Running head: PROBABILITY PRIMER

A Primer on Probabilistic Inference

Thomas L. Griffiths

Department of Psychology

University of California, Berkeley

Alan Yuille

Department of Statistics

University of California, Los Angeles

A Primer on Probabilistic Inference

Probabilistic models aim to explain human cognition by appealing to the principles of probability theory and statistics, which dictate how an agent should act rationally in situations that involve uncertainty. While probability theory was originally developed as a means of analyzing games of chance, it was quickly realized that probabilities could be used to analyze rational actions in a wide range of contexts (e.g., Bayes, 1763/1958; Laplace, 1795/1951). Probabilistic models have come to be used in many disciplines, and are currently the method of choice for an enormous range of applications, including artificial systems for medical inference, bio-informatics, and computer vision. Applying probabilistic models to human cognition thus provides the opportunity to draw upon work in computer science, engineering, mathematics, and statistics, often producing quite surprising connections.

There are two challenges involved in developing probabilistic models of cognition. The first challenge is specifying a suitable model. This requires considering the computational problem faced by an agent, the knowledge available to that agent, and the appropriate way to represent that knowledge. The second challenge is evaluating model predictions. Probabilistic models can capture the structure of extremely complex problems, but as the structure of the model becomes richer, probabilistic inference becomes harder. Being able to compute the relevant probabilities is a practical issue that arises when using probabilistic models, and also raises the question of how intelligent agents might be able to make similar computations.

In this chapter, we introduce some of the tools that can be used to address these challenges. By considering how probabilistic models can be defined and used, we aim to provide some of the background relevant to the other chapters in this volume. The plan of the chapter is as follows. First, we outline the fundamentals of Bayesian inference, which

is at the heart of many probabilistic models. We then discuss how to define probabilistic models that use richly structured probability distributions, introducing some of the key ideas behind *graphical models*, which can be used to represent the dependencies among a set of variables. Finally, we discuss two of the main algorithms that are used to evaluate the predictions of probabilistic models – the *Expectation-Maximization (EM)* algorithm, and *Markov chain Monte Carlo (MCMC)* – and some sophisticated probabilistic models that exploit these algorithms. Several books provide a more detailed discussion of these topics in the context of statistics (e.g., Berger, 1993; Bernardo & Smith, 1994; Gelman, Carlin, Stern, & Rubin, 1995), machine learning (e.g., Bishop, 2006; Duda, Hart, & Stork, 2000; Hastie, Tibshirani, & Friedman, 2001; Mackay, 2003), and artificial intelligence (e.g., Korb & Nicholson, 2003; Pearl, 1988; Russell & Norvig, 2002). Griffiths, Kemp, and Tenenbaum (in press) provide further information on some of the methods touched on in this chapter, together with examples of applications of these methods in cognitive science.

Fundamentals of Bayesian inference

Probabilistic models of cognition are often referred to as Bayesian models, reflecting the central role that Bayesian inference plays in reasoning under uncertainty. In this section, we will introduce the basic ideas behind Bayesian inference, and discuss how it can be used in different contexts.

Basic Bayes

Bayesian inference is based upon a simple formula known as *Bayes' rule* (Bayes, 1763/1958). When stated in terms of abstract random variables, Bayes' rule is a simple tautology of probability theory. Assume we have two random variables, A and B.[1] One of the principles of probability theory (sometimes called the *chain rule*) allows us to write the *joint probability* of these two variables taking on particular values a and b, $P(a, b)$, as the product of the *conditional probability* that A will take on value a given B takes on

value b, $P(a|b)$, and the *marginal probability* that B takes on value b, $P(b)$. Thus, we have

$$P(a, b) = P(a|b)P(b). \tag{1}$$

There was nothing special about the choice of A rather than B in factorizing the joint probability in this way, so we can also write

$$P(a, b) = P(b|a)P(a). \tag{2}$$

It follows from Equations 1 and 2 that $P(a|b)P(b) = P(b|a)P(a)$, which can be rearranged to give

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)}. \tag{3}$$

This expression is Bayes' rule, which indicates how we can compute the conditional probability of b given a from the conditional probability of a given b.

In the form given in Equation 3, Bayes' rule seems relatively innocuous (and perhaps rather uninteresting!). Bayes' rule gets its strength, and its notoriety, by making some assumptions about the variables we are considering and the meaning of probability. Assume that we have an agent who is attempting to infer the process that was responsible for generating some data, d. Let h be a hypothesis about this process, and $P(h)$ indicate the probability that the agent would have ascribed to h being the true generating process, prior to seeing d (known as a *prior probability*). How should that agent go about changing his beliefs in the light of the evidence provided by d? To answer this question, we need a procedure for computing the *posterior probability*, $P(h|d)$.

Bayes' rule provides just such a procedure. If we are willing to consider the hypotheses that agents entertain to be random variables, and allow probabilities to reflect subjective degrees of belief, then Bayes' rule indicates how an agent should update their beliefs in light of evidence. Replacing a with d and b with h in Equation 3 gives

$$P(h|d) = \frac{P(d|h)P(h)}{P(d)}, \tag{4}$$

which is the form in which Bayes' rule is normally presented. The probability of the data given the hypothesis, $P(d|h)$, is known as the *likelihood*.

Probability theory also allows us to compute the probability distribution associated with a single variable (known as the *marginal probability*) by summing over the other variables in a joint distribution (known as *marginalization*), e.g., $P(b) = \sum_a P(a, b)$. Using this principle, we can rewrite Equation 4 as

$$P(h|d) = \frac{P(d|h)P(h)}{\sum_{h' \in \mathcal{H}} P(d|h')P(h')}, \tag{5}$$

where $\mathcal{H}$ is the set of all hypotheses considered by the agent, sometimes referred to as the *hypothesis space*. This formulation of Bayes' rule makes it apparent that the posterior probability of h is directly proportional to the product of the prior probability and the likelihood. The sum in the denominator simply ensures that the resulting probabilities are normalized to sum to one.

The use of probabilities to represent degrees of belief is known as *subjectivism*, as opposed to *frequentism*, in which probabilities are viewed as the long-run relative frequencies of the outcomes of non-deterministic experiments. Subjectivism is strongly associated with Bayesian approaches to statistics, and has been a subject of great controversy in foundational discussions on the nature of probability. A number of formal arguments have been used to justify this position (e.g., Cox, 1961; Jaynes, 2003). In the remainder of this section, we will focus on some of the practical applications of Bayes' rule.

Comparing two simple hypotheses

The setting in which Bayes' rule is usually discussed in introductory statistics courses is for the comparison of two simple hypotheses. For example, imagine that you are told that a box contains two coins: one that produces heads 50% of the time, and one that produces heads 90% of the time. You choose a coin, and then flip it ten times, producing

the sequence HHHHHHHHHH. Which coin did you pick? What would you think if you had flipped HHTHTHTTHT instead?

To translate this problem into one of Bayesian inference, we need to identify the hypothesis space, $\mathcal{H}$, the prior probability of each hypothesis, $P(h)$, and the probability of the data under each hypothesis, $P(d|h)$. We have two coins, and thus two hypotheses. If we use θ to denote the probability that a coin produces heads, then h_0 is the hypothesis that $\theta = 0.5$, and h_1 is the hypothesis that $\theta = 0.9$. Since we have no reason to believe that we would be more likely to pick one coin than the other, the prior probabilities are $P(h_0) = P(h_1) = 0.5$. The probability of a particular sequence of coinflips containing N_H heads and N_T tails being generated by a coin which produces heads with probability θ is

$$P(d|\theta) = \theta^{N_H}(1-\theta)^{N_T}. \tag{6}$$

The likelihoods associated with h_0 and h_1 can thus be obtained by substituting the appropriate value of θ into Equation 6.

We can take the prior and likelihoods defined in the previous paragraph, and plug them directly into Equation 4 to compute the posterior probability of each of our two hypotheses. However, when we have just two hypotheses it is often easier to work with the *posterior odds*, which are just the ratio of the posterior probabilities. If we use Bayes' rule to find the posterior probability of h_0 and h_1, it follows that the posterior odds in favor of h_1 are

$$\frac{P(h_1|d)}{P(h_0|d)} = \frac{P(d|h_1)}{P(d|h_0)}\frac{P(h_1)}{P(h_0)} \tag{7}$$

where we have used the fact that the denominator of Equation 4 is constant. The first and second terms on the right hand side are called the *likelihood ratio* and the *prior odds* respectively. Returning to our example, we can use Equation 7 to compute the posterior odds of our two hypotheses for any observed sequence of heads and tails. Using the prior and likelihoods from the previous paragraph gives odds of approximately 357:1 in favor of

h_1 for the sequence HHHHHHHHHH and 165:1 in favor of h_0 for the sequence HHTHTHTTHT.

