Boston University
Department of Electrical and Computer Engineering

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

Matlab Exercise 2

Issued: Tue 20 Oct 2015 **Due:** 4pm Tue 3 Nov 2015 PHO440 box + 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 to blackboard (in a single zipped file appropriately named as described below).
- **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>_matlab2`. For example, if your BU email address is `mary567@bu.edu` you would submit a single directory named: `mary567_matlab2.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., `mary567_<filename>.m`. The file name will typically contain the problem number and subpart, e.g., for problem 2.1b, the file name would be `mary567_matlab2_1b.m`. Note that the dot . in 2.1 is replaced with an underscore (this is important).

Problem 2.1 San Francisco Crime Prediction: (~1–2hrs run time)
In this problem we will work with a recent real-world dataset that can be downloaded from the following website: `https://www.kaggle.com/c/sf-crime`. This dataset is hosted by Kaggle as part of a competition for the machine learning community to use for fun and practice. The competition started at 8:51 pm on Tuesday 2 June 2015 UTC and ends at 11:59 pm, Monday 6 June 2016 UTC (370 total days). The dataset contains information about various incidents ranging from January 2003 to March 2015 derived from the San Francisco Police Department Crime Incident Reporting system. The raw data has been pre-processed into two MATLAB .mat files: `data_SFcrime_train.mat` and `data_SFcrime_test.mat`. Each incident contains the following attributes:

- **Date**: The time the crime occurred. We will only extract the "Hour" (in 24-hour format).
- **Category**: The type of crime. This is the class label and is only available for the training set.
- **DayOfWeek**: Weekday of crime: Sunday, Monday, Tuesday, etc.
- **PdDistrict**: Police department district.
- **Address**: Street address of crime. We are not going to use this attribute in this MATLAB exercise.
- **X** and **Y**: GPS coordinates (**X** = Longitude, **Y** = Latitude) of the crime location. We are not going to use this attribute in this MATLAB exercise. You can, however, use them to visualize the incidents on a map of San Francisco.

(a) In this exercise we are going to make use of Hour (extracted from Date), DayOfWeek, and PdDistrict to build a classifier. We will treat all of them as *categorical* variables even though some of them like Hour and DayOfWeek have some quantitative and ordinal aspects to them. We will convert these three categorical variables into real-valued vectors as follows. The variable DayOfWeek will be represented as a 7-dimensional vector with "Sunday"= $[1, 0, 0, 0, 0, 0, 0]$, "Monday"= $[0, 1, 0, 0, 0, 0, 0]$, and so on. Similarly, the Hour variable will be represented as a 24-dimensional vector of zeros with a single one in the slot corresponding to the hour. The PdDistrict variable will likewise be a 10-dimensional vector of zeros with a single one in the slot corresponding to the police district (there are 10 police districts in the dataset). We will then concatenate all three of these vectors (Hour, DayOfWeek, PdDistrict) into one long binary feature vector of length $24 + 7 + 10 = 41$. Perform these pre-processing steps for both the training and the test dataset.

   (i) **Submit** your script for processing the categorical features with name `<yourBUemailID>_matlab2_1a.m`.

   (ii) **Plot** three histograms (one for Hour, one for DayOfWeek, and one for PdDistrict) for all the incidents contained in the file `data_SFcrime_train.mat`.

   (iii) Based on the incidents in `data_SFcrime_train.mat`, **find and report** the most likely hour of occurrence of each type of crime.

   (iv) Based on the incidents in `data_SFcrime_train.mat`, **find and report** the most likely type of crime within each PdDistrict.

   Visualization aid: For the purpose of visualization, we have provided you with a map visualization tool: `plot_google_map.m`. If you discover any interesting patterns or create any informative figures for visualization, please include them in your report.

(b) ℓ_2-regularized *multi-class logistic regression classifier*: Now each incident is represented as a 41-dimensional vector $\mathbf{x}_i$ whose elements are either 0 or 1 (viewed as real numbers). Write a MATLAB script to implement the gradient descent algorithm to learn the parameters $\theta := \{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$ of an ℓ_2-regularized multi-class logistic regression. Recall that the negative log-likelihood function of logistic regression for m classes and its gradients are given by:

$$\mathrm{NLL}(\theta) = \sum_{j=1}^{n} \ln\left(\sum_{k=1}^{m} e^{\mathbf{w}_k^\top \mathbf{x}_j}\right) - \sum_{k=1}^{m} \mathbf{w}_k^\top \left(\sum_{j=1}^{n} 1(y_j = k)\mathbf{x}_j\right)$$

$$\nabla_{\mathbf{w}_y} \mathrm{NLL}(\theta) = \sum_{j=1}^{n} \left(\frac{e^{\mathbf{w}_y^\top \mathbf{x}_j}}{\sum_{k=1}^{m} e^{\mathbf{w}_k^\top \mathbf{x}_j}} - 1(y_j = y)\right)\mathbf{x}_j, \quad y = 1, \ldots, m.$$

