CSCI 5525: Machine Learning (Fall'16)
Homework 1, Due 09/30/16 11:55pm

1. **(15 points)** The expected loss of a function $f(x)$ in modeling y using loss function $\ell(f(x), y)$ is given by

$$E_{(x,y)}[\ell(f(x), y)] = \int_x \int_y \ell(f(x), y) p(x, y) dy dx = \int_x \left\{ \int_y \ell(f(x), y) p(y|x) dy \right\} p(x) dx \ .$$

   (a) (7 points) What is the optimal $f(x)$ when $\ell(f(x), y) = (f(x) - y)^2$.

   (b) (8 points) What is the optimal $f(x)$ when $\ell(f(x), y) = |f(x) - y|$, where $|\cdot|$ represents absolute value.

2. **(15 points)** Consider a 2-class classification problem with features $\mathbf{x} \in \mathbb{R}^d$ and labels $y \in \{-1, +1\}$ and $(\mathbf{x}, y) \sim D$, where D is a fixed (but unknown) distribution on $\mathbb{R}^d \times \{-1, +1\}$. Assume $p(y = +1) = \frac{3}{4}, p(y = -1) = \frac{1}{4}$. Consider the classifier given by

$$f^*(\mathbf{x}) = \begin{cases} +1 \ , & \text{if } P(y = +1|\mathbf{x}) \geq \frac{1}{2} \\ -1 \ , & \text{otherwise.} \end{cases}$$

   The error-rate of the classifier is $L(f^*) = P(f^*(\mathbf{x}) \neq y) = E_{(\mathbf{x},y)\sim D}[\mathbb{1}(f^*(\mathbf{x}) \neq y)]$, where $\mathbb{1}$ is the indicator function (for 0-1 loss), and $L(\cdot)$ is the expected 0-1 loss or true error rate. Professor Bays claims the following:

   For any classifier $f : \mathbb{R}^d \mapsto \{-1, +1\}$, we have $L(f^*) \leq L(f)$.

   Is Professor Bays correct? Clearly justify your answer with technical details. (A simple yes/no answer will receive zero credit. You have provide a detailed and correct justification.)

Programming assignments: The next two problems involve programming. We will be considering two datasets for these assignments:

(a) `Boston`: The Boston housing dataset comes prepackaged with scikit-learn. The dataset has 506 points, 13 features, and 1 target (response) variable. You can find more information about the dataset here:
https://archive.ics.uci.edu/ml/datasets/Housing

While the original dataset is for a regression problem, we will create two classification datasets for the homework. Note that you only need to work with the `target` t to create these classification dataset, the `data` X should not be changed.

   i. `Boston50`: Let τ_{50} be the median (50th percentile) over all t (response) values. Create a 2-class classification problem such that $y = 1$ if $t \geq \tau_{50}$ and $y = 0$ if $t < \tau_{50}$. By construction, note that the class priors will be $p(y = 1) \approx \frac{1}{2}, p(y = 0) \approx \frac{1}{2}$.

ii. `Boston75`: Let τ_{75} be the 75th percentile over all t (response) values. Create a 2-class classification problem such that $y = 1$ if $t \geq \tau_{75}$ and $y = 0$ if $t < \tau_{75}$. By construction, note that the class priors will be $p(y = 1) \approx \frac{1}{4}, p(y = 0) \approx \frac{3}{4}$.

(b) `Digits`: The digits dataset comes prepackaged with scikit-learn. The dataset has 1797 points, 64 features, and 10 classes corresponding to ten numbers $0, 1, \ldots, 9$. The dataset was (likely) created from the following dataset:
http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

***These two datasets are provided on Moodle course page in csv format. The last column shows target values. A target value is either discrete (e.g., class label) or continuous (e.g., house price) and it is what we are trying to predict or classify. All other columns are data values. Each row is an input point.**

3. **(30 points)** We consider Fisher's linear discriminant analysis (LDA) for this problem. Train and evaluate the following classifiers using 10-fold cross-validation:

   (i) (10 points) For the `Boston50` dataset, apply LDA in the general case, i.e., both S_B and S_W computed from the data, to project data to $\mathbb{R}$ (one dimension), followed by suitable thresholding to classify points.

   (5 points) Using LDA, can you project the `Boston50` data to $\mathbb{R}^2$ (two dimensions)? Clearly justify your answer with technical details. (A simple yes/no answer will receive zero credit. You have provide a detailed and correct justification.)

   (ii) (15 points) For the `Digits` dataset, apply LDA in the general case, i.e., both S_B and S_W computed from the data, to project data to $\mathbb{R}^2$ (two dimensions), followed by bi-variate Gaussian generative modeling to do 10-class classification, i.e., by estimating and using class priors π_k and parameters $(\mu_k, \Sigma_k), k = 1, \ldots, 10$.

