CS 475 Machine Learning: Homework 4
The EM Algorithm (and more)

Due: Monday November 9, 2015, 11:59pm

100 Points Total Version 1.0

Make sure to read from start to finish before beginning the assignment.

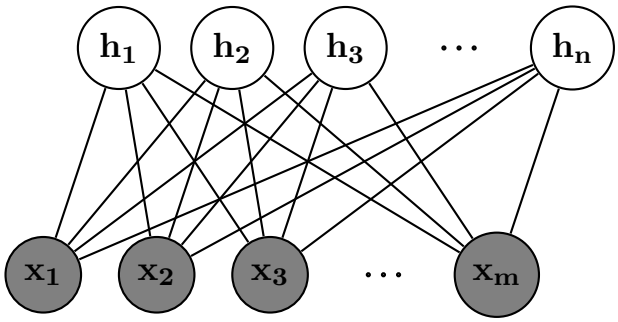
1 Programming (60 points)

A popular type of recurrent neural network is the Boltzmann machine, the development of which goes back to the mid 1980s. Boltzmann machines are neural networks that can learn internal representations. While learning Boltzmann machines are challenging, a common variant, the Restricted Boltzmann Machine (RBM) allows for efficient learning algorithms. For this reason, and others, many deep networks are composed of RBMs. A RBM is characterized by two layers: a hidden layer and an observed layer. These two layers form a bipartite graph, i.e., there are no connections within a layer. The observed layer is the input and the learned representation is captured by the hidden layer.

In this assignment, you will implement the gradient based learning algorithm for a RBM given its training data. Since RBMs have both observed and latent variables, the EM algorithm is used for learning. In the M-step, you will use gradient based optimization to update the parameters. In the E-step, you will compute the expectations of the variables using the current model parameters. For simplicity, we will use small networks where exact inference is possible for the E-step. In practice, more complex networks require approximate inference techniques for efficient learning.

1.1 RBM

Consider the following RBM configuration with n latent variables and m observed variables, all of which are binary valued.



Probability in RBM models is often expressed as "energy", where maximizing the probability means minimizing the energy. The joint probability distribution for the variables

in this RBM is given by:

$$p(\mathbf{x}, \mathbf{h}; \theta) = \frac{1}{Z} \exp\{-E_\theta(\mathbf{x}, \mathbf{h})\} \tag{1}$$

where $E_\theta(\mathbf{x}, \mathbf{h})$ is the energy function and Z is the partition function that normalizes the distribution. In this notation, $\mathbf{h}$ are the hidden variables and $\mathbf{x}$ are the observed variables. The energy function is defined as:

$$E_\theta(\mathbf{x}, \mathbf{h}) = -\mathbf{x}^T \mathbf{W} \mathbf{h} - \mathbf{b}^T \mathbf{x} - \mathbf{d}^T \mathbf{h} \tag{2}$$

We define the model parameters as $\theta := \{\mathbf{W}, \mathbf{b}, \mathbf{d}\}$, where $\mathbf{W} = (w_{i,j})$ is a matrix of weights associated with the connection between the visible unit x_i and the hidden unit h_j. $\mathbf{b}$ and $\mathbf{d}$ are vectors, where the ith position of $\mathbf{b}$ (b_i) and and the jth position of $\mathbf{d}$ (d_j) correspond to the bias parameters for the observed variable x_i and the latent variable h_j respectively.

The partition function Z is used to normalize the energy function to obtain a probability distribution:

$$Z = \sum_{(\mathbf{x}, \mathbf{h})} \exp\{-E_\theta(\mathbf{x}, \mathbf{h})\} \tag{3}$$

which sums over all possible combinations of the vectors $\mathbf{x}$ and $\mathbf{h}$.