The ℓ_2-regularized objective function to be minimized and its gradients are given by:

$$f(\theta) := \mathrm{NLL}(\theta) + \frac{\lambda}{2}\sum_{k=1}^{m} \|\mathbf{w}_k\|_2^2, \quad \lambda > 0,$$

$$\nabla_{\mathbf{w}_y} f(\theta) = \nabla_{\mathbf{w}_y}\mathrm{NLL}(\theta) + \lambda \mathbf{w}_y, \quad y = 1, \ldots, m.$$

The pseudocode of the gradient descent algorithm appears below.

```
Initialize: $\theta_0 := \{\mathbf{w}_1^{(0)}, \ldots, \mathbf{w}_m^{(0)}\}$
For $t = 1, 2, \ldots, t_{\max}$, do:
  Evaluate gradients: $\nabla_{\mathbf{w}_k} f(\theta_t), k = 1, \ldots, m$
  Update weights: $\mathbf{w}_k^{(t+1)} = \mathbf{w}_k^{(t)} - \eta_t \nabla_{\mathbf{w}_y} f(\theta_t), k = 1, \ldots, m$
Endfor
```

Choose a fixed step size η (recommended: 10^{-5}) and initialize all $\mathbf{w}_k$'s to be zero. We recommend using $\lambda = 1000$. Use the first 60% of data samples in `data_SFcrime_train.mat` as your training set and the remaining 40% as the "test" set.

(i) **Plot** the value of the objective function $f(\theta_t)$ (y-axis) against the number of iterations t (x-axis) for $t = 1, 2, \ldots, t_{\max} = 1000$.

(ii) **Plot** the test *logloss* and the test CCR versus the number of iterations respectively. The *logloss* on the "test" set is defined as:

$$\text{logloss} = -\frac{1}{n_{\text{test}}}\sum_{j=1}^{n_{\text{test}}} \log p(y_j|\mathbf{x}_j, \theta)$$

where $\mathbf{x}_j$ is the test data point and y_j is its ground truth label, and $p(y|\mathbf{x}, \theta) = \frac{e^{\mathbf{w}_y^\top \mathbf{x}}}{\sum_{k=1}^m e^{\mathbf{w}_k^\top \mathbf{x}}}$. In order to avoid taking the logarithm of extremely small numbers, any value $p(y_j|\mathbf{x}_j, \theta)$ that is smaller than 10^{-10} should be treated as 10^{-10}.

(c) (**Bonus**) **Cross-validate** the regularization parameter λ using all the data in `data_SFcrime_train.mat`. **Report** the best value of λ. **Predict** the labels for the **real** test data points in `data_SFcrime_test.mat` (the ground truth labels for the real test data have been withheld by the competition). **Submit** your results to the original competition website and find out and **report** the real test error from the website. Report the strategy that you used and your results on the leader board of the competition. **Attach** screen-shots of your submission to the website and the leaderboard.

Problem 2.2 SVM Classifier for Text Documents: (~1–2hrs run time)
In this problem we will use SVMs to classify documents in the 20 Newsgroup dataset which we have encountered before in Matlab Exercise 1.2. The dataset provided in `data_20news.zip` contains the following files:

1. `vocabulary.txt` : a list of all the words that can appear in the documents. The line number is the Word_ID.

2. `newsgrouplabels.txt` : the names of the 20 classes. The line number is the label_ID.

3. `train.label` and `test.label`: the labels of all the training and test documents.

4. `train.data` and `test.data` : files containing words along with their counts appearing in training and test documents. Each line in the files is of the form: "Document_ID, Word_ID, word count". Word count is the number of times the word: Word_ID appears in document: Document_ID.

5. `stoplist.txt` : a list of commonly used "stop words" such as "the", "is", "on", etc.

For this problem, we will **remove** all the stop words from the original documents. We represent the j-th document as a **word-frequency vector** $\mathbf{x}_j \in \mathbb{R}^W$ over the entire vocabulary of size W: $\mathbf{x}_{w,j} = n_{w,j}/n_j$ where $n_{w,j}$ is the number times word w appears in document j and n_j is the total number of words in document j. We will use the following built-in MATLAB implementations for SVM training and testing: `svmtrain` and `svmpredict`.

Binary Classification:

(a) Train a *binary* SVM classifier using the `linear` kernel to distinguish between the following two classes: *alt.atheism* (class 1) and *talk.religion.misc* (class 20). Here the so-called "`boxconstraint`" parameter C is unspecified and needs to be determined via cross-validation. This is typically done through an exhaustive grid-search. Create one 5-fold cross-validation (CV) partitioning by splitting the training data into 5 equal-sized disjoint parts uniformly at random. Then for different choices of C's in the range $2^{-5}, 2^{-4}, \ldots, 2^{15}$, calculate the CV-CCR. For one value of C, the CV-CCR is calculated by first successively training on 4 out of the 5 folds and testing on the remaining one to obtain a CCR value, and then averaging the 5 CCR values (one CCR for each test fold).

   (i) **Plot** of the CV-CCR (y-axis) as function of C (x-axis, use log-scale).
   (ii) **Report** the best C^*, i.e., the value of C that achieves the maximum CCR.
   (iii) Use C^* to train a `linear` SVM classifier on all the training data, and test it on the test dataset. **Report** the test CCR.
   (iv) **Submit** your script with name `<yourBUemailID>_matlab2_2a.m`.

