Comp 596 Assignment 1

Jonathan Pearce 260672004

February 21, 2020

Section 1

Problem 1.1

1. The Boltzmann machine has n units or neurons: {1, ..., n}. m visible units {1, ..., m} and k hidden units {1, ..., k}, where $n = m + k$.

2. The state of activation of unit i at time step t in the Boltzmann machine is as follows,

$$a_i(t) \in [0, 1] \ \forall i, t$$

3. The output function for the Boltzmann machine is a stochastic function defined below.

$$x \sim \mathbb{U}[0, 1]$$

$$o_i(t) = f(a_i(t), x) = \begin{cases} 1, & \text{if } a_i(t) \geq x \\ 0, & \text{if } a_i(t) < x \end{cases}$$

   Where $o_i(t)$ is the output for unit i at time step t and $\mathbb{U}[0, 1]$ is a uniform distribution from 0 to 1.

4. The units in a Boltzmann machine are connected according to the following rules:

   - Hidden units connect to all visible units
   - Hidden units connect to each other
   - There are no connections between visible units.

   A connection between unit i and unit j has weight W_{ij}. These weights follow one rule:

   - All weights are symmetric (i.e. $W_{ij} = W_{ji}$).

   Putting all this information together we can create a general form for the weight matrix $\mathbf{W}$ which represents the pattern of connectivity between all units. $\mathbf{W}$ is a symmetric matrix where the first k rows represent the connection weights of the k hidden units, and the next m rows represent the connection weights of the m visible units.

$$\mathbf{W} = \begin{bmatrix} 0 & W_{12} & \cdots & W_{1k} & W_{1(k+1)} & \cdots & W_{1(k+m)} \\ W_{21} & 0 & \cdots & W_{2k} & W_{2(k+1)} & \cdots & W_{2(k+m)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ W_{k1} & W_{k2} & \cdots & 0 & W_{k(k+1)} & \cdots & W_{k(k+m)} \\ W_{(k+1)1} & W_{(k+1)2} & \cdots & W_{(k+1)k} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ W_{(k+m)1} & W_{(k+m)2} & \cdots & W_{(k+m)k} & 0 & \cdots & 0 \end{bmatrix}$$

5. The propagation rule for sending activity between units in a Boltzmann machine is as follows,

$$\text{net}_i(t) = \sum_j W_{ij} o_j(t-1) + \theta_i$$

Where $\text{net}_i(t)$ is the total input that unit i receives at time step t and θ_i is the threshold for unit i.

6. The activation rule for combining the propagated activity with current activity in a Boltzmann machine is as follows,

$$a_i(t) = F(net_i(t), T) = \frac{1}{1 + exp\frac{-\text{net}_i(t)}{T}}$$

Where $a_i(t)$ is the activation of unit i at time step t and T is the temperature parameter.

7. The learning rules that stipulate how the connection weights and thresholds change based on current weights and activity information for the Boltzmann machine are as follows,

$$\Delta W_{ij} = \frac{\eta}{T}(< o_i o_j >^+ - < o_i o_j >^-)$$

$$\Delta \theta_i = -\frac{\eta}{T}(< o_i >^+ - < o_i >^-)$$

8. The environment that provides inputs to some subsets of the units, are the MNIST images, whose pixel values are clamped to the visible units of the Boltzmann machine during the wake phase.

I spoke with Prof. Richards in office hours and discussed the distinction between the output and activation functions of the Boltzmann machine with respect to the PDP framework. In the PDP framework the binary variable is the unit's output, however for ease of analysis and to remain consistent with the course slides the remainder of this report considers the state of activation of units the binary variable.

Problem 1.2 The loss function that we want to reduce is the Kullback-Leibler divergence between $p^+(\mathbf{v}^k)$ and $p^-(\mathbf{v}^k)$. It measures the number of bits required to encode data generated from $p^+(\mathbf{v}^k)$ using $p^-(\mathbf{v}^k)$. It is zero if and only if $p^+(\mathbf{v}^k) = p^-(\mathbf{v}^k)$. The KL divergence formula is as follows,

$$\mathcal{L}_{KL} = \sum_k p^+(\mathbf{v}^k) ln \frac{p^+(\mathbf{v}^k)}{p^-(\mathbf{v}^k)}$$

Where $\mathbf{v^k}$ represents the setting for the visible units when stimulus k is presented, $p^+(\mathbf{v}^k)$ represents the probability of $\mathbf{v^k}$ according to the world, and $p^-(\mathbf{v}^k)$ represent the probability of $\mathbf{v^k}$ according to the Boltzmann machine when it's free running (i.e. "dreaming").

