Boston University
Department of Electrical and Computer Engineering

ENG EC 500 B1 (Ishwar) Introduction to Learning from Data

Matlab Exercise 1

Issued: Tue 22 Sep 2015 **Due:** Fri 9 Oct 2015 in box outside PHO440 + Blackboard
Required reading: Your notes from lectures and additional notes on website on classification.

- **Advise**: This homework assignment requires a large amount of time and effort. We urge you to start right away. This assignment cannot be finished by staying up all night just before the deadline.
- This homework assignment requires programming in MATLAB. If you are new to MATLAB programming, please refer to the following link for a primer:
`http://www.math.ucsd.edu/~bdriver/21d-s99/matlab-primer.html`
- You will be making two submissions: (1) A paper submission in the box outside PHO440. (2) An electronic submission of all your matlab code (in a single zipped file appropriately named as described below) to blackboard.
- **Paper submission:** This must include all plots, figures, tables, numerical values, derivations, explanations (analysis of results and comments), and also printouts of all the matlab .m files that you either created anew or modified. Submit color printouts of figures and plots whenever appropriate. Color printers are available in PHO307 and PHO305. Be sure to annotate figures, plots, and tables appropriately: give them suitable *titles* to describe the content, label the *axes*, indicate *units* for each axis, and use a *legend* to indicate multiple curves in the plots. Please also explain each figure properly in your solution.
- **Blackboard submission:** All the matlab .m files (and only .m files) that you either create anew or modify must be appropriately named and placed into a **single** directory which should be zipped and uploaded into the course website. Your directory must be named as follows: `<yourBUemailID>_matlab1`. For example, if your BU email address is `charles500@bu.edu` you would submit a single directory named: `charles500_matlab1.zip` which contains all the matlab code (and only the code).
- **File naming convention:** Instructions for file names to use are provided for each problem. As a general rule, each file name must begin with your BU email ID, e.g., `charles500_<filename>.m`. The file name will typically contain the problem number and subpart, e.g., for problem 1.1b, the file name would be `charles500_matlab1_1b.m`. Note that the dot . in 1.1 is replaced with an underscore (this is important).

Problem 1.1 Gaussian Discriminant Analysis:
In this problem we consider using various Gaussian Discriminant Analysis classifiers.

(a) Write 4 generalized MATLAB "helper" functions for training and testing a Quadratic Discriminant Analysis (QDA) classifier and a Linear Discriminant Analysis (LDA) classifier. We have provided 4 template MATLAB function files (which specifies what inputs and outputs you should use) for you to write, complete, and **submit**:

1. `QDA_train.m`: outputs a trained QDA classifier given input training data and labels
2. `QDA_test.m`: outputs labels given a trained QDA classifier and testing data
3. `LDA_train.m`: outputs a trained LDA classifier given input training data and labels
4. `LDA_test.m`: outputs labels given a trained LDA classifier and testing data

Rename these helper functions to: <yourBUemailID>_<helper file name>.m. Example, `charles500_QDA_train.m`. Comment your code as necessary. Do **not** use built-in MATLAB functions for discriminant analysis such as `fitcdiscr`.

Dimension of samples $\gg$ *the dimension of features*: Load the dataset provided in `data_iris.mat`. In this dataset, the rows of variable `X` correspond to samples (150 total), and the columns correspond to feature dimension (4 total). The elements of variable `Y` are the ground truth class labels for each data sample.

(b) Classification involves the following steps (1) splitting the data into a training set (in this problem 100 samples) and a test set (the remaining 50 samples), (2) learning a classifier on the training set, (3) predicting the class labels of the samples in the test set, and (4) calculating a performance metric.

Create one training set by choosing 100 samples (picked uniformly at random without replacement) from the entire dataset. Repeat this 10 times for 10 different training/test splits. For each split, conduct the remaining 3 steps (training, testing, performance evaluation). Calculate the Correct Classification Rates (CCR) and the Confusion Matrices using the generalized QDA and LDA functions you have implemented in part (*a*) for each training/testing split.
Report the following results:

- Mean vectors of training samples for each class (common to both LDA and QDA), averaged over 10 splits.
- The variances of all the 4 dimensions (the diagonal terms of the covariance matrix) for each class of training set (in QDA), averaged over 10 splits.
- The overall variances of all the 4 dimensions (in LDA), averaged over 10 splits. **Note:** all classes share the same covariance matrix in LDA.
- The mean and standard deviation of all 10 test CCRs.
- The confusion matrices of the two splits which have the best and the worst CCR (in LDA).
- **Submit** your script with name <yourBUemailID>`_matlab1_1b.m` that generates all above results.