The form of Equation 7 helps to clarify how prior knowledge and new data are combined in Bayesian inference. The two terms on the right hand side each express the influence of one of these factors: the prior odds are determined entirely by the prior beliefs of the agent, while the likelihood ratio expresses how these odds should be modified in light of the data d. This relationship is made even more transparent if we examine the expression for the log posterior odds,

$$\log \frac{P(h_1|d)}{P(h_0|d)} = \log \frac{P(d|h_1)}{P(d|h_0)} + \log \frac{P(h_1)}{P(h_0)} \tag{8}$$

in which the extent to which one should favor h_1 over h_0 reduces to an additive combination of a term reflecting prior beliefs (the log prior odds) and a term reflecting the contribution of the data (the log likelihood ratio). Based upon this decomposition, the log likelihood ratio in favor of h_1 is often used as a measure of the evidence that d provides for h_1. The British mathematician Alan Turing, arguably one of the founders of cognitive science, was also one of the first people to apply Bayesian inference in this fashion: he used log likelihood ratios as part of a method for deciphering German naval codes during World War II (Good, 1979).

Comparing infinitely many hypotheses

The analysis outlined above for two simple hypotheses generalizes naturally to any finite set, although posterior odds are less useful when there are multiple alternatives to be considered. However, Bayesian inference can also be applied in contexts where there are (uncountably) infinitely many hypotheses to evaluate – a situation that arises surprisingly often. For example, imagine that rather than choosing between two alternatives for the probability that a coin produces heads, θ, we were willing to consider any value of θ between 0 and 1. What should we infer about the value of θ from a sequence such as HHHHHHHHHH?

In frequentist statistics, from which much of the statistical doctrine used in the analysis of psychology experiments derives, inferring θ is treated a problem of estimating a fixed parameter of a probabilistic model, to which the standard solution is *maximum-likelihood* estimation (see, e.g., Rice, 1995). The maximum-likelihood estimate of θ is the value $\hat{\theta}$ that maximizes the probability of the data, as given in Equation 6. It is straightforward to show that this is $\hat{\theta} = \frac{N_H}{N_H + N_T}$, which gives $\hat{\theta} = 1.0$ for the sequence `HHHHHHHHHH`.

The case of inferring the bias of a coin can be used to illustrate some properties of maximum-likelihood estimation that can be problematic. First of all, the value of θ that maximizes the probability of the data might not provide the best basis for making predictions about future data. To continue the example above, inferring that $\theta = 1.0$ after seeing the sequence `HHHHHHHHHH` implies that we should predict that the coin would never produce tails. This might seem reasonable, but the same conclusion follows for any sequence consisting entirely of heads. Would you predict that a coin would produce only heads after seeing it produce a head on a single flip?

A second problem with maximum-likelihood estimation is that it does not take into account other knowledge that we might have about θ. This is largely by design: the frequentist approach to statistics was founded on the belief that subjective probabilities are meaningless, and aimed to provide an "objective" system of statistical inference that did not require prior probabilities. However, while such a goal of objectivity might be desirable in certain scientific contexts, most intelligent agents have access to extensive knowledge that constrains their inferences. For example, many of us might have strong expectations that a coin would produce heads with a probability close to 0.5.

Both of these problems are addressed by the Bayesian approach to this problem, in which inferring θ is treated just like any other Bayesian inference. If we assume that θ is a

random variable, then we can apply Bayes' rule to obtain

$$p(\theta|d) = \frac{P(d|\theta)p(\theta)}{P(d)} \tag{9}$$

where

$$P(d) = \int_0^1 P(d|\theta)p(\theta)\, d\theta. \tag{10}$$

The key difference from Bayesian inference with finitely many hypotheses is that the posterior distribution is now characterized by a *probability density*, and the sum over hypotheses becomes an integral.

The posterior distribution over θ contains more information than a single point estimate: it indicates not just which values of θ are probable, but also how much uncertainty there is about those values. Collapsing this distribution down to a single number discards information, so Bayesians prefer to maintain distributions wherever possible (this attitude is similar to Marr's (1982, p. 106) "principle of least commitment"). However, there are two methods that are commonly used to obtain a point estimate from a posterior distribution. The first method is *maximum a posteriori (MAP)* estimation: choosing the value of θ that maximizes the posterior probability, as given by Equation 9. The second method is computing the *posterior mean* of the quantity in question. For example, we could compute the posterior mean value of θ, which would be

$$\bar{\theta} = \int_0^1 \theta\, p(\theta|d)\, d\theta. \tag{11}$$

For the case of coinflipping, the posterior mean also corresponds to the probability of obtaining heads under the *posterior predictive distribution*: the probability with which one should predict the next toss of the coin will produce heads.

The choice of which estimator to use for θ depends on the nature of the problem being solved. Bayesian decision theory (e.g., Berger, 1993) approaches this problem by introducing a loss function $L(\theta, \alpha(d))$ for the cost of making a decision $\alpha(d)$ when the

input is d and the true value of the parameter is θ. From this perspective, we should select the decision rule $\alpha^*(.)$ that minimizes the risk, or expected loss:

$$R(\alpha) = \sum_{\theta,d} L(\theta, \alpha(d))p(\theta, d). \tag{12}$$

where the distribution on θ and d captures both the natural variation in d and the uncertainty of the learner about θ given d, as represented in the posterior distribution. One loss function has the same penalty is paid for all wrong decisions: $L(\theta, \alpha(d)) = 1$ if $\alpha(d) \neq \theta$ and $L(\theta, \alpha(d)) = 0$ if $\alpha(d) = \theta$. For this loss function, the best decision rule is the MAP estimator. Alternatively, if the loss function is the square of the error $L(\theta, \alpha(d)) = (\theta - \alpha(d))^2$ then the best decision rule is the posterior mean. This approach can be extended to dynamical systems where decisions need to be made over time. This leads to optimal control theory (Bertsekas, 2000) where the goal is to minimize a cost functional to obtain a control law (analogous to the risk and the decision rule respectively). Optimal control theory lays the groundwork for rational models of animal learning and motor control.

Regardless of the estimator being used, different choices of the prior, $p(\theta)$, will lead to different guesses at the value of θ. A first step might be to assume a *uniform* prior over θ, with $p(\theta)$ being equal for all values of θ between 0 and 1. This case was first analyzed by Laplace (1795/1951). Using a little calculus, it is possible to show that the posterior distribution over θ produced by a sequence d with N_H heads and N_T tails is

$$p(\theta|d) = \frac{(N_H + N_T + 1)!}{N_H!\ N_T!}\ \theta^{N_H}(1-\theta)^{N_T}. \tag{13}$$

This is actually a distribution of a well known form, being a beta distribution with parameters $N_H + 1$ and $N_T + 1$, denoted Beta$(N_H + 1, N_T + 1)$ (e.g., Pitman, 1993). Using this prior, the MAP estimate for θ is the same as the maximum-likelihood estimate, being $\frac{N_H}{N_H+N_T}$, but the posterior mean is $\frac{N_H+1}{N_H+N_T+2}$. Thus, the posterior mean is sensitive to the fact that we might not want to put as much weight in a single head as a sequence of

ten heads in a row: on seeing a single head, we should predict that the next toss will produce a head with probability $\frac{2}{3}$, while a sequence of ten heads should lead us to predict that the next toss will produce a head with probability $\frac{11}{12}$.

Finally, we can also use priors that encode stronger beliefs about the value of θ. For example, we can take a Beta$(V_H + 1, V_T + 1)$ distribution for $p(\theta)$, where V_H and V_T are positive integers. This distribution has a mean at $\frac{V_H+1}{V_H+V_T+2}$, and gradually becomes more concentrated around that mean as $V_H + V_T$ becomes large. For instance, taking $V_H = V_T = 1000$ would give a distribution that strongly favors values of θ close to 0.5. Using such a prior, we obtain the posterior distribution

$$p(\theta|d) = \frac{(N_H + N_T + V_H + V_T + 1)!}{(N_H + V_H)!\ (N_T + V_T)!}\ \theta^{N_H+V_H}(1-\theta)^{N_T+V_T}, \tag{14}$$

which is Beta$(N_H + V_H + 1, N_T + V_T + 1)$. Under this posterior distribution, the MAP estimate of θ is $\frac{N_H+V_H}{N_H+N_T+V_H+V_T}$, and the posterior mean is $\frac{N_H+V_H+1}{N_H+N_T+V_H+V_T+2}$. Thus, if $V_H = V_T = 1000$, seeing a sequence of ten heads in a row would induce a posterior distribution over θ with a mean of $\frac{1011}{2012} \approx 0.5025$.

