Technical Introduction:  
A primer on probabilistic inference

Thomas L. Griffiths	extsuperscript{1} and Alan Yuille	extsuperscript{2}

	extsuperscript{1}Department of Cognitive and Linguistic Sciences, Brown University, Providence, RI, USA  
	extsuperscript{2}Department of Statistics, University of California, Los Angeles, CA, USA  

Corresponding authors: Griffiths, T.L. (tom_griffiths@brown.edu); Yuille, A. (yuille@stat.ucla.edu)

Research in computer science, engineering, mathematics and statistics has produced a variety of tools that are useful in developing probabilistic models of human cognition. We provide an introduction to the principles of probabilistic inference that are used in the papers appearing in this special issue. We lay out the basic principles that underlie probabilistic models in detail, and then briefly survey some of the tools that can be used in applying these models to human cognition.

Introduction

Probabilistic models aim to explain human cognition by appealing to the principles of probability theory and statistics, which dictate how an agent should act in situations that involve uncertainty. Although probability theory was originally developed as a means of analyzing games of chance, it was quickly realized that probabilities could be used as a guide to rational action \cite{1,2}. Probabilistic models are used in many disciplines, and are the method of choice for an enormous range of applications, including artificial systems for medical inference (e.g. \cite{3}), bioinformatics (e.g. \cite{4}) and computer vision (e.g. \cite{5}). Applying probabilistic models to human cognition thus provides the opportunity to draw upon work in computer science, engineering, mathematics, and statistics, often revealing surprising connections.

Developing probabilistic models of cognition involves two challenges. The first challenge is specifying a suitable model. This requires considering the computational problem faced by an agent and the knowledge available to that agent \cite{6,7}. 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 people might be able to make similar computations.

In this article, which is a supplementary article to the TICS July Special Issue on probabilistic models of cognition, we will introduce some of the tools that can be used to address these challenges. The plan of the article is as follows. First, we introduce Bayesian inference, which is at the heart of many probabilistic models. We then consider how to define 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 algorithms that are used to evaluate the predictions of probabilistic models: the Expectation-Maximization (EM) algorithm, and Markov chain Monte Carlo (MCMC). Several books provide a more detailed discussion of these topics in the context of statistics \cite{8–10}, machine learning \cite{11–13}, and artificial intelligence \cite{14–16}.

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.
we need a procedure for computing the agent’s beliefs in the light of the evidence provided by a hypothesis about this process, and is the hypothesis that.

The likelihoods associated with each hypothesis are:

\[
P(d|\theta) = \theta^N (1 - \theta)^T
\]  

(Eqn 3)

The likelihoods associated with \( h_0 \) and \( h_1 \) can thus be obtained by substituting the appropriate value of \( \theta \) into Equation 3.
We can place the priors and likelihoods defined above directly into Equation 1 to compute the posterior probability of each of our hypotheses. However, with just two hypotheses it is often easier to work with the posterior odds, 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

\[
\frac{P(h_1|d)}{P(h_0|d)} = \frac{P(d|h_1) P(h_1)}{P(d|h_0) P(h_0)} \tag{Eqn 4}
\]

where we have used the fact that the denominator of Equation 1 is constant. The first and second terms on the right hand side are called the likelihood ratio and the prior odds, respectively.

Equation 4 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, whereas the likelihood ratio expresses how these odds should be modified in light of the data \( d \). Returning to our example, we can use Equation 4 to compute the posterior odds of our two hypotheses for any observed sequence of heads and tails. Using the priors and likelihood ratios from the previous paragraph gives odds of around 357:1 in favor of \( h_1 \) for the sequence HHHHHHHHHH and 165:1 in favor of \( h_0 \) for the sequence HHTHTHTTHT.

### Comparing infinitely many hypotheses

The analysis outlined above for two simple hypotheses generalizes naturally to other finite (or countably infinite) sets. 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, \( \theta \), we were willing to consider any value of \( \theta \) between 0 and 1. What should we infer about the value of \( \theta \) from a sequence such as HHHHHHHHHH?