Helpful MATLAB functions: `randperm, confusionmat`

Dimension of data samples $\ll$ *dimension of feature*: Load the dataset provided in `data_cancer.mat`. Rows of variable `X` correspond to data samples (216). The elements of variable `Y` are the corresponding ground truth class labels. In this dataset, the dimension of features (4000) is much larger than the number of samples (216). As a result, the estimates of the covariance matrices in LDA/QDA $\widehat{\Sigma}$ will be singular. One strategy to account for this is to consider a *Regularized* Discriminant Analysis (RDA) in which the common covariance matrix in LDA is replaced by,

$$\widehat{\Sigma}_{\text{reg}} = \lambda \text{diag}(\widehat{\Sigma}_{\text{LDA}}) + (1 - \lambda)\widehat{\Sigma}_{\text{LDA}} \tag{1}$$

where $\lambda \in [0, 1]$ controls the amount of regularization and $\text{diag}(\mathbf{A})$ is a matrix whose diagonal entries are the same as $\mathbf{A}$ where the off-diagonal entries are 0.

(c) Implement your code for RDA using the provided "helper" function templates `RDA_train.m` and `RDA_test.m`. Comment your code and submit the completed MATLAB functions. As before, rename these helper functions to: `<yourBUemailID>_<helper file name>.m`.
Helpful MATLAB function: `diag`

(d) Split the data into a training set (of size 150) and use the remaining as test data (of size $216-150 = 66$). Choose the 150 training samples uniformly at random from the entire dataset as you did in part (b). For each choice of λ in the range of $[0.1 : 0.05 : 1]$ (x-axis), calculate the training and test CCR (y-axis) and report them on the same plot. **Submit** your script with name `<yourBUemailID>_matlab1_1d.m` along with its corresponding figure. Make sure that if the grader runs your code, it will generate the exact same results that you are reporting. This can ensured by fixing the random seed. See MATLAB function: randstream.

Problem 1.2 Naive Bayes Text Document Classifiers: In this problem we classify text documents using Naive Bayes. We use the classic *20 newsgroup* dataset. Here, each document is labeled as one of 20 different classes and is provided in `data_20news.zip`. This directory contains the following files:

1. `vocabulary.txt`: a list of all the words that can possibly appear in the documents. The line number of a word is its ID (Word_ID).
2. `newsgrouplabels.txt`: the names of the 20 classes. The line number of a class (label) is its ID.
3. `train.data` and `test.data`: the words in each of the documents. Each line of each file is of the form: "Document_ID, Word_ID, Word_count". Word_count is the number of times word Word_ID appears in document Document_ID.
4. `train.label` and `test.label`: the labels of the training and testing documents.
5. `stoplist.txt`: a list of so-called "stop words" such as "the", "is", "on", etc. This will be used only in part (f).

Let $\{1, \ldots, W\}$ denote the W distinct words in the vocabulary. We view each document $\mathbf{x}_j$, $j = 1, \ldots, n$ as a collection of words $\{w_{1,j}, \ldots, w_{d_j,j}\}$ where d_j is the number of words in document $\mathbf{x}_j$ and y_j is the label of the document. Each word $w_{i,j}$ takes values in $\{1, \ldots, W\}$. The Naive Bayes classifier assumes that:

$$P(Y_j = c|\mathbf{x}_j) \propto P(Y_j = c) \prod_{i=1}^{d_j} P(w_{i,j}|Y = c), \quad c = 1, \ldots, 20 \tag{2}$$

We denote $P(Y_j = c) = \pi_c, c = 1, \ldots, 20$ and $P(w_{i,j} = w|Y = c) = \beta_{w,c}, w = 1, \ldots, W, c = 1, \ldots, 20$. Note that by definition, $\sum_{w=1}^{W} \beta_{w,c} = 1$, for each $c = 1 \ldots, 20$.

(a) Write a MATLAB script to load and parse the data. Name this file `<yourBUemailID>_matlab1_2a.m`. **Report** the following statistics

- The total number of **unique** words that appear in the training set, the testing set, and the entire dataset, respectively.
- The average document length (in terms of number of words) in the training and testing sets, respectively.
- The total number of unique words that appear in the testing set, but not in the training set.

Helpful MATLAB functions: `textread, sparse`.

