More details on variational methods

Variational methods used to be complicated. After writing down the standard KL-divergence objective (previous note), researchers would try to derive clever fixed-point update equations to optimize it. For some models, including simple ones like logistic regression, this strategy didn't work out. Special-case variational objectives would be crafted for particular models. As a result, most text-book treatments of the applications of variational methods are fairly complicated, and beyond what's required for this course.

Fortunately the stochastic variational inference (SVI) methods developed in the last few years are simpler to understand, more general, and scale to enormous datasets. This note will outline enough of the idea to explain the demonstration code on the website. The demonstration is applied to logistic regression, but in principle its log-likelihood and gradients could be replaced with any other model with real-valued parameters.

As a reminder, we wish to minimize

$$J(\mathbf{m}, V) = \mathbb{E}_q[\log q(\mathbf{w})] - \mathbb{E}_q[\log p(\mathcal{D} \mid \mathbf{w})] - \mathbb{E}_q[\log p(\mathbf{w})], \tag{1}$$

with respect to our variational parameters $\{\mathbf{m}, V\}$, the mean and covariance of our Gaussian approximate posterior. For any $\{\mathbf{m}, V\}$ we have a lower bound on the log-marginal likelihood, or the model evidence[1]:

$$\log p(\mathcal{D}) \geq -J(\mathbf{m}, V). \tag{2}$$

We would like to maximize the model likelihood $p(\mathcal{D})$ with respect to any hyperparameters, such as the prior variance on the weights, σ_w^2. We can't do that exactly, but we can instead minimize J with respect to these parameters. So, we will jointly minimize J with respect to the variational distribution and model hyperparameters, aiming for a tight bound[2] and a large model likelihood.

1 Unconstrained optimization

Stochastic gradient descent on parameters V and σ_w^2 will sometimes set negative variances and covariances that aren't positive definite. Instead we should optimize unconstrained quantities, such as $\log \sigma_w$.

To optimize a covariance matrix, we can first write it in terms of its Cholesky decomposition: $V = LL^\top$. The diagonal elements of L are positive[3], the other elements are unconstrained. We take the log of the diagonal elements of the Cholesky decomposition, leaving the other elements equal to the values in the Cholesky decomposition, and optimize that unconstrained matrix.

2 The entropy terms

The negative entropy term $\mathbb{E}_q[\log q(\mathbf{w})]$ and cross-entropy term $\mathbb{E}_q[\log p(\mathbf{w})]$ can both be computed in closed form, if the prior is Gaussian. Substituting in the definition of a general multivariate Gaussian we get:

$$\mathbb{E}_{\mathcal{N}(\mathbf{w};\mathbf{m},V)}[\log \mathcal{N}(\mathbf{w};\, \mu,\, \Sigma)] = \mathbb{E}_{\mathcal{N}(\mathbf{w};\mathbf{m},V)}\left[-\frac{1}{2}(\mathbf{w}-\mu)^\top \Sigma^{-1}(\mathbf{w}-\mu)\right] - \frac{1}{2}\log|2\pi\Sigma|\,. \tag{3}$$

1. Where this "Evidence Lower BOund" is often called the ELBO.
2. The bound won't become tight in the sense of exact unless the true posterior is Gaussian.
3. The diagonal elements could be negative in the same sense that we could set σ_w negative and get a positive σ_w^2. However, optimizing the Cholesky decomposition L directly, allowing negative diagonal values, is not a good idea: the gradients become extreme when a diagonal element approaches zero.

The remaining expectation is of a quadratic form, and so can be expressed in terms of $\mathbf{m}$ and V. For the negative entropy term, things simplify considerably:

$$\mathbb{E}_{\mathcal{N}(\mathbf{w};\mathbf{m},V)}[\log \mathcal{N}(\mathbf{w};\,\mathbf{m},V)] = -\frac{D}{2} - \frac{1}{2}\log|2\pi V|, \tag{4}$$

where as usual, D is the number of parameters in $\mathbf{w}$. For a spherical Gaussian prior, the cross entropy term turns out to be:

$$-\,\mathbb{E}_{\mathcal{N}(\mathbf{w};\mathbf{m},V)}\left[\log \mathcal{N}(\mathbf{w};\,\mathbf{0},\,\sigma_w^2\mathbb{I})\right] = \frac{1}{2\sigma_w^2}(\mathrm{Trace}(V) + \mathbf{m}^\top\mathbf{m}) + \frac{D}{2}\log(2\pi\sigma_w^2). \tag{5}$$

