HW 2

CS6510: Applied Machine Learning
IIT-Hyderabad
Aug-Nov 2016

Max Points: 60
Due: 10th Oct 2016 11:59 pm (No extensions)

Instructions

- Please use Google Classroom to upload your submission by the deadline mentioned above. Your submission should comprise of a single ZIP file with all your solutions, including code.
- For late submissions, 10% is deducted for each day (including weekend) late after an assignment is due. Note that each student begins the course with 5 grace days for late submission of assignments. Late submissions will automatically use your grace days balance, if you have any left. You can see your balance on the CS6510 Marks and Grace Days document under the course Google drive.
- You should use PYTHON for the programming assignments.
- Please read the department plagiarism policy. Do not engage in any form of cheating - strict penalties will be imposed for both givers and takers. Please talk to instructor or TA if you have concerns.

1 Theory [30 points]

1. **($5 \times 2 = 10$ points)** Let k_1 and k_2 be valid kernel functions. Comment about the validity of the following kernel functions:

(a) $k(x, z) = k_1(x, z) + k_2(x, z)$

(b) $k(x, z) = k_1(x, z)k_2(x, z)$

(c) $k(x, z) = h(k_1(x, z))$ where h is a polynomial function with positive co-efficients

(d) $k(x, z) = \exp(k_1(x, z))$

(e) $k(x, z) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{z}\|_2^2}{\sigma^2}\right)$

2. **(2 + 2 = 4 points)** The XOR-problem is not linearly seperable and hence, not implementable by a single perceptron. Here is the truth table for the XOR operation:

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	1

(a) Construct a 2-layer perceptron with 2 inputs, 4 hidden units, and 1 output. Each hidden unit codes for one particular input in the truth table. Use the outputs of the (sparse) hidden neurons as inputs to the output perceptron. We allow negative weights. As the nonlinear activation function, the following simple step function should be used:

$$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 1 \end{cases}$$

(b) Can you think of a 2-layer perceptron with only 2 hidden units, that is solving the problem ?

3. **(4 points)** Let X be a set of n linearly separable features. The objective is to find θ and b, such that $|\theta^T X + b|_i > 1$ for all $i \in \{1, \cdots, n\}$. Let $(\theta, b) = (\theta_0, b_0)$ satisfy the above objective. Show that if the RHS of the above objective is replaced by an arbitrary $\gamma > 0$, then the solution does not change.

4. **(12 points)** We all know that Support Vector Machines (SVMs) can be applied for the regression setting also. In this question, you have to derive the dual form of SV regression (SVR). A frequently used loss function for regression is the epsilon sensitive loss:

$$L_\epsilon(x, y, f) = |y - f(x)|_\epsilon = \max(0, |y - f(x)| - \epsilon)$$

Here, x is the input, y is the output, and f is the function used for predicting the label. Using this notation, the SVR cost function is defined as:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} L_\epsilon(x_i, y_i, f)$$

where f(x) = $w^T x$, and $C, \epsilon > 0$ are parameters

(a) Introduce appropriate slack variables, and rewrite this problem as a quadratic problem (i.e. quadratic objective with linear constraints). This form is called the primal form of support vector regression. [*3 points*]

(b) Write down the Lagrangian function for the above primal form. [*1 point*]

(c) Using the Karush Kunh Tucker conditions, derive the dual form. [*3 points*]

(d) Write down the equation that can be used for predicting the label of an unseen sample X. [*1 point*]

(e) How would you define support vectors in this problem? [*1 point*]

(f) Is it possible to kernelize this algorithm? If yes, how? [*2 points*]

(g) What happens if we change ϵ and C? [*1 point*]

2 Programming (30 points)

1. **(9 points)**

(a) Write your own random forest classifier (this should be relatively easy, given you have written your own decision tree code) to apply to the *Spam* dataset [data, information]. Use 30% of the provided data as test data and the remaining for training. Compare

your results in terms of accuracy and time taken with Scikitlearn's built-in random forest classifier. [*5 points*]

(b) Explore the sensitivity of Random Forests to the parameter m (the number of features used for best split). [*2 points*]

(c) Plot the OOB (out-of-bag) error (you have to find what this is, and read about it!) and the test error against a suitably chosen range of values for m. [*2 points*]

2. **(6 points)** In this problem, you will implement a naive Bayes classifier. This classifier will be used to classify fortune cookie messages into two classes: messages that predict what will happen in the future and messages that just contain a wise saying. We will label messages that predict what will happen in the future as class 1 and messages that contain a wise saying as class 0. For example, "Never go in against a Sicilian when death is on the line" would be a message in class 0. "You will get an A in CS6510" would be a message in class 1.
There are three sets of data files provided:

- Training data and labels: traindata.txt: This is the training data consisting of fortune cookie messages; and trainlabels.txt: This file contains the class labels for the training data.
- Test data and labels: testdata.txt: This is the testing data consisting of fortune cookie messages; and testlabels.txt: This file contains the class labels for the testing data. These are only used to determine the accuracy of the classifier.
- A list of stopwords: stoplist.txt.