   Recommendations: Set 'autoscale' to be 'false' so that svmtrain does not automatically re-scale each feature dimension. For the 'boxconstraint' in MATLAB, set it to be a vector $C * \texttt{ones}(n, 1)$ where n is the number of training samples.

(b) Train a *binary* SVM classifier using the `RBF` kernel to distinguish between the *alt.atheism* and *talk.religion.misc* classes. Here you will have to determine two free parameters: the "`boxconstraint`" C and the "`rbf-sigma`". You should choose the best pair of parameters through an exhaustive search on a 2-D grid. As described in part (a), create a 5-fold cross-validation partitioning and calculate the CV-CCR for all combinations of values of the `boxconstraint` (values: $2^{-5}, 2^{-4}, \ldots, 2^{15}$) and `rbf-sigma` (values: $2^{-13}, 2^{-13}, \ldots, 2^{3}$).

   (i) **Plot** the CV-CCRs as a function of the `boxconstraint` (x-axis) and the `rbf-sigma` (y-axis) using a 2-D contour plot. Use a log-scale for both the `boxconstraint` and `rbf-sigma`. Remember to include the colorbar in your figure.
   (ii) **Report** the best (`boxconstraint`, `rbf-sigma`) pair.
   (iii) Use the best pair of parameters that you found in the previous part to train an `RBF` kernel SVM classifier on the entire training set and test it on the test dataset. **Report** the test CCR.
   (iv) **Submit** your script with name `<yourBUemailID>_matlab2_2b.m`.

Useful MATLAB functions: `cvpartition, contour, colorbar`.
Recommendations: Set 'autoscale' to be 'false'. For the 'boxconstraint' in MATLAB, set it to be a vector $C * \texttt{ones}(n, 1)$ where n is the number of training samples. Be patient; it might take a long time.

(c) We will now train a *binary* SVM (using a `linear` kernel) to separate class #17 (positive samples) from the remaining classes (the remaining 19 classes taken together are the negative samples). One practical issue here is that the number of samples in the positive class is far less than the number of samples in the negative class, i.e., the dataset is inherently unbalanced. Recall from class that the SVM solution can be reformulated as the unconstrained minimizer of the following objective function:

$$\min_{\mathbf{w},b}\left[\frac{1}{2}\|\mathbf{w}\|_2^2 + C_+ \sum_{j:y_j=``+"} (1 - y_j(\mathbf{w}^\top \mathbf{x}_j + b))_+ + C_- \sum_{j:y_j=``-"} (1 - y_j(\mathbf{w}^\top \mathbf{x}_j + b))_+\right]$$

As discussed in the class, the training objective function will be dominated by samples from the negative class if we follow the standard setting where $C_+ = C_- = C$. A commonly used practical strategy to mitigate this problem is to set different penalties C_+ and C_- for different classes. Specifically, $C_+ = n \times C/n_+$ and $C_- = n \times C/n_-$, where n_+ and n_- are number of positive and negative samples and $n = n_+ + n_-$ is the total number of training samples. Use this strategy and the same setting as in part (a), i.e., use a linear kernel and set autoscale = false. **Report** the same results as required in part (a). **In addition, report** the 2×2 confusion matrix on test data. **Comment on** what you observe about the confusion matrix.
Useful facts: if you set the '`boxconstraint`' to be a *scalar* in `svmtrain`, then C_+ and C_- are automatically scaled as described above by default. Parts (c) and (d) can be run together to save time.

(d) The CCR in part (c) might not be a good performance metric since the dataset is highly unbalanced. For the same setting as part (c):

(i) **Plot** the cross-validation **precision**, **recall**, and **F-score** as functions of the regularization (penalty) parameter C (on the same plot).

(ii) **Report** the best values of C in terms of both **recall** and **F-score**. Also **report** the confusion matrices associated with these two best values of C.

One-versus-one (OVO) multi-class classification: We will next build a multi-class SVM classifier using the following strategy: Train $m(m-1)/2$ binary SVMs for all the class pairs. At test time, apply all these $m(m-1)/2$ binary SVMs to the test sample and predict the label as the one with the most votes.

(e) Apply OVO to the entire 20 News Group dataset using the `linear` kernel with the *default* regularization parameter C in MATLAB.

(i) **Report** the overall CCR.
(ii) **Report** your training and test times (use MATLAB functions `tic,toc` for this part).
(iii) **Report** the confusion matrix on the test set.
(iv) **Submit** your script with name <`yourBUemailID`>`_matlab2_2e.m`

(f) Apply OVO to the entire 20 News Group data using the `rbf` kernel with the *default* regularization parameters C, σ in MATLAB.

(i) **Report** the overall CCR.
(ii) **Report** your training and test times (use MATLAB functions `tic,toc`).
(iii) **Report** the confusion matrix on the test set.
(iv) **Submit** your script with name <`yourBUemailID`>`_matlab2_2f.m`