## Congratulations! You passed!

Next Item

1 / 1 points

1. The Boltzmann Machine learning algorithm involves computing two expectations -

- $\langle s_i s_j \rangle_{data}$: Expected value of $s_i s_j$ at equilibrium when the visible units are fixed to be the data.
- $\langle s_i s_j \rangle_{model}$: Expected value of $s_i s_j$ at equilibrium when the visible units are not fixed.

When applied to a general Boltzmann Machine (not a Restricted one), this is an approximate learning algorithm because

- [ ] The first expectation can be computed exactly, but the second one cannot be.

**Un-selected is correct**

- [x] There is no efficient way to compute the first expectation exactly.

**Correct**
Computing $\langle s_i s_j \rangle_{data}$ is hard in general. It usually involves sampling from the model conditioned on the data.

- [x] There is no efficient way to compute the second expectation exactly.

**Correct**
Computing $\langle s_i s_j \rangle_{model}$ is hard in general. It usually involves sampling from the model.

- [ ] The first expectation cannot be computed exactly, but the second one can be.

**Un-selected is correct**

1 / 1 points

2. Throughout the lecture, when talking about Boltzmann Machines, why do we talk in terms of computing the **expected** value of $s_i s_j$ and not the value of $s_i s_j$ ?

- ( ) It is possible to compute the exact value but it is computationally inefficient.
- ( ) The expectation only refers to an average over all training cases.
- (x) It does not make sense to talk in terms of a unique value of $s_i s_j$ because $s_i$ and $s_j$ are random variables and the Boltzmann Machine defines a probability distribution over them.

**Correct**

- ( ) It is not possible to compute the exact value no matter how much computation time is provided. So all we can do is compute an approximation.

1 / 1 points

3. When learning an RBM, we decrease the energy of data particles **and** increase the energy of fantasy particles. Brian insists that the former is not needed. He claims that it is should be sufficient to just increase the energy of negative particles and the energy of all other regions of state space would have decreased relatively. Then we can get away with not having to clamp the inputs and doing all the work to compute $\langle s_i s_j \rangle$. What is wrong with this intuition ?

- ( ) Since total energy is constant, some particles must loose energy for others to gain energy.
- ( ) The sum of all updates must be zero so we need to increase the energy of negative particles to balance things out.
- ( ) There is nothing wrong with the intuition. This method is an alternative way of learning a Boltzmann Machine.
- (x) If the model was not decreasing the energy of the positive particles, it will not be using the data at all. The negative particles would roam around freely.

**Correct**
The algorithm uses the data by lowering the energy of positive particles. Without that the model would just be sampling from the initial distribution.

1 / 1 points

4. Restricted Boltzmann Machines are easier to learn than Boltzmann Machines with arbitrary connectivity. Which of the following is a contributing factor ?

- (x) For RBMs, $\langle s_i s_j \rangle_{data}$ can be computed exactly, whereas this is not the case for general BMs

**Correct**
Since there are no connections between hidden units directly or through any unobserved units, each hidden unit is conditionally independent of all others given the data.

- ( ) It is possible to run a persistent Markov chain in RBMs but not in general BMs.
- ( ) RBMs are more powerful models, i.e., they can model more probability distributions than general BMs.
- ( ) The energy of any configuration of an RBM is a linear function of its states. This is not true for a general BM.

1 / 1 points

5. PCD a better algorithm than CD1 when it comes to training a good generative model of the data. This means that samples drawn from a freely running Boltzmann Machine which was trained with PCD (after enough time) are likely to look more realistic than those drawn from the same model trained with CD1. Why does this happen ?

- ( ) In PCD, the persistent Markov chain can remember the state of the positive particles across mini-batches and show them when sampling. However, CD1 resets the Markov chain in each update so it cannot retain information about the data for a long time.
- (x) In PCD, the persistent Markov chain explores different regions of the state space. However, CD1 lets the Markov chain run for only one step. So CD1 cannot explore the space of possibilities much and can miss out on increasing the energy of some states which ought to be improbable. These states might be reached when running the machine for a long time leading to unrealistic samples.

**Correct**

- ( ) In PCD, only a single Markov chain is used throughout learning, whereas CD1 starts a new one in each update. Therefore, PCD is a more consistent algorithm.
- ( ) In PCD, many Markov chains are used throughout learning, whereas CD1 uses only one. Therefore, samples from PCD are an average of samples from several models. Since model averaging helps, PCD generates better samples.

1 / 1 points

6. I'ts time for some math now!

In RBMs, the energy of any configuration is a linear function of the state.

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j W_{ij}$$

and this eventually leads to

$$\Delta W_{ij} \propto \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

If the energy was non-linear, such as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} f(v_i, h_j) W_{ij}$$

for some non-linear function $f$, which of the following would be true.

- ( ) $\Delta W_{ij} \propto f(\langle v_i \rangle_{data}, \langle h_j \rangle_{data}) - f(\langle v_i \rangle_{model}, \langle h_j \rangle_{model})$
- ( ) $\Delta W_{ij} \propto \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$
- ( ) The weight update can no longer be written as a difference of data and model statistics.
- (x) $\Delta W_{ij} \propto \langle f(v_i, h_j) \rangle_{data} - \langle f(v_i, h_j) \rangle_{model}$

**Correct**

$$p(v) = \exp(-E(\mathbf{v}, \mathbf{h}))/Z$$
$$\Rightarrow \log(p(\mathbf{v})) = -E(\mathbf{v}, \mathbf{h}) - \log(Z)$$
$$\Rightarrow \frac{\partial \log(p(\mathbf{v}))}{\partial W_{ij}} = f(v_i, h_j) - \sum_{\mathbf{v}', \mathbf{h}'} P(v', h') f(v'_i, h'_j)$$

Averaging over all data points,

$$\frac{\partial \log(p(\mathbf{v}))}{\partial W_{ij}} = \langle f(v_i, h_j) \rangle_{data} - \langle f(v_i, h_j) \rangle_{model}$$
$$\Delta W_{ij} \propto \frac{\partial \log(p(\mathbf{v}))}{\partial W_{ij}}$$
$$\Rightarrow \Delta W_{ij} \propto \langle f(v_i, h_j) \rangle_{data} - \langle f(v_i, h_j) \rangle_{model}$$

1 / 1 points

7. In RBMs, the energy of any configuration is a linear function of the state.

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j W_{ij}$$

and this eventually leads to

$$P(h_j = 1 | \mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} v_i - b_j)}$$

If the energy was non-linear, such as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} f(v_i, h_j) W_{ij}$$

for some non-linear function $f$, which of the following would be true.

- ( ) $P(h_j = 1 | \mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} v_i - b_j)}$
- ( ) $P(h_j = 1 | \mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} (f(v_i, 1) + f(v_i, 0)) - b_j)}$
- ( ) None of these is correct.
- (x) $P(h_j = 1 | \mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} (f(v_i, 1) - f(v_i, 0)) - b_j)}$

**Correct**