Stony Brook University
CSE512 – Machine Learning – Spring 16
Homework 3, Due: Wednesday Mar 30, 2016, 11:59PM

This homework contains 4 questions. The last two questions require programming. The maximum number of points is 100 plus 10 bonus points.

1 Question 1 – Boosting (20 points)

We learned about boosting in lecture and the topic is covered in Murphy 16.4. On page 555 Murphy claims that "it was proved that one could boost the performance (on the training set) of any weak learner arbitrarily high, provided the weak learned could always perform slightly better than chance." We will now verify this in the AdaBoost framework.

1. (*5 points*) Given a set of N observations (x^j, y^j) where y^j is the label $y^j \in \{-1, 1\}$, let $h_t(x)$ be the weak classifier at step t and let α_t be its weight. First we note that the final classifier after T steps is defined as:

$$H(x) = sgn\left\{\sum_{t=1}^{T} \alpha_t h_t(x)\right\} = sgn\{f(x)\}$$

Where

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Show that:

$$\epsilon_{\text{Training}} = \frac{1}{N}\sum_{j=1}^{N} \delta(H(x^j) \neq y^j) \leq \frac{1}{N}\sum_{j=1}^{N} \exp(-f(x^j)y^j)$$

Where $\delta(H(x^j) \neq y^j)$ is 1 if $H(x^j) \neq y^j$ and 0 otherwise.

2. (*5 points*) The weight for each data point j at step $t+1$ can be defined recursively by:

$$w_j^{(t+1)} = \frac{w_i^{(t)} \exp(-\alpha_t y^j h_t(x^j))}{Z_t}$$

Where Z_t is a normalizing constant ensuring the weights sum to 1:

$$Z_t = \sum_{j=1}^{N} w_j^t \exp(-\alpha_t y^j h_t(x^j))$$

Show that:

$$\frac{1}{N}\sum_{j=1}^{N} \exp(-f(x^j)y^j) = \prod_{t=1}^{T} Z_t$$

3. (*10 points*) We showed above that training error is bounded above by $\prod_{t=1}^{T} Z_t$. At step t the values $Z_1, Z_2, \ldots, Z_{t-1}$ are already fixed therefore at step t we can choose α_t to minimize Z_t. Let

$$\epsilon_t = \sum_{j=1}^{N} w_j^t \delta(h_t(x^j) \neq y^j)$$

be the weighted training error for weak classifier $h_t(x)$ then we can re-write the formula for Z_t as:

$$Z_t = (1 - \epsilon_t)\exp(-\alpha_t) + \epsilon_t \exp(\alpha_t)$$

(a) First find the value of α_t that minimizes Z_t then show that

$$Z_t^{opt} = 2\sqrt{\epsilon_t(1 - \epsilon_t)}$$

(b) Assume we choose Z_t this way. Then re-write $\epsilon_t = \frac{1}{2} - \gamma_t$ where $\gamma_t > 0$ implies better than random and $\gamma_t < 0$ implies worse than random. Then show that:

$$Z_t \leq \exp(-2\gamma_t^2)$$

You may want to use the fact that $\log(1 - x) \leq -x$ for $0 \leq x < 1$

Thus we have:

$$\epsilon_{\text{training}} \leq \prod_{t=1}^{T} Z_t \leq \exp(-2\sum_{t=1}^{T} \gamma_t^2)$$

(c) Finally, show that if each classifier is better than random (e.g. $\gamma_t \geq \gamma$ for all t and $\gamma > 0$) that:

$$\epsilon_{\text{training}} \leq \exp(-2T\gamma^2)$$

Which shows that the training error can be made arbitrarily small with enough steps.

2 PCA via Successive Deflation [20 points]

(Adapted from Murphy Exercise 12.7)

Suppose we have a set of n data points $x_1, \ldots, x_n$, where each x_i is represented as a d-dimensional column vector.

Let $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_n]$ be the $(d \times n)$ matrix where column i is equal to $\mathbf{x}_i$. Define $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ to be the covariance matrix of $\mathbf{X}$, where $c_{ij} = \sum_{l=1}^{n} x_{il}x_{jl} = covar(i, j)$.

