards or not, the convergence property is the answer to where these increases eventually lead. Ideally, the point of the convergence is the global maximum, or at the very least a local minimum close to the global likelihood maximum. Unfortunately, this is not an easy exercise.

The standard work on EM, that of McLachlan and Krishnan discusses the theoretical convergence of EM in detail [McLachlan and Krishnan2007]. The primary work motivating their discussion comes from a series of classical papers by Wu [Wu et al.1983]. The papers note that under certain regularity conditions, EM will monotonically increase until it reaches a saddle-point, however no guarantee is given as to this being the global optimum. A relatively recent paper ensures that EM will converge to the MLE estimate within a finite set of iterations, both using a theoretical and empirical approach [Balakrishnan et al.2017]. A very recent by Wu et al. further gives concrete proofs of convergence rates and their relationship to sample size [Wu et al.2016].

5.1.3 EM for Hidden Markov Models

From the discussion on the evaluation and decoding problems in HMMs, three important quantities were defined[15]:

$$\begin{aligned}
&\text{Forwards} \quad \alpha_t(q_{i,t}) \equiv P(o_1, \ldots, o_t, q_t \,|\, \theta) \\
&\text{Backwards} \quad \beta_{t+1}(q_{j,t}) \equiv P(o_{t+1}, \ldots, o_T \,|\, q_t, \theta) \\
&\text{Posterior} \quad \gamma(q_t) \equiv P(q_t \,|\, O, \theta) \propto \alpha(q_t)\beta(q_t)
\end{aligned}$$

For the EM algorithm, typically referred to as the Baum-Welch algorithm[16], a fourth quantity needs to be defined, namely the two-time slice posterior; read as the probability of taking state $q_{i,t}$ and then taking state $q_{j,t+1}$ given the observation sequence (see Fig. 5.1):

$$\text{2TS Posterior} \quad \xi(q_t, q_{t+1}) \equiv P(q_t, q_{t+1} \,|\, O, \theta). \tag{87}$$

[15] Note the addition of conditional dependence on the model parameters. Where it was implied in previous chapters, this chapter seeks to explicitly infer these values.

[16] The work of Baum and colleagues for the specific application to HMMs [Baum and Petrie1966] [Baum et al.1970] precedes the more general, but seminal, work of Dempster-Laird-Ruben [Dempster et al.1977]

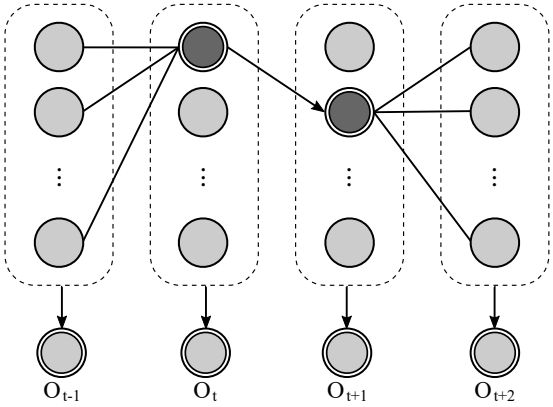

Fig. 5.1: A graphical depiction of the reasoning behind the two-time slice posterior.

Using reasoning common to Ch. 4, one can derive a formulation of the 2TS-posterior as follows,

$$\xi(q_{i,t}, q_{j,t+1}) = \frac{\alpha_t(q_{i,t}) a_{i,j} b_j(o_{t+1}) \beta_{t+1}(q_{j,t})}{P(O \mid \theta)} \tag{88}$$

$$= \frac{\alpha_t(q_{i,t}) a_{i,j} b_j(o_{t+1}) \beta_{t+1}(q_{j,t})}{\sum_i \sum_j \alpha_t(q_{i,t}) a_{i,j} b_j(o_{t+1}) \beta_{t+1}(q_{j,t})}. \tag{89}$$

The principle behind the two time slice posterior is given in Fig. 5.1, and an graphical depiction using the "Balls in Containers" example is given in Fig. 5.2. Note the shading of the edges as opposed to the shading of the nodes in Fig. 5.2.

Consider the posterior as the steady-state frequency of visiting state q_i at time t. The sum over all t renders this as the frequency of transitions into q_i for all t,

$$\sum_{\forall t} \gamma(q_{i,t}) = E(N_i) = \text{Expected frequency of transitions into } q_i$$

$$\sum_{\forall t} \xi(q_{i,t}, q_{j,t+1}) = E(N_{i,j}) = \text{Expected frequency of transitions from } q_i \text{ to } q_j$$

Expectation Step Recall the definition of $P(O, Q \mid \lambda)$, such that rewritten in terms of the collection of observations sequences $\mathcal{O}$, the likelihood and log-likelihood function takes the form of

$$\mathcal{L}(\theta \mid O) = \prod_{O=1}^{|\mathcal{O}|} \pi_i b_{q_i}(o_1) \prod_{t=1}^{|T|} a(i,j) b_{q_i}(o_t) \tag{90}$$

$$\ell(\theta \mid O) = \sum_{O=1}^{|\mathcal{O}|} [\log \pi_i + \sum_{t=2}^{|T|} \log a_t(i,j) + \sum_{t=1}^{|T|} \log b_t(q_i)]. \tag{91}$$

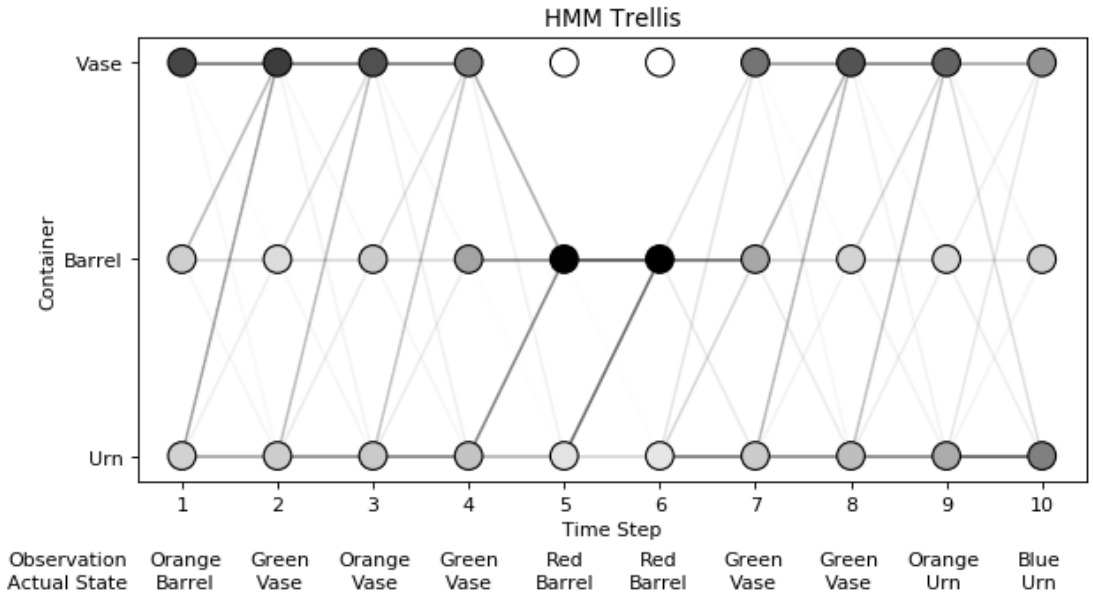

Fig. 5.2: A graphical overview of the posterior and 2TS-posterior distributions in a trellis diagram representing an HMM.

Note the definition of the expectation function as Eq. 78. The discussion of HMMs has so far limited itself to multinomial discrete as opposed to continuous distributions, implying that the expected value integral in Eq. 78 be replaced by a sum. Furthermore, rather than training a set of θ, the Baum-Welch algorithm specifically trains the HMM $\lambda = \{\pi_i, A, B\}$. Inserting the log-likelihood function as Eq. 91 into the expectation function gives

$$\begin{aligned} Q(\lambda, \lambda_t) &= E(\ell(\theta \,|\, O)) \\ &= E(\sum_{O=1}^{|\mathcal{O}|} \log \pi_i) + E(\sum_{O=1}^{|\mathcal{O}|} \sum_{t=2}^{|T|} \log a_t(i,j)) + E(\sum_{O=1}^{|\mathcal{O}|} \sum_{t=1}^{|T|} \log b_t(q_i)) \qquad (92) \\ &= \sum_{i=1}^{|Q|} \sum_{O=1}^{|\mathcal{O}|} \log \pi_i P(q_{i,t} \,|\, O, \lambda) + \sum_{i=1}^{|Q|} \sum_{O=1}^{|\mathcal{O}|} \sum_{t=2}^{|T|} \log a_t(i,j) P(q_{i,t} \,|\, O, \lambda) \\ &\quad + \sum_{i=1}^{|Q|} \sum_{O=1}^{|\mathcal{O}|} \sum_{t=1}^{|T|} \log b_t(q_i) P(q_{i,t} \,|\, O, \lambda). \qquad (93) \end{aligned}$$

The probability functions used as weighting terms can be rewritten to reflect the quantities in each term

$$\begin{aligned} Q(\lambda, \lambda_t) = &\sum_{i=1}^{|Q|} \sum_{O=1}^{|\mathcal{O}|} \log \pi_i P(q_{i,t=1} \,|\, O, \lambda) + \sum_{i=1}^{|Q|} \sum_{O=1}^{|\mathcal{O}|} \sum_{t=2}^{|T|} \log a_t(i,j) P(q_{i,t}, q_{j,t+1} \,|\, O, \lambda) \\ &+ \sum_{i=1}^{|Q|} \sum_{O=1}^{|\mathcal{O}|} \sum_{t=1}^{|T|} \log b_t(q_i) P(q_i \,|\, O, \lambda) I(o_t). \end{aligned} \qquad (94)$$

Algorithm 4: Baum-Welch Learning

Input : N states, $|\mathcal{O}| \cdot T$ observations, Initial HMM: $\lambda = \{\hat{A}, \hat{B}, \hat{\pi}\}$, Tolerance limit ϵ
Output: Trained HMM: $\lambda = \{\hat{A}, \hat{B}, \hat{\pi}\}$
while $\Delta Q(\lambda, \lambda_t) \geq \epsilon$ **do**
  Expectation Step
  for $\forall O$ **do**
    for $\forall t, i, j$ **do**
      $\alpha_{i,1:T} = \texttt{forwards}(\hat{A}, \hat{B}, \hat{\pi})$
      $\beta_{i,1:T} = \texttt{backwards}(\hat{A}, \hat{B}, \hat{\pi})$
      $\gamma(q_{i,t}) = (\alpha_{i,1:t}\beta_{i,t:T})/(\sum_i \alpha_{i,1:t}\beta_{i,t+1:T})$
      $\xi(q_{i,t}, q_{j,t}) = (\alpha_{i,1:t}a(q_i, q_j)b(q_i, o_t)\beta_{i,t+1:T})/(\sum_i \alpha_{i,1:t}a(q_i, q_j)b(q_i, o_t)\beta_{i,t+1:T})$
    end
  end
  Maximisation Step
  $\pi_i = \frac{1}{|\mathcal{O}|}\sum_{O=1}^{|\mathcal{O}|}\gamma(q_{i,1})$
  $a_{ij} = (\sum_{t=1}^{T}\xi(q_{i,t}, q_{j,t+1}))/(\sum_{t=1}^{T}\gamma(q_{i,t}))$
  $b_t(q_i) = (\sum_{t=1}^{T}\gamma(q_{i,1})I(o_t))/(\sum_{t=1}^{T}\gamma(q_{i,t}))$
end
return $\lambda = \{\hat{A}, \hat{B}, \hat{\pi}\}$

Hence, given the previous estimate of λ, the expectation function is easily found using $\gamma(q_{i,t=1})$ and $\xi(q_{i,t}, q_{j,t+1})$. The E-step of the Baum-Welch algorithm can thus be summarised simply as, calculating the quantities $\gamma(q_{i,t}), \xi(q_{i,t}, q_{j,t+1})$.

