CS 6190: Probabilistic Modelling Spring 2019

Homework 5

Abhinav Kumar (u1209853)

Practice [100 points + 50 bonus]

1. [20 points] You will implement MCMC algorithms to sample from the distribution

$$p(z) \propto \exp(-z^2)\sigma(10z+3).$$

To reach the burn-in stage, please run your chain for $100K$ iterations(i.e., generate $100K$ samples). Then continue to run $50K$ iterations, pick every 10-th sample to obtain your final samples. Set your initial sample to 0.

(a) [9 points] Implement Metroplis-Hasting, with Gaussian proposal, $q(z_{n+1}|z_n) = \mathcal{N}(z_{n+1}|z_n, \tau)$. Vary τ from $\{0.01, 0.1, 0.2, 0.5, 1\}$. Run your chain. For each setting of τ, record the acceptance rate (i.e., how many candidate samples are accepted/the total number of candidate samples generated). Draw a figure, where the x-axis represents the setting of τ, and y-axis the acceptance rate. What do you observe? For each setting of τ, draw a figure, show a normalized histogram of the $5K$ samples you collected. Also, please draw the ground-truth density curve (obtained via quadrature)

MCMC algorithms are needed in the case when the form of likelihood and prior is known while the model evidence can not be calculated because of the integration and therefore the normalization being intractable. The overall idea is to sample from these distributions even when there is no knowledge of the normalization. Note that the prior and likelihood form is known and therefore there is no need of sampled data.

In this question, we can use ideas of Riemannian integral to get the normalization constant as well. However, this method can not be scaled to higher dimensions. Metropolis Hastings requires the knowledge of joint distribution which has been produced. The plots are shown in figure 1.

We observe that the acceptance rate decreases as we increase τ since increasing τ results in more bigger steps. As a result the newer sampled points decrease the likelihood causing it to be rejected more number of times.

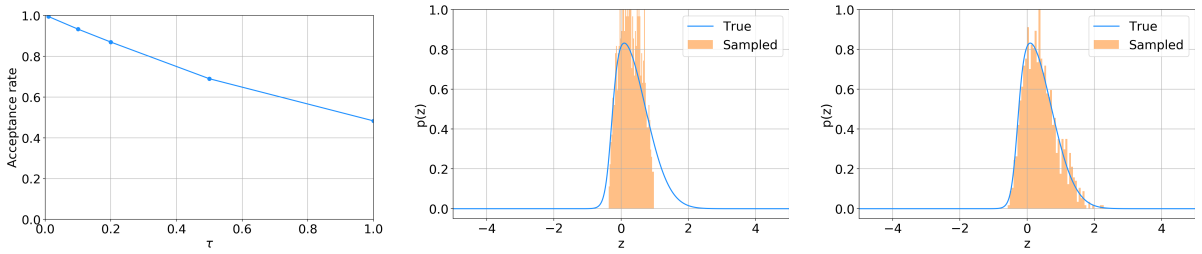
(b) [9 points] Implement Hybrid Monte-Carlo sampling with Leapfrog. Let us fix $L = 10$, and vary ϵ from $\{0.005, 0.01, 0.1, 0.2, 0.5\}$. Run your chain. Similar to the above, for each setting of ϵ, record the acceptance rate,and draw a figure showing ϵ *v.s.* acceptance rate. What do you observe? For each setting of ϵ, draw the normalized hist-gram (50 bins)of collected $5K$ samples. What do you observe?

We have

$$p(z) = \frac{1}{Z}\exp(-z^2)\sigma(10z+3) = \frac{1}{Z}p'(z)$$

The log joint is given by

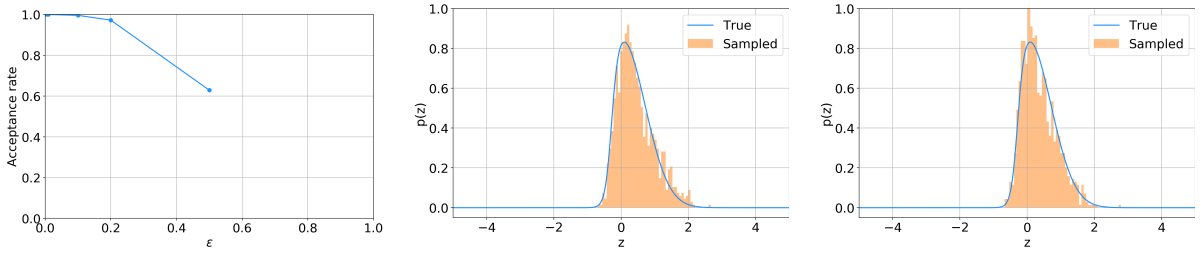
$$\ln \mathcal{L} \propto -z^2 + \ln(\sigma(10z+3))$$