We next train a Naive Bayes classifier. We first learn the parameters $\beta_{w,c}$'s using Maximum Likelihood Estimation (MLE):

$$\beta_{w,c} \propto n_{w,c} \tag{3}$$

where $n_{w,c}$ is the total number of the word w that appears in documents with the label c.

(b) Train a Naive Bayes classifier using the MLE rule from Eq. 3 and the training set. Afterwards, evaluate this classifier on the test set. Do not use the built-in MATLAB function for the Naive Bayes classifier. **Submit** your script with name `<yourBUemailID>_matlab1_2b.m`. **Report** the following results:

- Among the $W \times 20$ estimated parameters (the $\beta_{w,c}$'s), how many of them are non-zero?
- For some test documents, $P(Y = c|\mathbf{x}) = 0$ for all $c = 1, \ldots, 20$. What is the total number of such test documents? Can you explain why their probabilities are zero?
- The test CCR.

(c) One strategy to overcome the issue encountered in part (b) is to remove words that only appear in the testing documents (but not in the training documents). Remove these words. Repeat training and testing using the MLE rule in (b) and **report:**

- The total number of non-zero estimated $\beta_{w,c}$'s.
- The testing CCR.

To learn the parameters $\beta_{w,c}$'s we can also use the Maximum Aposteriori Probability (MAP) rule. It is common practice to impose a Dirichlet prior on the $\beta_{w,c}$'s and as such, Eq. 3 becomes:

$$\beta_{w,c} \propto n_{w,c} + \alpha \tag{4}$$

for some small $\alpha > 0$.

(d) Train a Naive Bayes classifier using the MAP rule in Eq. 4 with $\alpha = 1/W$, and evaluate it on the test set. Do not use the built-in MATLAB function for the Naive Bayes classifier. **Submit** your script with name `<yourBUemailID>_matlab1_2d.m`. **Report** the following results:

- The test CCR.
- The 20×20 confusion matrix. Make sure you annotate the confusion matrix appropriately.

(e) Calculate the test error for different choices of α's in the range $10^{-5}, 10^{-4.5}, 10^{-4}, 10^{-3.5}, ..., 10^{1}, 10^{1.5}$. Plot the test CCRs (y-axis) as functions of α (x-axis). Use a log-scale for α.

(f) In classification tasks with text observation, the most common words (*stop words*) such as "the","is", etc., may be less informative. A common practice is to remove such stop words. The file `stopword.txt` contains a list of such stop words. Remove these words from all the document and the vocabulary and then repeat the steps in part (*d*). Report:

- The size of the new dictionary (the new value of W) after removing the stop words.
- The average document length (in terms of number of words) in the training and test sets respectively.
- The test CCR.

Problem 1.3 Nearest Neighbor Classifier: In this problem we will apply a k- Nearest Neighbor classifier based on the Euclidean distance to two datasets. First, we consider a simulated data set provided in `data_knnSimulation.mat`. The rows of variable `Xtrain` are data points and each data point has 2 feature attributes.

(a) Create a scatter plot of all the training data (with the first column as the x-axis and the second column as the y-axis). Color the data points in the 1st, 2nd, and 3rd classes with red, green, and blue colors, respectively.
Helpful MATLAB function: `gscatter`

(b) For each point on this 2-D grid $[-3.5 : 0.1 : 6] \times [-3 : 0.1 : 6.5]$, calculate its probability of being class 2 using $k = 10$ nearest neighbors. Plot the probabilities of these points on a 2-D colored map using the MATLAB default colormap. Repeat this for class 3. **Submit** the two figures. Add a colorbar in the figures to indicate the color range.
Helpful MATLAB functions: `imagesc, contourf, colormap, colorbar`.

(c) For all the points on the same 2-D grid used in part (b), predict its class label using a k-NN classifier with $k = 1$. Color-code the decisions for each grid point using the same color coding scheme used in part (a). Repeat this for $k = 5$. **Submit** the two figures. Comment on their differences.

We next look at the classical handwritten digit recognition problem. The dataset is provided in `data_mnist_train.mat` and `data_mnist_test.mat`. Each data point represents a 28×28 grayscale image for hand written digits from 0 to 9. To visualize this data, for example, the 200-th training image, you can use `imshow(reshape(X_train(200,:), 28,28)')`.

(d) Apply a 1-Nearest Neighbor classifier to this dataset. **Report** the test CCRs and the confusion matrix. **Submit** your script with name `<yourBUemailID>_matlab1_4d.m`. Your script should be able to display in the command window the CCR that you reported.
Hint: Try to make your code time efficient. The dataset is large.