Some reflection upon the results in the previous paragraph yields two observations: first, that the prior and posterior are from the same family of distributions (both being beta distributions), and second, that the parameters of the prior, V_H and V_T, act as "virtual examples" of heads and tails, which are simply combined with the real examples tallied in N_H and N_T to produce the posterior. These two properties are not accidental: they are characteristic of a class of priors called *conjugate priors* (Bernardo & Smith, 1994). The likelihood determines whether a conjugate prior exists for a given problem, and the form that the prior will take. The results we have given in this section exploit the fact that the beta distribution is the conjugate prior for the Bernoulli or binomial likelihood (Equation 6) – the uniform distribution on $[0, 1]$ is also a beta distribution, being Beta$(1, 1)$. Conjugate priors exist for many of the distributions commonly used in

probabilistic models, such as Gaussian, Poisson, and multinomial distributions, and greatly simplify many Bayesian calculations. Using conjugate priors, posterior distributions can be computed analytically, and the interpretation of the prior as contributing virtual examples is intuitive.

The example of tossing a coin serves to illustrate how the posterior distribution forms a compromise between the prior and the information provided by the data, even when we move from a small number of hypotheses to a continuum. This compromise is quite explicit in Equation 14 and the estimators based upon it, where the number of heads and tails exhibited in the data combine directly with the number of heads and tails expected under the prior. Conjugate priors yield this kind of result in many models, but the underlying principle is more general: the posterior distribution will always represent a synthesis of the data, filtered through the likelihood, and the expectations of the learner, as reflected in the prior.

Comparing hypotheses that differ in complexity

Whether there were a finite number or not, the hypotheses that we have considered so far were relatively homogeneous, each offering a single value for the parameter θ characterizing our coin. However, many problems require comparing hypotheses that differ in their complexity. For example, the problem of inferring whether a coin is fair or biased based upon an observed sequence of heads and tails requires comparing a hypothesis that gives a single value for θ – if the coin is fair, then $\theta = 0.5$ – with a hypothesis that allows θ to take on any value between 0 and 1.

Using observed data to chose between two probabilistic models that differ in their complexity is often called the problem of *model selection* (Myung & Pitt, 1997; Myung, Forster, & Browne, 2000). In frequentist statistics, this problem is addressed via hypothesis testing, a complex and counter-intuitive method that will be familiar to many

readers. In contrast, the Bayesian approach to model selection is a seamless application of the methods discussed so far. Hypotheses that differ in their complexity can be compared directly using Bayes' rule, once they are reduced to probability distributions over the observable data (see Kass & Raftery, 1995).

To illustrate this principle, assume that we have two hypotheses: h_0 is the hypothesis that $\theta = 0.5$, and h_1 is the hypothesis that θ takes a value drawn from a uniform distribution on $[0, 1]$. If we have no a priori reason to favor one hypothesis over the other, we can take $P(h_0) = P(h_1) = 0.5$. The likelihood of the data under h_0 is straightforward to compute, using Equation 6, giving $P(d|h_0) = 0.5^{N_H+N_T}$. But how should we compute the likelihood of the data under h_1, which does not make a commitment to a single value of θ?

The solution to this problem is to compute the marginal probability of the data under h_1. As discussed above, given a joint distribution over a set of variables, we can always sum out variables until we obtain a distribution over just the variables that interest us. In this case, we define the joint distribution over d and θ given h_1, and then integrate over θ to obtain

$$P(d|h_1) = \int_0^1 P(d|\theta, h_1)p(\theta|h_1)\, d\theta \tag{15}$$

where $p(\theta|h_1)$ is the distribution over θ assumed under h_1 – in this case, a uniform distribution over $[0, 1]$. This does not require any new concepts – it is exactly the same kind of computation as we needed to perform to compute the normalizing constant for the posterior distribution over θ (Equation 10). Performing this computation, we obtain $P(d|h_1) = \frac{N_H!\ N_T!}{(N_H+N_T+1)!}$, where again the fact that we have a conjugate prior on θ provides us with a neat analytic result. Having computed this likelihood, we can apply Bayes' rule just as we did for two simple hypotheses. Figure 1 (a) shows how the log posterior odds in favor of h_1 change as N_H and N_T vary for sequences of length 10.

The ease with which hypotheses differing in complexity can be compared using Bayes' rule conceals the fact that this is actually a very challenging problem. Complex hypotheses have more degrees of freedom that can be adapted to the data, and can thus always be made to fit the data better than simple hypotheses. For example, for any sequence of heads and tails, we can always find a value of θ that would give higher probability to that sequence than the hypothesis that $\theta = 0.5$. It seems like a complex hypothesis would thus have a big advantage over a simple hypothesis. The Bayesian solution to the problem of comparing hypotheses that differ in their complexity takes this into account. More degrees of freedom provide the opportunity to find a better fit to the data, but this greater flexibility also makes a worse fit possible. For example, for d consisting of the sequence `HHTHTTHHHT`, $P(d|\theta, h_1)$ is greater than $P(d|h_0)$ for $\theta \in (0.5, 0.694]$, but is less than $P(d|h_0)$ outside that range. Marginalizing over θ averages these gains and losses: a more complex hypothesis will be favored only if its greater complexity consistently provides a better account of the data. This penalization of more complex models is known as the "Bayesian Occam's razor" (Jeffreys & Berger, 1992; Mackay, 2003), and is illustrated in Figure 1 (b).

Representing structured probability distributions

Probabilistic models go beyond "hypotheses" and "data". More generally, a probabilistic model defines the joint distribution for a set of random variables. For example, imagine that a friend of yours claims to possess psychic powers – in particular, the power of psychokinesis. He proposes to demonstrate these powers by flipping a coin, and influencing the outcome to produce heads. You suggest that a better test might be to see if he can levitate a pencil, since the coin producing heads could also be explained by some kind of sleight of hand, such as substituting a two-headed coin. We can express all possible outcomes of the proposed tests, as well as their causes, using the binary random

variables X_1, X_2, X_3, and X_4 to represent (respectively) the truth of the coin being flipped and producing heads, the pencil levitating, your friend having psychic powers, and the use of a two-headed coin. Any set of of beliefs about these outcomes can be encoded in a joint probability distribution, $P(x_1, x_2, x_3, x_4)$. For example, the probability that the coin comes up heads ($x_1 = 1$) should be higher if your friend actually does have psychic powers ($x_3 = 1$).

Once we have defined a joint distribution on X_1, X_2, X_3, and X_4, we can reason about the implications of events involving these variables. For example, if flipping the coin produces heads ($x_1 = 1$), then the probability distribution over the remaining variables is

$$P(x_2, x_3, x_4 | x_1 = 1) = \frac{P(x_1 = 1, x_2, x_3, x_4)}{P(x_1 = 1)}. \tag{16}$$

This equation can be interpreted as an application of Bayes' rule, with X_1 being the data, and X_2, X_3, X_4 being the hypotheses. However, in this setting, as with most probabilistic models, any variable can act as data or hypothesis. In the general case, we use probabilistic inference to compute the probability distribution over a set of *unobserved* variables (here, X_2, X_3, X_4) conditioned on a set of *observed* variables (here, X_1).

While the rules of probability can, in principle, be used to define and reason about probabilistic models involving any number of variables, two factors can make large probabilistic models difficult to use. First, it is hard to simply write down a joint distribution over a set of variables which expresses the assumptions that we want to make in a probabilistic model. Second, the resources required to represent and reason about probability distributions increases exponentially in the number of variables involved. A probability distribution over four binary random variables requires $2^4 - 1 = 15$ numbers to specify, which might seem quite reasonable. If we double the number of random variables to eight, we would need to provide $2^8 - 1 = 255$ numbers to fully specify the joint distribution over those variables, a much more challenging task!

Fortunately, the widespread use of probabilistic models in statistics and computer science has led to the development of a powerful formal language for describing probability distributions which is extremely intuitive, and simplifies both representing and reasoning about those distributions. This is the language of *graphical models*, in which the statistical dependencies that exist among a set of variables are represented graphically. We will discuss two kinds of graphical models: *directed* graphical models, and *undirected* graphical models.

Directed graphical models

Directed graphical models, also known as Bayesian networks or Bayes nets, consist of a set of nodes, representing random variables, together with a set of directed edges from one node to another, which can be used to identify statistical dependencies between variables (e.g., Pearl, 1988). Typically, nodes are drawn as circles, and the existence of a directed edge from one node to another is indicated with an arrow between the corresponding nodes. If an edge exists from node A to node B, then A is referred to as the "parent" of B, and B is the "child" of A. This genealogical relation is often extended to identify the "ancestors" and "descendants" of a node.

The directed graph used in a Bayes net has one node for each random variable in the associated probability distribution. The edges express the statistical dependencies between the variables in a fashion consistent with the *Markov condition*: conditioned on its parents, each variable is independent of all other variables except its descendants (Pearl, 1988; Spirtes, Glymour, & Schienes, 1993). This has an important implication: a Bayes net specifies a canonical factorization of a probability distribution into the product of the conditional distribution for each variable conditioned on its parents. Thus, for a set of variables $X_1, X_2, \ldots, X_M$, we can write $P(x_1, x_2, \ldots, x_M) = \prod_i P(x_i|\mathrm{Pa}(X_i))$ where $\mathrm{Pa}(X_i)$ is the set of parents of X_i.