In Bayesian statistics, inferring \( \theta \) is treated just like any other Bayesian inference. If we assume that \( \theta \) is a random variable, then we can apply Bayes’ rule to obtain

\[
p(\theta|d) = \frac{P(d|\theta)p(\theta)}{P(d)} \tag{Eqn 5}
\]

where

\[
P(d) = \int_0^1 P(d|\theta)p(\theta)d\theta \tag{Eqn 6}
\]

The key difference from Bayesian inference with a finite or countably infinite set of hypotheses is that the posterior distribution is now characterized by a probability density, which we indicate by using \( p(\cdot) \) instead of \( P(\cdot) \), and the sum over hypotheses becomes an integral.

The posterior distribution over \( \theta \) contains more information than a single point estimate; it indicates not just which values of \( \theta \) 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 ‘principle of least commitment’ [6]). 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 \( \theta \) that maximizes the posterior probability, as given by Equation 5. The second method is computing the posterior mean of the quantity in question. For example, we could compute the posterior mean value of \( \theta \), which would be

\[
\bar{\theta} = \int_0^1 \theta p(\theta|d)d\theta \tag{Eqn 7}
\]

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

Different choices of the prior, \( p(\theta) \), will lead to different guesses at the value of \( \theta \). A first step might be to assume a uniform prior over \( \theta \) with \( p(\theta) \) being equal for all values of \( \theta \) between 0 and 1. Using a little calculus, it is possible to show that the posterior distribution over \( \theta \) produced by a sequence \( d \) with \( Nh \) heads and \( Nt \) tails is
The solution to this problem is to compute the marginal probability of the data under $h_i$. Applying the principle of marginalization mentioned above, we can define the joint distribution over $d$ and $\theta$ given $h_i$, and then integrate over $\theta$ to obtain

$$P(d|h_i) = \int \theta p(d|\theta, h_i) p(\theta|h_i) d\theta$$  \hspace{1cm} (Eqn 10)
where \( p(\theta|h_1) \) is the distribution over \( \theta \) 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 \( \theta \) (Equation 6). 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 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 1a 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 a difficult problem. Complex hypotheses have more degrees of freedom that can be adapted to the data, and can thus always be made to fit better than simple hypotheses. For example, for any sequence of heads and tails, we can always find a value of \( \theta \) 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 HHTTHHHTHHTT, \( P(d|\theta, h_1) \) 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 \( \theta \) 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 complex models is known as the ‘Bayesian Occam’s razor’ \([21,13]\), and is illustrated in Figure 1b.

**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 (\( \theta \)) 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 \( \theta \) with respect to the prior, \( p(\theta) \). This averaging takes into account the fact that hypotheses with greater flexibility can produce both better and worse predictions. The red line shows the probability of the sequence HHTTHHHTHHT for different values of \( \theta \), and the dotted line is the probability of any sequence of length 10 under \( h_0 \). On average, the greater flexibility of \( h_1 \) results in lower probabilities, so \( h_1 \) is favored over \( h_0 \). By contrast, a wide range of values of \( \theta \) result in higher probability for the sequence HHTHHHTHHH, as shown by the blue line, and \( h_1 \) is favored over \( h_0 \).
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, because the coin producing heads could also be explained by having substituted a two-headed coin. We can express all possible outcomes of the proposed tests, as well as their causes, using the binary random variables distribution over a set of hypotheses. In the general case, we use probabilistic inference to compute the probability being the hypotheses. However, in this setting, as with most probabilistic models, any variable can act as data or hypothesis. For example, if flipping the coin produces heads (x₁ = 1) should be higher if your friend actually does have psychic powers (x₃ = 1).