Maximisation Step The maximisers of expectation function are found by setting the derivative to zero. While relevant, the derivation using Lagrangian multipliers is lengthy and takes away from the discussion of Hidden Markov Models. The derivation may be found in Appendix A. The estimated quantities used in the iterative steps of EM are

$$\pi_i = \frac{1}{|\mathcal{O}|}\sum_{O=1}^{|\mathcal{O}|}\gamma(q_{i,1}), \tag{95}$$

$$a_{ij} = \frac{\sum_{t=1}^{T}\xi(q_{i,t}, q_{j,t+1})}{\sum_{t=1}^{T}\gamma(q_{i,t})}, \tag{96}$$

$$b_t(q_i) = \frac{\sum_{t=1}^{T}\gamma(q_{i,1})I(o_t)}{\sum_{t=1}^{T}\gamma(q_{i,t})}. \tag{97}$$

A pseudo-code depiction of the whole algorithm is present in Algorithm 4. Here ϵ gives the lower bound to the change in the auxiliary function. In actual applications, rather than looking at whether $Q(.)$ converges (with a lot of sums over the data), it is common to evaluate the accuracy or some error function instead.

Baum-Welch While the proof of monotonicty of EM, of which Baum-Welch is just a special case, and the limited proofs of convergence to local optima is crucial for the use

of EM as a parameter estimation technique, it does imply great reliance on initial conditions. While the algorithm could theoretically run without any labelled data, the distance between the eventual convergence point and the global optimum value is dependent on the starting positions provided. As such, it is common to use a priori knowledge or simple estimation techniques to initialise the HMM before EM iterating.

Rabiner notes that using simple uniform estimates for the A and π parameters will usually work well enough, but that it is especially the B parameter that requires good initial estimates [Rabiner1989]. Proposed methods include manual annotation, segmenting via means of observations within states or the K-means clustering algorithm (perhaps ironically, using EM for EM). Within contemporary speech and language processing, the initial state and transition parameters are set by hand, and only the emmissions parameters are optimized via Baum-Welch [Martin and Jurafsky2018]. Murphy further recommends using a process similar to cross-validation or determinstic annealing to mitigate the effect of potential poor local optima [Murphy2012].

5.2 Learning for CRFs

While CRFs also typically still use MLE or similar methods, the log-likelihood function is sufficiently complex that it cannot be solved in closed form. As such, approximate gradient descent methods are commonly used, like the quasi-Newtonian BFGS algorithm or algorithms from the stochastic gradient descent family. Regardless, training CRFs, whether with or without latent variables, is incredibly computationally expensive. In fact, training is typically only tractable for linear chain and tree architectures [Sutton et al.2012].

The following sections will provide the likelihood function that needs to be optimised and a high-level overview of some methods for minimising. However, these methods fall far outside of the scope of this thesis, and are described only very briefly.

5.2.1 Likelihood Functions for CRFs

Recall the definition of a general CRF, such that the log-likelihood of follows as

$$P(\mathbf{Y} \mid \mathbf{X}, \theta) = \frac{1}{Z(\mathbf{X})} \exp\{\theta_y f(y_t, y_{t-1}, o_t)\} \implies$$
$$\ell(\theta \mid \mathcal{O}) = \sum_{\mathcal{O}} \sum_{t \in \mathbf{O}} \sum_{\mathbf{Y}} \theta_y f(y_t, y_{t-1}, o_t) - \sum_{\mathcal{O}} \log Z(\mathbf{X}). \tag{98}$$

Where Z is the standard partition function, or simply the left term with the label variables $\mathbf{Y}$ marginalised out. This is the likelihood function that needs to be maximised, implying that the optimal solution for θ is found using the first derivative. The partial derivative

w.r.t. to the parameters is

$$\begin{aligned}
&\arg\max_{\theta_y} \ell(\theta \mid \mathcal{O}) \implies \frac{\partial}{\partial \theta_y}\ell(\theta \mid \mathcal{O}) = 0 \\
&\frac{\partial}{\theta_y}\ell(\theta \mid \mathcal{O}) = \sum_{\mathcal{O}}\sum_{t\in\mathbf{O}} f(y_t, y_{t-1}, o_t) - \frac{\partial}{\partial \theta_y}\sum_{\mathcal{O}} \log Z \\
&\to \frac{\partial}{\partial \theta_y}\log Z = \sum_{\mathbf{O}}\sum_{Y} f(y_t, y_{t-1}, o_t) P(y \mid \mathbf{O}) \\
&\frac{\partial}{\partial \theta_y}\ell(\theta \mid \mathcal{O}) = \sum_{\mathcal{O}}\sum_{t\in\mathbf{O}} f(y_t, y_{t-1}, o_t) - E(f(y_t, y_{t-1}, o_t)).
\end{aligned} \tag{99}$$

Here the partition function is replaced by the expectation value of the feature functions under the model distribution, a standard result for all exponential family distributions. The left term can instead be interpreted as the expectation value of the feature functions under the empirical distribution that matches the labels to the data [Sutton et al.2012]. Thus, at a maximum of the log-likelihood, the expected value of the feature functions defined by the data is equivalent to that of the expected value defined by the parameters.

For CRF models with latent variables, the log-likelihood takes a very similar form, except that a sum must be ran over the hidden variables prior to computing the log-likelihood gradient, or symbolically,

$$\ell(\theta \mid \mathcal{O}) = \log \sum_{Q} P(Y, Q \mid X, \theta). \tag{100}$$

Without derivation, the gradient of the log-likelihood is again given by the difference of two expectations, now one in terms of the hidden variables, and one in terms of the observations:

$$\begin{aligned}
\frac{\partial}{\partial \theta_y}\ell(\theta \mid O) &= \sum_{Q} f(h_t, y_t, y_{t+1}, o_t) P(H \mid Y, X) - \sum_{Q}\sum_{Y} f(h_t, y_t, y_{t+1}, o_t) P(H, Y \mid X) \\
&= E_{H \mid O,Y}(f(h_t, y_t, y_{t+1}, o_t)) - E_{H,Y \mid O}(f(h_t, y_t, y_{t+1}, o_t)).
\end{aligned} \tag{101}$$

A natural choice for maximising these likelihood functions, given the Baum-Welch algorithm being the default learning algorithm for HMMs, would be a variant of EM. This, however, is not necessarily possible. McLachlan and Krishnan note that the EM algorithm applied to the exponential family of distributions is already difficult to carry out, even with purely numerical methods [McLachlan and Krishnan2007]. A CRF is considerably more difficult, and as such rarely suitable. They do note that using a suitable Gibbs sampler can lead to tractable learning. As such, it is hardly surprising that model developers tend to choose more general learning methods.

Quasi-Newtonian Methods One of the simplest methods for unconstrained optimisation methods is gradient descent (not to be confused with *stochastic* gradient descent), which is simply given as,

$$\theta_{i+1} = \theta_i - \eta \nabla \ell(\theta \mid \mathbf{O}).$$

The next estimate of the parameters is given by a 'step' towards the minimum given by the gradient of the likelihood. Thus, given many iterations, it is expected that gradient descent falls towards the minimum of the function. Intuitively, gradient descent generates a linear approximation of the function to be minimised, placing the next estimate of the parameters a step-size η along the linearised function.

A faster converging method is one that approximates the function as a quadratic or second-order curve. Newton's method is the simplest method making use of such second-order approximations, however also incredibly computationally expensive due to the inclusion of the inverse Hessian (i.e. the second-order Jacobian or nabla operator). It's updates follow as,

$$\theta_{i+1} = \theta_i - \eta \frac{\nabla \ell(\theta \mid \mathbf{O})}{\nabla^2 \ell(\theta \mid \mathbf{O})}.$$

In many applications, computing the Hessian and its inverse is too computationally expensive. In attempts to still apply a quadratic approximation for parameter updating, quasi-Newtonian methods exist. The most common of these is the (L-)BFGS method. Rather than computing the Hessian inverse at every iteration, it is estimated from the gradients of previous iterations, something like,

$$\theta_{i+1} = \theta_i - \eta C \nabla \ell(\theta \mid \mathbf{O}).$$

where $C \approx (\nabla^2 \ell(\theta \mid \mathbf{O}))^{-1}$.

Stochastic Gradient Descent The Newtonian and quasi-Newtonian methods fall in the class of *batch* algorithms: the whole data set is needed to compute the loss and gradient at each iteration. It takes little convincing that this can be a costly process, with a great number of operations needed just to perform inference for all observations. Most of these observations will most likely only provide minimal additional information, with similar samples likely describing a similar loss and gradient given the current model parameters. As such, a viable and practical approximation to using the whole training dataset is to use a very small subset, randomly selected, and iterating a set number of times. Stochastic gradient descent (SGD), and its many variants, make use of exactly this assumption. The gradient for a CRF was already derived in the batch setting, but this can be re-expressed for a single data point,

$$\frac{\partial}{\theta_y} \ell(\theta \mid \mathbf{O}) = \sum_{t \in \mathbf{O}} f(y_t, y_{t-1}, o_t) - \sum_{\mathbf{O}} \sum_{Y} f(y_t, y_{t-1}, o_t) P(y \mid \mathbf{O}),$$

where the only difference is the loss of the sum over the whole batch, $\mathcal{O}$. The update rule used by vanilla SGD is to step in the direction of the gradient based on the stochastically chosen sample, mediated by a hyper-parameter that determines learning speed, here depicted as ν. Symbolically, this may be depicted as,

$$\theta_{i+1} = \theta_i - \eta \frac{\partial}{\theta_y} \ell(\theta \mid \mathbf{O}) \tag{102}$$

SGD is a very common optimisation technique that has found success in a wide array of machine learning applications. This includes parameter estimation problems for conditional random fields. The first application of SGD (and a variant called stochastic meta descent) is attributed to Vishwanathan et al. [Vishwanathan et al.2006], indicating performance better than BFGS. In general the choice between BFGS and SGD is an interesting one, especially when considering the large genre of literature on speeding up and parallelising SGD methods: while BFGS achieves better improvement at every *single* iteration, SGD can compute worse iterations at a much faster pace [Sutton et al.2012].