Next, order the eigenvectors of $\mathbf{C}$ by their eigenvalues (largest first), and let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ be the first k eigenvectors. These satisfy

$$\mathbf{v}_i^T \mathbf{v}_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

$\mathbf{v}_1$ is the first principal eigenvector of $\mathbf{C}$ (the eigenvector with the largest eigenvalue), and as such satisfies $\mathbf{C}\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$. Now define $\tilde{\mathbf{x}}_i$ as the orthogonal projection of $\mathbf{x}_i$ onto the space orthogonal to $\mathbf{v}_1$:

$$\tilde{\mathbf{x}}_i = (\mathbf{I} - \mathbf{v}_1\mathbf{v}_1^T)\mathbf{x}_i$$

Finally, define $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1; \ldots; \tilde{\mathbf{x}}_n]$ as the **deflated matrix** of rank $d - 1$, which is obtained by removing from the d-dimensional data the component that lies in the direction of the first principal eigenvector:

$$\tilde{\mathbf{X}} = (\mathbf{I} - \mathbf{v}_1\mathbf{v}_1^T)\mathbf{X}$$

1. [5 points] Show that the covariance of the deflated matrix,

$$\tilde{\mathbf{C}} = \frac{1}{n}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$$

is given by

$$\tilde{\mathbf{C}} = \frac{1}{n}\mathbf{X}\mathbf{X}^T - \lambda_1 \mathbf{v}_1 \mathbf{v}_1^T$$

(Hint: Some useful facts: $(\mathbf{I} - \mathbf{v}_1\mathbf{v}_1^T)$ *is symmetric,* $\mathbf{X}\mathbf{X}^T\mathbf{v}_1 = n\lambda_1\mathbf{v}_1$*, and* $\mathbf{v}_1^T\mathbf{v}_1 = 1$*. Also, for any matrices* $\mathbf{A}$ *and* $\mathbf{B}$*,* $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$*.)*

2. [5 points] Show that for $j \neq 1$, if $\mathbf{v}_j$ is a principal eigenvector of $\mathbf{C}$ with corresponding eigenvalue λ_j (that is, $\mathbf{C}\mathbf{v}_j = \lambda_j\mathbf{v}_j$), then $\mathbf{v}_j$ is also a principal eigenvector of $\tilde{\mathbf{C}}$ with the same eigenvalue λ_j.

3. [5 points] Let $\mathbf{u}$ be the first principal eigenvector of $\tilde{\mathbf{C}}$. Explain why $\mathbf{u} = \mathbf{v}_2$. (You may assume $\mathbf{u}$ is unit norm.)

4. [5 points] Suppose we have a simple method f for finding the leading eigenvector and eigenvalue of a positive-definite matrix, denoted by $[\lambda, \mathbf{u}] = f(\mathbf{C})$. Write some pseudocode for finding the first k principal basis vectors of $\mathbf{X}$ that only uses the special f function and simple vector arithmetic.

   (Hint: This should be a simple iterative routine that takes only a few lines to write. The input is $\mathbf{C}, k$*, and the function* f*, the output should be* $\mathbf{v}_j$ *and* λ_j *for* $j \in 1, \cdots, k$*)*

3 Programming Question (clustering with K-means) [30 points]

In class we discussed the K-means clustering algorithm. Your programming assignment this week is to implement the K-means algorithm on digit data.

3.1 The Data

There are two files with the data. The first `digit.txt` contains the 1000 observations of 157 pixels (a subset of the original 785) from images containing handwritten digits. The second file `labels.txt` contains the true digit label (either 1, 3, 5, or 7). You can read both data files in with

```
X = load('../hw3data/digits/digit.txt');
Y = load('../hw3data/digits/labels.txt');
```

Please note that there aren't IDs for the digits. Please assume the first line is ID 1, the second line is ID 2, and so on. The labels correspond to the digit file, so the first line of labels.txt is the label for the digit in the first line of digit.txt.

3.2 The algorithm

Your algorithm should be implemented as follows:

1. Select k starting centers that are points from your data set. You should be able to select these centers randomly or have them given as a parameter.

2. Assign each data point to the cluster associated with the nearest of the k center points.

3. Re-calculate the centers as the mean vector of each cluster from (2).

4. Repeat steps (2) and (3) until convergence or iteration limit.

Define convergence as no change in label assignment from one step to another **or** you have iterated 20 times (whichever comes first). Please count your iterations as follows: after 20 iterations, you should have assigned the points 20 times.

