

CS 475 Machine Learning: Homework 4
Non-Linear Supervised Learning (and more)
Due: Thursday November 6, 2014, 11:59pm
100 Points Total Version 1.0

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

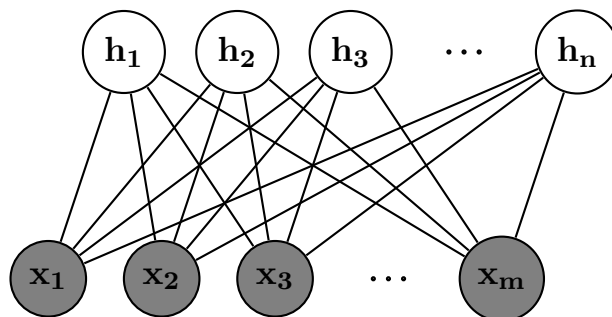
1 Programming (60 points)

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:

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

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:

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

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 i th position of \mathbf{b} (b_i) and the j th 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:

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

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:

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\partial}{\partial \mathbf{b}_i} \sum_{t=1}^T \log p(\mathbf{x}^{(t)}; \theta) = - \sum_{t=1}^T \mathbb{E}_{p(\mathbf{h}|\mathbf{x}^{(t)}; \theta)} [-x_i^{(t)}] + \sum_{t=1}^T \mathbb{E}_{p(\mathbf{x}, \mathbf{h}; \theta)} [-x_i] \tag{12}$$

$$\frac{\partial}{\partial \mathbf{d}_j} \sum_{t=1}^T \log p(\mathbf{x}^{(t)}; \theta) = - \sum_{t=1}^T \mathbb{E}_{p(\mathbf{h}|\mathbf{x}^{(t)}; \theta)} [-h_j] + \sum_{t=1}^T \mathbb{E}_{p(\mathbf{x}, \mathbf{h}; \theta)} [-h_j] \tag{13}$$

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:

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

$$= - \sum_{t=1}^T \sum_{h_j} p(h_j|\mathbf{x}^{(t)}; \theta) \left(-x_i^{(t)} h_j \right) + T \cdot \sum_{x_i, h_j} p(x_i, h_j; \theta) (-x_i h_j) \tag{15}$$

$$= - \sum_{t=1}^T p(h_j = 1|\mathbf{x}^{(t)}; \theta) \left(-x_i^{(t)} \right) - T \cdot p(x_i = 1, h_j = 1; \theta) \tag{16}$$

$$= - \sum_{t=1}^T \sigma(\mathbf{x}^{(t)T} \mathbf{W}_{-,j} + d_j) \left(-x_i^{(t)} \right) - T \cdot \sum_{\mathbf{x}_{-i}, \mathbf{h}_{-j}} p(\mathbf{x}_{-i}, \mathbf{h}_{-j}, x_i = 1, h_j = 1; \theta) \tag{17}$$

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) // sets the bias for node x_i
public boolean setHiddenBias(int i) // sets the bias for node h_i
public boolean setWeight(int i, int j) // 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 (40 points)

1) Expectation Maximization (14 points) You plan to use a Naive Bayes classifier for a binary classification task where each feature is given by a single digit integer ($x_i(j) \in \{0, 1, \dots, 9\}$). However, when the data arrives it only has X values; there are no labels. You still want to use your Naive Bayes classifier, so you decide to use the EM algorithm to handle the latent variables: Y .

- Write the log-likelihood for this setting.
- Derive the E and M steps for learning your model.
- How should you initialize the parameters of the model?
- Will parameter initialization matter for learning?

2) k -Nearest Neighbors (13 points) A problem with the k -NN algorithm is that computing the distance between each query x and all the training data examples can be very slow. Consider this solution to make this computation faster by performing some computations offline, before we are given a query example x to classify. Consider a binary classification task. We define μ_0 and μ_1 to be the centroids of the training data corresponding to the two classes: C_0 and C_1 .

- Propose a new algorithm that uses μ_0 and μ_1 for classification that is faster than the standard k -NN.
- Write the decision function for this new algorithm as $\text{sign}(\sum_{i=1}^n \alpha_i \mathbf{x}_i^\top x + b)$, where \mathbf{x}_i are the training examples and α and b are model parameters.
- Compare the computational cost of prediction of this new algorithm with k -NN.

3) Clustering (13 points) We learned that the distortion measure can be used as an objective function for clustering:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|x_n - \mu_k\|^2 \quad (19)$$

The k -means algorithm minimizes this objective in a coordinate-descent fashion. Each iteration of k -means makes two updates:

$$r_{nk} = 1 \quad \text{if } k = \arg \min_j \|x_n - \mu_j\|^2, \quad 0 \quad \text{otherwise} \quad (20)$$

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} \quad (21)$$

- For fixed centers μ_j , show that equation (20) minimizes J with respect to r_{nk} . (Hint: you can show that assigning an example to any other cluster would be worse with respect to J .)
- For fixed r_{nk} show that equation (21) minimizes J with respect to all centers μ_j . (Hint: take the derivative of J with respect to a single cluster center μ_k .)

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>.