Chapter Summary

This chapter was the last to focus on the theory of PGMs for structured data. Special attention was given to the Baum-Welch algorithm for training HMMs. This was derived from the general formulation of the EM algorithm. Basic properties of the EM algorithm were presented, namely those of monotonicity and convergence. While its senior, the Baum-Welch algorithm can be derived from the EM algorithm using the method of Lagrangian multipliers. The eventual maximisation step was presented as three relatively easy to estimate quantities already defined. The learning methods for CRFs were presented much more briefly. The log-likelihood function was presented for several topologies, taken mainly from literature on the topic. Lastly, two frameworks for optimising the CRF log-likelihood function were presented, given that EM is intractable for realistic applications.

Chapter 6

Probabilistic Graphical Models for Automatic Speech Recognition

Automatic speech recognition, in a sense connecting the most human form of communication to the most used communication method, has received interest from the artificial intelligence community for the past 5 decades. Despite the extensive efforts, true human-like recognition remains far out of reach.

That does not imply that success has been lacking. The initially defined tasks in the 1970s now achieve almost perfect accuracy, and more difficult tasks defined after, now achieve reliable accuracy. Specifically, when this chapter refers to ASR, it also implies the following properties:

1. **Large Vocabulary**: rather than focusing on a specific subset of the English language, like code-words or numerals, the training and testing utterances include a vocabulary sufficiently large enough to make learning the dictionary intractable
2. **Continuous Speech**: the training utterances closely reflect human speech, such that no conscious pause it taken between words. While sentences can be initiated and conclude with silence, and natural or voiced pauses occur between words, these are sufficiently short enough to represent a natural cadence
3. **Speaker Independence**: the training utterances are spoken by a large variety of people, of varying dialect regions and equally distributed over genders, such that recognition does not require a user-specific training period before achieving the desired accuracy
4. **Laboratory Set-up**: speech is generated in a laboratory setting, implying no background noise, a consistent recording methodology and close distance of speaker to the microphone. Thus, while the previous conditions are intended to emulate human-like verbal communication, this condition prevents robustness to situations in which speech occurs

The standard ASR pipeline (the system that takes an utterance in raw encoded audio and produces a textual representation of the sentence) is comprised of three components: a feature representation block, an acoustic model and a language model. The feature representation block takes the raw audio, by necessity digitised, and converts the waveform into a set of trainable features. The acoustic model performs classification, generally at the level of phonemes[17], on the generated features. The language model takes the classification results and brings it to a understandable level, thus going from

[17] The individual, context independent sounds whose combinations form words.

a collection of digitised sounds to a full set of words of sentences. Otherwise, the acoustic model ensures accurate prediction of individual sounds, whereas the language model ensures these sounds form a syntactically correct and coherent utterance. The different tasks within ASR within the context of the TIMIT corpus are also summarised in Table 5.

This chapter will first discuss the necessary steps in creating a PGM-based large-vocabulary continuous speech recognition (LVCSR) pipeline before showing an application on the classical TIMIT dataset. First speech and its transformation to a classifiable features. This follows the standard 39-dimensional MFCC process. The section after discusses advances of PGMs, specifically HMMs and CRFs, for speech recognition. The papers highlighted follow a directly comparable experimental set-up and are all trained and evaluated using the same TIMIT dataset. Lastly, Sec. 6.4 discusses in detail the experimental procedure used for both a HMM and HCRF acoustic models, before presenting results comparable to those of discussed papers.

6.1 Speech Representation

A common, if not ubiquitous, feature generation method used in automatic speech recognition tasks are the Mel-Frequency Cepstral Coefficients (MFCC) (see for example [Huang et al.2001]). This technique combines a number of advances within digital signal and speech processing to condense a time-series waveform into machine readable data that emulates human perceived hearing. Going from waveform input to MFCC features typically involves 4 steps. Each step is outlined briefly below, providing some motivation behind them, under the assumption that the reader already possesses some understanding of digital signal processing.

Framing and Windowing Rather than analysing the entirety of the signal, whose non-stationary behaviour is exactly the phenomenon one wishes to model, the waveform of the utterance is segmented into many small 'frames'. A typical frame duration is 20ms, with a small overlap between neighbouring frames. Ultimately, these frames are converted to their frequency components and analysed. However, it should be noted that the choice of the framing will already impact the eventual frequency spectrum returned.

Windowing is the practise of multiplying a signal sample in the time domain, to attenuate or amplify regions in the frequency domain. The naive slicing of the waveform into frames is effectively the equivalent to using many rectangular, low-pass filters with overlap. While the signal remains unchanged, the discrete Fourier transform can show 'spectral leakage'; a phenomenon where non-existent frequencies are amplified while frequencies that are present are attenuated. As opposed to showing a single peak for a single tone signal, as an example, the peak is spread over multiple frequencies. Common causes are noisy or discontinuous signals or including a non-integer amount of periods in the frame. Indicative of spectral leakage are raised 'side-lobes'; non-dominant frequencies that obfuscate peaks in the frequency plot. Common windows applied to speech signals

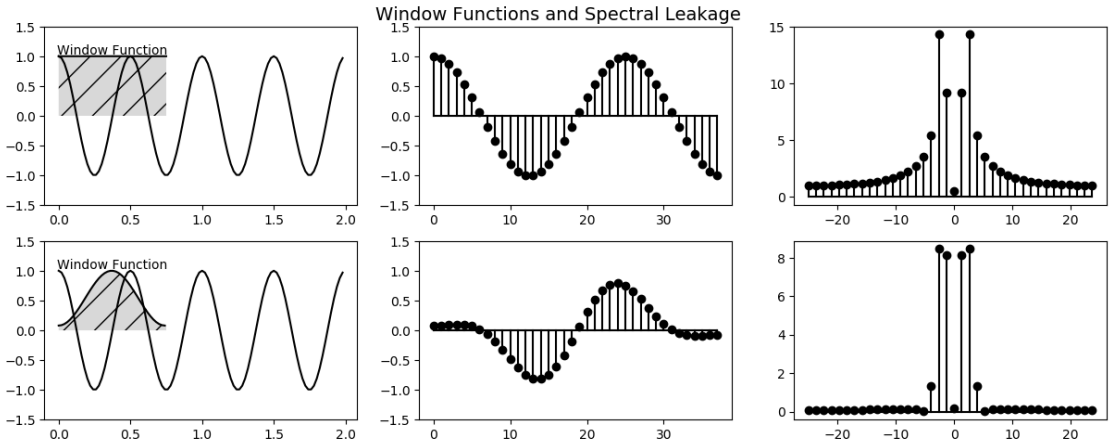

Fig. 6.1: Spectral leakage and counteracting using a Hamming window of a simple cosine function. The first column shows the functions and the respective windowing function, the second the effective signal analysed and lastly the frequency representation. Note the reduced magnitude of the 'side' frequencies in the signal treated with a Hamming window.

are the Hamming (double 'm') and Hanning (double 'n') functions. For an overview of window functions and their properties, consult Harris' [Harris1978]. For a more visual, practical explanation consult Lyons' article [Lyons1998]. A visual example of spectral leakage using a simple cosine wave and a Hamming window is shown in Fig. 6.1.

Short-Time Discrete Fourier Transform The next step, after having collected the windowed frames, is converting each frame to its frequency representation using the discrete Fourier transform (DFT). Briefly, without considering motivation of derivation, the DFT follows as,

$$\mathcal{F}(x) = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn}. \tag{103}$$

Thus, DFT models the framed signal as a series of superimposed cycloids, and analysis moves to the frequency domain as opposed to the time domain. The DFT output is another signal except with x-values being frequencies and their y-values as the dominance of the frequency in the original time signal. Spectrograms, see the second subfigure of Fig. 6.3, join the transpose of many short-window Fourier transforms together to create a graph that already shows patterns that can distinguish voice signals. Rather than looking at the frequency domain of each window individually, an expert looks at patterns across many windows, observing transitions between dominant frequencies.

Mel Filterbank While the Fourier transform is already much closer to creating computer readable voice signals, these are not similar to how humans perceive sound. The

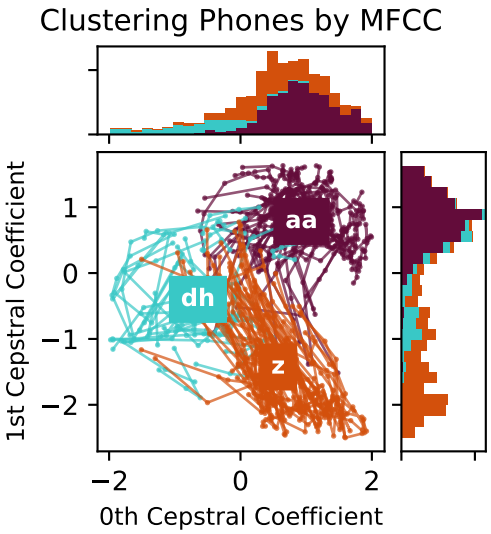

Fig. 6.2: Discriminating phones by their MFCC profile. Here only the 0th and 1st cepstral coefficients are used, for only 3 phones. In actuality, this would be done with 39 MFCC coefficients and 40+ phones. Each colour denotes a phone, with each dot a frame, connected by a line to directly neighbouring frames. The histograms represent a discrete attempt at capturing the marginal distributions of MFCC coefficient.

field of psycho-acoustics is entirely dedicated to transformations of physical signals to something more akin to that of human perceived sound [Stevens et al.1937]. A very common transformation is the Mel filter bank, a series of triangular filters applied to the frequency that attenuate higher frequencies in favour for lower frequencies. The motivation for the Mel frequency is typically attributed to Stevens and Volkmann in 1937 [Stevens et al.1937], and while more of a heuristic measure based on very limited experimental evidence, it remains recommended in standard works like *Spoken Language Processing* [Huang et al.2001]. The third subfigure in Fig. 6.3 shows the Mel filters applied to the spectogram immediately above it.