There are two steps to this problem: the *pre-processing step* and the *classification step.*
Pre-processing Step:

(a) Form the vocabulary. The vocabulary consists of the set of all the words that are in the training data with stop words removed (stop words are common, uninformative words such as "a" and "the" that are listed in the file `stoplist.txt`). Maintain the vocabulary in alphabetical order. The vocabulary will be used to derive your actual training data. [*1 point*]

(b) Now, convert the training data into a set of features. Let M be the size of your vocabulary. For each fortune cookie message, you will convert it into a feature vector of size M+1. Each slot in that feature vector takes the value of 0 or 1. For the first M slots, if the i^{th} slot is 1, it means that the i^{th} word in the vocabulary is present in the fortune cookie message; otherwise, if it is 0, then the i^{th} word is not present in the message. Most of the first M feature vector slots will be 0. Since you are keeping the vocabulary in alphabetical order, the first feature will be the first word alphabetically in the vocabulary. The $(M+1)^{th}$ slot corresponds to the class label. A 1 in this slot means the message is from class 1 while a 0 in this slot means the message is from class 0. [*1 point*]

(c) Output the pre-processed training data to a file called `preprocessed.txt` (to be submitted). The first line should contain the words in the vocabulary, separated by commas. The lines that follow the vocabulary words should be the featurized versions of the fortune cookie messages in the training data, with the features separated by commas. Your file should look something like: [*1 point*]

```
a,aardvark,almost,anticipate,...
0,0,1,0,...
0,1,0,1,...
```

Classification Step:
Build a naive Bayes classifier as described in class. Feel free to use any built-in functions in Scikitlearn, or write your own code. Some points to note [*3 points*]:

- You will need to convert the fortune cookie messages in the testing data also into a feature vector, just like the training data.
- If you encounter a word in the testing data that is not present in your vocabulary, ignore that word. Note that the feature vector is only of size M because the class labels are not part of the testing data.
- Output the accuracy of the naive Bayes classifier by comparing the predicted class label of each message in the testing data to the actual class label.

Your results must be stored in a file called **results.txt** (to be submitted).

3. **(15 points)** We will now implement Linear Regression to predict the age of Abalone (a type of snail). The data set is made available as part of the provided zip archive (linregdata). You can read more about the dataset at the UCI repository link. We are interested in predicting the last column of the data that corresponds to the age of the abalone using all the other attributes.

   (a) The first column in the data denotes the attribute that encodes-female, infant and male as 0, 1 and 2 respectively. The numbers used to represent these values are symbols and therefore should not be ordinal. Transform this attribute into a three column binary representation. For example, represent female as (1, 0, 0), infant as (0, 1, 0) and male as (0, 0, 1). [*0.5 points*]

   (b) Before performing linear regression, we must first standardize the independent variables, which includes everything except the last attribute (target attribute). Standardizing means subtracting each attribute by its mean and dividing by its standard deviation. Standardization will transform the attributes to possess zero mean and unit standard deviation. You can use this fact to verify the correctness of your code. [*0.5 points*]

   (c) Implement the following functions: (i) `mylinridgereg(X, Y, λ)` that calculates the linear least squares solution with the ridge regression penalty parameter (λ) and returns the regression weights; (ii) `mylinridgeregeval(X, weights)` that returns a prediction of the target variable given the input variables and regression weights; and (iii) `meansquarederr(T, Tdash)` that computes the mean squared error between the predicted and actual target values. [*6 points*]

   (d) Partition the dataset into 80% training and 20% testing (Let's call this the partition fraction, in this case 0.2). Now, use your `mylinridgereg` with different λ values to fit the penalized linear model to the training data and predict the target variable for both training and testing data. [*1 point*]

   (e) Identify the λ with the best performance and examine the weights of the ridge regression model. Which are the most significant

attributes? Try removing two or three of the least significant attributes and observe how the mean squared errors change. [*1 point*]

(f) We now would like to ask the question: Does the effect of λ on error change for different partitions of the data into training and test sets? To do this, change the partition fraction (a value between 0 and 1, as defined earlier) with at least 4 other values. Repeat the following steps 25 times for each partition fraction:

- Randomly divide data into training and test sets.
- Standardize the training input variables.
- Standardize the testing input variables using the means and standard deviations from the training set.
- Follow step (d) for each such partition.

For each partition fraction, plot a figure with λ on the x-axis, and MSE on the y-axis. For each figure, include 2 graphs - one for the training MSE and one for the test MSE. (You should then have 5 figures in total, with 2 plots on each figure.) [*3 points*]

(g) Do the above figures give you clarity? Also, plot two more figures. In the first graph, plot the minimum average mean squared testing error versus the partition fraction values. In the second graph, plot the λ value that produced the minimum average mean squared testing error versus the partition fraction. [*1 point*]

(h) How good is your model? So far, we have been looking at only the mean squared error. We might also be interested in understanding the contribution of each prediction towards the error. Maybe the error is due to a few samples with large errors and others have tiny errors. One way to visualize this information is to a plot of predicted versus actual values. Use the best choice for the training fraction and λ, make two graphs corresponding to the training and testing set. The X and Y axes in these graphs will correspond to the predicted and actual target values respectively. If the model is good, then all the points will be close to a 45-degree line through the plot. [*2 points*]

Include all the plots and your observations in your submission.