(a) Acceptance rate (b) $\tau = 0.01$ (c) $\tau = 0.1$

(d) $\tau = 0.2$ (e) $\tau = 0.5$ (f) $\tau = 1$

Figure 1: Acceptance rate vs τ and sampled distribution for different values of τ for Metropolis Hastings.



(a) Acceptance rate (b) $\epsilon = 0.005$ (c) $\epsilon = 0.01$

(d) $\epsilon = 0.1$ (e) $\epsilon = 0.2$ (f) $\epsilon = 0.5$

Figure 2: Acceptance rate vs ϵ and sampled distribution for different values of ϵ for HMC.

The gradient of the negative log joint is

$$\nabla_z(-\ln \mathcal{L}) \propto 2z - 10(1 - \sigma(10z + 3))$$

HMC uses $\mathcal{L} = e^{\ln \mathcal{L}} = e^{-U}$ where U is the potential energy and so $U = -\ln \mathcal{L}$. Moreover, HMC keeps kinetic energy K with scaled identity mass or $\mathbf{M} = \alpha^{-1}\mathbf{I}$. Hence, we have

$$U(z) = -\log(p'(z)) = z^2 - \log(\sigma(10z + 3)) \tag{1}$$

$$K(p) = \frac{1}{2} p^\top M^{-1} p = \alpha \frac{1}{2} p^2 \tag{2}$$
$$H(z, p) = U(z) + K(p) \tag{3}$$
$$p(z, p) = \exp(-H(z, p)) \tag{4}$$
$$\nabla_z U(z) = \nabla_z(-\ln \mathcal{L}) = 2z - 10(1 - \sigma(10z + 3)) \tag{5}$$
$$\nabla_p K(p) = \alpha p \tag{6}$$

We keep $\alpha = 1$ in the experiments. The plots are shown in figure 2. We observe that the acceptance rate decreases as we increase ϵ since increasing ϵ results in more bigger steps. As a result the newer sampled points decrease the likelihood causing it to be rejected more number of times. We also observe that with $L = 10$, the HMC algorithm is robust to changes in the step size ϵ as it is able to sample from the entire distribution.

(c) [2 points] Now compare the results from the two MCMC algorithms, what do you observe and conclude?

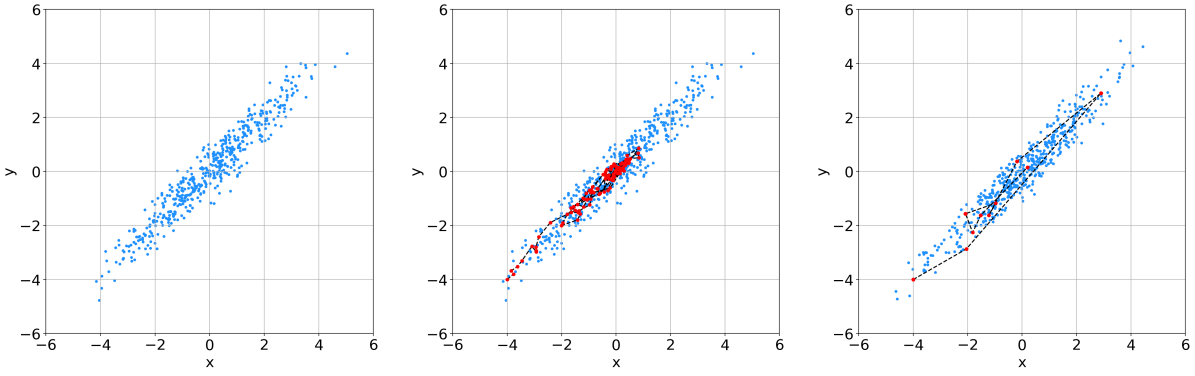
We can see that Metropolis-Hastings does not explore the distribution properly at $\tau = 0.01$ while the HMC gives good results at even $\epsilon = 0.005$. This is because of the fact that HMC uses multi steps and gradient information to get to the new point and also does not reject samples as frequently as Metropolis-Hastings.