Cepstral Features The last, but generally most important step, comes from digital signal processing developments by Oppenheim and Schafer, outlined in their 2004 paper [Oppenheim and Schafer2004]. The Mel-frequency short-time Fourier transformed signals potentially is expert readable, individually these typically remain highly periodic. Purely in terms of classification, this remains as useless as the original time signal. Ideally, the period transformed signal can be summarised in a small set of continuous variables common to all observations.

A potential reason for the periodic nature of the transformed signal is the presence of echo, caused by the vocal tract during production of speech. Echo in speech is believed to be an additive delayed signal in the time-domain, such that in the frequency spectrum,

$$x(t) = s(t) + \alpha s(t-\tau) \overset{\mathcal{F}}{\to} |X(f)|^2 = |s(f)|^2(1 + \alpha^2 + 2\alpha \cos 2\pi f\tau),$$

which is again a linear combination of two signal under a log-transform. Taking the Fourier transform for a second time, bringing the frequency to the frequency of the frequency (humorously coined the quefrency), the frequency domain signal can be analysed in terms of the true signal components and the echo components. Generally, the signal components are found in the lower quefrencies whereas those in the higher regions correspond to echo. This analysis of the spectrum of a spectrum is coined the cepstral analysis. This in turn can be used to filter the system and include only those components relevant to analysis (coined liftering).

In practise, a second Fourier transform is not needed. Instead, the frequency domain signal is is fed into a discrete cosine transform (DCT), which produces a set of weighted cosines that produce a smoothed but accurate representation of the frequency signal. The more DCT coefficients are kept, the more weight is given to the echo in the time signal. The choice for a cosine transform is very deliberate, as these very closely approximate the eigenvalues of the frequency signal [Logan et al.2000].

It is standard practise to take the first 13 DCT coefficients (the 0th corresponds to the energy in the signal, the 1st to 12th correspond to the eigenvalues). For classification, it has also proven useful to include the first and second order differences [Huang et al.2001]. This concludes the generation of MFCC feature vectors, and leaves on with a matrix with 39 columns, corresponding to the MFCC coefficients and many rows, corresponding to the windowed signals. Such a matrix, although transposed to match with the spectograms, is shown in the last sub-figure in Fig. 6.3. The ability to cluster phonetic segments by their MFCC features is depicted in Fig. 6.2.

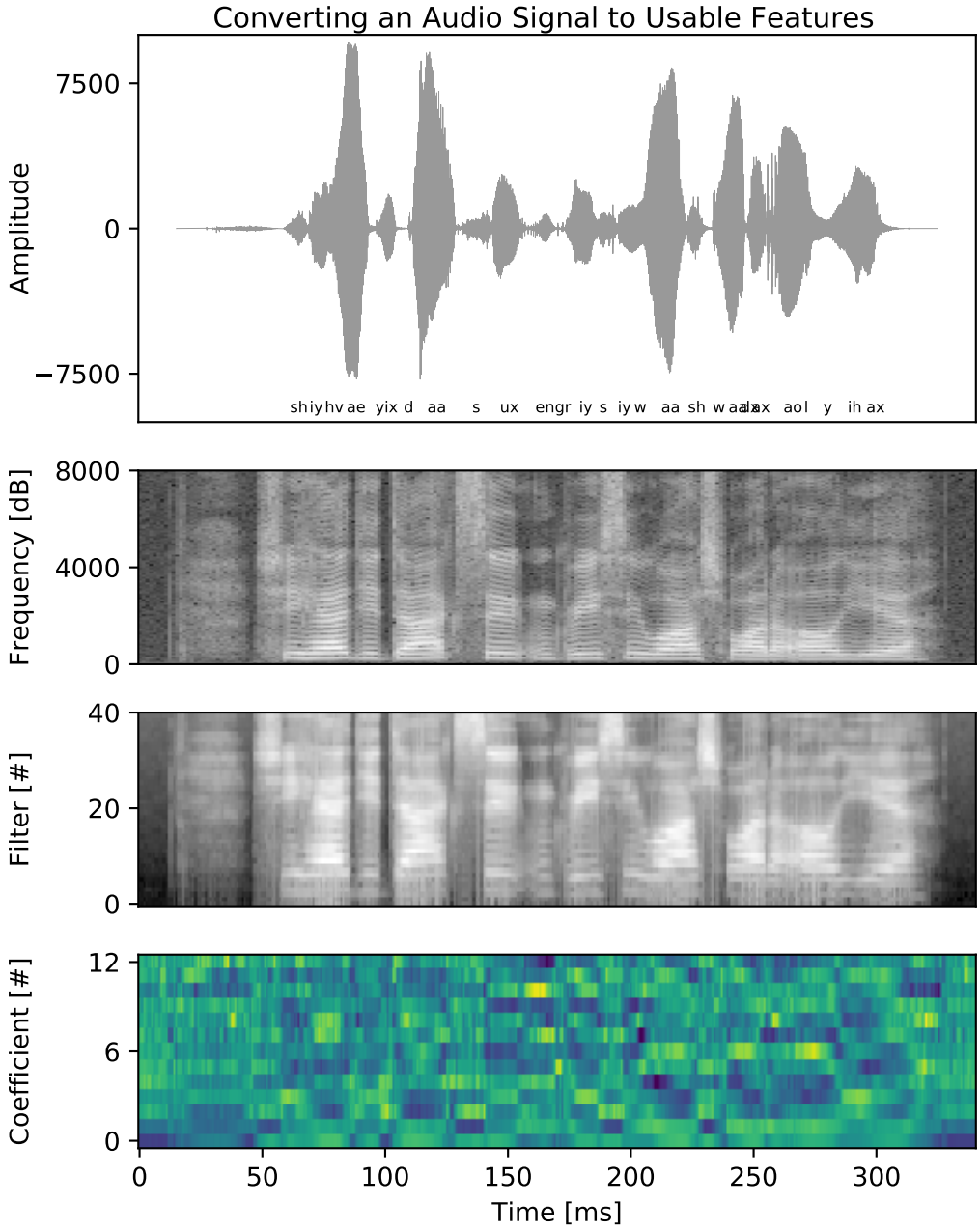

Fig. 6.3: The sentence "She had your dark suit in dirty wash water all year" represented as a discrete time signal, a spectogram, a log Mel-filterbank (N=40) and finally as cepstral coefficients.

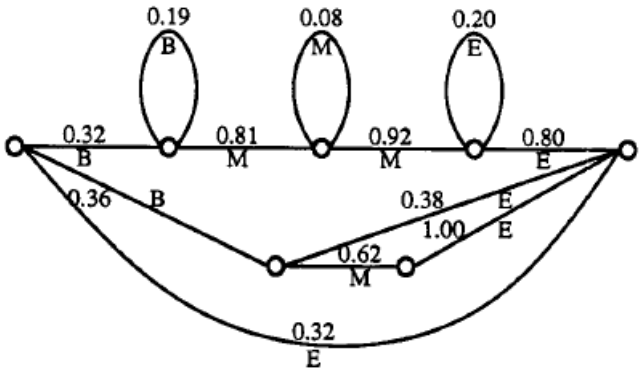

Fig. 6.4: The phonetic HMM for the phone /d/. Three primary states are defined, representing the beginning (B), the middle (M) and end (E) of the phonetic segment, with a start and end state further defined. Excluded are the empirical distributions for the emissions. Taken from [Lee and Hon1989].

6.2 Speech Modelling using Probabilistic Graphical Models

Throughout this thesis, a link between HMMs and speech recognition has been made. To some extent, the modern interpretation of HMMs as a sequential form of dynamic Bayesian networks is primarily motivated by their use within speech recognition models. Furthermore, despite their formulation in the 1970s and 1980s, when the available computational power forced the ASR models to limit itself to small vocabulary (e.g. the 0-9 digits, or a small subset of code-words) with speaker dependence, the technology proved extensible enough to form the core of many modern large-vocabulary continuous speech recognition pipelines.

The first HMM-based LVCSR structure is generally believed to be the DRAGON system [Baker1975]. Described by Baker are many of the fundamental components of modern systems, for example the use of linked models as a Markov process to represent phoneme segments, and a second model using the phone-predictions to decode lexical information. While the language used is somewhat archaic, there is clear reference to an HMM for the phone-level model, although no clear definition of the emission probabilities is made. The lexical-level model appears to be a form of a Bayesian network, although again, the details remains a little fuzzy.

First use of a HMM for the TIMIT corpus is attributed to Lee and Hon [Lopes and Perdigao2011, Lee and Hon1989]. Where the DRAGON system was vague in implementation details, Lee and Hon were explicit in virtually everything, and as such from the basis for subsequent state-of-the-art systems. The topology of the phone-level HMMs used by Lee and Hon are shown in Fig. 6.4. Characteristic of their phone models are the left-to-right transitions: a sub-phone state either transitions to itself or the next step (thus, the beginning of a phone either leads to a segment that is still the beginning, or the middle, etc.). The skip-transitions (in Fig. 6.4 all transitions below the top row) where

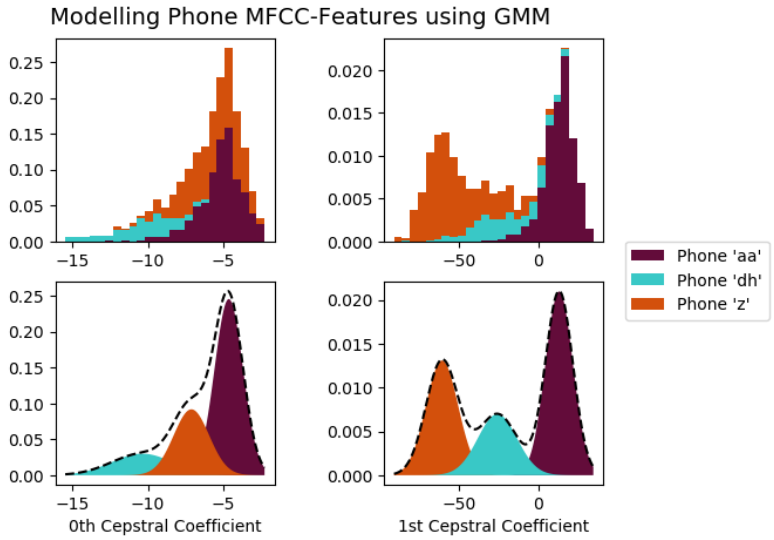

Fig. 6.5: The top row represent the marginal empirical distributions found in Fig. 6.2. The bottom row is achieved by modelling the MFCC features using a multivariate Gaussian. The dotted line represents the mixture of Gaussians fit over the undifferentiated distributions.

later found to not significantly increase model performance. Improvement in the phone classification system was booked by further considering context-dependence: the phone being modelled has its emission distributions tied to those neighbouring, overcoming surface level realisation differences due to the linguistic phenomenon of phonology[18]. This would, naively, require N^3 models of N phones. Keep in mind that beyond the cubed increase in computational complexity, this further requires greatly reducing the amount of data samples for each model. Lee and Hon approach this problem by using deleted interpolation to get the best of both: generalisability from the context independent models, and accuracy from the context dependent ones.