Figure 2 shows a Bayes net for the example of the friend who claims to have psychic powers. This Bayes net identifies a number of assumptions about the relationship between the variables involved in this situation. For example, X_1 and X_2 are assumed to be independent given X_3, indicating that once it was known whether or not your friend was psychic, the outcomes of the coin flip and the levitation experiments would be completely unrelated. By the Markov condition, we can write $P(x_1, x_2, x_3, x_4) = P(x_1|x_3, x_4)P(x_2|x_3)P(x_3)P(x_4)$. This factorization allows us to use fewer numbers in specifying the distribution over these four variables: we only need one number for each variable, conditioned on each set of values taken on by its parents. In this case, this adds up to 8 numbers rather than 15. Furthermore, recognizing the structure in this probability distribution simplifies some of the computations we might want to perform. For example, in order to evaluate Equation 16, we need to compute

$$\begin{aligned} P(x_1 = 1) &= \sum_{x_2}\sum_{x_3}\sum_{x_4} P(x_1 = 1, x_2, x_3, x_4) \\ &= \sum_{x_2}\sum_{x_3}\sum_{x_4} P(x_1 = 1|x_3, x_4)P(x_2|x_3)P(x_3)P(x_4) \\ &= \sum_{x_3}\sum_{x_4} P(x_1 = 1|x_3, x_4)P(x_3)P(x_4) \end{aligned} \tag{17}$$

where we were able to sum out X_2 directly as a result of its independence from X_1 when conditioned on X_3.

Bayes nets make it easy to define probability distributions, and speed up probabilistic inference. There are a number of specialized algorithms for efficient probabilistic inference in Bayes nets, which make use of the dependencies among variables (see Pearl, 1988; Russell & Norvig, 2002). In particular, if the underlying graph is a tree (i.e. it has no closed loops), dynamic programming algorithms (e.g., Bertsekas, 2000) can be used to exploit this structure. The intuition behind dynamic programming can be illustrated by planning the shortest route for a trip from Los Angeles to Boston. To determine the cost of going via Chicago, you only need to calculate the shortest route

from LA to Chicago and then, independently, from Chicago to Boston. Decomposing the route in this way, and taking into account the linear nature of the trip, gives an efficient algorithm with convergence rates which are polynomial in the number of nodes and hence are often feasible for computation. Equation 17 is one illustration for how the dynamic programming can exploit the structure of the problem to simplify the computation. These methods are put to particularly good use in hidden Markov models, which we discuss in further detail later in the chapter.

With a little practice, and a few simple rules (e.g., Schachter, 1998), it is easy to read the dependencies among a set of variables from a Bayes net, and to identify how variables will influence one another. One common pattern of influence is *explaining away*. Imagine that your friend flipped the coin, and it came up heads ($x_1 = 1$). The propositions that he has psychic powers ($x_3 = 1$) and that it is a two-headed coin ($x_4 = 1$) might both become more likely. However, while these two variables were independent before seeing the outcome of the coinflip, they are now dependent: if you were to go on to discover that the coin has two heads, the hypothesis of psychic powers would return to its baseline probability – the evidence for psychic powers was "explained away" by the presence of the two-headed coin.

Different aspects of directed graphical models are emphasized in their use in the artificial intelligence and statistics communities. In the artificial intelligence community (e.g., Korb & Nicholson, 2003; Pearl, 1988; Russell & Norvig, 2002), the emphasis is on Bayes nets as a form of knowledge representation and an engine for probabilistic reasoning. In statistics, graphical models tend to be used to clarify the dependencies among a set of variables, and to identify the *generative model* assumed by a particular analysis. A generative model is a step-by-step procedure by which a set of variables are assumed to take their values, defining a probability distribution over those variables. Any Bayes net specifies such a procedure: each variable without parents is sampled, then each

successive variable is sampled conditioned on the values of its parents. By considering the process by which observable data are generated, it becomes possible to postulate that the structure contained in those data is the result of underlying unobserved variables. The use of such *latent variables* is extremely common in probabilistic models, as we discuss further later in the chapter.

Recently, research has begun to explore the use of graphical models for the representation of causal relationships. Causal graphical models augment standard directed graphical models with a stronger assumption about the relationship indicated by an edge between two nodes: rather than indicating statistical dependency, such an edge is assumed to indicate a direct causal relationship (Pearl, 2000; Spirtes et al., 1993). This assumption allows causal graphical models to represent not just the probabilities of events that one might observe, but also the probabilities of events that one can produce through intervening on a system. The implications of an event can differ strongly, depending on whether it was the result of observation or intervention. For example, *observing* that nothing happened when your friend attempted to levitate a pencil would provide evidence against his claim of having psychic powers; *intervening* to hold the pencil down, and thus guaranteeing that it did not move during his attempted act of levitation, would remove any opportunity for this event to provide such evidence.

In causal graphical models, the consequences of intervening on a particular variable are be assessed by removing all incoming edges to the variable that was intervened on, and performing probabilistic inference in the resulting "mutilated" model (Pearl, 2000). This procedure produces results that align with our intuitions in the psychic powers example: intervening on X_2 breaks its connection with X_3, rendering the two variables independent. As a consequence, X_2 cannot provide evidence as to the value of X_3. Introductions to causal graphical models that consider applications to human cognition are provided by Glymour (2001) and Sloman (2005). There are also several good resources for more

technical discussions of learning the structure of causal graphical models (Heckerman, 1998; Glymour & Cooper, 1999).

Undirected graphical models

Undirected graphical models, also known as Markov Random Fields (MRFs), consist of a set of nodes, representing random variables, and a set of undirected edges, defining neighborhood structure on the graph which indicates the probabilistic dependencies of the variables at the nodes (e.g., Pearl, 1988). Each set of fully-connected neighbors (known as a *clique*) is associated with a *potential* function, which varies as the associated random variables take on different values. When multiplied together, these potential functions give the probability distribution over all the variables. Unlike directed graphical models, there need be no simple relationship between these potentials and the local conditional probability distributions. Moreover, undirected graphical models usually have closed loops (if they do not, then they can be reformulated as directed graphical models (e.g., Pearl, 1988)).

In many unsupervised learning problems, the observable data are believed to reflect some kind of underlying latent structure. For example, in a clustering problem, we might only see the location of each point, but believe that each point was generated from one of a small number of clusters. Associating the observed data with random variables X_i and the latent variables with random variables Y_i, we might want to define a probabilistic model for X_i that explicitly takes into account the latent structure Y_i. Such a model can ultimately be used to make inferences about the latent structure associated with new datapoints, as well as providing a more accurate model of the distribution of X_i. In the following sections we will use X_i to refer to variables whose values can be directed observed and Y_i to refer to latent, or hidden, variables whose values can only be inferred. We will use the vector notation $\vec{x}$ and $\vec{y}$ to represent the values taken by these random

variables, being $x_1, x_2, \ldots$ and $y_1, y_2, \ldots$ respectively.

A standard model used in computer vision is of form: $P(\vec{x}|\vec{y}) = (\prod_i P(x_i|y_i))\, P(\vec{y})$ where the prior distribution on the latent variables is an MRF,

$$P(\vec{y}) = \frac{1}{Z} \prod_{i,j \in \Lambda} \psi_{ij}(y_i, y_j) \prod_i \psi_i(y_i) \tag{18}$$

where Z is a normalizing constant ensuring that the resulting distribution sums to 1 (e.g., Geman & Geman, 1984). Here, $\psi_{ij}(\cdot,\cdot)$ and $\psi_i(\cdot)$ are the potential functions, and the underlying graph is a lattice, with Λ being the set of connected pairs of nodes (see Figure 3). This model has many applications. For example, $\vec{x}$ can be taken to be the observed intensity values of a corrupted image and $\vec{y}$ the true image intensity, with $P(x_i|y_i)$ modeling the corruption of these intensity values. The prior $P(\vec{y})$ is used to put prior probabilities on the true intensity, for example that neighboring intensity values are similar (e.g. that the intensity is spatially smooth). A similar model can be used for binocular stereopsis, where the $\vec{x}$ correspond to the image intensities in the left and right eyes and $\vec{y}$ denotes the depth of the surface in space that generates the two images. The prior on $\vec{y}$ can assume that the depth is a spatially smoothly varying function.