3.3 Within group sum of squares

The goal of clustering can be thought of as minimizing the variation within groups and consequently maximizing the variation between groups. A good model has low sum of squares within each group.

We define sum of squares in the traditional way. Let $\mathcal{C}_k$ be the k^{th} cluster and let $\boldsymbol{\mu}_k$ be the mean of the observations in cluster $\mathcal{C}_k$. Then the within group sum of squares for cluster $\mathcal{C}_k$ is defined as:

$$SS(k) = \sum_{i \in \mathcal{C}_k} ||\mathbf{x}_i - \boldsymbol{\mu}_k||^2$$

Please note that the term $||\mathbf{x}_i - \boldsymbol{\mu}_k||^2$ is the euclidean distance between $\mathbf{x}_i$ and $\boldsymbol{\mu}_k$.

If there are K clusters total then the "total within group sum of squares" is just the sum of all K of these individual $SS(k)$ terms.

3.4 Pair-counting measures

Given that we know the actual assignment labels for each data point we can attempt to analyze how well the clustering recovered this. WE will performance criteria which are based on two principles: i) two data points that belong to the same class should be assigned to the same cluster; and ii) two data points that belong to different classes should be assigned to different clusters. Formally speaking, consider all pairs of same-class data points, let p_1 be the percentage of pairs of which both data points were assigned to the same cluster. Consider all pairs of different-class data points, let p_2 be the percentage of pairs of which two data points are assigned to different clusters. Let p_3 be the average of these two values $p_3 = (p_1 + p_2)/2$, which summarizes the clustering performance.

3.5 Questions

When you have implemented the algorithm please report the following:

1. [8pts] The values of the total within group sum of squares and pair-counting measures (p_1, p_2, p_3) for $k = 2$, $k = 4$ and $k = 6$. Start your centers with the first k points in the dataset. So, if $k = 2$, your initial centroids will be ID 1 and ID 2, which correspond to the first two lines in the file.

2. [7pts] The number of iterations that k-means ran for $k = 6$, starting the centers as in the previous item. Make sure you count the iterations correctly. If you start with iteration $i = 1$ and at $i = 4$ the cluster assignments don't change, the number of iterations was 4, as you had to do step 2 four times to figure this out.

3. [7pts] A plot of the total within group sum of squares versus k for $k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$. Start your centers randomly (choose k points from the dataset at random).

4. [8pts] A plot of p_1, p_2, p_3 versus k for $k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$. Start your centers randomly (choose k points from the dataset at random).

For the last two items, you should run k-means algorithm several times (e.g., 10 times) and average the results. For each question, submit a single plot, which is the average of the runs.

4 Programming Question (scene classification) [30 points + 10 bonus]

This question gives you the opportunities to learn LibSVM, which is one of the best software tool for classification problem. You will train SVMs with different kernels and use them to classify scenes from The Big Bang Theory, your favorite TV series. To classify an image, you will use Bag-of-Word representation, which is one of the most popular image representation. This representation views an images as the histogram of image features, or visual words. You MUST use LibSVM and your K-means implementation from Question 3.

4.1 LibSVM installation

First download LibSVM `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`. Follow the instruction in README to install LibSVM for Matlab. Two main functions of LibSVM that you should pay attention to are: `svmtrain` and `svmpredict`. Note that Matlab also has a machine learning toolbox that comes with these two functions with exactly the same names. However, Matlab's SVM implementation is not as good as LibSVM, so you need to make sure that you are using svmtrain and svmpredict from LibSVM. To check if you have installed the program correctly, in Matlab do:

```
>> which svmtrain
>> which svmpredict
```

Matlab should return the paths to the `svmtrain` and `svmpredict` of LibSVM. To learn how to use these functions, type the names of the function in Matlab:

```
>> svmtrain
>> svmpredict
```

4.2 Data

Training and test images are provided in the subdirectory `bigbangtheory`. The training image ids and labels are given in `train.mat`. This file contains two variables: `imgIds` and `lbs`. `imgIds` is a column vector and each row has a name of image in the training set. `lbs` is a matrix denoting the label for the image with the corresponding index. There are total 8 classes for the dataset: living room (1), kitchen (2), hallway (3), Penny's living room (4), cafeteria (5), Cheesecake factory (6), laundry room (7), and comic bookstore (8).