Once we have defined a joint distribution on x₁, x₂, x₃, and x₄, we can reason about the implications of events involving these variables. For example, if flipping the coin produces heads (x₁ = 1), then the probability distribution over the remaining variables is

\[ P(x₂, x₃, x₄ | x₁ = 1) = \frac{P(x₁ = 1, x₂, x₃, x₄)}{P(x₁ = 1)} \]  
(Eqn 11)

This equation can be interpreted as an application of Bayes' rule, with X₁ being the data, and X₂, X₃, X₄ 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, X₂, X₃, X₄) conditioned on a set of observed variables (here, X₁).

Although 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 write down simply a sensible joint distribution over a large set of variables. 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. Fortunately, research in statistics and computer science has produced a formal language for describing probability distributions that simplifies both representation and reasoning. This is the language of graphical models, in which the statistical dependencies that exist among a set of variables are represented graphically.

Directed graphical models

The most common kind of graphical models are directed graphical models, also known as Bayesian networks (or Bayes nets), which consist of a set of nodes, representing random variables, together with a set of directed edges from one node to another [16]. 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. Bayes nets use acyclic graphs, meaning that no node can be its own ancestor.

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 [16,22]. 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 X₁, X₂, ..., Xₘ, we can write:

\[ P(x₁, x₂, ..., xₘ) = \prod_i P(x_i | Pa(X_i)) \]  

where Pa(Xᵢ) is the set of parents of Xᵢ,
Figure 2 shows a Bayes net for the example of the friend who claims to have psychic powers. This Bayes net identifies several 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. There are several specialized algorithms for efficient probabilistic inference in Bayes nets, which make use of the dependencies among variables [16,14].

**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 a neighborhood structure on the graph which indicates the probabilistic dependencies of the variables at the nodes [16]. Each set of fully-connected neighbors 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. These models are widely used in computer vision [5], and some kinds of artificial neural networks inspired by ideas from statistical physics can be interpreted as undirected graphical models [23].
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 [33,22]. 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 assessed by removing all incoming edges to the variable that was intervened on, and performing probabilistic inference in the resulting ‘mutilated’ model [33]. This procedure produces results that align with our intuitions in the psychic powers example: intervening on \( X_1 \) breaks its connection with \( X_2 \), rendering the two variables independent. As a consequence, \( X_1 \) cannot provide evidence as to the value of \( X_2 \). Good summaries exist for both the psychological [34–36] and technical uses of causal graphical models [37,38].

**Box 2. Causal graphical models**

Different aspects of graphical models are emphasized in their use in the artificial intelligence and statistics communities. In the artificial intelligence community [15,16,14], the emphasis is on Bayes nets as a form of knowledge representation and an engine for probabilistic reasoning. Recently, research has begun to explore the use of graphical models for the representation of causal relationships (see Box 2). 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 (see Box 3), and is at the heart of probabilistic models of language (see Box 4).

**Box 3. Latent variables and mixture models**

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 \( X \) and the latent variables with random variables \( Z \), we might want to define a probabilistic model for \( X \) that explicitly takes into account the latent structure \( Z \). 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 \( X \).

A common example of a latent variable model is a mixture model, in which the distribution of \( X \) is assumed to be a mixture of several other distributions. For example, in the case of clustering, we might believe that our data were generated from two clusters, each associated with a different Gaussian (i.e. normal) distribution. If we let \( z \) denote the cluster from which the datapoint \( x \) was generated, and assume that there are \( K \) such clusters, then the probability distribution over \( x \) is

\[
P(x_i) = \sum_{k=1}^{K} P(x_i|z=k)P(z=k) \quad \text{(Eqn II)}
\]

where \( P(x_i|z=k) \) is the distribution associated with cluster \( k \), and \( P(z=k) \) 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 \( z \) for any datapoint \( x \). An example of a Gaussian mixture model (also known as a mixture of Gaussians) appears in Figure 3 in the main text.