Another example of an MRF is the Boltzmann Machine, which has been very influential in the neural network community (Dayan & Abbott, 2001; Mackay, 2003). In this model the components x_i and y_i of the observed and latent variables $\vec{x}$ and $\vec{y}$ all take on values 0 or 1. The standard model is

$$P(\vec{y}, \vec{x}|\vec{\omega}) = \frac{1}{Z} \exp\{-E(\vec{y}, \vec{x}, \vec{\omega})/T\} \tag{19}$$

where T is a parameter reflecting the "temperature" of the system, and E depends on unknown parameters $\vec{\omega}$ which are weighted connections ω^h_{ij} between hidden variables y_i, y_j and ω^o_{ij} between observed and hidden variables x_i, y_j,

$$E(\vec{y}, \vec{x}, \vec{\omega}) = \sum_{ij} \omega^o_{ij} x_i y_j + \sum_{ij} \omega^h_{ij} y_i y_j. \tag{20}$$

In this model, the potential functions are of the form $\exp\{-\omega^o_{ij} x_i y_j\}$ and $\exp\{-\omega^h_{ij} y_i y_j\}$, and the underlying graph connects pairs of observed and hidden variables and pairs of hidden variables (see Figure 3). Training the Boltzmann Machine involves identifying the correct potential functions, learning the parameters $\vec{\omega}$ from training examples. Viewing Equation 19 as specifying the likelihood of a statistical model, inferring $\vec{\omega}$ can be formulated as a problem of Bayesian inference of the kind discussed above.

Algorithms for inference

The presence of latent variables in a model poses two challenges: inferring the values of the latent variables, conditioned on observable data (i.e. computing $P(\vec{y}|\vec{x})$), and learning the probability distribution $P(\vec{x}, \vec{y})$ from training data (e.g., learning the parameters $\vec{\omega}$ of the Boltzmann Machine). In the probabilistic framework, both these forms of inference reduce to inferring the values of unknown variables, conditioned on known variables. This is conceptually straightforward but the computations involved are difficult and can require complex algorithms (unless the problem has a simple graph structure which allows dynamic programming to be used). In this section, we will discuss two algorithms that can be used to solve this problem.

The Expectation-Maximization algorithm

A standard approach to solving the problem of estimating probability distributions involving latent variables from training data is the Expectation-Maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977). Imagine we have a model for data $\vec{x}$ that has parameters θ, and latent variables $\vec{y}$. A mixture model is one example of such a model, in which the distribution of X_i is assumed to be a mixture of several other distributions, each responsible for a cluster of observations. For example, we might believe that our data were generated from two clusters, each associated with a different Gaussian (i.e. normal) distribution. If we let y_i denote the cluster from which the datapoint x_i was generated,

and assume that there are K such clusters, then the probability distribution over x_i is

$$P(x_i) = \sum_{k=1}^{K} P(x_i|y_i = k)P(y_i = k) \tag{21}$$

where $P(x_i|y_i = k)$ is the distribution associated with cluster k, and $P(y_i = k)$ is the probability that a point would be generated from that cluster. If we can estimate the parameters that characterize these distributions, we can infer the probable cluster membership (y_i) for any datapoint (x_i).

The likelihood for a mixture model is $P(\vec{x}|\theta) = \sum_{\vec{y}} P(\vec{x}, \vec{y}|\theta)$ where the latent variables $\vec{y}$ are unknown. The EM algorithm is a procedure for obtaining a maximum-likelihood (or MAP estimate) for θ, without resorting to generic methods such as differentiating $\log P(\vec{x}|\theta)$. The key idea is that if we knew the values of the latent variables $\vec{y}$, then we could find θ by using the standard methods for estimation discussed above. Even though we might not have perfect knowledge of $\vec{y}$, we can still assign probabilities to $\vec{y}$ based on $\vec{x}$ and our current guess of θ, $P(\vec{y}|\vec{x}, \theta)$. The EM algorithm for maximum-likelihood estimation proceeds by repeatedly alternating between two steps: evaluating the expectation of the "complete log-likelihood" $\log P(\vec{x}, \vec{y}|\theta)$ with respect to $P(\vec{y}|\vec{x}, \theta)$ (the E-step), and maximizing the resulting quantity with respect to θ (the M-step). For many commonly used distributions, it is possible to compute the expectation in the E-step without enumerating all possible values for $\vec{y}$. The EM algorithm is guaranteed to converge to a *local* maximum of $P(\vec{x}|\theta)$ (Dempster et al., 1977), and both steps can be interpreted as performing hillclimbing on a single "free energy" function (Neal & Hinton, 1998). An illustration of EM for a mixture of Gaussians appears in Figure 4 (a).

Markov chain Monte Carlo

The EM algorithm represents an intuitive solution to the problem of parameter estimation with latent variables, and is effective for a range of applications, but it only provides a solution to one of the inference problems faced in probabilistic models. Indeed,

the EM algorithm requires that we be able to solve the other problem – computing $P(\vec{y}|\vec{x},\theta)$ – in order to be able to perform the E-step (although see Jordan, Ghahramani, Jaakkola, & Saul, 1999, for methods for dealing with this issue in complex probabilistic models). Another class of algorithms, Markov chain Monte Carlo (MCMC) methods, provide a means of obtaining samples from complex distributions, which can be used both for inferring the values of latent variables and for identifying the parameters of models.

MCMC algorithms were originally developed to solve problems in statistical physics (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953), and are now widely used both in physics (e.g., Newman & Barkema, 1999) and in statistics (e.g., Gilks, Richardson, & Spiegelhalter, 1996; Mackay, 2003; Neal, 1993). As the name suggests, Markov chain Monte Carlo is based upon the theory of Markov chains – sequences of random variables in which each variable is independent of all of its predecessors given the variable that immediately precedes it (e.g., Norris, 1997). The probability that a variable in a Markov chain takes on a particular value conditioned on the value of the preceding variable is determined by the *transition kernel* for that Markov chain. One well known property of Markov chains is their tendency to converge to a *stationary distribution*: as the length of a Markov chain increases, the probability that a variable in that chain takes on a particular value converges to a fixed quantity determined by the transition kernel.

In MCMC, a Markov chain is constructed such that its stationary distribution is the distribution from which we want to generate samples. Since these methods are designed for arbitrary probability distributions, we will stop differentiating between observed and latent variables, and just treat the distribution of interest as $P(\vec{x})$. An MCMC algorithm is defined by a transition kernel $K(\vec{x}|\vec{x}')$ which gives the probability of moving from state $\vec{x}$ to state $\vec{x}'$. In order for the Markov chain to have the target distribution $P(\vec{x})$ as its stationary distribution, the transition kernel must be chosen so that the $P(\vec{x})$ is invariant to the kernel. Mathematically this is expressed by the condition $\sum_{\vec{x}} P(\vec{x})K(\vec{x}|\vec{x}') = P(\vec{x}')$.

If this is the case, once the probability that the chain is in a particular state is equal to $P(\vec{x})$, it will continue to be equal to $P(\vec{x})$ – hence the term "stationary distribution". A variety of standard methods exist for constructing transition kernels that satisfy this criterion, including *Gibbs sampling* and the *Metropolis-Hastings algorithm* (see Gilks et al., 1996). The algorithm then proceeds by repeatedly sampling from the transition kernel $K(\vec{x}|\vec{x}')$, starting from any initial configuration $\vec{x}$. The theory of Markov chains guarantees that the states of the chain will ultimately be samples from the distribution $P(\vec{x})$. These samples enable us to estimate properties of the distribution such as the most probable state or the average state. The results of an MCMC algorithm (in this case, Gibbs sampling) for mixtures of Gaussians are shown in Figure 4 (b).

Applying MCMC methods to distributions with latent variables, or treating the parameters of a distribution as a random variable in itself (as is standard in Bayesian statistics, as discussed above), makes it possible to solve both of the problems of inference faced by users of probabilistic models. The main problems with MCMC methods are that they can require a long time to converge to the target distribution, and assessing convergence is often difficult. Designing an MCMC algorithm is more of an art than a science. A poorly designed algorithm will take a very long time to converge. There are, however, design principles which often lead to fast MCMC. For example, transition kernels can be carefully constructed to guide the Markov chain so that it explores important parts of the space of possible $\vec{x}$.

More complex probabilistic models

The basic ideas outlined in the previous section can translate into some quite expressive and powerful probabilistic models. In this section, we will illustrate this by briefly reviewing two of these models: hidden Markov models and probabilistic context-free grammars. These models have largely been used in computational linguistics,

so we will focus on their application to language, but the underlying principles can be used to define probabilistic models in any domain. A more detailed account of the implications of probabilistic models of language appears in Chater and Manning (2006).