   You will have to submit (a) **summary of methods and results** report and (b) **code** for each algorithm:

   (a) **Summary of methods and results**: Briefly describe the approaches in (i) and (ii) above, along with relevant equations. Also, report the training and test set error rates and standard deviations from 10-fold cross validation for the methods on the datasets.

   (b) **Code**: For part (i), you will have to submit code for `LDA1dThres(filename,num_crossval)` (main file). This main file has **input**: (1) a filename including extension (exactly the same as your downloaded file) and absolute path (e.g., in Unix: /home/mywork/Boston50.data). The content of input data must be exactly the same as the downloaded file. (2) the number of folds for cross-validation, and **output**: (1) the training and test set error rates and standard deviations printed to the terminal (stdout).
   For part (ii), you will have to submit code for `LDA2dGaussGM(filename,num_crossval)`, with all other guidelines staying the same.

4. **(40 points)** The goal is to evaluate the results reported in the paper "On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes" by A. Ng and M. Jordan, using the `Boston50`, `Boston75`, and `Digits` datasets. You will train and evaluate two classifiers:

(i) (20 points) Logistic regression (LR), and

(ii) (20 points) Naive-Bayes with marginal Gaussian distributions (GNB)

on all three datasets. Evaluation will be done using 10 random class-specific 80-20 train-test splits, i.e., for each class, pick 80% of the data at random for training, train a classifier using training data from all classes, use the remaining 20% of the data from each class as testing, and repeat this process 10 times. We will be doing a learning curve, similar to the Ng-Jordan paper—please see guidelines below.

You will have to submit (a) **summary of methods and results** report and (b) **code** for each algorithm:

(a) **Summary of methods and results**: Briefly describe the approaches in (i) and (ii) above, along with (iterative) equations for parameter estimation. Clearly state which method you are using for logistic regression. For each dataset and method, create a plot of the test set error rate illustrating the relative performance of the two methods with increasing number of training points (see instructions below). The plots will be similar in spirit to Figure 1 in the Ng-Jordan paper, along with error-bars with standard deviation of the errors.

Instructions for plots: Your plots will be based on 10 random 80-20 train-test splits. For each split, we will always evaluate results on the same test set (20% of the data), while using increasing percentages of the training set (80% of the data) for training. In particular, we will use the following training set percentages: [10 25 50 75 100], so that for each 80-20 split, we use 10%, 25%, all the way up to 100% of the training set for training, and always report results on the same test set. We will repeat the process 10 times, and plot the mean and standard deviation (as error bars) of the test set errors for different training set percentages.

(b) **Code**: For logistic regression, you will have to submit code for `logisticRegression(filename, num_splits, train_percent)`. This main file has **input**: (1) a filename including extension (exactly the same as your downloaded file) and absolute path (e.g., in Unix: /home/mywork/digits.data), (2) the number of 80-20 train-test splits for evaluation, (3) and a vector containing percentages of training data to be used for training (use [10 25 50 75 100] for the plots), and **output**: (1) test set error rates for each training set percent printed to the terminal (stdout). The test set error rates should include both the error rates for each split for each training set percentage as well as the mean of the test set error rates across all splits for each training set percentage (print the mean error rates at the end).

For naive Bayes, you will have to submit code for `naiveBayesGaussian(filename, num_splits, train_percent)`, with all other guidelines staying the same.

Additional instructions: Code can only be written in Python or Matlab; no other programming languages will be accepted. One should be able to execute all programs from matlab/python command prompt or the code can be a jupyter notebook. Your code must be runnable on a CSE lab machine (e.g., csel-kh1260-01.cselabs.umn.edu). Please specify instructions on how to run your program in the README file. Information on the size of the datasets, including number of data points and dimensionality of features, as well as number of classes can be readily extracted from the dataset text file.

Each function must take the inputs in the order specified in the problem and display the textual output via the terminal. The input data file for your function must be exactly the same as the original downloaded file, which will be used as input for grading.

For each part, you can submit additional files/functions (as needed) which will be used by the main file. In your code, you **cannot** use machine learning libraries such as those available from scikit-learn for learning the models or for cross-validation. However, you may use libraries for basic matrix computations. Put comments in your code so that one can follow the key parts and steps in your code.

Instructions

Follow the rules strictly. If we cannot run your code, you will not get any credit.

- **Things to submit**
  1. hw1.pdf: A document which contains the solution to Problems 1, 2, 3, and 4 which including the summary of methods and results.
  2. LDA1dThres and LDA2dGauss: Code for Problem 3.
  3. logisticRegression and naiveBayesGauss: Code for Problem 4.
  4. README.txt: README file that contains your name, student ID, email, instructions on how to compile (if necessary) and run your code, any assumptions you are making, and any other necessary details.
  5. Any other files, except the data, which are necessary for your code.