2. [10 points] Let us work with a 2-dimensional Gaussian distribution,

$$p(z_1, z_2) = \mathcal{N}(\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} | \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 & 2.9 \\ 2.9 & 3 \end{bmatrix})$$

(a) [1 point] Draw 500 samples from this distribution and show the scatter plot. What do you observe?

I see an elliptical skewed kind of scatter in Figure 3(a) which is because of highly correlated axes.



(a) Sampled (b) Gibbs Sampler (c) HMC Sampler

Figure 3: Sampled Distribution and trajectories of using Gibbs as well as HMC sampler.

(b) [4 points] Implement Gibbs sampling to alternatively sample z_1 and z_2. Set your initial sample to $(-4, -4)$. Run your Gibbs sampler for 100 iterations. Draw the trajectory of the samples. What do you observe?

The complete data likelihood is

$$p(\mathbf{z}|D) \propto p(D|\mathbf{z})p(\mathbf{z})$$

$$\propto \prod e^{-0.5(\mathbf{x}_i-\mathbf{z})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\mathbf{z})}$$
$$\propto e^{-0.5\sum_i(\mathbf{x}_i-\mathbf{z})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\mathbf{z})}$$

The sampling equations for z_1 are

$$z_1 \sim \mathcal{N}(\mu_1, \sigma_1) \quad \text{where}$$
$$\mu_1 = \frac{\sum \mathbf{x}_{i1}}{N} + \sigma_{12}\sigma_{22}^{-1}\left(z_2 - \frac{\sum \mathbf{x}_{i2}}{N}\right) \tag{7}$$
$$\sigma_1 = \sigma_{11} - \sigma_{12}\sigma_{22}^{-1}\sigma_{21} \tag{8}$$

The sampling equations for z_2 are

$$z_2 \sim \mathcal{N}(\mu_2, \sigma_2) \quad \text{where}$$
$$\mu_2 = \frac{\sum \mathbf{x}_{i2}}{N} + \sigma_{12}\sigma_{11}^{-1}\left(z_1 - \frac{\sum \mathbf{x}_{i1}}{N}\right) \tag{9}$$
$$\sigma_2 = \sigma_{22} - \sigma_{12}\sigma_{11}^{-1}\sigma_{21} \tag{10}$$

Here, σ_{ij} represents the ij^{th} entry of the sample covariance matrix.
The result of using this sampling scheme is shown in Figure 3(b).
References: https://kieranrcampbell.github.io/blog/2016/05/15/gibbs-sampling-bayesian-linear-regression.html

(c) [5 points] Implement HMC with Leapfrog, set $\epsilon = 0.1$ and $L = 20$. Run your HMC for 100 iterations. Set your initial sample to $(-4, -4)$. Draw the trajectory of the samples. What do you observe? Compare with the results of Gibbs sampling, what do you conclude?

Note that we do not use sampled mean and covariance but instead use the true mean and true covariance. The log joint is given by

$$\ln \mathcal{L} \propto -\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu})$$

The gradient of the negative log joint is

$$\nabla_z(-\ln \mathcal{L}) \propto \boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu})$$

HMC uses $\mathcal{L} = e^{\ln \mathcal{L}} = e^{-U}$ where U is the potential energy and so $U = -\ln \mathcal{L}$. Moreover, HMC keeps kinetic energy K with scaled identity mass or $\mathbf{M} = \alpha^{-1}\mathbf{I}$. Hence, we have

$$U(\mathbf{z}) \propto \frac{1}{2}(\mathbf{z}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu}) \tag{11}$$
$$K(\mathbf{p}) = \frac{1}{2}\mathbf{p}^\top\mathbf{M}^{-1}\mathbf{p} = \alpha\frac{1}{2}(p_1^2 + p_2^2) \tag{12}$$
$$H(\mathbf{z}, \mathbf{p}) = U(\mathbf{z}) + K(\mathbf{p}) \tag{13}$$
$$p(\mathbf{z}, \mathbf{p}) = \exp(-H(\mathbf{z}, \mathbf{p})) \tag{14}$$
$$\nabla_{\mathbf{w}}U(\mathbf{w}) = \nabla_{\mathbf{w}}(-\ln \mathcal{L}) = \boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu}) \tag{15}$$
$$\nabla_{\mathbf{p}}K(\mathbf{p}) = \alpha\mathbf{p} \tag{16}$$

We keep $\alpha = 1$ in the experiments. The result of using this sampling scheme is shown in Figure 3(c). We see that HMC is sometimes able to explore low density regions as well while Gibbs Sampling in general takes many iterations to explore the low density region. We thus conclude that HMC is a better algorithm than Gibbs sampling.

