Supplementary Material

Payam Delgosha
University of California, Berkeley
pdelgosha@eecs.berkeley.edu

Naveen Goela
Tanium, Data Science
ngoela@alum.mit.edu

A Overview

This document contains supplementary material for the paper "Deep Switch Networks for Generating Discrete Data and Language" accepted to the AISTATS 2019 conference.

B Approximate Gradient Calculation for the Two-Layer Network

In this section, we propose an approximate method to compute the gradients of the empirical likelihood of the two-layer switch network with respect to the model parameters. Recall from the main paper that in the two layer setup, the empirical log-likelihood has the following form

$$\begin{aligned} L^{(k)} &:= \frac{1}{I}\sum_{i=1}^I \log p\big(X_{k+1} = x^{(i)}_{k+1} | x^{(i)}_{[1:k]}; \theta^{(k)}\big) \\ &= \frac{1}{I}\sum_{i=1}^I \log \sum_{f^{(k)}_{[1:l]}} p\big(f^{(k)}_{[1:l]} | x^{(i)}_{[1:k]}; \theta^{(k)}_1\big) \times \\ &\qquad p\big(X_{k+1} = x^{(i)}_{k+1} | f^{(k)}_{[1:l]}; \theta^{(k)}_2\big). \end{aligned} \tag{1}$$

B.1 Likelihood Gradient

Differentiating $L^{(k)}$ with respect to $\theta^{(k)}_1$, the parameters of the first layer, we get

$$\begin{aligned} \frac{\partial L^{(k)}}{\partial \theta^{(k)}_1} &= \frac{1}{I}\sum_{i=1}^I \sum_{f^{(k)}_{[1:l]}} \frac{p(x^{(i)}_{k+1}|f^{(k)}_{[1:l]};\theta^{(k)}_2)\frac{\partial p(f^{(k)}_{[1:l]}|x^{(i)}_{[1:k]};\theta^{(k)}_1)}{\partial \theta^{(k)}_1}}{p(x^{(i)}_{k+1}|x^{(i)}_{[1:k]};\theta^{(k)})} \\ &= \frac{1}{I}\sum_{i=1}^I \sum_{f^{(k)}_{[1:l]}} \frac{p(x^{(i)}_{k+1}|f^{(k)}_{[1:l]};\theta^{(k)}_2)p(f^{(k)}_{[1:l]}|x^{(i)}_{[1:k]};\theta^{(k)}_1)}{p(x^{(i)}_{k+1}|x^{(i)}_{[1:k]};\theta^{(k)})} \\ &\qquad \times \frac{\partial}{\partial \theta^{(k)}_1} \log p(f^{(k)}_{[1:l]}|x^{(i)}_{[1:k]};\theta^{(k)}_1). \end{aligned} \tag{2}$$

Note that for each $1 \le i \le I$, the inner summation over $f^{(k)}_{[1:l]}$ could be written as an expectation with respect to the distribution p_i on $F^{(k)}_{[1:l]}$ defined as

$$p_i(f^{(k)}_{[1:l]};\theta^{(k)}) := \frac{p(x^{(i)}_{k+1}|f^{(k)}_{[1:l]};\theta^{(k)}_2)p(f^{(k)}_{[1:l]}|x^{(i)}_{[1:k]};\theta^{(k)}_1)}{W_i(\theta^{(k)})},$$

with the normalizing constant

$$W_i(\theta^{(k)}) := \sum_{\tilde{f}^{(k)}_{[1:l]}} p(x^{(i)}_{k+1}|\tilde{f}^{(k)}_{[1:l]};\theta^{(k)}_2)p(\tilde{f}^{(k)}_{[1:l]}|x^{(i)}_{[1:k]};\theta^{(k)}_1).$$