Using the conditional independence assumptions inherent in this model, we obtain the following probabilities:

$$p(\mathbf{x}|\mathbf{h}; \theta) = \prod_{i=1}^{m} p(x_i|\mathbf{h}; \theta) \tag{4}$$

$$p(\mathbf{h}|\mathbf{x}; \theta) = \prod_{j=1}^{n} p(h_j|\mathbf{x}; \theta) \tag{5}$$

The conditional probability of a single variable (e.g., $p(x_i = 1|\mathbf{h}; \theta)$) is represented using a sigmoid activation function $\sigma(z)$:

$$p(h_j = 1|\mathbf{x}; \theta) = \sigma(\mathbf{x}^T \mathbf{W}_{-,j} + d_j) \tag{6}$$

$$p(x_i = 1|\mathbf{h}; \theta) = \sigma(\mathbf{h}^T \mathbf{W}_{i,-}^T + b_i) \tag{7}$$

where $\mathbf{W}_{-,j}$ is the jth column of $\mathbf{W}$, $\mathbf{W}_{i,-}$ is the ith row of $\mathbf{W}$, and $\sigma(z)$ is defined as:

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{8}$$

The likelihood of a single example $\mathbf{x}$ is given by:

$$p(\mathbf{x}; \theta) = \sum_{\mathbf{h}} \frac{1}{Z} \exp\{-E_\theta(\mathbf{x}, \mathbf{h})\} \tag{9}$$

With T training examples, the log likelihood is given by:

$$\sum_{t=1}^{T} \log p(\mathbf{x}^{(t)}; \theta) = \sum_{t=1}^{T} \log \sum_{\mathbf{h}} \frac{1}{Z} \exp\{-E_\theta(\mathbf{x}^{(t)}, \mathbf{h})\} \tag{10}$$

Gradient-based optimization requires the gradients of the parameters, which for the RBM is given by:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{W}_{i,j}} \sum_{t=1}^{T} \log p(\mathbf{x}^{(t)};\theta) &= \frac{\partial}{\partial \mathbf{W}_{i,j}} \sum_{t=1}^{T} \log \frac{\sum_{\mathbf{h}} \exp\{-E_\theta(\mathbf{x}^{(t)},\mathbf{h})\}}{\sum_{(\mathbf{x},\mathbf{h})} \exp\{-E_\theta(\mathbf{x},\mathbf{h})\}} \\
&= \sum_{t=1}^{T} \frac{\partial}{\partial \mathbf{W}_{i,j}} \log \sum_{\mathbf{h}} \exp\{-E_\theta(\mathbf{x}^{(t)},\mathbf{h})\} - \sum_{t=1}^{T} \frac{\partial}{\partial \mathbf{W}_{i,j}} \log \sum_{(\mathbf{x},\mathbf{h})} \exp\{-E_\theta(\mathbf{x},\mathbf{h})\} \\
&= \sum_{t=1}^{T} \frac{\sum_{\mathbf{h}} \exp\{-E_\theta(\mathbf{x}^{(t)},\mathbf{h})\}(-\frac{\partial}{\partial \mathbf{W}_{i,j}} E_\theta(\mathbf{x}^{(t)},\mathbf{h}))}{\sum_{\mathbf{h}} \exp\{-E_\theta(\mathbf{x}^{(t)},\mathbf{h})\}} \\
&\quad - \sum_{t=1}^{T} \frac{\sum_{(\mathbf{x},\mathbf{h})} \exp\{-E_\theta(\mathbf{x},\mathbf{h})\}(-\frac{\partial}{\partial \mathbf{W}_{i,j}} E_\theta(\mathbf{x},\mathbf{h}))}{\sum_{(\mathbf{x},\mathbf{h})} \exp\{-E_\theta(\mathbf{x},\mathbf{h})\}} \\
&= -\sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{h}|\mathbf{x}^{(t)};\theta)} \left[\frac{\partial}{\partial \mathbf{W}_{i,j}} E_\theta(\mathbf{x}^{(t)},\mathbf{h})\right] + \sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{x},\mathbf{h};\theta)} \left[\frac{\partial}{\partial \mathbf{W}_{i,j}} E_\theta(\mathbf{x},\mathbf{h})\right] \\
&= -\sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{h}|\mathbf{x}^{(t)};\theta)} \left[-x_i^{(t)} h_j\right] + \sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{x},\mathbf{h};\theta)} \left[-x_i h_j\right] \qquad (11) \\
\frac{\partial}{\partial \mathbf{b}_i} \sum_{t=1}^{T} \log p(\mathbf{x}^{(t)};\theta) &= -\sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{h}|\mathbf{x}^{(t)};\theta)} \left[-x_i^{(t)}\right] + \sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{x},\mathbf{h};\theta)} \left[-x_i\right] \qquad (12) \\
\frac{\partial}{\partial \mathbf{d}_j} \sum_{t=1}^{T} \log p(\mathbf{x}^{(t)};\theta) &= -\sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{h}|\mathbf{x}^{(t)};\theta)} \left[-h_j\right] + \sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{x},\mathbf{h};\theta)} \left[-h_j\right] \qquad (13)
\end{aligned}$$