Validation set is not provided for this question. You have to do cross validation to find the parameter for the best performance. You can implement cross validation by yourself, or you can use LibSVM functionality. Image ids for test set are given in `test.mat`.

4.3 Helper functions

A number of Matlab helper functions are provided. Also, some functions are left unfinished and you have to complete them.

1. Use `HW3_BoW.learnDictionary()` to learn visual vocabulary for scene representation. You have to fill your K-means implementation from Question 3 in this function.

2. Use `HW3_BoW.cmpFeatVecs()` to compute feature vectors. This function will return the histogram of visual words for each image, which is our feature vector.

4.4 What to implement?

1. Complete the function `HW3_BoW.learnDictionary()`. This function learns a visual vocabulary by running k-means on random image patches. Learn a visual dictionary with $K = 1000$ clusters.

2. (5 points) Using SVM with RBF kernel, report the 5-fold cross-validation accuracy for the default kernel parameters for C and γ.

3. (10 points) Tune C and γ for SVM with RBF kernel using 5-fold cross validation. Report the values of C, γ, and the 5-fold cross-validation accuracy.

4. Unfortunately, LibSVM does not support exponential $\mathcal{X}^2$ kernel, so you will need to implement it. Implement a function:
   `[trainK, testK] = cmpExpX2Kernel(trainD, testD, gamma)`

that takes train and test data and the kernel parameter gamma and return the train and test kernels. Recall that the exponential $\mathcal{X}^2$ kernel value for two d-dimensional feature vector $\mathbf{x}$ and $\mathbf{y}$ is:

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{\gamma}\sum_{i=1}^{d}\frac{(x_i - y_i)^2}{x_i + y_i + \epsilon}\right) \tag{1}$$

The ϵ term is added to avoid division by 0.

5. (10 points) LibSVM allows using pre-computed kernels. Train an SVM with exponential $\mathcal{X}^2$ kernel. Report the 5-fold cross validation accuracy with the best tuned values of C and γ. Note that a good default value of γ is the average of $\mathcal{X}^2$ distance between training data points, as discussed in the lecture.

6. (5 points) Use your model to predict on the test set. Save the predicted labels on a CSV file `predTestLabels.csv` (see 5.2 for the format). The order of predicted labels should correspond to the order of the reviews in `test.mat`. Submit this `predTestLabels.csv` file on Kaggle and report classification accuracy.

7. (10 bonus points) Submit `predTestLabels.csv` and enter our Kaggle competition for fame. You must use your Stony Brook email to register on Kaggle. We will maintain a leader board, and the top three entries at the end of the competition (due date) will receive 10 bonus points. The ranking is based on classification accuracy in your PDF submission file.

   To prevent exploiting test data, you are allowed to make a maximum of 2 submissions per 24 hours. Your submission will be evaluated immediately and the leader board will be updated.

5 What to submit?

5.1 Blackboard submission

You will need to submit both your code and your answers to questions on Blackboard. Do not submit the provided data. Put the answer file and your code in a folder named: SUBID_FirstName_LastName (e.g., *10947XXXX_heeyoung_kwon*). Zip this folder and submit the zip file on Blackboard. Your submission must be a zip file, i.e, SUBID_FirstName_LastName.zip. The answer file should be named: answers.pdf, and it should contain:

1. Answers to Question 1 and 2
2. Answers to Question 3.5, including the requested plots.
3. Answers to Question 4.4

5.2 Kaggle submission

For Questions 4.4.4 and 4.4.5, you must submit a CSV file to get the classification accuracy from the competition site `inclass.kaggle.com/c/hw3-scene-classification/`. A submission file should contain two columns: ImgId and Prediction. The file should contain a header and have the following format.

$$\begin{array}{cc} ImgId, & Prediction \\ 001778, & 2 \\ 001779, & 5 \\ \ldots & \ldots \end{array} \tag{2}$$

A sample submission file is available from the competition site and our handout.

6 Cheating warnings

Don't cheat. You must do the homework yourself, otherwise you won't learn. You must use your real name to register on Kaggle. Do not create multiple accounts to bypass the submission limitation per 24 hours. Doing so will be considered cheating.