Using this, we may write (2) as

$$\frac{\partial L^{(k)}}{\partial \theta^{(k)}_1} = \frac{1}{I}\sum_{i=1}^I \mathbb{E}_{p_i}\left[\frac{\partial}{\partial \theta^{(k)}_1} \log p(F^{(k)}_{[1:l]}|x^{(i)}_{[1:k]};\theta^{(k)}_1)\right], \tag{3}$$

where in the expectation, on the right hand side, $F^{(k)}_{[1:l]}$ has the distribution p_i defined above. Similar calculation shows that

$$\frac{\partial L^{(k)}}{\partial \theta^{(k)}_2} = \frac{1}{I}\sum_{i=1}^I \mathbb{E}_{p_i}\left[\frac{\partial}{\partial \theta^{(k)}_2} \log p(x^{(i)}_{k+1}|F^{(k)}_{[1:l]};\theta^{(k)}_2)\right]. \tag{4}$$

B.2 Metropolis-Hastings Algorithm

Now, we may use the Metropolis–Hastings algorithm to approximate the expectation for each i. The reason is that the ratio of probabilities $p_i(f^{(k)}_{[1:l]};\theta^{(k)})/p_i(\tilde{f}^{(k)}_{[1:l]};\theta^{(k)})$ for two configurations $f^{(k)}_{[1:l]}$ and $\tilde{f}^{(k)}_{[1:l]}$ does not depend on the normalizing constant $W_i(\theta^{(k)})$ and can be computed efficiently. More precisely, for $1 \le i \le I$, we design a Markov Chain Monte Carlo (MCMC) algorithm with the proposal distribution

$$g_i(\tilde{f}^{(k)}_{[1:l]}|f^{(k)}_{[1:l]}) = p(\tilde{f}^{(k)}_{[1:l]}|x^{(i)}_{[1:k]};\theta^{(k)}_1).$$

With the current sample $f_{[1:l]}^{(k)}$ and the new sample $\tilde{f}_{[1:l]}^{(k)}$, the acceptance ratio takes the following form

$$A_i(\tilde{f}_{1:l}^{(k)}|f_{[1:l]}^{(k)}) = \min\left(1, \frac{p_i(\tilde{f}_{[1:l]}^{(k)};\theta^{(k)})}{p_i(f_{[1:l]}^{(k)};\theta^{(k)})}\frac{g_i(f_{[1:l]}^{(k)}|\tilde{f}_{[1:l]}^{(k)})}{g_i(\tilde{f}_{[1:l]}^{(k)}|f_{[1:l]}^{(k)})}\right)$$
$$= \min\left(1, \frac{p(x_{k+1}^{(i)}|\tilde{f}_{[1:l]}^{(k)};\theta_2^{(k)})}{p(x_{k+1}^{(i)}|f_{[1:l]}^{(k)};\theta_2^{(k)})}\right).$$

We can interpret this procedure as generating samples given the previous k symbols, and then re–weight based on the likelihood of the symbol $k+1$.

With this setup, we iterate the above Markov chain for t steps, and repeat this procedure for r rounds. Finally, we take the average of these independent r outcomes, where each of them is the result of a Markov chain after t iterations, to approximate each term in the gradients of (3) and (4). Figure 1 illustrates the performance of this approach for one bit in the `MNIST` dataset and for different values of the parameters r and t. As we can see, by increasing the values of r and t, the approximate gradient converges to the actual gradient.

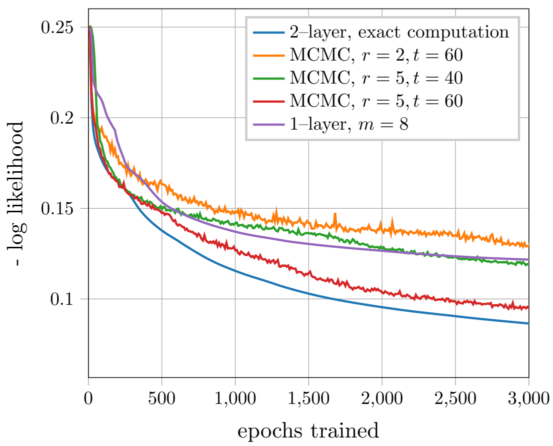

Figure 1: The performance of the MCMC algorithm for the two-layer switch network for bit index 629 of the `MNIST` data. The size of the switch network in this example is $(m_1, l, m_2) = (8, 8, 4)$. We run the Markov chain t steps for r independent runs and take the average to approximate the gradients. By increasing the values of r and t, the likelihood performance approaches that of the exact gradient computation. The performance of the single-layer switch network with $m = 8$ is illustrated for reference.