Beyond just using HMMs for phone classification, another larger-HMM was constructed for phone recongition. The end and start states of the individual phone model were tied together, such that each state of the recognition HMM is itself a smaller classification HMM. For decoding the whole utterance, a variant of the Viterbi algorithm was used (see Sec. 4.4). The achieved accuracy of their system was 64.07% for context independent training, and 73.80% for the context dependent models.

While other HMM speech recognition systems were created, and also specifically for the TIMIT corpus, there was little innovation with regards to PGMs until the use of the hidden CRF (HCRF) specifically for phone classification [Gunawardana et al.2005]. The Microsoft research team replaced the HMM with a CRF that had a layer of hidden subphone states feeding into a classifier node. Gunawardana et al. briefly showed that with the right feature choice, not one they ended up using, the HCRF was totally equivalent to

[18] Briefly, the pronunciation of phones changes based on the context around it.

the conditional HMM (see Sec. 3.2). Where Lee and Hon had represented their emission using the empirical distribution of features for their pre-segmented class, Gunawardana et al. opted for using the continuous Gaussian mixture model (GMMs). These models mix a set number of independent multivariate Gaussians to form a more general (i.e. multi-modal) distribution. The use of a multivariate Guassian for the phone MFCC dimensions is depicted in Fig. 6.5, corresponding to Fig. 6.2. This figure shows the Gaussians come close to modelling the individual emission distributions, but that a multi-modal mix of Gaussian can much more accurately model these features.

The HCRFs were left-to-right transitioning, much like the HMMs, with between 10 and 40 mixtures of Gaussians for each sub-phone state. Note that these models are purely for classification, and a recognition model much like the HMM case needed to be trained also. Gunawardana et al. used both L-BFGS and SGD training for their HCRFs, and compared their evaluative results against comparable HMMs. Their SGD trained CRFs achieved 79.7% and 78.3% accuracy on the development and test subsets respectively, whereas the best Baum-Welch trained HMMs achieved 74.9% and 73.6% instead. Thus, for phone classification, the conditional HCRF greatly outperformed the generative HMMs. Furthermore, note that HMMs with GMMs as emission probabilities and without context dependence, outperformed Lee and Hon's context dependent models on exact same data[19].

The last truly innovative PGM for LVCSR is the generalised HCRF defined by Sung and Jurafsky [Sung and Jurafsky2009]. Within the computer vision community, where the application of CRFs remains common, these models are referred to as latent-dynamic CRFs (LD-CRF). Rather than separating the classification and recognition models, with one training on sub-phone segments and one on classified phones, an LD-CRF trains both simultaneously, with a hidden layer of sub-phone states and a final layer of phone classification. Thus, where the HMM and the HCRF would train to classify single phones Y, the LD-CRF can classify a series of phones from the full utterance as $\mathcal{Y} = \{Y_1, \ldots, Y_T\}$. Furthermore, each hidden variable in the hidden layer contains not only sub-phones, but also a set of mixture components for modelling the observation. Much like the HCRF, the LD-CRFs were trained with between 8 and 64 Gaussian mixtures. In their experimental set-up, the LD-CRF achieved 71.7% accuracy, outperforming equivalent HMMs (68.4%) and CRFs (70.7%).

The models described, for the phone classification task, are also graphically depicted in Fig. 6.6. The three PGMs for LVCSR discussed have been chosen not just for their seminal status, but also their comparable experimental set-ups. Each paper used the same feature generation process (i.e. MFCC generation with specific parameters), on the TIMIT corpus, with similar train/development/test data subsets.

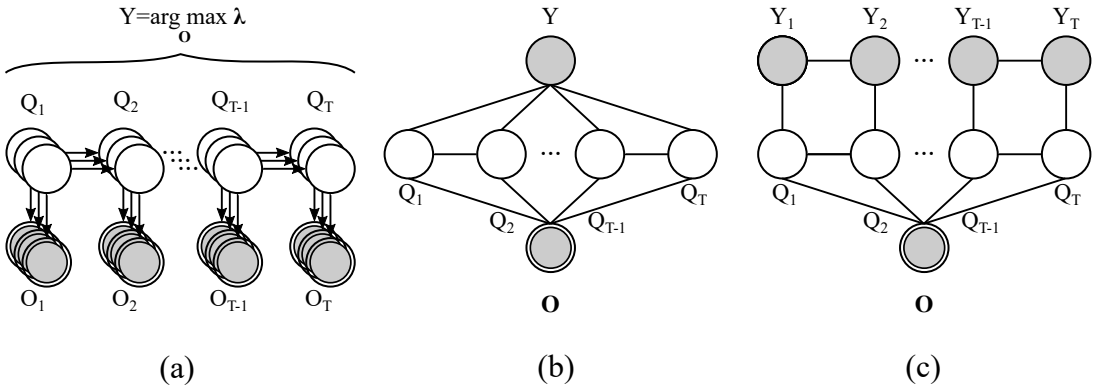

Fig. 6.6: Some common architectures of PGMs for speech classification. Subfigure (a) depicts a stack of phoneme level HMMs, where every HMM is trained for one specific phone, stacked to allow for word decoding, with typically the highest probability phone-model pair pointing to model used, i.e. a one-vs-all architecture. Subfigure (b) represents a Hidden CRF model, with a hidden layer of phones feeding in to an observed in training, but hidden in evaluation label node. Lastly, subfigure (c) represents a Latent Dynamic CRF, where two factorial layers are added. The first is the hidden layer, also found in HMMs and HCRFs, and the second is the label layer. Note that this model architecture combines an acoustic and language model into one, training on whole utterances (i.e. sentences) as opposed to isolated words.

6.3 TIMIT Dataset

The TIMIT (1988) corpus is a standard speech dataset used in automatic speech recognition. Developed as a joint effort by Texas Instruments (TI), MIT and the Stanford Research Institute, and verified by the National Institute of Standards and Technology, it contains 6300 sentences, or roughly 4 hours, of read sentences. Participants are equally distributed over 8 major dialect regions within the U.S.; from New England and New York City to Southern and the Midlands. The 630 speakers are roughly uniformly distributed across the dialect regions, with 70% being male. All sentences were transcribed, both for words and phonemes, and aligned to the utterance. The samples were taken using a single channel microphone at 16 kHz.

The sentences were deliberately designed to be phonetically and phonologically diverse, but balanced. Three classes are present. The "SA" are specifically meant to expose dialectical variations in speech. Two such sentences exist, one of which is displayed in Fig. 6.3, and each is uttered by every speaker. The 8 remaining sentences for each speaker are composed from the compact "SX" (5 per speaker) and the diverse "SI" (3 per speaker)

[19] It should be noted that this refers to the training data. Lee and Hon used a *very* small testing set, whereas Gunawardana et al. used the standard provided test set.

Table 5: The three tasks within TIMIT research, as described by Lopes and Perdigão [Lopes and Perdigao2011]. Reported metrics are accuracies of state-of-the-art systems within the noted year, or those of the cited PGM for ASR papers.

Task	Description	2011	PGMs
Segmentation	Finding boundaries between phones in utterances, made difficult by phonological interference.	93%	-
Classification	Correctly classifying the pre-segmented and labelled phone, typically using a competitive set of classifiers.	83%	79.3%
Recognition	Finding the most similar sequence of phones based of an utterance. Typically requires an additional language model.	79%	71.7%

classes. The compact class are specifically created for this task to allow for good coverage of phones, focusing on phonetic contexts assumed to be difficult, whereas the diverse class is taken from existing corpora, providing more natural and varying phonetic classes. Each "SX" sentence is recorded by 7 different speakers, while the "SI" only by one.

With the TIMIT files a suggested train-test subdivision is supplied. The test set contains sentences generated with very specific criteria [Garofolo et al.1993]:

1. Between 20 and 30% of data should be dedicated to training
2. No speaker should appear in both the training and testing subsets
3. All dialect regions should be included, with at least 1 male and female speaker
4. The amount of overlap of textual material between the training and testing subsets should be minimised
5. All phonemes should be covered in differing contexts

A minimal test set was generated to abide by these criteria. This 'core test set' includes 24 speakers for a total of 192 total texts. The "SA" class of sentences are removed to abide by the 4th testing criterion. The remaining test set speakers are made available for more comprehensive testing, although in practise this set is reserved for hyperparameter optimisation (i.e. a development set).

Lopes and Perdigão provide a modern overview of standard practises when dealing with the TIMIT corpus [Lopes and Perdigao2011]. Related to the distinctions made at the beginning of this chapter, they identify three broad domains of tasks for which the TIMIT corpus has been used as a benchmark. These have been summarised in Table 5. Of most relevance to this chapter is the classification task, for which the HCRFs performed most optimally [Gunawardana et al.2005]. Lopes and Perdigão describe phone classification as

> "Phonetic classification is an artificial but instructive problem in ASR. It takes the correctly segmented signal, but with unknown labels for the segments. The problem is to correctly identify the phones in those segments. Phone models compete against each other in an attempt to set their label to the respective segment. The label of the winning model is compared with the corresponding TIMIT label and a

> hit or an error occurs. Nevertheless, phone classification allows a good evaluation of the quality of the acoustic modelling, since it computes the performance of the recognizer without the use of any kind of grammar [...]" [Lopes and Perdigao2011]

The phone reduction mapping proposed by Lee and Hon is further expanded and modernised. The 39 typical remaining phones are dispayed in Appendix B, Table 7.

6.4 Experimental Results

This section seeks to validate the use of probabilistic graphical models for automatic speech recognition, specifically two model architectures for the TIMIT phone *classification* task, using isolated and transcribed phone segments. This task is the most simple of large-vocabulary speech recognition, and provides a good testing ground for the discussed models without delving too much into feature engineering and parameter optimisation. As such, the presented results do no come close to state-of-the-art or those presented as final results in the consulted literature, and are not expected to generalise to more modern systems.

