Homework Assignment 1

CSE 190: Neural Networks

Fall 2015

Instructions

Due on Friday October 16^{th}:

1. Please hand in a hard copy for your assignment. You are required to make a LaTeXreport for each assignment. You are free to choose the template you want.
2. You may use a language of your choice (Python and MATLAB are recommended) but all source code used in the assignment must be attached in the appendix. Please keep the code clean with explanatory comments, as they may be reused in the future.
3. Using the MATLAB .toolbox for neural networks or any off-the-shelf code is strictly prohibited.
4. Any form of cheating, lying, or plagiarism will not be tolerated. Discussions on course materials and homework solutions are encouraged, but you should write the final solutions alone. Books, notes, and Internet resources can be consulted, but not copied from. Working together on homeworks must follow the (spirit of) Gilligan's Island Rule (Dymond, 1986): No notes can be made during a discussion, and you must watch one hour of Gilligan's Island or equivalent before writing anything down. Suspected cheating will be reported to the Dean.

Perceptron (25 points)

1. (*5 points*) Recall the perceptron activation rule:

$$y = \begin{cases} 1 & \text{if } \sum_i w_i x_i \geqslant \theta \\ 0 & \text{else} \end{cases}$$

   Please derive the 2-D decision boundary for the perceptron (*2 points*), and prove that the distance from the decision boudary to the origin is given by:

$$l = \frac{w^T x}{\|x\|}$$

   (*3 points*)

2. (*7 points*) In the class, we learned how to learn "OR" function using perceptron learning rule. Now we want to learn "NAND" using four patterns , as shown in Table 1. Please complete the following task:

Input	Output
0 0	1
0 1	1
1 0	1
1 1	0

Table 1: The "NAND" function

(a) Write down the perceptron learning rule. (*1 point*)

(b) Create a table with 8 columns: | x_1 | x_2 | w_0 | w_1 | Net | Output | Teacher | Threshold (θ) |. Initialize w_0, w_1 and θ all to be 0 and learning rate is fixed to be 1. Add one row for each randomly selected pattern (training example) for the perceptron to learn. Stop when the learning converges. Please present the resulting table and show the final learned weights and threshold. You may pick the order to make the learning converge faster, if you can.(*4 points*)

(c) Is the solution unique? Why or why not? (*2 points*)

3. **Programming Assignment (*13 points*). The data is in hw1_iris.tar.gz under resources on piazza** In this problem, we will attempt to make a flower classifier. We shall use a modified version of the popular UCI Iris dataset provided in the file *iris_train.data* within the iris folder. The file describes a data set with the following characteristics:

   (a) Number of Attributes: 4 numeric attributes and the class label.

   (b) Attribute Information:

      i. sepal length in cm
      ii. sepal width in cm
      iii. petal length in cm
      iv. petal width in cm

   (c) The last column is the class label (Iris-setosa, Iris-versicolor)

   (d) You are required to perform the following tasks:

      i. Z-score the each attribute (feature) of data by the equation:

$$X_n = (X - \text{mean}(X))./(\text{standard-deviation}(X))$$

      You will find that z-score is a common pre-processing step when dealing with datasets. Can you explain why it is useful in our application ? (*2 points*)

      ii. Using the MATLAB function scatter(X,Y), plot each of the 2 dimensional feature space (i.e., sepal length vs. sepal width, sepal width vs. petal width, etc.). Are the classes linearly separable in each of the feature space? Why? (*3 points*)

      iii. Train a perceptron as a classifier using the delta learning rule on the training dataset *iris_train.data*, using all four features. You are free to choose the stopping criterion when necessary (i.e., a patience parameter or some test that the error is no longer going down). Be sure to include your source code in the appendix. (*4 points*)

      iv. Classify the test data of file *iris_test.data* and report the average error rate. The average error rate is simply the number of misclassified test data points averaged over the number of test data points. (*2 points*)

      v. What is your learning rate? What will happen if you raise/lower your learning rate? (*2 points*)

Logistic and Softmax Regression (25 points)

In this problem, we will classify handwritten digits from Yann LeCun's at MNIST Database. Please download the four files found there, these will be used for this problem. To reduce computation time, we will subset these files. Please use only the first 20,000 training images/labels and only the first 2,000 testing images/labels.