where we have the expectations with respect to $p(\mathbf{h}|\mathbf{x}^{(t)};\theta)$, and $p(\mathbf{x},\mathbf{h};\theta)$. Intuitively, the gradient acts to move the two expectations close to each other until the gradient is 0, at which point the gradient of the energy functions have equal expectations with $\mathbf{x}$ sampled from the training distribution or with $\mathbf{x}$ sampled from the model. In other words, the distribution from the model and the training data agree with each other.

The M-step is comprised of updating the model parameters using the gradients defined in Eq. (11,12,13). As is typical in the EM algorithm, this M-step depends on computing the expectation of the hidden variables according to the model parameters $p(\mathbf{h}|\mathbf{x}^{(t)};\theta)$ and the joint of the hidden and observed variables $p(\mathbf{x},\mathbf{h};\theta)$. Note that we need to compute these distributions for all possible combinations of $\mathbf{x},\mathbf{h}$ given the current estimate of the parameters θ. Obviously for large networks, we could never sum over all possible combinations. However, in this homework we will restrict ourselves to small networks (i.e. with very few binary-valued nodes) where these summations are tractable using Eq. (1) and Bayes rules.

Note that before computing the expectations, we can first marginalize over variables that do not appear inside the expectations. For example, you can replace $p(\mathbf{h}|\mathbf{x}^{(t)})$ with $p(h_j|\mathbf{x}^{(t)};\theta)$, and $p(\mathbf{x},\mathbf{h};\theta)$ with $p(x_i,h_j;\theta)$ in Eq. (11) since the functions inside the expectations only involve x_i and h_j.

To give you a specific example, Eq. (11) would be derived as:

$$
\begin{aligned}
& -\sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{h}|\mathbf{x}^{(t)};\theta)} \left[-x_i^{(t)} h_j\right] + \sum_{t=1}^{T} \mathbb{E}_{p(\mathbf{x},\mathbf{h};\theta)} \left[-x_i h_j\right] \\
= & -\sum_{t=1}^{T} \mathbb{E}_{p(h_j|\mathbf{x}^{(t)};\theta)} \left[-x_i^{(t)} h_j\right] + T \cdot \mathbb{E}_{p(x_i,h_j;\theta)} \left[-x_i h_j\right] && (14) \\
= & -\sum_{t=1}^{T} \sum_{h_j} p(h_j|\mathbf{x}^{(t)};\theta) \left(-x_i^{(t)} h_j\right) + T \cdot \sum_{x_i,h_j} p(x_i,h_j;\theta) \left(-x_i h_j\right) && (15) \\
= & -\sum_{t=1}^{T} p(h_j = 1|\mathbf{x}^{(t)};\theta) \left(-x_i^{(t)}\right) - T \cdot p(x_i = 1, h_j = 1;\theta) && (16) \\
= & -\sum_{t=1}^{T} \sigma(\mathbf{x}^{(t)T} \mathbf{W}_{-,j} + d_j) \left(-x_i^{(t)}\right) - T \cdot \sum_{\mathbf{x}_{-i},\mathbf{h}_{-j}} p(\mathbf{x}_{-i}, \mathbf{h}_{-j}, x_i = 1, h_j = 1;\theta) && (17)
\end{aligned}
$$

where we denote all variables in $\mathbf{x}$ (and $\mathbf{h}$) except x_i (and h_j) as $\mathbf{x}_{-i}$ (and $\mathbf{h}_{-j}$) in Eq. (17), and the expectations in Eq. (14) are expanded as in Eq. (15) by definition. E.g., $\mathbb{E}_{p(h_j|\mathbf{x}^{(t)};\theta)} \left[-x_i^{(t)} h_j\right]$ is the expected value of $-x_i^{(t)} h_j$ when $h_j \sim p(h_j|\mathbf{x}^{(t)};\theta)$.

