# Assignment 1 - Boltzmann Machine

**COMP 596 - Brain-Inspired Artificial Intelligence**

**LE, Nhat Hung**
McGill ID: 260793376
Date: February 12, 2020
Due date: February 21, 2020

Prof. Blake Richards
Winter 2020

---

## Part 1 – Understanding Boltzmann machines (30 marks total)

**Question 1.1 (8 marks):** In writing, define a Boltzmann machine using the eight elements of the PDP framework described in week 3 of the class (see the lecture slides and the readings).

1. Processing units: *n* hidden and visible units/neurons

$$\mathbf{a} = [a_1, ..., a_n]^T$$

2. State of activations: each unit have a binary state 0 or 1

$$a_i = \{0, 1\}$$

3. Pattern of connectivity defined by weight matrices: symmetric weights, each hidden unit connected to all other units, each visible unit is connected to all hidden units

$$W_{ij} \in \mathbb{R}^{n \times n}$$
$$W_{ij} \text{ weight between units } i \text{ and } j$$

4. Set of propagation rules

$$net_i = \sum_j W_{ij} a_i + \theta_i$$
$$\Rightarrow \mathbf{net} = \mathbf{W}\mathbf{a} + \mathbf{\Theta}$$

5. Set of output functions

$$\mathbf{o}(t) = [o_1(t) = \sigma(net_1), ..., o_n(t) = \sigma(net_n)]^T$$
$$\sigma(net_i) = \frac{1}{1 + e^{-net_i}} \text{ (sigmoid function)}$$

6. Set of activation rules: the probability that unit *i* is activated (=1) is

$$p(a_i(t) = 1 | \mathbf{a}(t-1)) = \sigma\left(net_i / T\right)$$
$$\theta_i \text{ bias}, T \text{ temperature}$$

7. Learning rules

$$\Delta W_{ij} = \eta \frac{1}{T} [\langle a_i a_j \rangle^+ - \langle a_i a_j \rangle^-]$$
$$\langle a_i a_j \rangle^{+/-} \text{ expected co-activation counts of units } i, j \text{ when training data is clamped/not clamped to visible units, re}$$

$$\Delta \mathbf{\Theta} = \eta \frac{1}{T} [\langle A \rangle^+ - \langle A \rangle^-]$$
$$\langle A \rangle^{+/-} \text{ expected activation counts when training data is clamped/not clamped to visible units, respectively}$$

8. An environment: series of inputs e.g. images, unlabeled as the Boltzmann machine is an unsupervised learning model

**Question 1.2 (2 marks):** What is the loss function that a Boltzmann machine minimizes?

$$L = \sum_K p^+(\mathbf{v}^k) \ln \frac{p^+(\mathbf{v}^k)}{p^-(\mathbf{v}^k)}$$

**Question 1.3 (2 marks):** What is the partial derivative of the loss function with respect to a synaptic weight?

$$\frac{\partial L}{\partial W_{ij}} = -\sum_k \frac{p^+(\mathbf{v}^k)}{p^-(\mathbf{v}^k)} \frac{\partial p^-(\mathbf{v}^k)}{\partial W_{ij}}$$

$$= -\sum_k \frac{p^+(\mathbf{v}^k)}{p^-(\mathbf{v}^k)} \frac{1}{T} \left[ \sum_l p^-(\mathbf{v}^k \wedge \mathbf{h}^l) a_i^{kl} a_j^{kl} - p^-(\mathbf{v}^k) \sum_{r,s} p^-(\mathbf{v}^r \wedge \mathbf{h}^s) a_i^{rs} a_j^{rs} \right]$$

$$= \frac{1}{T} [\langle a_i a_j \rangle^+ - \langle a_i a_j \rangle^-]$$

**Question 1.4 (4 marks):** Explain in words what the various terms of the equation from Q1.3 represent. What do you have to do to calculate them?

The term

$$\langle a_i a_j \rangle^+$$

is the expected co-activation count of units i,j  during the wake phase i.e. when training data is clamped to the visible units.

$$\langle a_i a_j \rangle^-$$

is the same but during the dream phase, when training data isn't camped to the visible units.

Each can be kept track of with a co-activation matrix, which counts the average co-activation of each pair of units during Gibbs sampling, both in the wake phase and in the dream phase.

**Question 1.5 (4 marks):** What is the “temperature” of a Boltzmann machine, i.e. what does it control and what impact does it have on the network’s behaviour?