We can evaluate both entropy terms, and they are both differentiable. The terms involving V are simple functions of the Cholesky decomposition: $\frac{1}{2}\log|V| = \sum_i \log L_{ii}$, and $\mathrm{Trace}(V) = \sum_{ij} L_{ij}^2$, making it easy to get the gradients we need.

3 The log-likelihood term

The final term, the average negative log-likelihood under the variational posterior,

$$-\,\mathbb{E}_{\mathcal{N}(\mathbf{w};\mathbf{m},V)}[\log p(\mathcal{D}\,|\,\mathbf{w})] = -\mathbb{E}_{\mathcal{N}(\mathbf{w};\mathbf{m},V)}\left[\sum_{n=1}^{N}\log p(y^{(n)}\,|\,\mathbf{x}^{(n)},\mathbf{w})\right] \tag{6}$$

cannot be computed in closed form. We could convert the integral of each term into a 1D integral and compute it numerically. However, that is expensive.

We only need unbiased estimates of the gradients to perform stochastic gradient descent. We can get a get a simple "Monte Carlo" unbiased estimate by sampling a random weight from the variational posterior:

$$-\,\mathbb{E}_{\mathcal{N}(\mathbf{w};\mathbf{m},V)}[\log p(\mathcal{D}\,|\,\mathbf{w})] \approx -\sum_{n=1}^{N}\log p(y^{(n)}\,|\,\mathbf{x}^{(n)},\mathbf{w}), \quad \mathbf{w}\sim\mathcal{N}(\mathbf{m},V). \tag{7}$$

We can also replace the sum by scaling up the contribution of a random example, and still get an unbiased estimate:

$$-\,\mathbb{E}_{\mathcal{N}(\mathbf{w};\mathbf{m},V)}[\log p(\mathcal{D}\,|\,\mathbf{w})] \approx -N\log p(y^{(n)}\,|\,\mathbf{x}^{(n)},\mathbf{w}), \quad \mathbf{w}\sim\mathcal{N}(\mathbf{m},V), n\sim\mathrm{Uniform}\{1..N\}. \tag{8}$$

Alternatively we could take the average log-likelihood for a random minibatch of M examples, and scale it up by N. The demonstration code just uses all of the data in each update, as the number of datapoints was small.

4 Gradients of the log-likelihood term

We could obtain gradients of the log-likelihood term by differentiating the expectation symbolically, and then finding a Monte Carlo approximation of the result. However, it's easier, and lower-variance to use a "reparameterization trick".

The standard way to sample a random weight $\mathbf{w}$ from the variational posterior is to sample a vector of standard normals $\boldsymbol{\nu}\sim\mathcal{N}(\mathbf{0},\mathbb{I})$ and transform it: $\mathbf{w} = \mathbf{m} + L\boldsymbol{\nu}$. Similarly, the expectation can be rewritten:

$$\mathbb{E}_{\mathcal{N}(\mathbf{w};\mathbf{m},V)}[f(\mathbf{w})] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\nu};\mathbf{0},\mathbb{I})}[f(\mathbf{m}+L\boldsymbol{\nu})]\,. \tag{9}$$