1.2 Initialization

We need to randomly initialize the parameters $\theta^{(0)} = \{\mathbf{W}^{(0)}, \mathbf{b}^{(0)}, \mathbf{d}^{(0)}\}$. To ensure that everyone gets the same values of all the parameters, you must use the following procedure *exactly*.

$$
\mathbf{W}_{i,j}^{(0)} = \begin{cases} 1 & \text{if } j \text{ is even} \\ 0 & \text{otherwise} \end{cases} \quad \mathbf{b}_i^{(0)} = \begin{cases} 1 & \text{if } i \text{ is even} \\ 0 & \text{otherwise} \end{cases} \quad \mathbf{d}_j^{(0)} = \begin{cases} 1 & \text{if } j \text{ is even} \\ 0 & \text{otherwise} \end{cases} \tag{18}
$$

1.3 The Framework of learning RBM

Using the above derivations, learning an RBM network proceeds as follows.

1. Set learning rate η and number of iterations K.
2. Initialize all the parameters $\theta^{(0)} = \{\mathbf{W}^{(0)}, \mathbf{b}^{(0)}, \mathbf{d}^{(0)}\}$ according to Sec. 1.2.
3. E-Step: compute the probabilities $p(\mathbf{h}|\mathbf{x}^{(t)};\theta)$ and $p(\mathbf{x},\mathbf{h};\theta)$ based on current model parameters.
4. M-Step: update parameters $\theta^{(k)} = \theta^{(k-1)} + \eta \frac{\partial}{\partial \theta^{(k-1)}} \sum_{t=1}^{T} \log p(\mathbf{x}^{(t)};\theta^{(k-1)})$ using Eq. (11,12,13).
5. Repeat E-Step and M-Step for K iterations.

1.4 Implementation

IMPORTANT: You are *only* allowed to *add* your own code in `cs475.RBM.RBMEnergy`. *Do not* delete any existing code.

We have provided you 3 samples of the data file, `RBM_data.txt`. The format is `"m n T"` on the first line, where m is the number of input(visible) nodes $\mathbf{x}$, n is the number of the hidden nodes $\mathbf{h}$ and T is the number of training examples, followed by T lines of training examples, each of which is an observation of length m.

You will be able to get the values for m and n by calling the following functions in `cs475.RBM.RBMParameters`:

```
public int numVisibleNodes()   // returns m
public int numHiddenNodes()  // returns n
public int numExamples()  // returns T
```

We have also provided functions in `cs475.RBM.RBMParameters` that give you the values of the visible nodes, the bias parameters and the weights, and functions that update these parameters,

```
public double getExample(int t, int i)//returns the value of node x_i for example t
public double getVisibleBias(int i)  // returns the bias for node x_i
public double getHiddenBias(int i)  // returns the bias for node h_i
public double getWeight(int i, int j) // returns the weight(i,j)
public boolean setVisibleBias(int i, double val)  // sets the bias for node x_i
public boolean setHiddenBias(int i, double val)  // sets the bias for node h_i
public boolean setWeight(int i, int j, double val) // sets the weight W_(i,j)
```

1.4.1 Command Line Arguments

We have already added 3 command line options in `cs475.RBM.RBMTester` for you.

```
registerOption("data", "String", true, "The data to use.");
registerOption("gd_eta", "double", true, "The constant scalar for learning rate. default: 0.1");
registerOption("gd_iterations", "int", true, "The number of iterations.default: 20");
```

1.4.2 What You Need to Implement

We have provided you with the class `cs475.RBM.RBMEnergy` with one method left blank that you will need to implement:

```
public class RBMEnergy {
     public double learning() {

     // TODO: Add code here

     }
}
```

1.5 How We Will Run Your Code

We will run your code using the data file and the following command line arguments:

```
java cs475.RBM.RBMTester -data RBMData.txt  \
-gd_iterations K -gd_eta eta
```

Note: we may use data files with different values of m and n, so make sure your code works for any reasonable input.