Problem 1.3 The partial derivative of the loss function with respect to a synaptic weight is as follows,

$$\frac{\partial \mathcal{L}_{Kl}}{\partial W_{ij}} = \frac{1}{T}(< a_i a_j >^+ - < a_i a_j >^-)$$

Problem 1.4 $< a_i a_j >^+$ indicates the expected value of the co-activation of unit i and j during the "wake" phase and $< a_i a_j >^-$ indicates the expected value of the co-activation of unit i and j during the "dreaming" phase. In order to calculate $< a_i a_j >^+$ and $< a_i a_j >^-$ we need to be able to sample from the probability distributions $p^+(a)$ and $p^-(a)$, we do this via Gibbs sampling. Gibbs sampling is a procedure where we select a neuron at random and update its activation and output according to the activation rule and output function (outlined in 1.1). We continue this process until the system is at equilibrium, at which point we can begin sampling from the Boltzmann machine distributions in order to estimate $< a_i a_j >^+$ and $< a_i a_j >^-$. I discuss the implementation details of how this is done in Q2.2. T is the temperature of the system at equilibrium, while we are sampling from the Boltzmann machine distributions.

Problem 1.5 The temperature in a Boltzmann machine is a parameter used in the activation (and learning) rule. In the activation rule the temperature T effects how random the new activation values are.

$$\lim_{T\to\infty} \mathbb{P}(a_i(t) = 1|\mathbf{a}(t-1)) = \lim_{T\to\infty} \frac{1}{1 + exp\frac{-\text{net}_i(t)}{T}} = \frac{1}{2}$$

$$\lim_{T\to 0} \mathbb{P}(a_i(t) = 1|\mathbf{a}(t-1)) = \lim_{T\to 0} \frac{1}{1 + exp\frac{-\text{net}_i(t)}{T}} = 1$$

A very high temperature T makes the probability of activation approximately 50% regardless of the current activations, weights and thresholds, In other words a high temperature makes the activations completely random. Where as a very low temperature T makes the probability of activation close to 100% regardless of the current activations, weights and thresholds, therefore a very low temperature ensures the unit will become activated. Therefore the temperature controls the entropy of the system (infinite temperature is max entropy, zero temperature is no entropy) and how predictable the activation's of units are.

Problem 1.6 An annealing schedule of a Boltzmann machine controls the temperature parameter T during Gibbs sampling. More specifically, the annealing schedule specifies how many iterations of Gibbs sampling should be performed and what the temperature is at each iteration of Gibbs sampling. Usually the schedule is designed to start at a high temperature and slowly lower the temperature over time. When used with an appropriate schedule and well tuned temperatures, simulated annealing can help improve the system's chances of avoiding local minima and reaching the equilibrium state. This is important because when we begin computing our estimates of $< a_i a_j >^+$ and $< a_i a_j >^-$ we want to be sampling from the equilibrium distributions.

Problem 1.7 The goal of training the Boltzmann machine is to minimize the loss function, specifically the KL divergence between $p^+(v)$ and $p^-(v)$. The equation from Q1.3 is the partial derivative of the loss function with respect to a synaptic weight. This equation allows us to perform gradient descent in the configuration space of the weight matrix $\mathbf{W}$, in order to try and find the optimal weight matrix. The equation from Q1.3 encourages our Boltzmann machine to make the patterns seen in the wake phase more likely while making the patterns observed in the dream phase less likely. This is strongly connected to a topic in the readings called reverse learning (or unlearning). Reverse learning is a microbiological theory related to the idea that when people dream the brain parses through recently gathered information and discards unwanted material. In other words, people dream in order to forget, and the idea of reverse learning enables this process. The equation in Q1.3 can roughly be considered a numerical representation of the reverse learning (unlearning) process. As expressed in the course slides the equation from Q1.3 encourages the Boltzmann machine to remember its reality and forget its dreams.

Section 2

Problem 2.1 The first set of variables I included in the Boltzmann machine class were the model parameters; number of hidden units, number of visible units, number of units, learning rate and annealing schedule information. I defined these variables in my class because I thought that the model parameters would frequently be used in multiple methods and therefore wanted to avoid having to constantly pass them into functions as a parameters. This was very useful for parameters such as number of hidden units and number of units which are both used fairly frequently in for-loops and matrix vector calculations across many different methods. Some of these model parameters, such as learning rate are only used in one method in the Boltzmann machine. However, defining learning rate in the same manner and location as the other model parameters in the Boltzmann machine class makes it very easy to edit, adjust and keep track of parameter values. The next set of variables I included in the Boltzmann machine class were the variables that represented the Boltzmann machine data; weight matrix, threshold values and current activation vector. These three data items are used across a lot of the Boltzmann machine functions and therefore defining them in the class and avoiding passing them through to each function made the code slightly easier to read and follow. Finally the last variable I defined in the class was the vector that represented $< a >^-$, the expected activation of all units during the dream phase. I defined this in the class because it is the only estimated quantity that is required for updating weights (in this case threshold values) and computing the loss function for images.