Hidden Markov models

Hidden Markov models (or HMMs) are an important class of graphical models that have been used for problems such as speech and language processing. For example, in speech recognition a hidden Markov model might be used to capture the pattern of speech sounds that make up an individual word. As shown in Figure 5 (a), the HMM for a word W assumes that there is a sequence of T observations $\{x_t : t = 1, ..., T\}$ (taking L values) generated by a set of hidden states $\{y_t : t = 1, ..., T\}$ (taking K values). The joint probability distribution is defined by $P(\{y_t\}, \{x_t\}, W) = P(W)P(y_1|W)P(x_1|y_1, W) \prod_{t=2}^{T} P(y_t|y_{t-1}, W)P(x_t|y_t, W)$. The HMM for W is defined by the probability distributions $P(y_1|W)$, the $K \times K$ probability transition matrix $P(y_t|y_{t-1}, W)$, the $K \times L$ observation probabilty matrix $P(x_t|y_t, W)$, and the prior probability of the word $P(W)$. Using hidden Markov models to recognize words requires solving three related inference tasks. First, we need to learn the models (i.e. $P(x_t|y_t, W)$ and $P(y_t|y_{t-1}, W)$) for each word W. Second, we need to evaluate the probability $P(\{x_t\}, W) = \sum_{\{y_t\}} P(\{y_t\}, \{x_t\}, W)$ for the observation sequence $\{x_t\}$ for each word W. Third, we must recognize the word by model selection to estimate $W^* = \arg\max_W \sum_{\{y_t\}} P(\{y_t\}, W|\{x_t\})$. These problems can be solved using probabilistic inference algorithms based on dynamic programming and the principles behind the EM algorithm (Rabiner, 1989).

Figure 5 (b) shows a simple example of a hidden Markov model. The representation of the HMM used in this panel is not a graphical model, as in Figure 5 (a), but a *state transition diagram*, indicating the states of the model, the observations these states

generate, and the probabilities of transitioning between states and producing observations. In this model, it is assumed that somebody has two coins, one biased and the other fair, and produces a sequence of heads and tails by flipping these coins, with the coins being switched occasionally. The observable values 0 and 1 indicate whether the coin comes up heads or tails. The hidden states A and B indicate which coin is used on a given flip. There are (unknown) transition probabilities between the hidden states A and B, and (unknown) probabilities for the observations $0, 1$ conditioned on the hidden states. Given this structure, the learning, or training, task of the HMM is to estimate the probabilities from a sequence of measurements. The HMM can then be used to estimate the hidden states and the probability that the model generated a particular sequence of observations (so that it can be compared to alternative models for classification).

Hidden Markov models were first described by Baum and Petrie (1966). HMMs are widely used for processing text and speech (e.g., Charniak, 1993; Jurafsky & Martin, 2000). One of their most successful applications, other than speech recognition, is part-of-speech tagging (e.g., Charniak, Hendrickson, Jacobson, & Perkowitz, 1993), in which the latent states are trained to match the syntactic categories, such as nouns and verbs, that comprise the most basic level of linguistic structure. These analyses are similar to the "distributional clustering" methods that cognitive scientists have claimed may play a role in the acquisition of syntactic categories by children (e.g., Redington, Chater, & Finch, 1998).

Probabilistic context-free grammars

Hidden Markov models are equivalent to probabilistic finite state automata, and consequently are a form of probabilistic regular grammar (e.g., Manning & Shutze, 1999). Generative models can be defined in terms of probabilistic versions of other grammars in Chomsky's (1956) hierarchy of formal languages. In particular, computational linguists

have explored probabilistic context-free grammars (PCFGs; Baker, 1979, Charniak 1993, Jurafsky & Martin, 2000, Manning & Shutze, 1999, Geman & Johnson, 2004). A PCFG can be used to generate sentences and to parse them. The probability of a sentence, or a parse, is defined to be the product of the probabilities of the production rules used to generate the sentence.

For example, we can define a PCFG as follows (see Figure 5 (c)). We define non-terminal nodes $S, NP, VP, AT, NNS, VBD, PP, IN, DT, NN$ where S is a sentence, VP is a verb phrase, VBD is a verb,NP is a noun phrase, NN is a noun, and so on (for more details, see Manning & Schütze, 1999). The terminal nodes are words from a dictionary (e.g. "the", "cat", "sat", "on", "mat".) We define production rules which are applied to non-terminal nodes to generate child nodes (e.g. $S \mapsto NP, VP$ or $NN \mapsto$ "cat"). Finally, we specify probabilities for the production rules, so that the probabilities assigned to production rules with the same non-terminal symbol on the left hand side sum to one.

These production rules enable us to generate a sentence starting from the root node S. We sample a production rule from the distribution over rules with S on the left hand side, and apply it to generate child nodes. We repeat this process on the child nodes and stop when all the nodes are terminal (i.e. all are words). The result is a "parse tree" of the kind shown in Figure 5 (c) – a hierarchical representation of the structure of a sentence. As a result of following this procedure for generating parse trees, the probability of obtaining a particular parse tree (with an accompanying sequence of words) is just the product of the probabilities of the rules that produce that parse tree. Multiple parse trees can result in the same sequence of words. To parse an input sentence, we use dynamic programming to compute the most probable way that sequence of words could have been generated by the production rules.

As with hidden Markov models, we can learn the probabilities associated with

production rules from a set of observed sentences. This can be done in a supervised fashion, using sentences for which the true parse structure is known, or an unsupervised fashion, working just from the sentences. The supervised learning problem is similar to the problem of estimating the probability with which a coin will produce heads, although on a much larger scale. The unsupervised learning problem involves latent variables – the parse trees – making it an attractive target for methods related to the EM algorithm (Lari & Young, 1990) and Markov chain Monte Carlo (Johnson, Griffiths, & Goldwater, 2007).

Probabilistic context-free grammars are used throughout computational linguistics, and are the subject of many formal results concerning the tractability of language acquisition (e.g., Horning, 1969; Angluin, 1988). The formalism is also beginning to be used in psycholinguistics, in modeling people's treatment of phenomena disambiguation and attachment (e.g., Baldewein & Keller, 2004; Jurafsky, 1996). Recent work in bioinformatics has also begun to exploit the capacity of PCFGs to capture complex dependencies among the elements of sequences (e.g., Ding, Chan, & Lawrence, 2005). However, the strong independent assumptions that underlie PCFGs, as well as their restriction to context-free languages, mean that they are largely a convenient approximation to the structure of human languages, and linguists continue to explore new probabilistic models that remove some of these constraints.

Conclusion

Probabilistic models provide a unique opportunity to develop a rational account of human cognition that combines statistical learning with structured representations. However, specifying and using these models can be challenging. The widespread use of probabilistic models in computer science and statistics has led to the development of valuable tools that address some of these challenges. Graphical models provide a simple and intuitive way of expressing a probability distribution, making clear the assumptions

about how a set of variables are related, and can greatly speed up probabilistic inference. The EM algorithm and Markov chain Monte Carlo can be used to estimate the parameters of models that incorporate latent variables, and to work with complicated probability distributions of the kind that often arise in Bayesian inference. These tools make it possible to work with richly structured probabilistic models, such as hidden Markov models and probabilistic context-free grammars. Current work in computer science and statistics is extending the scope of probabilistic models yet further, exploring models in which the number of clusters or features expressed in a dataset is unbounded (e.g., Neal, 1998; Griffiths & Ghahramani, 2005), where hypotheses are defined at multiple levels of abstraction (resulting in "hierarchical" Bayesian models) allowing information to be generalized across datasets (e.g., Tenenbaum, Griffiths, & Kemp, 2006), and where the properties of an object depend probabilistically both on other properties of that object and on properties of related objects (e.g., Friedman, Getoor, Koller, & Pfeffer, 1999; Kemp, Griffiths, & Tenenbaum, 2004). One of the great strengths of probabilistic models is this capacity to combine structured representations with statistical methods, providing a set of tools that can be used to explore how structure and statistics are combined. By using these tools and continuing to draw on advances in the many disciplines where probabilistic models are used, it may ultimately become possible to define models that can capture the complexity of human cognition.

References

Angluin, D. (1988). *Identifying languages from stochastic examples* (Tech. Rep. No. YALEU/DCS/RR-614). Yale University, Department of Computer Science.

Baker, J. (1979). Trainable grammars for speech recognition. In J. J. Wolf & D. H. Klatt (Eds.), *Speech communication papers presented at the 97th meeting of the acoustical society of america* (p. 547-550). MIT, Cambridge, Massachusetts.

Baldewein, U., & Keller, F. (2004). Modeling attachment decisions with a probabilistic parser: The case of head final structures. In *Proceedings of the 26th Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum.

Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, *37*, 1554-1563.

Bayes, T. (1763/1958). Studies in the history of probability and statistics: IX. Thomas Bayes's Essay towards solving a problem in the doctrine of chances. *Biometrika*, *45*, 296-315.

Berger, J. O. (1993). *Statistical decision theory and Bayesian analysis.* New York: Springer.

Bernardo, J. M., & Smith, A. F. M. (1994). *Bayesian theory.* New York: Wiley.