3. [70 points] Let us work on a real-world dataset we have met before. Please download the data from the folder"data/bank-note".

(a) [20 points] We assign the feature weight vector $\mathbf{w}$ a standard normal prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Write down the joint probability of the Bayesian logistic regression model. Now, implement HMC with Leapfrog. Set your initial sample to be $\mathbf{0}$. Run your chain for $100K$ iterations to reach the burn-in state; then continue to run $10K$ iterations, pick every 10-th sample to obtain your posterior samples. Vary ϵ from $\{0.005, 0.01, 0.02, 0.05\}$ and L from $\{10, 20, 50\}$. As we did before, we can test the predictive accuracy and predictive log-likelihood. List a table showing different combinations of ϵ and L and the resulting predictive accuracy, predictive log-likelihood, and acceptance rate. What do you observe and conclude?

The joint probability is given by

$$\begin{aligned} p(\mathbf{w}, \mathbf{x}, t) &= p(\mathbf{x}, t|\mathbf{w})p(\mathbf{w}) \\ &= \prod_{i=1}^{N} \sigma(\mathbf{w}^T\mathbf{x}_i)^{t_i}(1 - \sigma(\mathbf{w}^T\mathbf{x}_i))^{1-t_i} p(\mathbf{w}) \end{aligned}$$

The log joint is given by

$$\begin{aligned} \ln \mathcal{L} &= \sum_{i=1}^{N} \left[t_i \ln \sigma(\mathbf{w}^T\mathbf{x}_i) + (1 - t_i) \ln(1 - \sigma(\mathbf{w}^T\mathbf{x}_i)) \right] + \ln p(\mathbf{w}) \\ &= \sum_{i=1}^{N} \left[t_i \ln \sigma(\mathbf{w}^T\mathbf{x}_i) + (1 - t_i) \ln(1 - \sigma(\mathbf{w}^T\mathbf{x}_i)) \right] - \frac{1}{2}\mathbf{w}^T\mathbf{w} \end{aligned}$$

The gradient of the negative log joint is

$$\begin{aligned} \nabla_{\mathbf{w}}(-\ln \mathcal{L}) &= \sum_{n=1}^{N} (y_n - t_n)\boldsymbol{\phi}_n + \mathbf{w} \\ \implies \nabla_{\mathbf{w}}(-\ln \mathcal{L}) &= \boldsymbol{\Phi}^T(\mathbf{y} - \mathbf{t}) + \mathbf{w} \end{aligned}$$

where $\boldsymbol{\Phi}$ is a $n \times d$ matrix and each $\boldsymbol{\phi}_n$ is a row vector.
HMC uses $\mathcal{L} = e^{\ln \mathcal{L}} = e^{-U}$ where U is the potential energy and so $U = -\ln \mathcal{L}$. Moreover, HMC keeps kinetic energy K with scaled identity mass or $\mathbf{M} = \alpha^{-1}\mathbf{I}$. Hence, we have

$$U(\mathbf{w}) = -\sum_{i=1}^{N} \left[t_i \ln \sigma(\mathbf{w}^T\mathbf{x}_i) + (1 - t_i) \ln(1 - \sigma(\mathbf{w}^T\mathbf{x}_i)) \right] + \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{17}$$

$$K(\mathbf{p}) = \frac{1}{2}\mathbf{p}^\top\mathbf{M}^{-1}\mathbf{p} = \alpha\frac{1}{2}\sum_i p_i^2 \tag{18}$$

$$H(\mathbf{w}, \mathbf{p}) = U(\mathbf{w}) + K(\mathbf{p}) \tag{19}$$

$$p(\mathbf{w}, \mathbf{p}) = \exp(-H(\mathbf{w}, \mathbf{p})) \tag{20}$$

$$\nabla_{\mathbf{w}}U(\mathbf{w}) = \nabla_{\mathbf{w}}(-\ln \mathcal{L}) = \boldsymbol{\Phi}^T(\mathbf{y} - \mathbf{t}) + \mathbf{w} \tag{21}$$

$$\nabla_{\mathbf{p}}K(\mathbf{p}) = \alpha\mathbf{p} \tag{22}$$

Table 1 lists out the predictive accuracy, predictive log-likelihood, and acceptance rate for different combinations of ϵ and L.