Experiments were run on a 64 bit Windows machine, with 8 GB of RAM and an Intel Core i7 8550U CPU. Some experiments were instead run on Google's Colab cloud computing service. While the hardware made available to users is not always visible, a 25 GB RAM CPU (no hardware accelerator) run time was used for experiments. Python 3.7.3 was the primary used language, although some inferential algorithms were rewritten in Cython 3.0. Important packages used were `TIMIT Utils` and `Python Speech Features` for access to the TIMIT data directories and fast speech processing, `Pomegranate` for (GMM-)HMM models and a rewritten version of `pyHCRF`, along with the standard packages for scientific computing within Python (`NumPy`, `SciPy`, `Pandas`, `Matplotlib`, `Seaborn` and `Sklearn`).

A phone reduction method, as proposed by Lee and Hon [Lee and Hon1989], folding overlapping phones and silences into 39 more distinct labels was performed. The NIST recommended train/test/development for TIMIT was employed. Phones were treated as isolated units, with MFCC features being labelled by the phone closest to the start of the windowed data. Frames were collected via a Hamming window every 25 ms with a 10 ms overlap. Spectral analysis was performed with a 40 channel Mel filter bank, ranging from 64 to 8 kHz (half the sampling rate). A pre-emphasis coefficient of 0.97, a standard, was applied. The Cepstral coefficients were generated via a DCT, with the 0th and the first 12 coefficients kept. These vectors were appended with the first and second order differences. Finally, these were standardised. The same standardisation parameters were used for the train/test/development subsets. This left 963,888 phonetic sequences for training, and some 49,170 for testing. All in all, feature extraction and saving the complete data file to a JSON format took roughly 120 minute, leaving a 1 GB data file.

The first set of models used a HMM architecture. A left-to-right transition matrix with three states (Beginning, Middle, End) was used, without skip transitions. A start and

Table 6: Model phone prediction accuracy for the three TIMIT subsets.

Evaluation	HMM	HCRF
Training	56.64 %	67.96 %
Development	55.07 %	66.41 %
Testing	54.60 %	65.73 %

end state were also included. Each state was modelled using a 39-dimensional multivariate Gaussian with a diagonal covariance matrix (each dimension is independent of all other dimensions). Model initiation was conducted using the K-means algorithm, a close variant to the EM algorithm. The Baum-Welch algorithm was used for training. In total 40 (39 for phones, 1 for silence) models were trained and put into a single stack. Models that resulted in the highest predicted log-likelihood phone sequences were used as the predicted label.

The second set of models used a HCRF architecture. The actual models used stem from the computer vision community, initially described by Quattoni et al. in [Louis et al.] and [Quattoni et al.2005]. The transition model was unspecified, and thus potentially allowed to be fully ergodic. A multivariate Gaussian distribution was again used for the emission probabilities.

To avoid overfitting L2-regularisation was employed. This entails including a constraint term that penalises model complexity. For strong regularisation parameters, the more the model tends towards uniformity, whereas weak regularisation encourages closer fit to the provided training data. The importance of regularisation for HCRFs in speech recognition tasks was discussed by Sung et al. [Sung et al.2007]. A coarse to fine search with only a few iterations was performed on the development set prior to training, with the parameter that showed the best performance on unseen data being selected as optimal. A weight of 150 was found to be optimal for HCRFs on the TIMIT development set.

`SciPy`'s L-BFGS implementation was used to optimise the HCRF log-likelihood. Each iteration took roughly 9 minutes, with 100 needed before accuracy on the training data converged. Total training, including saving and evaluation, thus taking around 15 hours to complete. The complexity of parameter estimation for HCRFs is made clear when considering that 39 HMMs converged within 15 minutes[20]. All models and training data gave been available as a Google Colab iPython notebook[21].

The results of the models on phone classification are displayed in Table 6. The use of the conditional random fields are clearly considerably better than those of the generative hidden Markov model. Again, note that this is only these models are relevant only for the classification task, with no additional language model being constructed; the presented results may not be generalisable to the recognition task. Furthermore, the difference

[20] It should be noted that `Pomegranate`, the package used for HMM modelling, is professionally maintained and highly optimised, while the adapted `pyHCRF` code for HCRFs definitely is not.

[21] https://colab.research.google.com/drive/1M4v9$_C$$TugCz_I$$KWiEUh_X$$uPEqBzllpZH?usp = sharing$

between the HMM and the HCRF technologies are much more pronounced than for the initial application of HCRFs to the TIMIT dataset. Where Gunawardana et al. report a $\approx 5\%$ difference, here this is closer to 10%. A potential reason for this might be their use of many (30 was found optimal) mixtures of Gaussians, whereas the presented models used a single multivariate Gaussians. Given that HMMs are generative, these models will generally suffer more under violation of assumptions (see discussion on generative and discriminative models in Chp. 1).

Regardless, these results show the promise of using PGMs for speech classification. While the presented results are far away from the results presented in literature, these models represent the simplest form of an ASR system. While PGMs are shown to be effective, for speech recognition, methods like modelling context dependency, including language models, improved feature engineering and stronger representational power by using GMMs for emission probabilities will be able to increase the presented accuracy towards more modern performance standards.

Chapter Summary

Where previous chapters were revolved around the theoretical underpinnings of probabilistic graphical models for structured data, this chapter finally saw application to a standard field. Some concepts within automatic speech recognition were discussed, with distinctions being made between the various sub-tasks within the discipline. Much attention was spent on converting speech as a waveform to a trainable set of data, specifically using the common MFCC methodology. The logic of this feature set was visualised, showing both the ability to cluster into latent classes and represent these features as (mixtures of) Gaussians. A brief literature review of applications of PGMs to ASR was presented. This highlighted the difference between the classification and recognition tasks, while further showing the development of model accuracy over decades of development. The TIMIT corpus was introduced, and standard practises regarding training and evaluation were presented. Lastly, an application of the discussed models was presented. The code and results of this application has been made openly available. These show the promise of applying PGMs to ASR tasks, while leaving room for future research.

Chapter 7

Discussion and Conclusion

This thesis has presented the general framework of probabilistic graphical modelling, with the specific focus on structured data. Throughout, two foundational assumptions were stressed; first, the domain being modelled can be expressed probabilistically, and second, the specific structure and a priori domain knowledge inherent to the modelled process can be leveraged to ensure succint expression of the probability space.

The use of probabilities is a general method for expressing and handling uncertainty in intelligent systems. The large literature on the properties of probabilities, both mathematical and philosophical, allow for strong theoretical motivation for PGMs. However, more than rigour and convenient notation, the use of probability also induces reasoning patterns within the trained that closely emulate human behaviour.

The use of graphs as the core data structure comes from the second of the foundational assumptions. The naturally extend the probability functions to an easily interpretable and aesthetically pleasing display. Far more importantly, the extensive study of graphs within the mathematical and computing sciences ensures properties are well understood, allowing for efficient computation. Given the exponential explosion in parameters that occurs when combining multiple probability functions, efficiency is far more than just an economical consideration.

In their epilogue, Koller and Friedman summarise PGMs as providing "support for natural representation, effective inference, and feasible model" acquisition. [Koller and Friedman2009]. As such, a natural application of the probabilistic graphical model framework is that of structured data. While standard machine learning tasks, ones that train on the present data, are difficult enough, no matter the complexity of the model, without consideration of structures inherent to the data generating process, failure is inevitable. PGMs can overcome this by further specifying relationships between variables, whether latent or not, that are believed to be present but not explicit. The discussed representational forms, Bayesian networks and Markov random fields, express the relationships and variables differently, but ultimately derive from the same motivation, and thus model phenomenons' structures in a comparable manner.

Beyond the motivational arguments for using probabilistic graphical models, the topics discussed in this thesis include:

1. preliminary background material that discusses basic principles and motivation behind concepts used throughout this thesis. Sec. 1.2 - 1.4 specifically gave a whirlwind introduction to probability functions, conditional probability, Bayesian updating, in-

formation theory and the principle of maximum entropy. Most important was the derivation of the Naive Bayes and Maximum Entropy classifiers, a generative and discriminative model respectively. While these used very different motivational arguments, both can be seen as the simplest form of *sequential* probabilistic graphical models, with similarities and differences between these models being made explicit in later chapters.

2. Bayesian networks as a causal representational form of PGMs. Sec. 2.1 was steeped in PGM concepts, stemming largely from the work of Judea Pearl, making explicit how the joint-probability space related to their graphical depiction. The concept of directed separation, a graphical form of statistical (in)dependence, proved crucial in proving equivalence in a concept known as factorisation. The intuitive use of causal inference and Bayes' theorem that comes from using Bayesian networks was briefly highlighted.
3. dynamic relationships within Bayesian networks, leading ultimately to the formulation hidden Markov models in Sec. 2.2. The route most likely taken by Baum and colleagues during their definition of HMMs was taken, starting with the random walk over a graph model, and extending this by further introducing an emission parameter. Here the famous "Balls from Urns" problem was used as a motivational example.
4. Markov random fields as a different, undirected, representational form of PGMs in Sec. 3.1. Due to the relaxation of criteria related to the topology and nature of relationships in Bayesian networks, the use of DAGs and probability functions was no longer warranted. However, use of factor functions and redefining the concept of directed separation, allowed for pseudo-probabilistic expressions in the form of Gibbs distributions. The use of a global normalisation constant however meant loss of local interpretation of probability, and further loss of the relatively compact Bayesian network representation.
5. conditional random fields as the conditional generalisation to the HMM, and the sequential form of Markov random fields in Sec. 3.2. Rather than using first principles, the relationship between the HMM and the CRF was made very explicit by using the HMM definition as a start point for the derivation of CRFs. Extensions of feature functions and parameters were discussed within the context of Boltzmann distributions. Lastly, a brief example showed the relative *representational* ease with which a CRF could extend to more complex (latent grids) graphical forms.
6. inferential problems stemming from Rabiner's formulation of the basic problems for HMMs. The forward-backward algorithm was discussed for both the evaluation task (how likely is an observation sequence?) and the smoothing task (how likely is a state at t given a sequence?) in Sec. 4.1 - 4.2. The extension to belief propagation was made in Sec. 4.3, allowing the same inference on general PGMs rather than just linear chains. The equivalence between the forwards-backwards algorithm and belief propagation was shown using two examples. Lastly, the decoding task (what is the most likely latent state sequence given the observation sequence?) was introduced in Sec. 4.4. The difference between MPM and MAP inference was briefly discussed, before providing the Viterbi algorithm. Given that the Viterbi algorithm slightly reformulates the forwards-backwards algorithm as a shortest-path problem, the extension to general graphs using a slightly reformulated belief-propagation algorithm was shown.