The expectation is now under a constant distribution, so it's easy to write down derivatives with respect to the variational parameters:

$$\nabla_{\mathbf{m}}\,\mathbb{E}_{\mathcal{N}(\mathbf{w};\mathbf{m},V)}[f(\mathbf{w})] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\nu};\mathbf{0},\mathbb{I})}[\nabla_{\mathbf{m}} f(\mathbf{m}+L\boldsymbol{\nu})] \tag{10}$$

$$\approx \nabla_{\mathbf{m}} f(\mathbf{m}+L\boldsymbol{\nu}), \quad \boldsymbol{\nu}\sim\mathcal{N}(\mathbf{0},\mathbb{I}), \tag{11}$$

and

$$\nabla_L \mathbb{E}_{\mathcal{N}(\mathbf{w};\mathbf{m},V)}[f(\mathbf{w})] \approx \nabla_L f(\mathbf{m} + L\boldsymbol{\nu}), \quad \boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}) \tag{12}$$

$$= [\nabla_{\mathbf{w}} f(\mathbf{w})]\boldsymbol{\nu}^\top, \quad \mathbf{w} = \mathbf{m} + L\boldsymbol{\nu}, \;\; \boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \mathbb{I}). \tag{13}$$

To estimate both gradients, we just need to be able to evaluate gradients of the log-likelihood function with respect to the weights, which we already know how to do if we can do maximum likelihood fitting.

5 Black box variational inference

Unbiased estimates of gradients of "the ELBO", the variational lower bound on the marginal likelihood, don't require any new derivations for new models. Tricks were required to make the parameterization of the Gaussian unconstrained, and to derive the working above. However, all of that work only needs doing once. In the demonstration code these details are put into one standard function. This routine can be used as a "black box" — like a generic optimizer. We pass the routine a function that evaluates our log likelihood and its gradients, and we can perform variational inference in the new model.

6 Check your understanding

If we have derivatives of a cost with respect to the prior variance σ_w^2, how do we convert them into derivatives with respect to $s_w = \log \sigma_w$? (Converting derivatives for the unconstrained version of the Cholesky factor is similar, just more details to keep track of.)

When attempting to fit $\mathbf{m}$ and V by stochastic gradient descent, you might find that the updates are quite noisy, and the variational parameters don't converge. What are two ways that you might fix this issue?

Do you recall how to use $\mathbf{m}$ and V to make probabilistic predictions for logistic regression? If you just wanted to make hard decisions, reporting whether $P(y=1 \mid \mathbf{x}, \mathcal{D}) > 0.5$, explain why you don't need V. Is there any point doing variational inference in this case?

If you want to reproduce all of the derivations in this note, you might find the trace trick from the maths cribsheet in the background materials useful.[4] However, I wouldn't make you do these long and detailed derivation on the exam.

7 References

Shakir Mohamed has recent tutorial slides on variational inference. The final slide has a reading list of both the classic and modern variational inference papers that discovered the theory in this note.

Black-box stochastic variational inference in five lines of Python, David Duvenaud, Ryan P. Adams, NIPS Workshop on Black-box Learning and Inference, 2015. The associated Python code and neural net demo require autograd.

---

4. As an example, here is how to find the following expectation over a D-dimensional vector $\mathbf{z}$:

$$\mathbb{E}_{\mathcal{N}(\mathbf{z};0,V)}[\log \mathcal{N}(\mathbf{z}; 0, V)]. \tag{14}$$

Using standard manipulations, including the "trace trick":

$$\mathbb{E}_{\mathcal{N}(\mathbf{z};0,V)}[\log \mathcal{N}(\mathbf{z}; 0, V)] + \frac{1}{2}\log|2\pi V| = \mathbb{E}_{\mathcal{N}(\mathbf{z};0,V)}\left[-\frac{1}{2}\mathrm{Tr}\left(\mathbf{z}^\top V^{-1}\mathbf{z}\right)\right] \tag{15}$$

$$= \mathbb{E}_{\mathcal{N}(\mathbf{z};0,V)}\left[-\frac{1}{2}\mathrm{Tr}\left(\mathbf{z}\mathbf{z}^\top V^{-1}\right)\right] \tag{16}$$

$$= -\frac{1}{2}\mathrm{Tr}\left(\mathbb{E}_{\mathcal{N}(\mathbf{z};0,V)}\left[\mathbf{z}\mathbf{z}^\top\right] V^{-1}\right) \tag{17}$$

$$= -\frac{1}{2}\mathrm{Tr}(VV^{-1}) = -\frac{1}{2}\mathrm{Tr}(\mathbb{I}_D) = -\frac{D}{2}. \tag{18}$$