Bertsekas, D. P. (2000). *Dynamic programming and optimal control.* Nashua, NH: Athena Scientific.

Bishop, C. M. (2006). *Pattern recognition and machine learning.* New York: Springer.

Charniak, E. (1993). *Statistical language learning.* Cambridge, MA: MIT Press.

Charniak, E., Hendrickson, C., Jacobson, N., & Perkowitz, M. (1993). Equations for part-of-speech tagging. In *Proceedings of the tenth national conference on artificial intelligence (AAAI-93)* (p. 784-789). Washington, D.C.: AAAI Press.

Chater, N., & Manning, C. D. (2006). Probabilistic models of language processing and acquisition. *Trends in Cognitive Sciences*, *10*, 335-344.

Chomsky, N. (1956). Threee models for the description of language. *IRE Transactions on Information Theory*, *2*, 113-124.

Cox, R. T. (1961). *The algebra of probable inferencce.* Baltimore, MD: Johns Hopkins University Press.

Dayan, P., & Abbott, L. (2001). *Theoretical neuroscience.* Cambridge, MA: MIT Press.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, *39*.

Ding, Y., Chan, C. Y., & Lawrence, C. E. (2005). RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA*, *11*, 1157-1166.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification.* New York: Wiley.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In T. Dean (Ed.), *Proceedings of the 16th international joint conference on artificial intelligence (IJCAI)* (p. 1300-1309). San Francisco, CA: Morgan Kaufmann.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (1995). *Bayesian data analysis.* New York: Chapman & Hall.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 721-741.

Geman, S., & Johnson, M. (2004). Probability and statistics in computational linguistics, a brief review. In M. Johnson, S. P. Khudanpur, M. Ostendorf, & R. Rosenfeld (Eds.), *Mathematical foundations of speech and language processing* (p. 1-26). New York: Springer.

Gilks, W., Richardson, S., & Spiegelhalter, D. J. (Eds.). (1996). *Markov chain Monte Carlo in practice.* Suffolk, UK: Chapman and Hall.

Glymour, C. (2001). *The mind's arrows: Bayes nets and graphical causal models in psychology.* Cambridge, MA: MIT Press.

Glymour, C., & Cooper, G. (1999). *Computation, causation, and discovery.* Cambridge, MA: MIT Press.

Good, I. J. (1979). A. M. Turing's statistical work in World War II. *Biometrika*, *66*, 393-396.

Griffiths, T. L., & Ghahramani, Z. (2005). *Infinite latent feature models and the Indian buffet process* (Tech. Rep. No. 2005-001). Gatsby Computational Neuroscience Unit.

Griffiths, T. L., Kemp, C., & Tenenbaum, J. B. (in press). Bayesian models of cognition. In R. Sun (Ed.), *Cambridge handbook of computational cognitive modeling.* Cambridge: Cambridge University Press.

Griffiths, T. L., & Yuille, A. (2006). A primer on probabilistic inference. *Trends in Cognitive Sciences*, *10*(7).

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning: Data mining, inference, and prediction.* New York: Springer.

Heckerman, D. (1998). A tutorial on learning with Bayesian networks. In M. I. Jordan (Ed.), *Learning in graphical models* (p. 301-354). Cambridge, MA: MIT Press.

Horning, J. J. (1969). *A study of grammatical inference.* Unpublished doctoral dissertation, Stanford University.

Jaynes, E. T. (2003). *Probability theory: The logic of science.* Cambridge: Cambridge University Press.

Jeffreys, W. H., & Berger, J. O. (1992). Ockham's razor and Bayesian analysis. *American Scientist*, *80*(1), 64-72.

Johnson, M., Griffiths, T. L., & Goldwater, S. (2007). Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of the North American Conference on Computational Linguistics (NAACL '07).*

Jordan, M. I., Ghahramani, Z., Jaakkola, T., & Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, *37*, 183-233.

Jurafsky, D. (1996). A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, *20*, 137-194.

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing.* Upper Saddle River, NJ: Prentice Hall.

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, *90*, 773-795.

Kemp, C., Griffiths, T. L., & Tenenbaum, J. B. (2004). *Discovering latent classes in relational data* (Tech. Rep. No. AI Memo 2004-019). Cambridge, MA: Massachusetts Institute of Technology.

Korb, K., & Nicholson, A. (2003). *Bayesian artificial intelligence.* Boca Raton, FL: Chapman and Hall/CRC.

Laplace, P. S. (1795/1951). *A philosophical essay on probabilities* (F. W. Truscott & F. L. Emory, Trans.). New York: Dover.

Lari, K., & Young, S. (1990). The estimation of Stochastic Context-Free Grammars using the Inside-Outside algorithm. *Computer Speech and Language*, *4*(35-56).

Mackay, D. J. C. (2003). *Information theory, inference, and learning algorithms.* Cambridge: Cambridge University Press.

Manning, C., & Schütze, H. (1999). *Foundations of statistical natural language processing.* Cambridge, MA: MIT Press.

Marr, D. (1982). *Vision.* San Francisco, CA: W. H. Freeman.

Metropolis, A. W., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, *21*, 1087-1092.

Myung, I. J., Forster, M. R., & Browne, M. W. (2000). Model selection [special issue]. *Journal of Mathematical Psychology*, *44*.

Myung, I. J., & Pitt, M. A. (1997). Applying Occam's razor in modeling cognition: A Bayesian approach. *Psychonomic Bulletin and Review*, *4*, 79-95.

Neal, R. M. (1993). *Probabilistic inference using Markov chain Monte Carlo methods* (Tech. Rep. No. CRG-TR-93-1). University of Toronto.

Neal, R. M. (1998). *Markov chain sampling methods for Dirichlet process mixture models* (Tech. Rep. No. 9815). Department of Statistics, University of Toronto.

Neal, R. M., & Hinton, G. E. (1998). A view EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan (Ed.), *Learning in graphical models.* Cambridge, MA: MIT Press.

Newman, M. E. J., & Barkema, G. T. (1999). *Monte carlo methods in statistical physics.* Oxford: Clarendon Press.

Norris, J. R. (1997). *Markov chains.* Cambridge, UK: Cambridge University Press.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems.* San Francisco, CA: Morgan Kaufmann.

Pearl, J. (2000). *Causality: Models, reasoning and inference.* Cambridge, UK: Cambridge University Press.

Pitman, J. (1993). *Probability.* New York: Springer-Verlag.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257-286.

Redington, M., Chater, N., & Finch, S. (1998). Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, *22*, 425-469.

Rice, J. A. (1995). *Mathematical statistics and data analysis* (2nd ed.). Belmont, CA: Duxbury.

Russell, S. J., & Norvig, P. (2002). *Artificial intelligence: A modern approach* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.

Schachter, R. (1998). Bayes-ball: The rational pasttime (for determining irrelevance and requisite information in belief networks and influence diagrams. In *Proceedings of the fourteenth annual conference on uncertainty in artificial intelligence (uai 98).* San Francisco, CA: Morgan Kaufmann.

Sloman, S. (2005). *Causal models: How people think about the world and its alternatives.* Oxford: Oxford University Press.

Spirtes, P., Glymour, C., & Schienes, R. (1993). *Causation prediction and search.* New York: Springer-Verlag.

Tenenbaum, J. B., Griffiths, T. L., & Kemp, C. (2006). Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Science*, *10*, 309-318.

Author Note

This chapter is an extended version of Griffiths and Yuille (2006), a tutorial on probabilistic inference that appeared as an online supplement to a special issue of *Trends in Cognitive Sciences* on probabilistic models of cognition (Volume 10, Issue 7).

Footnotes

[1]We will use uppercase letters to indicate random variables, and matching lowercase variables to indicate the values those variables take on. When defining probability distributions, the random variables will remain implicit. For example, $P(a)$ refers to the probability that the variable A takes on the value a, which could also be written $P(A = a)$. We will write joint probabilities in the form $P(a, b)$. Other notations for joint probabilities include $P(a\&b)$ and $P(a \cap b)$.

Figure Captions

Figure 1. Comparing hypotheses about the weight of a coin. (a) The vertical axis shows log posterior odds in favor of h_1, the hypothesis that the probability of heads (θ) is drawn from a uniform distribution on $[0, 1]$, over h_0, the hypothesis that the probability of heads is 0.5. The horizontal axis shows the number of heads, N_H, in a sequence of 10 flips. As N_H deviates from 5, the posterior odds in favor of h_1 increase. (b) The posterior odds shown in (a) are computed by averaging over the values of θ with respect to the prior, $p(\theta)$, which in this case is the uniform distribution on $[0, 1]$. This averaging takes into account the fact that hypotheses with greater flexibility – such as the free-ranging θ parameter in h_1 – can produce both better and worse predictions, implementing an automatic "Bayesian Occam's razor". The solid line shows the probability of the sequence `HHTHTTHHHT` for different values of θ, while the dotted line is the probabiltiy of any sequence of length 10 under h_0 (equivalent to $\theta = 0.5$). While there are some values of θ that result in a higher probability for the sequence, on average the greater flexibility of h_1 results in lower probabilities. Consequently, h_0 is favored over h_1 (this sequence has $N_H = 6$). In contrast, a wide range of values of θ result in higher probability for for the sequence `HHTHHHTHHH`, as shown by the dashed line. Consequently, h_1 is favored over h_0, (this sequence has $N_H = 8$).