**Box 4. Algorithms for inference**

The presence of latent variables in a model poses two problems: inferring the values of the latent variables conditioned on observed data, and learning the probability distribution characterizing both the observable and latent variables. 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.
Latent variable models for language

Latent variables are particularly useful in models of language, where they can be used to capture some of the unobserved structure that governs the words appearing in sentences. Hidden Markov models (HMMs) are one class of latent variable models that have been used for problems such as speech and language processing [39]. In an HMM, each word \( W_i \) is assumed to be generated from a latent class \( Z_i \), where \( Z_i \) is chosen from a distribution that depends on the class of the previous word, \( Z_{i-1} \). The classes thus form a Markov chain, which is ‘hidden’ because only the words are observable. The graphical model for an HMM is shown in Figure 1.

A probabilistic context-free grammar (PCFG) is a context-free grammar that associates a probability with each of its production rules [40]. The joint probability of a sentence and its parse tree is the product of the probabilities of the production rules used to generate the sentence. For example, we can define a PCFG as follows (and see Figure 1). The 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 [40]. The terminal nodes are words from a dictionary (e.g. ‘the’, ‘cat’, ‘sat’, ‘on’, ‘mat’). We then define production rules which are applied to non-terminal nodes to generate child nodes (e.g. \( S \rightarrow NP \), \( VP \) or \( NN \rightarrow ‘cat’ \)), and specify probabilities for these production rules. The production rules enable us to generate a sentence starting from the root node \( S \). We sample a production rule starting with \( S \) 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. words). To parse a sentence, we compute the most probable way the sentence could have been generated by the production rules. We can learn the production rules, and their probabilities, directly from sentences. This can be done in a supervised way, where the correct parses are known, or in an unsupervised way, as described in the article by Chater and Manning in this issue.

A standard approach to solving the problem of estimating probability distributions involving latent variables is the Expectation-Maximization (EM) algorithm [24]. Imagine we have a model for data \( x \) that has parameters \( \theta \) and latent variables \( z \). A mixture model is one example of such a model (see Box 3). The likelihood for this model is \( P(x \mid \theta) = \sum_z P(x, z \mid \theta) \) where the latent variables \( z \) are unknown. The EM algorithm is a procedure for obtaining a maximum-likelihood (or MAP) estimate for \( \theta \), without resorting to generic methods such as differentiating \( \log P(x \mid \theta) \). The key idea is that if we knew the values of the latent variables \( z \), then we could find \( \theta \) by using the standard methods for estimation discussed above. Even though we might not have perfect knowledge of \( z \), we can still assign probabilities to \( z \) based on \( x \) and our current guess of \( \theta \), \( P(z \mid x, \theta) \). The EM algorithm for maximum-likelihood estimation proceeds by repeatedly alternating between two steps: evaluating the expectation of the ‘complete log-likelihood’ \( \log P(x, z \mid \theta) \) with respect to \( P(z \mid x, \theta) \) (the E-step), and maximizing the resulting quantity with respect to \( \theta \) (the M-step). This algorithm is guaranteed to converge to a local maximum of \( P(x \mid \theta) \) [24], and both steps can be interpreted as performing hillclimbing on a single ‘free energy’ function [25]. An illustration of EM for a mixture of Gaussians appears in Figure 3a.
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. These algorithms were originally developed to solve problems in statistical physics [26], and are now widely used both in physics [27] and in statistics [28,13,29]. 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 [30]. 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. In MCMC, a Markov chain is constructed such that its stationary distribution is the distribution from which we want to generate samples. For example, we could construct a Markov chain that would converge to the posterior distribution over the parameters of a mixture of Gaussians given the data. The states of the Markov chain would be the parameter values, and could be used like samples from the posterior distribution once the Markov chain converged. There are a variety of standard MCMC algorithms, including Gibbs sampling [5], in which the Markov chain moves between states by drawing each of a set of variables from its distribution when conditioned on the values of all other variables, and the more general Metropolis–Hastings algorithm [26,31]. The results of applying Gibbs sampling to a mixture of Gaussians are shown in Figure 3b.

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. By using these tools and continuing to draw on advances in the many disciplines where probabilistic models are used, it might ultimately become possible to define models that can capture some of the complexity of human cognition.
References

2. Laplace, P.S. (1795/1951) A Philosophical Essay on Probabilities, Dover