Your output should just be the results of the print statements in the code given. **Do not print anything else in the version you hand in.**

2 Analytical Questions (40 points)

1. Overfitting in Clustering (10 points) Given the data set $x_1, ..., x_n$, we want to do clustering by the K-means algorithm. The K-means algorithm aims to partition the n observations into k sets ($k < n$) $S = \{S_1, S_2, ..., S_k\}$ so as to minimize the within-cluster sum of squares

$$\min_{S=\{S_1,...,S_k\}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_j - \mu_j\|_2^2$$

where μ_j is the mean of points in S_j.

(a) Prove that the objective value does not increase in each iteration of the K-means algorithm.

(b) Let γ_k denote the global optimal objective value, prove γ_k is non-increasing in k.

2. Curse-of-dimensionality (10 points) In this problem, we study why K-NN could fail in high dimensions by means of a very simple example. Consider a sphere of radius r in d-dimensions together with a concentric hypercube of side $2r$. The sphere touches the hypercube at the center of each of its sides.

(a) V_c is the volume of the cube and V_s is the volume of the sphere, where the volume of a d-dimensional sphere with radius r is given as

$$V_s = \frac{r^d \sqrt{\pi}^d}{\Gamma(d/2+1)},$$

where $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} \mathrm{d}t$. Note that $\Gamma(z) = (z-1)!$ (the factorial of $z-1$) if z is a positive integer. Please show that:

$$\lim_{d \to \infty} \frac{V_s}{V_c} = 0 \tag{19}$$

Note that this relies on algebra and will not require any complex calculus (just some basic facts on limits). You may find the following limit useful:

$$\lim_{z \to \infty} \frac{\Gamma(z+1)}{\sqrt{2\pi z} e^{-z} z^z} = 1$$

(b) What is the connection between (19) and the curse of dimensionality?

3. Semi-supervised EM algorithm (10 points) Suppose that some of your observed data are labelled. You have $x_1, ..., x_n \in \mathbb{R}^d$ and $x_{n+1}, ..., x_{n+m}$. Meanwhile, you also know the labels corresponding to $x_{n+1}, ..., x_{n+m}$, i.e., $y_{n+1}, ..., y_{n+m} \in \{1, ..., K\}$, where K is the number of clusters. Please design a Gaussian Mixture Model-based EM algorithm to cluster the data. [Hint: For $x_{n+1}, ..., x_{n+m}$, the corresponding labels are no longer missing values]

(a) Write the new likelihood objective for this new algorithm.

(b) Write the new update rules in each iteration.

4. Modified EM (10 points) The EM algorithm we learned about in class is just one of several different general EM algorithms, all with similar goals and structures. In this problem we will consider an alternative EM algorithm which modifies the M-step. Instead of maximizing $\mathcal{L}(q, \theta)$ with respect to θ, the algorithm selects a single parameters $\theta_i \in \theta$ and modifies it to increase $\mathcal{L}(q, \theta)$.

(a) Will this new EM algorithm yield the same solution as the normal EM algorithm?

(b) Will this new EM algorithm converge (assuming lack of singularities)? If yes, then prove convergence. If no, then give a counterexample illustrating why not.

3 What to Submit

In each assignment you will submit two things.

1. **Code:** Your code as a zip file named `library.zip`. **You must submit source code (.java files)**. We will run your code using the exact command lines described above, so make sure it works ahead of time. Remember to submit all of the source code, including what we have provided to you.
2. **Writeup:** Your writeup as a **PDF file** (compiled from latex) containing answers to the analytical questions asked in the assignment. Make sure to include your name in the writeup PDF and use the provided latex template for your answers.

Make sure you name each of the files exactly as specified (library.zip and writeup.pdf).

To submit your assignment, visit the "Homework" section of the website (http://www.cs475.org/.)

4 Questions?

Remember to submit questions about the assignment to the appropriate group on the class discussion board: http://bb.cs475.org.