From the learning rules

$$\Delta W_{ij} = \eta \frac{1}{T} [\langle a_i a_j \rangle^+ - \langle a_i a_j \rangle^-]$$
$$\Delta \mathbf{\Theta} = \eta \frac{1}{T} [\langle A \rangle^+ - \langle A \rangle^-]$$

The higher the temperature *T*, the smaller the weights and thresholds update will be, and vice-versa. Therefore, a higher temperature implies more exploration/less exploitation of the loss function, and a lower temperature implies the opposite.

**Question 1.6 (4 marks):** In your own words, what is the “annealing schedule” of a Boltzmann machine and why is it important?

Per the simulated annealing strategy, the annealing schedule enables finer control of the exploration-exploitation ratio throughout training, which is important to ensure convergence of the loss. Typically, training starts off with more exploration, and gradually gives way to exploitation as it progresses.

**Question 1.7 (6 marks):** What does the equation in Q1.3 have to do with dreaming? What is the implied theory of what dreams are and why we forget them?
If the hidden units capture the correct constraints, then when the system runs without any sensory inputs it should generate data on the visual units that looks like data from the real world. Put another way: the goal of learning is to make the network’s dreams look as much like reality as possible.

The loss function that we want to reduce is a measurement of the difference between the “awake” probabilities and the “dream” probabilities. The two terms in the equation in Q1.3 are the Hebbian plasticity during the awake phase, and anti-Hebbian

plasticity during the dreaming phase - hence remembering reality and forgetting dreams.

## Part 2 – Designing your Boltzmann machine (20 marks in total)

**Question 2.1 (4 marks):** What variables did you declare in your Boltzmann machine class? Why did you include these in the class, i.e. justify your design decisions?

Necessary for training (Gibbs sampling, weights and biases updates, calculating loss):

```
A: All activations as vector of binary states
W: Matrix of weights connecting units, initialized randomly
Theta: Vector of biases, initialized randomly
```

Easy access:

```
V: Activations of visible units as a window of activations vector A
H: Activations of hidden units as a window of activations vector A
```

To remove training and test images as function arguments and simplify class functions:

```
train_images: All training images
test_images: All test images
```

Basic information to improve code legibility:

```
n: Total number of units
train_size: Number of training images
```

**Question 2.2 (4 marks):** How will you calculate the weight updates for the network?

For each training iteration, calculate the average co-activation matrices and activation count vectors through Gibbs sampling

```
# Obtain the average co-activation matrices AA[wake] and AA[dream], and activation count vectors A[wake] and A[dream]
# through Gibbs sampling
AA[wake], A[wake], AA[dream], A[dream] = gibbs()
```

Then, the weights and thresholds are updated as follows

```
# eta learning rate and T temperature
W += eta * (AA[wake] - AA[dream]) / T
Theta += eta * (A[wake] - A[dream]) / T
```

**Question 2.3 (4 marks):** How will you define the annealing and training schedule?

The annealing and training schedule is defined as

```
# Increasing temperature, selection of 100 temperatures
anneal_sched = [20, 30, 40, ..., 1010]

# The total number of iterations is 1000, 10 iterations for each temperature
iters = 1000
```

The temperature increases as weight updates are larger for lower temperatures, which is preferred at the beginning of training.

## Part 3 – Running your Boltzmann machine (20 marks in total + 10 available bonus marks)

**Question 3.2 (10 marks + 5 bonus marks):** If you run your Boltzmann machine freely (i.e. without clamping the visual units), what happens? What images are generated on the visible units? In your own words, describe what you think your Boltzmann machine has or hasn't learned, and how this might relate to a healthy or unhealthy brain. You will get 5 marks for your answers to these questions and the ideas behind them (write it in your pdf document). You will get another 5 marks if Raymond can run your image generation code and see some plotted images. You will get 5 bonus marks if the images your Boltzmann machine generates look anything like MNIST digits.

If trained correctly, the images generated on the visible units should look similar to MNIST digits depending on the activations of the hidden units. The Boltzmann machine then would have learned the higher-order, weak constraints between the pixels of the MNIST image, captured as weights of the Boltzmann machine. This can be related to a healthy brain with correct synaptic weights that can read or imagine handwriting that never looks the same, just similar.

If the machine is unable to generate images similar to MNIST digits, it means its weights have not captured these higher-order, weak constraints. This is analogous to an unhealthy brain with damaged neurons or synaptic weights, that would have trouble recognizing handwriting patterns.