16.6 Inference and Approximate Inference

One of the main ways we can use a probabilistic model is to ask questions about how variables are related to each other. Given a set of medical tests, we can ask what disease a patient might have. In a latent variable model, we might want to extract features $\mathbb{E}[\mathbf{h} \mid \mathbf{v}]$ describing the observed variables $\mathbf{v}$. Sometimes we need to solve such problems in order to perform other tasks. We often train our models using the principle of maximum likelihood. Because

$$\log p(\boldsymbol{v}) = \mathbb{E}_{\mathbf{h} \sim p(\mathbf{h} \mid \boldsymbol{v})} \left[\log p(\boldsymbol{h}, \boldsymbol{v}) - \log p(\boldsymbol{h} \mid \boldsymbol{v}) \right], \tag{16.9}$$

we often want to compute $p(\mathbf{h} \mid \boldsymbol{v})$ in order to implement a learning rule. All of these are examples of **inference** problems in which we must predict the value of some variables given other variables, or predict the probability distribution over some variables given the value of other variables.

Unfortunately, for most interesting deep models, these inference problems are intractable, even when we use a structured graphical model to simplify them. The graph structure allows us to represent complicated, high-dimensional distributions with a reasonable number of parameters, but the graphs used for deep learning are usually not restrictive enough to also allow efficient inference.

It is straightforward to see that computing the marginal probability of a general graphical model is #P hard. The complexity class #P is a generalization of the complexity class NP. Problems in NP require determining only whether a problem has a solution and finding a solution if one exists. Problems in #P require counting the number of solutions. To construct a worst-case graphical model, imagine that we define a graphical model over the binary variables in a 3-SAT problem. We can impose a uniform distribution over these variables. We can then add one binary latent variable per clause that indicates whether each clause is satisfied. We can then add another latent variable indicating whether all of the clauses are satisfied. This can be done without making a large clique, by building a reduction tree of latent variables, with each node in the tree reporting whether two other variables are satisfied. The leaves of this tree are the variables for each clause. The root of the tree reports whether the entire problem is satisfied. Due to the uniform distribution over the literals, the marginal distribution over the root of the reduction tree specifies what fraction of assignments satisfy the problem. While this is a contrived worst-case example, NP hard graphs commonly arise in practical real-world scenarios.

This motivates the use of approximate inference. In the context of deep learning, this usually refers to variational inference, in which we approximate the