Logistic Regression

Logistic regression is a binary classification method. Logistic regression can be modeled as using a single neuron reading in an input vector $x \in \mathbb{R}^p$ and parametrized by weight vector $\theta \in \mathbb{R}^p$, where the neuron outputs the probability of the class being $y = 1$ given x.

$$P(y = 1|x) = h_\theta(x) = \frac{1}{1 + \exp(-\theta^\top x)} \equiv \sigma(\theta^\top x) \tag{1}$$

$$P(y = 0|x) = 1 - P(y = 1|x) = 1 - h_\theta(x). \tag{2}$$

With the hypothesis function defined, we now use Cross Entropy Loss function 3 over our training examples. This equation measures how well our hypothesis function h_θ does over the N data points,

$$E(\theta) = -\sum_{i=1}^{N} \left(y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right). \tag{3}$$

Finally, we may seek to optimize this cost function via gradient descent.

Softmax Regression

Softmax regression is the generalization of logistic regression for multiple (K) classes. Now given an input x, the softmax regression will output is a vector, where each element represents the probability of that class.

$$h_\theta(x) = \begin{bmatrix} P(y = 0|x;\theta) \\ P(y = 1|x;\theta) \\ \vdots \\ P(y = K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=0}^{K} \exp(\theta^{(j)\top} x)} \begin{bmatrix} \exp(\theta^{(0)\top} x) \\ \exp(\theta^{(1)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix} \tag{4}$$

With our model defined, we now seek to define a cost function by summing over all N examples. Our cost function is described in Equation 5

$$E(\theta) = -\left[\sum_{i=1}^{N} \sum_{k=0}^{K} 1\left\{ y^{(i)} = k \right\} \log \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=0}^{K} \exp(\theta^{(j)\top} x^{(i)})} \right] \tag{5}$$

where we have defined the indicator function $1\left\{ y^{(i)} = k \right\}$ to take on value 1 when $y^{(i)}$ is equal to k and 0 otherwise.

Gradient Descent

Recall the gradient descent iterative algorithm.

Algorithm 1 Gradient Descent

```
procedure GRADIENT DESCENT
    θ ← 0
    for t = 0, ..., m do
        θ_{t+1} = θ_t − η Σ_{i=1}^{n} ∇E(θ)
    return θ
```

where η is the step size.

Problem (25 points)

1. **Show Gradient for Logistic Regression.** *(2 points)* We need the gradient of the cost function, Equation 3, with respect to the parameter θ. Show that for the logistic regression cost function, the gradient is:

$$\frac{\partial E(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} x_j^{(i)} (h_\theta(x^{(i)}) - y^{(i)}) \tag{6}$$

   Show work.

2. **Show Gradient for Softmax Regression.** *(3 points)* For softmax regression cost function, Equation 5, show that the gradient is:

$$\nabla_{\theta^{(k)}} E(\theta) = -\sum_{i=1}^{N} \left[x^{(i)} \left(1\{y^{(i)} = k\} - P(y^{(i)} = k|x^{(i)};\theta) \right) \right] \tag{7}$$

   Show work.

3. **Read in Data**. Read in the data from files. Each image is 28×28 pixels, so now our unraveled $x \in \mathbb{R}^{784}$, where each vector component represents the greyscale intensity of that pixel. For each image, append a '1' to the beginning of each x-vector; this will act as an intercept term.

4. **Logistic Regression via Gradient Descent.** *(10 points)* Now, using the gradient derived for Logistic Regression cross entropy loss, use gradient descent to classify given $x \in \mathcal{R}^{785}$ whether a digit's label is i or if it is some *other* digit, i.e. $i \notin \{0, \cdots, 9\}$. So for instance, if you are classifying $\{2\}$, you would designate $y = 2$ as the positive class, and all other integers as the negative class. For each image, do this 2-way classification for all 10-integers.

   (a) Report the test accuracy for each of the 10 2-way classifications on the test set.

   (b) For each image in the test set, report the overall test accuracy. An example will be considered labeled correctly if the logistic regression classification of the true label has the highest probability. So for instance, if the true label was $\{2\}$ for an image, you would count it as correctly classified if the logistic regression test of $\{2\}$ vs. $\{0, 1, 3, 4, 5, 6, 7, 8, 9\}$ had the highest probability of all the 10 2-way classifications.

5. **Softmax Regression via Gradient Descent.** *(10 points)* Now, using the gradient derived for Softmax Regression loss, use gradient descent to 10-way classify given $x \in \mathcal{R}^{785}$ what the digit's label is $y \in \{0, \cdots, 9\}$.

   (a) Plot the training accuracy on the training set vs. number of iterations of gradient descent.

   (b) Report the test accuracy on the test set.

   (c) Is the test accuracy lower or higher than the one-vs-all logistic regression approach? Why?