Step Size ϵ	**#Steps** L	**Predictive Accuracy** (%)	**Predictive Log Likelihood**	**Acceptance Rate**
0.005	10	99.00	-0.04	0.0023
0.005	20	99.00	-0.04	0.0560
0.005	50	99.00	-0.03	0.0604
0.01	10	99.00	-0.03	0.0471
0.01	20	98.96	-0.03	0.1355
0.01	50	98.97	-0.03	0.0757
0.02	10	98.94	-0.03	0.1196
0.02	20	98.96	-0.03	0.0727
0.02	50	98.96	-0.03	0.0956
0.05	10	44.20	-0.69	0.0000
0.05	20	44.20	-0.69	0.0000
0.05	50	44.20	-0.69	0.0000

Table 1: Predictive accuracy, predictive log-likelihood, and acceptance rate for different combinations of ϵ and L.

We observe that proper choice of ϵ is needed for algorithm to get correct posterior samples. Moreover, there is a wide range of $\epsilon = [0.005, 0.02]$ for which the convergence happens. At $\epsilon = 0.05$, we see that the new states have less energy and therefore the state does not get updated. The acceptance rate seems to be a function of ϵ and L.

(b) [20 points]. We will implement Gibbs sampling for linear classification. However, Bayesian logistic regression does not allow tractable conditional posteriors. So we will use the Bayesian probit model with augmented variables. Again, we will assign a standard normal prior over $\mathbf{w}$. For each training sample n, we introduce an auxiliary variable z_n. The joint probability is given by

$$p(\mathbf{w}, \mathbf{z}, \mathcal{D}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{I}) \prod_n \mathcal{N}(z_n|\mathbf{w}^\top \mathbf{x}_n, 1) \mathbb{1}\big((2y_n - 1)z_n \geq 0\big) \tag{23}$$

where $\mathbf{z} = [z_1, z_2, \ldots, z_N]^\top$, and $\mathbb{1}(\cdot)$ is the indicator function. Note that if we marginalize out $\mathbf{z}$, we will recover the original Probit model, $\phi(x) = \int_\infty^x \mathcal{N}(t|0, 1)\mathrm{d}t$. Now implement your Gibbs sampling algorithm based on the augmented version (23). Alternatively sample $\mathbf{w}$ and each z_n. Note that the conditional posterior of each z_n will be a truncated Gaussian.

Before coding, please list your derivation of the conditional posteriors. Run your chain for $100K$ iterations to reach the burn-in stage; then continue to run $10K$ iterations, pick every 10-th sample to obtain your final posterior samples. Now compute and report the predictive accuracy and log-likelihood with your posterior samples. Note that your predictive log-likelihood should be based on the original model, rather than the augmented one. How does the performance compare with HMC on Bayesian logistic regression model?

Gibbs sampling is frequently used in the following two (related) settings:

- When it is difficult or impossible to sample $x = (x1, ..., xp)$ directly, but is possible to conditionally sample $x_i|x_1, ..., x_{i-1}, x_{i+1}, ..., x_p \; \forall \; i = 1, ..., p$.
- When it is difficult or impossible to sample x directly, but there exists a "latent variable" z such that it is possible to conditionally sample $x|z$ and $z|x$.