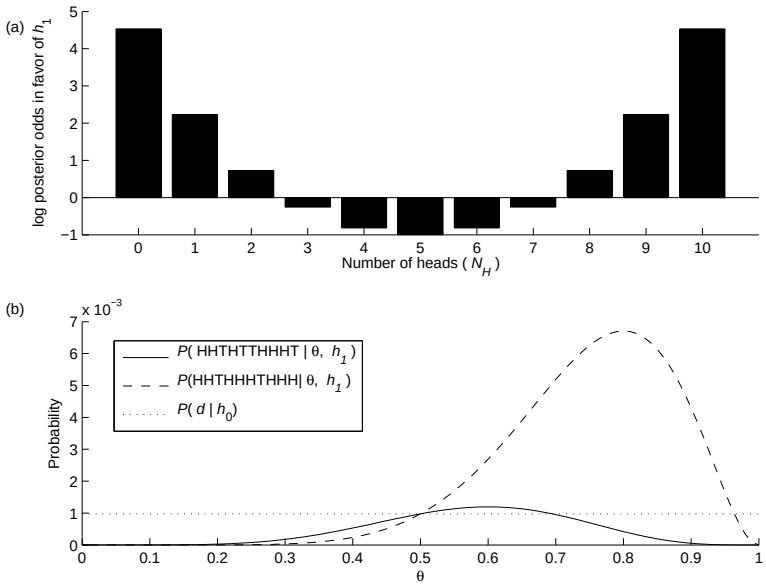
Figure 2. Directed graphical model (Bayes net) showing the dependencies among variables in the "psychic friend" example discussed in the text.

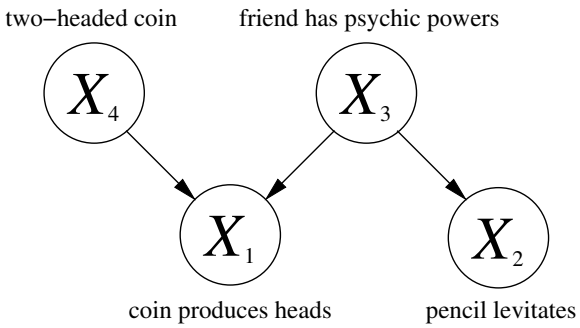
Figure 3. Markov random fields. (a) The undirected graph associated with a Markov random field where $\Lambda = \{(1, 2), (2, 3), (3, 4), (4, 1)\}$. (b) The undirected graph associated with a Boltzmann Machine, as discussed in the text.

Figure 4. Expectation-Maximization (EM) and Markov chain Monte Carlo (MCMC)

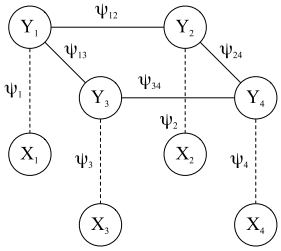
algorithms applied to a Gaussian mixture model with two clusters. Colors indicate the assignment of points to clusters (black and white), with intermediate greys representing probabilistic assignments. The ellipses are a single probability contour for the Gaussian distributions reflecting the current parameter values for the two clusters. (a) The EM algorithm assigns datapoints probabilistically to the two clusters, and converges to a single solution that is guaranteed to be a local maximum of the log-likelihood. (b) In contrast, an MCMC algorithm samples cluster assignments, so each datapoint is assigned to a cluster at each iteration, and samples parameter values conditioned on those assignments. This process converges to the posterior distribution over cluster assignments and parameters: more iterations simply result in more samples from this posterior distribution.

Figure 5. Probabilistic models of language. (a) The graphical model representation for a hidden Markov model (HMM). The Y_i are latent states, and the X_i observations. (b) A state transition diagram represetnation for an HMM. The circles represent the states of the model, with arrows connecting circles indicating transitions between states. Each state can also produce observations (in this case 0 and 1), with further arcs indicating the possible observations. The probabilities on the arcs represent the parameters of the model. (c) A parse tree from a probabilistic context-free grammar (PCFG).

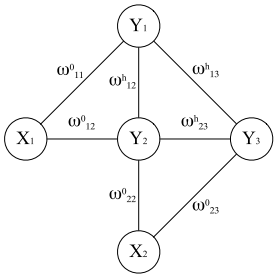

(a)
log posterior odds in favor of h_1
5
4
3
2
1
0
−1
0
1
2
3
4
5
6
7
8
9
10
Number of heads (N_H)
(b)
x 10^{-3}
7
6
5
4
3
2
1
0
Probability
P(HHTHTTHHHT | θ, h_1)
P(HHTHHHTHHH| θ, h_1)
P(d | h_0)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
θ

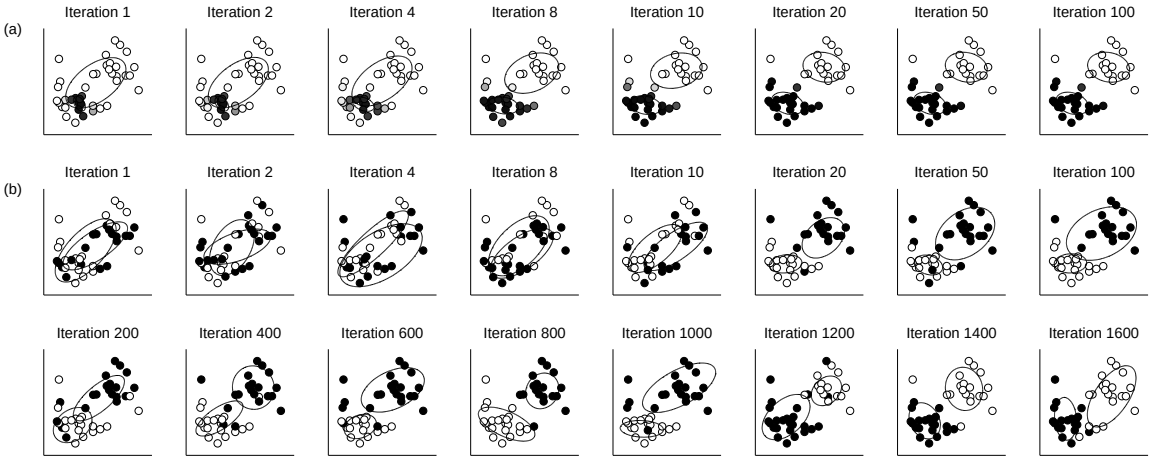

two−headed coin
friend has psychic powers
X_4
X_3
X_1
X_2
coin produces heads
pencil levitates

(a)

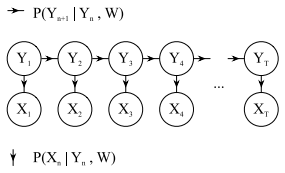

Y1
Y2
Y3
Y4
X1
X2
X3
X4
ψ12
ψ13
ψ24
ψ34
ψ1
ψ2
ψ3
ψ4

(b)

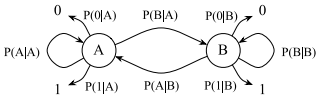

Y1
Y2
Y3
X1
X2
ω0 11
ω0 12
ω0 13
ω0 12
ω0 23
ω0 22
ω0 23


(a)
Iteration 1
Iteration 2
Iteration 4
Iteration 8
Iteration 10
Iteration 20
Iteration 50
Iteration 100
(b)
Iteration 1
Iteration 2
Iteration 4
Iteration 8
Iteration 10
Iteration 20
Iteration 50
Iteration 100
Iteration 200
Iteration 400
Iteration 600
Iteration 800
Iteration 1000
Iteration 1200
Iteration 1400
Iteration 1600

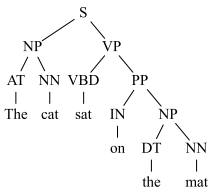
(a)


P(Y_{n+1} | Y_n , W)
Y1
Y2
Y3
Y4
...
YT
X1
X2
X3
X4
XT
P(X_n | Y_n , W)

(b)


0
P(0|A)
P(B|A)
P(0|B)
0
P(A|A)
A
B
P(B|B)
1
P(1|A)
P(A|B)
P(1|B)
1

(c)


S
NP
VP
AT
NN
VBD
PP
The
cat
sat
IN
NP
on
DT
NN
the
mat