CSCI 5525: Machine Learning (Fall 2020)

Homework 1

Due 10/1/2020 11:59 PM CDT

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. **(10 points)** A generalization of the least squares problem adds an affine function to the least squares objective,

$$\min_{\mathbf{w}} \|\mathbf{A}\mathbf{w} - \mathbf{b}\|_2^2 + \mathbf{c}^\top \mathbf{w} + d$$

   where $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{w} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n, d \in \mathbb{R}$. Assume the columns of $\mathbf{A}$ are linearly independent. This generalized problem can be solved by reducing it to a standard least squares problem, using a trick called *completing the square*.

   Show that the objective of the problem above can be expressed in the form

$$\|\mathbf{A}\mathbf{w} - \mathbf{b}\|_2^2 + \mathbf{c}^\top \mathbf{w} + d = \|\mathbf{A}\mathbf{w} - \mathbf{b} + \mathbf{f}\|_2^2 + g$$

   where $\mathbf{f} \in \mathbb{R}^m, g \in \mathbb{R}$. Then solve the generalized least squares problem by finding the $\mathbf{w}$ that minimizes $\|\mathbf{A}\mathbf{w} - (\mathbf{b} - \mathbf{f})\|_2^2$.

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 data points, 13 features, and 1 target (response) variable. You can find more information about the dataset here: `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html`.

   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.

   First, load the dataset in with the following commands:

```
import sklearn as sk
X, t = sk.datasets.load_boston(return_X_y=True)
```

Then, create the two following data sets.

i. `Boston50`: Let τ_{50} be the median (50th percentile) over all t (response) values. Create a 2-class classification problem such that one class corresponds to label $y = 1$ if $t \geq \tau_{50}$ and the other class corresponds to label $y = 0$ 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 one class corresponds to label $y = 1$ if $t \geq \tau_{75}$ and the other class corresponds to label $y = 0$ 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 data points, 64 features, and 10 classes corresponding to ten numbers $0, 1, \ldots, 9$. You can find more information about the dataset here: `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html`.

3. **(35 points)** In this problem, we consider Fisher's linear discriminant analysis (LDA) for this problem. Implement[1], train, and evaluate the following classifiers using 10-fold cross-validation:

   (i) **(15 points)** For the `Boston50` dataset, apply LDA in the general case, i.e., compute both the between-class and within-class covariance matrices S_B and S_W, respectively, from the training data, project the data onto $\mathbb{R}$ (one dimension), and then find a suitable threshold (one that minimizes classification error) to classify the training samples correctly.

   (ii) **(20 points)** For the `Digits` dataset, apply LDA in the general case, i.e., compute S_B and S_W from the data, project the data to $\mathbb{R}^2$ (two dimensions), then use bi-variate Gaussian generative modeling to do 10-class classification, i.e., estimate and use 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(num_crossval)` (main file). This main file has **input**: the number of folds for cross-validation, and **output**: 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(num_crossval)`, with all other guidelines staying the same.

[1] You must implement all algorithms in this homework from scratch; you cannot use toolboxes like scikit-learn.

4. **(40 points)** In this problem, 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[2], using the `Boston50`, `Boston75`, and `Digits` datasets. Implement, 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 creating 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(num_splits, train_percent)`. This main file has **input**: 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**: 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(num_splits, train_percent)`, with all other guidelines staying the same.

Additional instructions: Code can only be written in Python 3.6+; no other programming languages will be accepted. One should be able to execute all programs from the Python command prompt or terminal. Please specify instructions on how to run your program in the README file.

[2] `https://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf`

Each function must take the inputs in the order specified in the problem and display the textual output via the terminal and plots/figures should be included in the report.

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.

Your code must be runnable on a CSE lab machine (e.g., csel-kh1260-01.cselabs.umn.edu). One option is to SSH into a machine. Learn about SSH at these links: `https://cseit.umn.edu/knowledge-help/learn-about-ssh`, `https://cseit.umn.edu/knowledge-help/choose-ssh-tool`, and `https://cseit.umn.edu/knowledge-help/remote-linux-applications-over-ssh`.

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 solutions to Problems 1, 2, 3, and 4, which including the summary of methods and results.
  2. `LDA1dThres` and `LDA2dGaussGM`: Code for Problem 3.
  3. `logisticRegression` and `naiveBayesGaussian`: Code for Problem 4.
  4. README.txt: README file that contains your name, student ID, email, instructions on how to 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.

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems, but each person must write up the final solutions individually. You need to list in the README.txt which problems were a collaborative effort and with whom. (2) Regarding online resources, you should **not**:

- Google around for solutions to homework problems,
- Ask for help on online,
- Look up things/post on sites like Quora, StackExchange, etc.