There is no clear way to sample $\mathbf{w}$ in the original Probit regression model as shown in Figure 4(a). Hence, we add a latent model z_i and thus the joint is given by

$$p(\mathbf{w}, \mathbf{z}, \mathcal{D}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{I}) \prod_n \mathcal{N}(z_n|\mathbf{w}^\top \mathbf{x}_n, 1) \mathbb{1}_{(2y_n-1)z_n \geq 0}$$

Now, we write the conditionals first

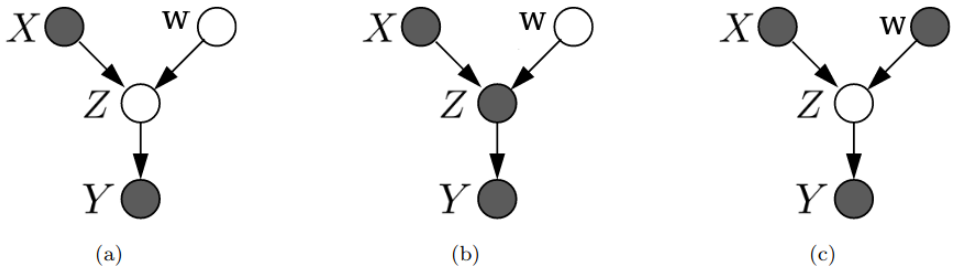


Figure 4: Probit regression with augmented variable (a) with the observed data shaded (b) 'observed' data associated with the Gibbs conditional $\mathbf{w}|X, Y, Z$ shaded (c) 'observed' data associated with the Gibbs conditional $z|\mathbf{w}, X, Y$ shaded.

$$\begin{aligned} p(\mathbf{w}|z_i, \mathbf{x}_i, y_i) &\propto p(\mathbf{w}) \prod_{i=1}^{N} \mathcal{N}(z_i|\mathbf{w}^t \mathbf{x}_i) \\ &= e^{-\frac{1}{2}\left[\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} (\mathbf{x}_i^T\mathbf{w} - z_i)^T(\mathbf{x}_i^T\mathbf{w} - z_i)\right]} \\ &= e^{-\frac{1}{2}\left[\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} \mathbf{w}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{w} - 2\sum_{i=1}^{N} z_i\mathbf{x}_i^T\mathbf{w} + C\right]} \end{aligned}$$

We again use the trick of completing the squares and therefore this should be equal to

$$e^{-\frac{1}{2}\left[\mathbf{w}^T\boldsymbol{\Sigma}^{-1}\mathbf{w} - 2\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{w} + C\right]}$$

On comparing the two,

$$\begin{aligned} \boldsymbol{\Sigma}^{-1} &= \mathbf{I} + \sum_{i=1}^{N} \mathbf{x}_i\mathbf{x}_i^T \\ \implies \boldsymbol{\Sigma} &= \left[\mathbf{I} + \sum_{i=1}^{N} \mathbf{x}_i\mathbf{x}_i^T\right]^{-1} \end{aligned} \tag{24}$$

and

$$\begin{aligned} \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1} &= \sum_{i=1}^{N} z_i\mathbf{x}_i^T \\ \implies \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} &= \sum_{i=1}^{N} z_i\mathbf{x}_i \\ \implies \boldsymbol{\mu} &= \boldsymbol{\Sigma}\left(\sum_{i=1}^{N} z_i\mathbf{x}_i\right) \end{aligned}$$

$$\implies \boldsymbol{\mu} = \left[\mathbf{I} + \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T\right]^{-1} \left(\sum_{i=1}^{N} z_i \mathbf{x}_i\right) \tag{25}$$

Hence,

$$\boxed{p(\mathbf{w}|z_i, \mathbf{x}_i, y_i) \sim \mathcal{N}\left(\left[\mathbf{I} + \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T\right]^{-1} \left(\sum_{i=1}^{N} z_i \mathbf{x}_i\right), \left[\mathbf{I} + \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T\right]^{-1}\right)} \tag{26}$$

On the other hand, $z_i|\mathbf{w}, \mathbf{x}_i, y_i$ would be normal, except we are also conditioning on y_i, which indicates the sign of z_i (see Figure 4(c)). Thus, $z_i|\mathbf{w}, \mathbf{x}_i, y_i$ is a truncated normal:

$$\boxed{p(z_i|\mathbf{w}, \mathbf{x}_i, y_i) = \begin{cases} \mathcal{TN}(\mathbf{w}^T\mathbf{x}_i, 1), & \text{for } y_i = 1, z_i \in [0, \infty) \\ \mathcal{TN}(\mathbf{w}^T\mathbf{x}_i, 1), & \text{for } y_i = 0, z_i \in (-\infty, 0] \end{cases}} \tag{27}$$

The predictive accuracy is 98.89% while the predictive log likelihood is -0.04.

The performance of Gibbs sampler with Variable Augmentation is slightly bad than properly chosen HMC.