7. parameter estimation for both HMMs and CRFs in Ch. 5. Given the tremendously complex nature of the parameter estimation field, the presented derivations were shorter and less motivated than those for the previous chapters. The expectation algorithm was provided, and the properties of monotonic increase and convergence to minima were discussed. Using a slight extension of the forwards-backwards algorithm, the expectation step for the Baum-Welch algorithm was derived. This expectation step was then maximised using Lagrangian multipliers, resulting in a deceptively simple iterative learning scheme for HMMs. CRFs, however, are generally intractable under Baum-Welch, and as such general learning methods were discussed using log-likelihood functions taken from literature on (latent) CRFs.
8. an application of HMMs and CRFs to automatic speech recognition. ASR remains an active research domain, with PGMs forming the core technology for classical and modern systems, although being edged out by deep learning approaches by the early 2010s. Common tasks and concepts within ASR were presented, before discussing the almost ubiquitous Mel-Frequency Cepstral Coefficients feature generation methodology in Sec. 6.1. Specific PGM architectures for ASR were discussed in very limited literature review in Sec. 6.2, highlighting the one-vs-all scheme used by HMMs as opposed to an integrated classification method for CRFs, as well as the use of Gaussians and mixtures of Gaussians for emission modelling. The TIMIT speech corpus was presented, along with standard practises.
9. replicated experiments of PGMs as applied to ASR in Sec. 6.4. These experiments were conducted using the methods and concepts outlined in the previous chapters. Specifically, a HMM and a HCRF phone classifier was trained on the TIMIT corpus, achieving good, but not remotely state-of-the-art results. Some suggestions for future research were provided, given that the results validate the use of PGMs for ASR.

Despite this thesis being far more extensive than initially intended, the discussed topics have barely begun to scratch the surface of probabilistic graphical models, modelling structured data or automatic speech recognition. Within PGMs, there remain various core subjects not touched upon. Especially exciting might be ongoing research into causality, with tremendously important applications to statistics and science in general. Otherwise, the task of structure learning (given the data, what *form* of PGM best fits?) might be. Effective structure learning is both relevant for existing models, being able to generalise HMMs and CRFs beyond the training samples, and for defining new ones, removing the need for a priori definitions of PGM structure. Of course, within the topics actually dicussed within this thesis, there remains much to be said. For example, the reformulation of CRFs and HMMs as forms of structural support vector machines provides a new parameter learning paradigm for the discussed models. Such a model has already been used in Sec. 3.2.2, where the Python package `pyStruct` was used to train a max-margin CRF for a toy image segmentation task. It should take little convincing that these formulations of parameter estimation might prove beneficial for ASR (see for example Sha and Saul, achieving a classification accuracy of phone classification accuracy of 73% while using HMMs [Sha and Saul2007]). In general, see Koller and Friedman's Part. IV for a variety of innovative topics related to PGMs not discussed in this thesis, but likely to be central to AI efforts in the near future [Koller and Friedman2009].

The presented results of PGMs for the TIMIT corpus represent the simplest possible PGM classifier pipeline for ASR. Many extensions and topics remain to be discussed. Pre-segmented and labelled data was used, while this would not be the case for deployable models. For HMMs, using codebook quantisation provided a means for assigning individual subphone frames a phone sequence. Nowadays, deep auto-encoders for end-to-end speech recognition circumvent this issue entirely. The presented models in Sec. 6.4 used multivariate diagonal Gaussians, while it is standard to use the more complex mixture models for representing state emissions. Beyond phone classification, the use of language models could further extend the presented experiments to predict coherent *language* as opposed to accurately identifying sounds. All of these limitations further tie into the burgeoning field of natural language processing, where PGMs remain state-of-the-art for many tasks, and are certainly worthy of investigation.

Ultimately, this thesis has attempted to encapsulate an entire domain within artificial intelligence research, presenting results with just enough rigour and motivation for their application and discussion. While much work remains, especially considering how central the language of PGMs lie within machine learning, the presented work should suffice as a stand-alone introduction to anyone starting their foray into probabilistic graphical models for structured data.

Appendix

A. Maximising the Baum-Welch Expectation Function

This derivation originates from Stephen Tu, and uses much of their notation [Tu2015]. As such, the notation differs slightly from the rest of this thesis.

Recall the long, unwieldy definition of the expectation function as per Eq. 93,

$$Q(\lambda, \lambda_t) = \sum_{i=1}^{|Q|}\sum_{O=1}^{|\mathcal{O}|} \log \pi_i P(q_{i,t} \,|\, O, \lambda) + \sum_{i=1}^{|Q|}\sum_{j=1}^{|Q|}\sum_{O=1}^{|\mathcal{O}|}\sum_{t=2}^{T} \log a_t(i,j) P(q_{i,t}, q_{j,t-1} \,|\, O, \lambda)$$
$$\sum_{i=1}^{|Q|}\sum_{O=1}^{|\mathcal{O}|}\sum_{t=1}^{|T|} \log b_t(q_i) P(q_i \,|\, O, \lambda) I(o_t).$$

The formulation of probability gives rise to natural constraints for the maximisation procedure; namely, their sum must equal to certainty. As such, the Lagrangian function L[22] in terms of the Lagrangian multipliers μ[23] may be defined as,

$$L(\lambda, \lambda_t) = Q(\lambda, \lambda_t) - \mu_\pi(\sum_{i=1}^{|Q|} \pi_i - 1) - \sum_{i=1}^{|Q|} \mu_A(\sum_{i=1}^{|Q|} a_i - 1) - \sum_{i=1}^{|Q|} \mu_B(\sum_{i=1}^{|Q|} b_i - 1).$$

Rather than maximising $Q(\lambda, \lambda_t)$, now maximise $L(\lambda, \lambda_t)$ as,

$$\nabla L(\lambda, \lambda_t) = 0,$$

and considering the parameters in the expectation function are independent of each, the maximisation requires only one term from $Q(\lambda, \lambda_t)$ and its corresponding term in $L(\lambda, \lambda_t)$. The value for π_j follows as,

$$\frac{\partial}{\partial \pi_j} L(\lambda, \lambda_t) = \frac{\partial}{\partial \pi_j} \sum_{i=1}^{|Q|}\sum_{O=1}^{|\mathcal{O}|} \log \pi_i P(q_i \,|\, O, \lambda) + \frac{\partial}{\partial \pi_j} \mu_\pi (\sum_{i=1}^{|Q|} \pi_i - 1) = 0$$
$$\implies \sum_{O=1}^{|\mathcal{O}|} \frac{P(q_j \,|\, O, \lambda)}{\pi_j} + \mu_\pi = 0$$
$$\frac{\partial}{\partial \mu_{\pi_j}} L(\lambda, \lambda_t) = -(\sum_{j=1}^{|Q|} \pi_j - 1) = 0.$$

The sum over all states may be removed due to the fact that the derivative only focuses on π_j, such that all $i \neq j$ will simply be constants and may be discarded. Now

[22] Unlike the customary $\mathcal{L}(\cdot)$ as this is reserved for the likelihood function

[23] Again, unlike the customary λ as this is reserved for the ensemble of parameters that define HMMs

two equations in terms of two unknowns need to be solved. Using the intermediate results for each quantity, one can replace the Lagrangian multiplier leaving an identity for π_j solely in terms of the conditional probability functions:

$$\begin{aligned}
\frac{\partial}{\partial \pi_j} L(\lambda, \lambda_t) &= \frac{1}{\pi_j} \sum_{O=1}^{|\mathcal{O}|} P(q_j \,|\, O, \lambda) - \mu_\pi = 0 \\
&\implies \pi_j = \frac{\sum_{O=1}^{|\mathcal{O}|} P(q_j \,|\, O, \lambda)}{\mu_\pi} \\
\frac{\partial}{\partial \mu_{\pi_j}} L(\lambda, \lambda_t) &= \sum_{j=1}^{|Q|} \frac{\sum_{O=1}^{|\mathcal{O}|} P(q_j \,|\, O, \lambda)}{\mu_\pi} = 1 \\
&\implies \mu_\pi = \sum_{j=1}^{|Q|} \sum_{O=1}^{|\mathcal{O}|} P(q_j, O \,|\, \lambda) \\
&\implies \pi_j = \frac{\sum_{O=1}^{|\mathcal{O}|} P(q_j \,|\, O, \lambda)}{\sum_{j=1}^{|Q|} \sum_{O=1}^{|\mathcal{O}|} P(q_j \,|\, O, \lambda)}.
\end{aligned}$$

Solely using the laws of probability, the denominator may be treated as a constant after marginalising out,

$$\begin{aligned}
\pi_j &= \frac{\sum_{O=1}^{|\mathcal{O}|} P(q_j \,|\, O, \lambda)}{\sum_{j=1}^{|Q|} \sum_{O=1}^{|\mathcal{O}|} P(q_j \,|\, O, \lambda)} = \frac{\sum_{O=1}^{|\mathcal{O}|} P(q_j \,|\, O, \lambda)}{\sum_{O=1}^{|\mathcal{O}|} \sum_{j=1}^{|Q|} P(q_j \,|\, O, \lambda)} \\
&= \frac{\sum_{O=1}^{|\mathcal{O}|} P(q_j \,|\, O, \lambda)}{\sum_{O=1}^{|\mathcal{O}|} 1} = \frac{\sum_{O=1}^{|\mathcal{O}|} P(q_j \,|\, O, \lambda)}{|\mathcal{O}|} \\
\pi_j &= \frac{1}{|\mathcal{O}|} \sum_{O=1}^{|\mathcal{O}|} P(q_j \,|\, O, \lambda).
\end{aligned}$$

The initial position vector is thus simply the sample average of the state probability for $t = 1$.

A similar process may be used for the other parameters as well. Given that the general steps have been outlined, the derivation for these parameters will be less complete. The partial derivative of the Lagrangian in terms of the transition parameters follow as,

$$\begin{aligned}
\frac{\partial}{\partial A_{i,j}} L(\lambda, \lambda_t) &= \frac{\partial}{\partial A_{i,j}} \sum_{i=1}^{|Q|} \sum_{j=1}^{|Q|} \sum_{O=1}^{|\mathcal{O}|} \sum_{t=2}^{|T|} \log a_t(i,j) P(q_{i,t} \,|\, O, \lambda) - mu_A = 0 \\
&= \sum_{O=1}^{|\mathcal{O}|} \sum_{t=1}^{T} \frac{P(q_{i,t} \,|\, O, \lambda)}{A_{i,j}} \\
\frac{\partial}{\partial \mu_A} L(\lambda, \lambda_t) &= -(\sum_{i=1}^{|Q|} a_i - 1),
\end{aligned}$$

which again is a system of two equations in terms of two unknowns. Algebraic manipulation finally yields

$$A_{i,j} = \frac{\sum_{O=1}^{|\mathcal{O}|} \sum_{t=1}^{T} P(q_{i,t}, q_{j,t-1} \,|\, O, \lambda)}{\sum_{O=1}^{|\mathcal{O}|} \sum_{t=1}^{T} P(q_{j,t-1} \,|\, O, \lambda)}.$$

The interpretation of this result in terms of the state and two time slice posterior is found in Sec. 5.1.3.

