DATA AUGMENTATION FOR TRAINING OF STOCHASTIC NEURAL NETWORKS

RYAN LEE

1. h_t Proposal Ideas / ReLU Networks (10/5/17)

In our first meeting, we discussed a number of ideas, mainly involving: 1) smart proposal ideas for the hidden state h_t; 2) using ReLU $[x]^+$ networks instead of binary networks; 3) considering probit instead of sigmoid activation.

Currently, our ideas seem to be:

- **Binary networks**. Use either

$$h_j^k \sim \text{Bern}(\sigma((W^k h^{k-1})_j))$$

or

$$h_j^k \sim \text{Bern}(\Phi((W^k h^{k-1})_j))$$

and then update/sample using

(a) W update. Use 1) Bayesian logistic regression; or 2) SGD with Langevin dynamics.

(b) h update. This is the tricky part. We know the posterior for all $h^1, \dots, h^L$ is as given below, but if we do proposals on all of the h^k simultaneously, then we will almost surely not accept the proposal. So we discussed *layer-wise proposal* methods for h^k, which has a Gibbs flavor by keeping all of the other h^{k-1}, h^{k+1} fixed. Then, the posterior of h^k is

$$p(h^k | h^{k-1}, h^{k+1}, y, x, W) \propto p(h^{k+1} | h^k, W^{k+1}) p(h^k | h^{k-1}, W^k)$$

where the right-hand is just a product of Bernoulli terms. But we need to use a smart proposal for h^k, since again the realizations are dependent. At the current state, we seem to have come up with two possibilities for this.

1) *Backprop-and-Reject*. The key idea is that we "pretend" that the stochastic hidden layer h^k is actually simply a deterministic hidden layer. That is, we consider the "false" hidden layer g^k, where we suppose that:

$$h^{k+1} \sim \text{Bern}(\sigma(W^{k+1} g^k))$$
$$g^k = W^k h^{k-1}$$

so then we can use cross-entropy loss and then backpropagate that loss. Let

$$g^{k+1} \equiv \sigma(W^{k+1} g_0^k)$$

Then, we fix h^{k-1}, h^{k+1} as before, and then we can compute

$$g_0^k = W^k h^{k-1}$$

and the cross-entropy loss is

$$L(g^{k+1}, h^{k+1}) = \sum_j h_j^{k+1} \log(g_j^{k+1})$$

which is differentiable, and then we can update

$$g^k = g_0^k + \nabla_{g^k} L$$

Finally, for the proposal, we sample

$$h^k \sim \text{Bern}(g^k)$$

and we do acceptance-rejection using the layer-wise posterior above.

2) *Component-wise Gibbs.* In addition to fixing the other layers, we can also fix the other components in the layer to do full Gibbs sampling. That is,

$$p(h_i^k | h_{j \neq i}^k, h^{k-1}, h^{k+1}, y, x, W) \propto p(h^{k+1} | h^k, W^{k+1}) p(h_i^k | h^{k-1}, W^k)$$

Since we can't sample from the full conditional, we can again to a Metropolis/accept-reject step by proposing from, for example,

$$h_i^k \sim \text{Bern}(\sigma(W^k h^{k-1}))$$

and then using the above posterior to do acceptance. This should yield higher acceptance rates than doing the layer at once. We can also incorporate the backprop idea here, though it may be too costly to do so many backprop operations.

- **ReLU networks**. Rather than sigmoid or tanh, the most popular activation function currently is the rectified linear unit (ReLU):

$$ReLU(x) = [x]^+$$

so the idea is that we can again turn the network stochastic by assuming Gaussian noise in the underlying variable and by adding a latent variable Z^k for each layer

$$h^k \sim [\mathcal{N}(W^k h^{k-1}, \sigma^2)]^+ = [Z^k]^+$$

where

$$Z^k = W^k h^{k-1} + \epsilon^k$$

In this case, the updating steps have the following form.

(a) W update. If all of the Z^k are fixed, then so are the h^k, so we can use a conjugate prior on W^k to obtain a closed-form posterior for the W^k that we can sample from.

(b) h update. The idea that I proposed here is to take advantage of the fact that even though the entire network is quite nonlinear, within each layer, the operations are almost linear. Thus, one idea is to "pretend" as if the layer is a linear layer with no ReLU activation, use this to propose h^k, and then correct using acceptance-rejection. That is, we pretend that

$$Z^{k+1} = W^{k+1} Z^k + \epsilon^{k+1}$$
$$Z^k = W^k Z^{k-1} + \epsilon^k$$

and so we have that under these false assumptions,

$$p(Z^k|Z^{k-1}, Z^{k+1}, y, x, W) \propto p(Z^{k+1}|Z^k, W^{k+1})p(Z^k|Z^{k-1}, W^k) = \mathcal{N}(W^{k+1}Z^k, \sigma^2)\mathcal{N}(W^k Z^{k-1}, \sigma^2)$$

which, by conjugacy, again gives us a closed form to sample Z^k. However, these is not the correct posterior sample, since our model was deliberately wrong. We can thus treat this as a proposal, then correct for this using acceptance-rejection using the true posterior.

