Learning in undirected models

Let us now look at parameter learning in undirected graphical models. Unfortunately, as in the case of inference, the higher expressivity of undirected models also makes them significantly more difficult to deal with. Fortunately, maximum likelihood learning in these models can be reduced to repeatedly performing inference, which will allow us to apply all the approximate inference techniques that we have seen earlier.

Learning Markov random fields

Let us start with a Markov random field (MRF) of the form

$$p(x_1, .., x_n) = \frac{1}{Z(\varphi)} \prod_{c \in C} \phi_c(x_c; \varphi),$$

where

$$Z(\varphi) = \sum_{x_1, .., x_n} \prod_{c \in C} \phi_c(x_c; \varphi)$$

is the normalizing constant.

We can reparametrize p as follows:

$$\begin{aligned} p(x_1, .., x_n) &= \frac{1}{Z(\varphi)} \exp(\sum_{c \in C} \log \phi_c(x_c; \varphi)) \\ &= \frac{1}{Z(\varphi)} \exp(\sum_{c \in C} \sum_{x'_c} 1\{x'_c = x_c\} \log \phi_c(x'_c; \varphi)) \\ &= \frac{\exp(\theta^T f(x))}{Z(\theta)}, \end{aligned}$$

where $f(x)$ is a vector of indicator functions and θ is the set of all the model parameters, as defined by the $\log \phi_c(x'_c; \varphi)$.

Note that $Z(\theta)$ is different for each set of parameters θ; therefore, we also refer to it as the *partition function*. Bayesian networks are constructed such that $Z(\theta) = 1$ for all θ; MRFs do not make this modeling assumption, which makes them more flexible, but also more difficult to learn.

Exponential families

More generally, distributions of this form are members of the *exponential family* of distributions, about which you can learn more in CS229. Many other distributions are in the exponential family, including the Bernoulli, Multinomial, Normal, Poisson, and many other distributions.

Exponential families are widely used in machine learning[1]. Suppose that you have an exponential family distribution of the form[2]

$$p(x; \theta) = \frac{\exp(\theta^T f(x))}{Z(\theta)}.$$

Here are few facts about these distributions that you should know about.

Exponential families are log-concave in their *natural parameters* θ. The partition function $Z(\theta)$ is also log-convex in θ.

The vector $f(x)$ is called the vector of *sufficient statistics*; these fully describe the distribution p; e.g. if p is Gaussian, $f(x)$ contains (simple reparametrizations of) the mean and the variance of p.

Exponential families make the fewest unnecessary assumptions about the data distribution. More formally, the distribution maximizing the entropy $H(p)$ under the constraint $\mathbb{E}_p[\phi(x)] = \alpha$ (i.e. the sufficient statistics equal some value α) is in the exponential family.

Exponential families are also very convenient to work with computationally. Their sufficient statistics can summarize arbitrary amounts of i.i.d. variables from the same distribution, and they admit so-called conjugate priors which makes them easily applicable in variational inference. In short, it definitely pays off to learn more about exponential families!

Maximum likelihood learning of MRFs

Suppose now that we are given a dataset D and we want to estimate θ via maximum likelihood. Since we are working with an exponential family, the maximum likelihood will be concave. The log-likelihood is:

$$\frac{1}{|D|}\log p(D;\theta) = \frac{1}{|D|}\sum_{x\in D}\theta^T f(x) - \log Z(\theta).$$

The first term is linear in θ and is easy to handle. The second term equals

$$\log Z(\theta) = \log\sum_x \exp(\theta^T f(x)).$$

Unlike the first term, this one does not decompose across x. It is not only hard optimize, but it is hard to even evaluate that term, as we have already seen in previous chapters.

Now consider the gradient of the log likelihood. Obtaining the gradient of the linear part is obviously very easy. However, the gradient of $\log Z(\theta)$ takes a more complicated form:

$$\nabla_\theta \log Z(\theta) = \mathbb{E}_{x\sim p}[f(x)].$$

This expression can be derived using simple algebra.

