Homework #6

CS 260: Machine Learning Algorithms
Prof. Ameet Talwalkar
Due: 3/15/17, 10am
Please abide by the Academic Integrity Policy

1 Clustering

Given a set of data points $\{\mathbf{x}_n\}_{n=1}^N$, the k-means clustering minimizes the following distortion measure (or objective or clustering cost):

$$D = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2$$

where $\boldsymbol{\mu}_k$ is the prototype of the k-th cluster and r_{nk} is a binary indicator variable. If $\mathbf{x}_n$ is assigned to the cluster k, r_{nk} is 1 otherwise r_{nk} is 0. For each cluster, $\boldsymbol{\mu}_k$ is the representative for all the data points assigned to that cluster.

a. In the lecture, we stated but did not prove that $\boldsymbol{\mu}_k$ is the mean of all points associated with the kth cluster, thus motivating the name of the algorithm. You will now prove this statement. Assuming all r_{nk} are known (i.e., assuming you know the assignments of all N data points), show that if $\boldsymbol{\mu}_k$ is the mean of all data points assigned to cluster k, for any k, then the objective D is minimized. This justifies the iterative procedure of k-means[1].

b. We now change the distortion measure to

$$D = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|_1$$

In other words, we use the L_1 norm ($\|\mathbf{z}\|_1 = \sum_d |z_d|$) to measure the "closeness" of each point to the cluster prototypes. Under this new cost function, and again assuming that all r_{nk} are known, show that the $\{\boldsymbol{\mu}_k\}_{k=1}^K$ which minimize D are the elementwise medians of all data points assigned to the k-th cluster. Note that the elementwise median of a set of vectors is defined as a vector whose d-th element is the median of all vectors' d-th elements.

2 Neural Networks for MNIST Digit Recognition

In this problem, you will implement a neural network to classify handwritten digits using raw pixels as features. You will be using the classic MNIST digits dataset. The state-of-the-art error rate on this dataset using various learning methods is around 0.5% (see this leaderboard for details). However, you will be implementing a simple neutral network and should, with appropriate hyperparameter settings, get a test error of approximately 6% or better. Specifically, you will implement a neural network with a total of three layers: an input layer, a hidden layer, and an output layer. Please pay careful attention to the following implementation details.

[1] More rigorously, one would also need to show that if all $\boldsymbol{\mu}_k$ are known, then r_{nk} can be computed by assigning $\mathbf{x}_n$ to the nearest $\boldsymbol{\mu}_k$. You are not required to do so.

2.1 Details

- You will be using a hidden layer of size 200. Let $n_{in} = 784$, the number of raw pixel features. Let $n_{hid} = 200$, the size of the hidden layer. Finally, let $n_{out} = 10$, the number of output labels or classes. Then, you will have $n_{in} + 1$ units in the input layer, $n_{hid} + 1$ units in the hidden layer, and n_{out} units in the output layer. The input and hidden layers have one additional unit which always takes a value of 1 to represent bias. The output layer size is set to the number of classes. We will refer to the last layer as layer $L = 2$, and the input layer as layer 0.

- Each label will have to be transformed to a one-hot-encoding representation, i.e., a vector of length 10 that has a single 1 in the position of the true class and 0 everywhere else.

- Each layer is fully connected to the previous layer. The parameters of the model are the weights w_{ij}^{l}. For $l = 1$, the weights from the input to the hidden layer, there are n_{hid}-by-$(n_{in} + 1)$ weights, which represent the weights from the ith input node to the jth hidden node. For $l = L = 2$, the weights from the hidden layer to the output, there are n_{out}-by-$(n_{hid} + 1)$ weights, which represent the weights from the ith hidden node to the jth output node.

- You will be using the cross entropy error, given as

$$L = -\sum_{j=1}^{n_{out}} [y_j ln(x_j^{(L)}) + (1 - y_j) ln(1 - x_j^{(L)})]$$

  where $x_j^{(L)}$ is the output of the jth node at layer L.

- Hidden units should use $h(z) = tanh(z)$ as its activation function, and output units should use the sigmoid function $g(z) = \frac{1}{1+e^{-z}}$ as its activation function.

- Make sure you initialize your weights with random (small) values. This allows us to break symmetry that occurs when all weights are initialized to 0.

- It may be helpful to make your step size inversely proportional to the current training iteration.

- Datasets are provided as train.mat and test.mat. We also provide a train_small.mat which may be helpful in debugging and checking your code. Please report values for the full training set in train.mat.

- Function signatures are also provided for training (trainNeuralNetwork.m) and testing (testNeuralNetwork.m) your neural network.

2.2 Problems

a. Derive the stochastic gradient descent updates for all parameters w_{ij}^{l}.

b. Divide the full training dataset into training and validation sets (use an 80/20 split). Train this multi-layer neural network on the training data using stochastic gradient, and evaluate its accuracy on the validation set. As you see fit, tune various hyperparameters of the model, e.g., learning rate, when you stopped training, how you initialized the weights, and describe what you observe while performing this hyperparameter optimization.

c. After deciding on a good choice of hyperparameters, train the neural network on the *full training dataset*, and report the following results for this model:

   - Training accuracy and test accuracy.
   - Plots of training accuracy vs. iteration.

Submission Instructions

- Provide your answers to Problems 1 and 2.2 in hardcopy. The papers need to be stapled, and submitted at the beginning of class on the due date.
- For problem 2, please put all of your code in a single folder named `[lastname]_[firstname]_hw6`, and submit a single `.zip` file containing this folder called `[lastname]_[firstname]_hw6.zip` to **CCLE** by the due date. The only acceptable programming languages are MATLAB and Octave.
- You are encouraged to collaborate, but collaboration must be limited to discussion only and you need to write down / implement your solution on your own. You also need to list with whom you have discussed the HW problems.