Problem 2.2 The weight update equations for the entire weight matrix W and threshold vector θ are formulated as follows,

$$\Delta W = \frac{\eta}{T}(< aa >^+ - < aa >^-)$$

$$\Delta \theta = -\frac{\eta}{T}(< a >^+ - < a >^-)$$

η and T are parameters, whereas the co-activation expected values $< aa >^+$ and $< aa >^-$ as well as the activation expected values $< a >^-$ and $< a >^+$ are quantities that need to be estimated, all four of these expectations are estimated in Gibbs sampling. Whenever Gibbs sampling is run the activations are randomly reset on all non-clamped units. Gibbs sampling begins with the burn-in, which is the section of the annealing schedule where we are working towards the equilibrium state. During the burn-in we are not collecting activation and co-activation statistics. After the burn-in, when the Boltzmann machine has reached equilibrium we begin collecting activation and co-activation statistics, this procedure is the same regardless of whether we are in the wake or dream phase and we can therefore simplify our notation. $< aa >$ and $< a >$ are initialized as a zero matrix and zero vector respectively. In every iteration of Gibbs sampling after the burn-in, we update a unit's activation as usual, then add the outer product of the activation vector $a(t)$ to $< aa >$, and add the activation vector $a(t)$ to $< a >$. Explicitly each iteration after the burn-in we do the following add operations,

$$< aa > += a(t)a(t)^T$$

$$< a > += a(t)$$

At the end of Gibbs sampling we divide every value in $< aa >$ and $< a >$ by the number of iterations we collected statistics, in order to scale the values into the range $[0, 1]$. Gibbs sampling returns the estimated expected value of co-activation $< aa >$ and activation $< a >$ that are used in the weight update. Running Gibbs sampling in the dream phase provides us with $< aa >^-$ and $< a >^-$ and in the wake phase Gibbs sampling produces $< aa >^+$ and $< a >^+$.

Problem 2.3 The annealing schedule will be defined by a series of iteration and temperature pairs. For my Boltzmann machine the annealing schedule is as follows; 2 iterations at temperature 10, 4 at 5, 8 at 2, 50 at 1. The last iteration of the annealing schedule is when we collect statistics for calculating the weight update. Here one iteration is defined as the number of times we have to run the activation process so that each unclamped unit is run through the activation process once on average. During dreaming all units can change their activation, whereas only the hidden units can have their activation changed in the wake phase. Therefore in order to ensure we are always reaching the equilibrium state in Gibbs sampling the number of times we run the activation process per iteration is proportional to the number of units that are eligible to

change during Gibbs sampling.

The training schedule loops through a specified number of training epochs. For each epoch, we first run the dream phase and compute $< aa >^-$ and $< a >^-$, these remain fixed and are not recomputed during the epoch. Next we pick an image from the training set randomly and compute $< aa >^+$ and $< a >^+$ with the selected image's pixel values clamped to the visible units. We then update the weight matrix and the threshold values according to the learning rules. Finally we compute the training loss on that image and add to the total training loss for the current epoch. We repeat this procedure to ensure that every image in the training set has been selected once on average. We finish by dividing the total training loss for this epoch by the number of images in the training set to obtain the average training loss per image for this epoch of training.

Section 3

Problem 3.1 Submitted code takes approximately 15-20 minutes to run in google colab research notebook.

Problem 3.2 Below are two examples of images that are generated on the visible units of my Boltzmann machine when it runs freely after training.

Figure 1: Boltzmann machine generated images

My Boltzmann machine has learned how to effectively summarize the MNIST digits, and the images it generates are examples of this summary: pixels around the border are much less likely to be activated and center region pixels are quite likely to be activated by the Boltzmann machine. However, my Boltzmann machine has failed to learn any of the high order weak constraints that distinguish the 10 different digits from each other and instead just learned an approximate average of these 10 digits.

A healthy brain that has been able to train on the MNIST digits, if asked to recall what a typical MNIST digit looks like may produce something reasonably close to the following two images. A healthy brain understands

Figure 2: Digits that a healthy brain could generate

the high order weak constraints that make the digit 5 and the digit 0 different and can clearly reproduce images that respect those constraints. A very unhealthy brain that is not capable at reasoning about the constraints involved in the MNIST dataset may produce something reasonably close to the following two images if asked to recall what a typical MNIST digit looks like.

Figure 3: Digits that an unhealthy brain could generate

Clearly these images are just noise and the unhealthy brain clearly understands nothing about the MNIST digits general structure or more exact constraints that distinguish digits from each other.

After training my Boltzmann machine is about halfway between these two types of brains with respect to health. My Boltzmann machine has learned that pixels closer to the center of the image should be much more likely to be activated as opposed to pixels near the border. However my Boltzmann machine has failed to understand how to arrange activated pixels in a manner that attempts to replicate the general shape of specific numerical digits.