Computing the expectation on the right hand side of the equation requires inference with respect to p. For example, we could sample from p and construct a Monte-Carlo estimate of the expectation. However, as we have seen, inference in general is intractable, and therefore so is computing the gradient.

We can similarly derive an expression for the Hessian of $\log Z(\theta)$:

$$\nabla_\theta^2 \log Z(\theta) = \mathbb{E}_{x\sim p}[f(x)f(x)^T] - \mathbb{E}_{x\sim p}[f(x)]\mathbb{E}_{x\sim p}[f(x)]^T = \text{cov}[f(x)].$$

Since covariance matrices are always positive semi-definite, this proves that $\log Z(\theta)$ is convex (and therefore that the log-likelihood is concave). Recall that this was one of the properties of exponential families that we stated earlier.

In summary, even though the log-likelihood objective is convex, optimizing it is hard; usually non-convexity is what makes optimization intractable, but in this case, it is the computation of the gradient.

Approximate learning techniques

Interestingly, however, maximum-likelihood learning reduces to inference in the sense that we may repeatedly use inference to compute the gradient and determine the model weights using gradient descent.

This observation lets us apply to learning many of the approximate inference methods that we have seen in previous chapters, such as:

> Gibbs sampling from the distribution at each step of gradient descent; we then approximate the gradient using Monte-Carlo.
>
> Persistent contrastive divergence[3], a variant of Gibbs sampling which re-uses the same Markov Chain between iterations. After a step of gradient descent, our model has changed very little: hence we can essentially keep taking samples from the same Gibbs sampler, instead of starting a new one from scratch.

Pseudo-likelihood

Another popular approach to learning p is called *pseudo-likelihood*. The pseudo-likelihood replaces the likelihood

$$\frac{1}{|D|}\log p(D;\theta) = \frac{1}{|D|}\sum_{x\in D}\log p(x;\theta)$$

with the following approximation:

$$\ell_{\mathrm{PL}}(\theta;D) = \frac{1}{|D|}\sum_{x\in D}\sum_{i}\log p(x_i \mid x_{N(i)};\theta),$$

where x_i is the i-th variable in x and $N(i)$ is the set of neighbors of i in the graph of g (i.e. the Markov blanket of i).

Note that each term $\log p(x_i \mid x_{N(i)};\theta)$ only involves one variable x_i and hence its partition function is going to be tractable (we only need to sum over the values of one variable).

However, it is not equal to the likelihood. Note that the correct way to expand the likelihood would involve the chain rule, i.e. the terms would be $\log p(x_i \mid x_{-i};\theta)$ objective, where x_{-i} are variables preceding i in some ordering.

Intuitively, the pseudo-likelihood objective assumes that x_i depends mainly on its neighbors in the graph, and ignores the dependencies on other, more distant variables. Observe also that if pseudo-likelihood succeeds in matching all the conditional distributions to the data, a Gibbs sampler run on the model distribution will have the same invariant distribution as a Gibbs sampler run on the true data distribution, ensuring that they are the same.

More formally, we can show that the pseudo-likelihood objective is concave. Assuming the data is drawn from an MRF with parameters θ^*, we can show that as the number of data points gets large, $\theta_{\mathrm{PL}} \to \theta^*$.

Pseudo-likelihood often works very well in practice, although there are exceptions.

Moment matching

We will end with an interesting observation about the maximum likelihood estimate θ^* of the MRF parameters.

Recall that the log-likelihood of an MRF is

$$\frac{1}{|D|}\log p(D;\theta) = \frac{1}{|D|}\sum_{x\in D}\theta^T f(x) - \log Z(\theta).$$

Taking the gradient, and using our expression for the gradient of the partition function, we obtain the expression

$$\frac{1}{|D|}\sum_{x\in D} f(x) - \mathbb{E}_{x\sim p}[f(x)]$$

Note that this is precisely the difference between the expectations of the natural parameters under the empirical (i.e. data) and the model distribution. Let's now look at one component of $f(x)$. Recall that we have defined f in the context of MRFs to be the vector of indicator functions for the variables of a clique: one entry of f equals $\mathbb{I}[x_c = \bar{x}_c]$ for some $x_c, \bar{x}_c$. The gradient over that component equals

$$\frac{1}{|D|}\sum_{x\in D}\mathbb{I}[x_c = \bar{x}_c] - \mathbb{E}_{x\sim p}[\mathbb{I}[x_c = \bar{x}_c]]$$

This gives us an insight into how MRFs are trained. The log-likelihood objective forces the model marginals to match the empirical marginals.

