Assignment 2

CS6510: Applied Machine Learning
IIT-Hyderabad
Aug-Nov 2018

Max Points: 35
Due: 31 Oct 2018 11:59 pm

This homework is intended to cover programming exercises in the following topics:

- SVMs, Kernel functions, Naive Bayes, Neural Networks, Boosting

Instructions

- Please use Google Classroom to upload your submission by the deadline mentioned above. Your submission should comprise of a single file (PDF/ZIP), named `<Your_Roll_No>_Assign2`, with all your solutions.
- 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 15 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.
- We recommend using PYTHON for the programming assignments, although you can use Java/C/C++/R if you like too.
- 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 Questions

1. **Random Forests (15 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. ***(9 points)***

(b) Explore the sensitivity of Random Forests to the parameter m (the number of features used for best split). ***(3 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. ***(3 points)***

Deliverables:

- Code
- Brief report (PDF) with your solutions for the above questions

2. **Naive Bayes Classifier (10 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 choose a complex classifier when you can choose a simple one" 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. No test labels are provided, and we will use this to assess your performance.
- 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. ***(2 points)***

(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. ***(2 points)***

(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 ***(5 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).
Deliverables:

- Code
- results.txt
- Brief report (PDF) with any observations you would like to add

3. **(10 points)** Install the `pydataset` module (if you haven't already):

```
$ pip install pydataset
```

Then, in the python `REPL`, type in the following commands to get started

```
>>> from pydataset import data
>>> import pandas as pd
>>> melanoma_data = data('Melanoma', show_doc=True)
```

The `Melanoma` dataset consists of measurements of patients with malignant melanoma (a type of cancer). For each patient, the dataset specifies if the patient died or lived at the end of the trial. Moreover, some patients died due to causes unrelated to melanoma. Your task is to do the following:

(a) Remove those patients who died due to causes unrelated to Melanoma, and plot patient `status` vs `age` and patient `status` vs `thickness` - for your own understanding. ***(1 point)***

(b) Fit a model using any classification algorithm of your choice to predict the `status` of the patient. You can use any library of your choice. ***(5 points)***

(c) Split the data into 70% training and 30% test set. Compute and show accuracy, precision and recall. To ensure that there is no randomness bias, repeat this over 5 trials,

and plot the mean and standard deviation of the three quantities (accuracy, precision, recall). Do you observe any difference between the readings of your first trial, and the average of the 5 trials? Comment. ***(4 points)***

Deliverables:

- Code
- Brief report (PDF) with your solutions for the above questions

2 Theory and Practice Questions (No submission required)

The questions below are only for your practice - no submission required.

1. 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. 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.
3. 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 ?