2. Original Proposal

We would like to propose a data augmentation method for training stochastic neural networks. Several methods have recently been proposed to train *stochastic feedforward neural networks* (SFNN). These networks are standard feedforward neural networks (FNN) except that they sample instead of average, as explained below.

Let x be the input vector. In the standard FNN, using sigmoid activation, we have the l^{th} hidden layer defined deterministically as

$$h^l \equiv \sigma(W^l h^{l-1})$$

where σ is the sigmoid function, and if the network has L layers, then the output is

$$y \equiv W^O h^L$$

The stochastic version SFNN changes this by replacing the sigmoid by sampling. We can think of the standard version as taking an expectation:

$$h^l = E[\tilde{h}^l|W^l, h^{l-1}]$$

where

$$\tilde{h}^l \sim \text{Bern}(\sigma(W^l h^{l-1}))$$

So instead of taking this expectation, we can do sampling and define the l^{th} hidden layer as

$$\tilde{h}^l|W^l \sim \text{Bern}(\sigma(W^l\tilde{h}^{l-1}))$$
$$y \sim \mathcal{N}(W^O\tilde{h}^L, \sigma^2)$$

This led us to consider a *data augmentation scheme for SFNN*, which is the following algorithm:

- At step t, suppose we have a sample
$$\theta_t \equiv (W_t^1, W_t^2, \ldots, W_t^L, W_t^O)$$
- Then, we want to sample
$$h_t^1, \ldots, h_t^L \sim p(h^1, \ldots, h^L|y, x, \theta_t)$$

  We note that this can be done via Metropolis-Hastings or acceptance-rejection looking at the posterior:
$$p(h^1, \ldots, h^L|y, x, \theta) \propto p(y|h^1, \ldots, h^L, x, \theta)p(h^1, \ldots, h^L|x, \theta)$$
$$= \mathcal{N}(y|W^O h^L, \sigma^2) \cdot \text{Bern}(h^1|\sigma(W^1 x)) \cdots \text{Bern}(h^L|\sigma(W^L h^{L-1}))$$
  So we can use the following proposal procedure:
$$h_t^1|\theta_t \sim \text{Bern}(\sigma(W^1 x))$$
$$h_t^2|\theta_t, h_t^1 \sim \text{Bern}(\sigma(W^2 h_t^1))$$

$$\vdots$$

$$h_t^L|\theta_t, h_t^1, \ldots, h_t^{L-1} \sim \text{Bern}(\sigma(W^L h_t^{L-1}))$$

and then correct the entire sample using

$$\alpha = \min\left(1, \frac{\mathcal{N}(y|W^O h^L, \sigma^2)}{\mathcal{N}(y|W^O h'^L, \sigma^2)}\right)$$

where h' is the new proposal. We can run this chain for a few steps and use the last sample as the h_t.

- Now, given $(h_t^1, \ldots, h_t^L)$, we "sample" (or train) our parameters to yield θ_{t+1}. I can think of two possibilities, but there may be better solutions.
  1. *Logistic regression.* At each layer, use logistic regression:
  $$h_t^l \sim \text{Bern}(\sigma(W^l h_t^{l-1})) \Rightarrow W_{t+1}^l$$
  and then use linear regression on the last layer:
  $$y \sim \mathcal{N}(W^O h_t^L, \sigma^2) \Rightarrow W_{t+1}^O$$
  2. *SGD.* Similar to option (1), but instead of a deterministic regression method, just use stochastic gradient descent (SGD) to yield samples θ_{t+1}. This may be better for inducing mixing into the Gibbs chain.

There are a number of advantages of this proposed approach.

(a) It does not require *backpropagation*, which is generally the most costly step in training deep neural networks.
(b) The signal from y is propagated to all the hidden layers $h^1, \ldots, h^L$ at once during the sampling step, which means there is little chance for vanishing/exploding gradient problems.
(c) The training step for θ is done layer-by-layer, which has been shown to be much easier compared to full backpropagation.
(d) It yields samples $\theta_1, \ldots, \theta_T$, which can be used to do Bayesian inference on the network and predictions, and also yields variance estimates rather than point estimates.
(e) All steps are quite computationally straightforward, and inexpensive. Even the MCMC steps are trivial (computationally) due to the proposal moves.
(f) It makes the network naturally more robust to overfitting and small sample problems through stochasticity (i.e. many papers discuss stochastic networks due to their regularization properties).