We refer to the above property as *moment matching*. This property of maximum-likelihood learning is very general: whenever we choose distribution q to minimize the inclusive KL-divergence $KL(p$

q)</script></span> across q in an exponential family, the minimizer q^* will match the moments of the sufficient statistics to the corresponding moments of p. Recall that the MLE estimate is the minimizer over q of $$D(\hat{p}

q), *where*\hat{p}*istheempiricaldistributionofthedata*, *soMLEestimationisaspecialcaseofthisminimization*. *Thispropertyhasco* is known as M-projection (``moment projection'').

Learning in conditional random fields

Finally, let us look how maximum-likelihood learning extends to conditional random fields (CRFs), the other important type of undirected graphical models that we have seen.

Recall that a CRF is a probability distribution of the form

$$p(y \mid x) = \frac{1}{Z(x, \varphi)} \prod_{c \in C} \phi_c(y_c, x; \varphi),$$

where

$$Z(x, \varphi) = \sum_{y_1, \ldots, y_n} \prod_{c \in C} \phi_c(y_c, x; \varphi)$$

is the partition function. The feature functions now depend on x in addition to y. The x variables are fixed and the distribution is only over y; the partition function is thus a function of both x and φ.

We can reparametrize p as we did for MRFs:

$$p(y \mid x) = \frac{\exp(\theta^T f(x, y))}{Z(x, \theta)},$$

where $f(y, x)$ is again a vector of indicator functions and θ is a reparametrization of the model parameters.

The log-likelihood for this model given a dataset D is

$$\frac{1}{|D|} \log p(D; \theta) = \frac{1}{|D|} \sum_{x, y \in D} \theta^T f(x, y) - \frac{1}{|D|} \sum_{x \in D} \log Z(x, \theta).$$

Note that this is almost the same form as we had for MRFs, except that now there is a different partition function $\log Z(x, \theta)$ for each data point x, y.

The gradient is now

$$\frac{1}{|D|} \sum_{x, y \in D} f(x, y) - \frac{1}{|D|} \sum_{x \in D} \mathbb{E}_{y \sim p(y|x)} [f(x, y)]$$

Similarly, the Hessian is going to be the covariance matrix

$$\text{cov}_{y \sim p(y|x)} [f(x, y)]$$

The good news is that this means that the conditional log-likelihood is still a concave function. We can optimize it using gradient ascent as before.

The bad news is that computing the gradient now requires one inference per training data point x, y in order to compute the term

$$\sum_{x,y \in D} \log Z(x, \theta).$$

This makes learning CRFs more expensive that learning in MRFs. In practice, however, CRFs are much more widely used than MRFs because supervised learning is a more widely used type of learning, and discriminative models (like CRFs) often learn a better classifier than their generative counterparts (which model $p(x, y)$).

To deal with the computational difficulties introduced by the partition function, we may use simpler models in which exact inference is tractable. This was the approach taken in the OCR example introduced in our first discussion of CRFs. More generally, one should try to limit the number of variables or make sure that the model's graph is not too densely connected.

Finally, we would like to add that there exists another popular objective for training CRFs called the max-margin loss, a generalization of the objective for training SVMs. Models trained using this loss are called *structured support vector machines* or *max-margin networks*. This loss is more widely used in practice because it often leads to better generalization, and also it requires only MAP inference to compute the gradient, rather than general (e.g. marginal) inference, which is often more expensive to perform.

Index Previous Next