Finally, the emission parameters. Using the same formulation as before, the partial derivative in terms of the emission parameters follows as,

$$\begin{aligned}
\frac{\partial}{\partial B_i} L(\lambda, \lambda_t) &= \frac{\partial}{\partial B_{i,j}} \sum_{i=1}^{|Q|} \sum_{O=1}^{|\mathcal{O}|} \sum_{t=1}^{|T|} \log b_t(q_i) P(q_i \,|\, O, \lambda) I(o_t) - mu_B = 0 \\
&= \sum_{i=1}^{|Q|} \sum_{t=1}^{|T|} \frac{P(q_i \,|\, O, \lambda) I(o_t)}{b_t(q_i)} = 0 \\
\frac{\partial}{\partial \mu_B} L(\lambda, \lambda_t) &= -(\sum_{i=1}^{|Q|} b_t(q_i) - 1),
\end{aligned}$$

from which the final maximisation quantity follows as,

$$b_t(q_i) = \frac{\sum_{i=1}^{|Q|} \sum_{t=1}^{|T|} P(q_{i,t} \,|\, O, \lambda) I(o_t)}{\sum_{i=1}^{|Q|} \sum_{t=1}^{|T|} P(q_{i,t} \,|\, O, \lambda)}.$$

Thus, the emission parameters are the relative frequency of emissions of a particular state to all emissions in the sample.

B. TIMIT Phone Table

Table 7: The 61 to 39 phone-map reduction initially proposed by Lee Hon [Lee and Hon1989], and updated to match more modern procedures by Lopes Perdigão [Lopes and Perdigao2011]. The folded column includes similar phones that have been folded into the phone. The underlined letters in the examples represent the phone. Note that the /q/ phone is excluded.

Index	Phone	Example	Folded	Index	Phone	Example	Folded
1	/iy/	beat		21	/ng/	sing	/ng/ /eng/
2	/ih/	bit	/ih/ /ix/	22	/ch/	church	
3	/eh/	bet		23	/jh/	judge	
4	/ae/	bat		24	/dh/	dhey	
5	/ah/	butt	/ah/ /ax/ /ax-h/	25	/b/	bob	
6	/uw/	boot	/uw/ /ux/	26	/dh/	dad	
7	/uh/	book		27	/dx/	butter	
8	/aa/	caat	/aa/ /ao/	28	/g/	gag	
9	/ey/	bait		29	/p/	pop	
10	/ay/	bite		30	/t/	tot	
11	/oy/	boy		31	/k/	kick	
12	/aw/	bough		32	/z/	zoo	
13	/ow/	boat		33	/v/	very	
14	/l/	led	/l/ /el/	34	/f/	fief	
15	/r/	red		35	/th/	thief	
16	/y/	yet		36	/s/	sis	
17	/w/	wet		37	/sh/	shoe	/sh/ /zh/
18	/er/	bird	/er/ /axr/	38	/hh/	hay	/hh/ /hv/
19	/m/	mom	/m/ /em/	39	/sil/	silence	various closures, pauses and silence variants
20	/n/	non	/n/ /en/ /nx/				

References

[Baker1975] Baker, J. (1975). The dragon system–an overview. *IEEE Transactions on Acoustics, speech, and signal Processing*, 23(1):24–29.

[Balakrishnan et al.2017] Balakrishnan, S., Wainwright, M. J., Yu, B., et al. (2017). Statistical guarantees for the em algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1):77–120.

[Basharin et al.2004] Basharin, G. P., Langville, A. N., and Naumov, V. A. (2004). The life and work of aa markov. *Linear algebra and its applications*, 386:3–26.

[Baum and Petrie1966] Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563.

[Baum et al.1970] Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171.

[Bishop2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[Christensen2006] Christensen, R. (2006). *Log-linear models and logistic regression*. Springer Science & Business Media.

[Dempster et al.1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

[Fisher1936] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

[Garofolo et al.1993] Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., Pallett, D. S., Dahlgren, N. L., and Zue, V. (1993). Timit acoustic phonetic continuous speech corpus. *Linguistic Data Consortium, 1993*.

[Gibbs2014] Gibbs, J. W. (2014). *Elementary principles in statistical mechanics*. Courier Corporation.

[Gleich and Saunders2009] Gleich, D. F. and Saunders, M. (2009). *Models and algorithms for pagerank sensitivity*. Stanford University Stanford, CA.

[Gunawardana et al.2005] Gunawardana, A., Mahajan, M., Acero, A., and Platt, J. C. (2005). Hidden conditional random fields for phone classification. In *Ninth European Conference on Speech Communication and Technology*.

[Hammersley and Clifford1971] Hammersley, J. M. and Clifford, P. (1971). Markov fields on finite graphs and lattices. *Unpublished manuscript*, 46.

[Harris1978] Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83.

[Huang et al.2001] Huang, X., Acero, A., Hon, H.-W., and Foreword By-Reddy, R. (2001). *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR.

[Jaynes1957] Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical review*, 106(4):620.

[Keynes2013] Keynes, J. M. (2013). *A treatise on probability*. Courier Corporation.

[Koller and Friedman2009] Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

[Lafferty et al.2001] Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

[Lauritzen1996] Lauritzen, S. L. (1996). *Graphical models*, volume 17. Clarendon Press.
[Lauritzen and Spiegelhalter1988] Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194.
[Lee and Hon1989] Lee, K.-F. and Hon, H.-W. (1989). Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648.
[Logan et al.2000] Logan, B. et al. (2000). Mel frequency cepstral coefficients for music modeling. In *Ismir*, volume 270, pages 1–11.
[Lopes and Perdigao2011] Lopes, C. and Perdigao, F. (2011). Phone recognition on the timit database. *Speech Technologies/Book*, 1:285–302.
[Louis et al.] Louis, S. B. W. A. Q., Demirdjian, P. M. D., and Darrell, T. Hidden conditional random fields for gesture recognition.
[Lyons1998] Lyons, R. (1998). Windowing functions improve fft results-flexible windowing functions let you adjust frequency and leakage effects. *Test and Measurement World*, 18(10):53.
[Maathuis et al.2018] Maathuis, M., Drton, M., Lauritzen, S., and Wainwright, M. (2018). *Handbook of graphical models*. CRC Press.
[Martin and Jurafsky2018] Martin, J. H. and Jurafsky, D. (2018). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. 3 edition.
[McLachlan and Krishnan2007] McLachlan, G. and Krishnan, T. (2007). *The EM algorithm and extensions*, volume 382. John Wiley & Sons.
[Mezard and Montanari2009] Mezard, M. and Montanari, A. (2009). *Information, physics, and computation*. Oxford University Press.
[Mount2011] Mount, J. (2011). The equivalence of logistic regression and maximum entropy models. *URL: http://www. win-vector. com/dfiles/LogisticRegressionMaxEnt. pdf*.
[Müller and Behnke2014] Müller, A. C. and Behnke, S. (2014). Pystruct: learning structured prediction in python. *The Journal of Machine Learning Research*, 15(1):2055–2060.
[Murphy2012] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.
[Nowozin et al.2011] Nowozin, S., Lampert, C. H., et al. (2011). Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365.
[Oppenheim and Schafer2004] Oppenheim, A. V. and Schafer, R. W. (2004). From frequency to quefrency: A history of the cepstrum. *IEEE signal processing Magazine*, 21(5):95–106.
[Page et al.1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.
[Pearl1982] Pearl, J. (1982). *Reverend Bayes on inference engines: A distributed hierarchical approach*. Cognitive Systems Laboratory, School of Engineering and Applied Science ....
[Pearl2014] Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

[Pearl and Mackenzie2018] Pearl, J. and Mackenzie, D. (2018). *The book of why: the new science of cause and effect*. Basic Books.

[Pearl and Russel2011] Pearl, J. and Russel, S. (2011). Bayesian networks.

[Quattoni et al.2005] Quattoni, A., Collins, M., and Darrell, T. (2005). Conditional random fields for object recognition. In *Advances in neural information processing systems*, pages 1097–1104.

[Rabiner1989] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Rabiner and Juang1986] Rabiner, L. R. and Juang, B.-H. (1986). An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16.

[Sha and Saul2007] Sha, F. and Saul, L. K. (2007). Large margin hidden markov models for automatic speech recognition. In *Advances in neural information processing systems*, pages 1249–1256.

[Stevens et al.1937] Stevens, S. S., Volkmann, J., and Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190.

[Sung et al.2007] Sung, Y.-H., Boulis, C., Manning, C., and Jurafsky, D. (2007). Regularization, adaptation, and non-independent features improve hidden conditional random fields for phone classification. In *2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 347–352. IEEE.

[Sung and Jurafsky2009] Sung, Y.-H. and Jurafsky, D. (2009). Hidden conditional random fields for phone recognition. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 107–112. IEEE.

[Sutton et al.2012] Sutton, C., McCallum, A., et al. (2012). An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373.

[Truyen2008] Truyen, T. T. (2008). *On conditional random fields: applications, feature selection, parameter estimation and hierarchical modelling*. PhD thesis, Curtin University.

[Tu2015] Tu, S. (2015). Derivation of baum-welch algorithm for hidden markov models.

[Vishwanathan et al.2006] Vishwanathan, S., Schraudolph, N. N., Schmidt, M. W., and Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the 23rd international conference on Machine learning*, pages 969–976.

[Weissteina] Weisstein, E. W. Jensen's inequality.

[Weissteinb] Weisstein, E. W. Principle of insufficient reason.

[Wright1920] Wright, S. (1920). The relative importance of heredity and environment in determining the piebald pattern of guinea-pigs. *Proceedings of the National Academy of Sciences of the United States of America*, 6(6):320.

[Wright1923] Wright, S. (1923). The theory of path coefficients a reply to niles's criticism. *genetics*, 8(3):239.

[Wu et al.2016] Wu, C., Yang, C., Zhao, H., and Zhu, J. (2016). On the convergence of the em algorithm: A data-adaptive analysis. *arXiv preprint arXiv:1611.00519*.

[Wu et al.1983] Wu, C. J. et al. (1983). On the convergence properties of the em algorithm. *The Annals of statistics*, 11(1):95–103.

[Xing et al.2010] Xing, Z., Pei, J., and Keogh, E. (2010). A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, 12(1):40–48.