References: https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/lectures/lecture19.pdf

(c) [28 points] Finally, we implement a Bayesian neural network with two intermediate layers, and same number of nodes in each layer. Vary the number of nodes from $[10, 20, 50]$. Vary the activation function from {`tanh`, `RELU`}. We will use a factorized Gaussian posterior for the NN weights (check out the slides). We will assign a standard normal prior over each weight. The output of the NN will be thrown into a Sigmoid activation function, with which we obtain a Bernoulli likelihood. Please use Adam algorithm for stochastic optimization. You can tune the base learning rate from {1e-3, 0.5e-3, 1e-4, 1e-5} to find the best result for each layer-width and activation function setting. Initialize posterior mean and variance of each weight to be 0 and 1, respectively. Run the Adam algorithm for 1,000 iterations. For each layer width and activation function setting, calculate the predictive log-likelihood and accuracy. Use the variational posterior to generate 100 samples for the weight vector; With each sample, you can calculate the prediction accuracy and log-likelihood on the test dataset; finally, you report the average results. To check the behaviour of your BNN, let us pick up one setting: the number of node is 20 and the activation function is `tanh`. For each iteration, please use the current posterior mean of the weights to compute the average log-likelihood on the training set and test set respectively. Draw two plots, one plot showing how the training log-likelihood varies along with the number of iterations, the other showing how the test log-likelihood varies along with the number of iterations. What do you observe and conclude?

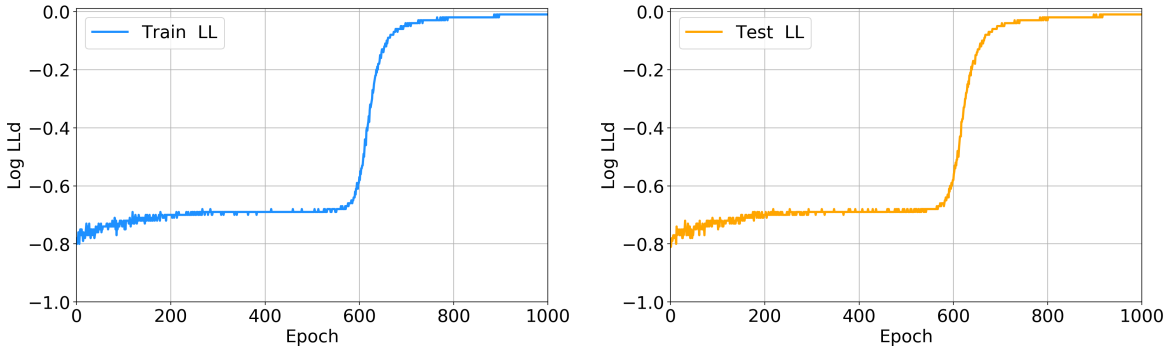
Table 2 lists out the predictive accuracy and predictive log-likelihood for different combinations of activations and number of nodes in each hidden layer for Bayesian MLP. Figure 5 shows the plot of log likelihood with the number of epochs.

Observations:

- The results depend on learning rate. If I used a very small learning rate, the convergence does not happen.

Activation Function	#Nodes per layer	Best LR	Predictive Accuracy (%)	Predictive Log Likelihood
ReLU	10	0.001	96.48	-0.21
	20	0.001	99.25	-0.10
	50	0.001	99.59	-0.08
tanh	10	0.001	69.48	-0.61
	20	0.001	99.86	-0.01
	50	0.001	99.94	-0.00

Table 2: Predictive accuracy and predictive log-likelihood for different combinations of activations and number of nodes in each hidden layer for Bayesian MLP after 1000 epochs.



(a) Train (b) Test

Figure 5: Plot of average of log likelihood vs epoch for training vs testing on 20 nodes and `tanh` activation

- In this problem, `tanh` seems to have better convergence than `ReLU` for width =20 and 50.
- Increasing the number of hidden units seem to fit the data better.
- As shown in Figure 5, the convergence is very slow initially. The log likelihood does not seem to improve till a certain number of epochs is reached and after that the convergence happens very quickly.

(d) [2 points] Compare the results from Bayesian linear classification and NN models. What do you observe and conclude?

The Bayesian neural network models fit the data better than their linear counterpart which is justified since the Linear models have a non-linearity only in the link function while the Neural Networks introduce non-linearity at each of the hidden layers.