CPTS 570 Machine Learning, Fall 2017
Homework #3
Due Date: Nov 3, Fri 4-5pm (office hours)

1. (**10 points**) Suppose $x = (x_1, x_2, \cdots, x_d)$ and $z = (z_1, z_2, \cdots, z_d)$ be any two points in a high-dimensional space (i.e., d is very large).

   a. (**5 points**) Try to prove the following, where the right-hand side quantity represent the standard Euclidean distance.

   $$\left(\frac{1}{\sqrt{d}}\sum_{i=1}^{d} x_i - \frac{1}{\sqrt{d}}\sum_{i=1}^{d} z_i\right)^2 \leq \sum_{i=1}^{d}(x_i - z_i)^2 \tag{1}$$

   Hint: Use Jensen's inequality – If X is a random variable and f is a convex function, then $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$.

   b. (**5 points**) We know that the computation of nearest neighbors is very expensive in the high-dimensional space. Discuss how we can make use of the above property to make the nearest neighbors computation efficient?

2. (**10 points**) We briefly discussed in the class about Locality Sensitive Hashing (LSH) algorithm to make the nearest neighbor classifier efficient. Please read the following paper and briefly summarize the key ideas as you understood:

   Alexandr Andoni, Piotr Indyk: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Communications of ACM 51(1): 117-122 (2008) `http://people.csail.mit.edu/indyk/p117-andoni.pdf`

3. (**10 points**) We know that we can convert any decision tree into a set of if-then rules, where there is one rule per leaf node. Suppose you are given a set of rules $R = \{r_1, r_2, \cdots, r_k\}$, where r_i corresponds to the i^{th} rule. Is it possible to convert the rule set R into an equivalent decision tree? Explain your construction or give a counterexample.

4. (**5 points**) You are provided with a training set of examples (see Figure 1). Which feature will you pick first to split the data as per the ID3 decision tree learning algorithm? Show all your work: compute the information gain for all the four attributes and pick the best one.

5. (**10 points**) Please read the following paper and briefly summarize the key ideas as you understood (You can skip the proofs, but it is important to understand the main results):

   Andrew Y. Ng, Michael I. Jordan: On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes. NIPS 2001: 841-848 `http://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf`

6. (**5 points**)

   a. Let us assume that the training data satisfies the Naive Bayes assumption (i.e., features are independent given the class label). As the training data approaches infinity, which classifier will produce better results, Naive Bayes or Logistic Regression? Please explain your reasoning.

   b. Let us assume that the training data does **NOT** satisfy the Naive Bayes assumption. As the training data approaches infinity, which classifier will produce better results, Naive Bayes or Logistic Regression? Please explain your reasoning.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Figure 1: Table with training examples. Each row corresponds to a single training example. There are four features, namely, outlook, temperature, humidity, and wind. "PlayTennis" is the class label.

7. (**5 points**)

   a. Can we compute $P(X)$ from the learned parameters of a Naive Bayes classifier? Please explain your reasoning.

   b. Can we compute $P(X)$ from the learned parameters of a Logistic Regression classifier? Please explain your reasoning.

8. (**10 points**) In the class, we looked at the log-likelihood derivation and the corresponding gradient ascent algorithm to find the parameters of a binary logistic regression classifier (see slide 12 and slide 13). We want to extend the log-likelihood derivation and parameter learning algorithm to the multi-class case. Suppose we have K different classes, and the posterior probability can be represented using the so-called soft-max function (see slide 18):

$$P(y = k|x) = \frac{exp(w_k \cdot x)}{\sum_{i=1}^{K} exp(w_i \cdot x)} \tag{2}$$

   a. Derive the log-likelihood and the corresponding gradient ascent algorithm to find the parameters.

   b. Add a regularization term to the log-likelihood objective (see slide 16), and derive the gradient ascent update rule with the additional change.

9. (**35 points**) Fortune Cookie Classifier[1]

   You will build a binary fortune cookie classifier. This classifier will be used to classify fortune cookie messages into two classes: messages that predict what will happen in the future (class 1) and messages that just contain a wise saying (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 Machine learning class" would be a message in class 1.

[1]Thanks to Weng-Keen Wong and his advisor Andrew Moore for sharing the data.

Files Provided There are three sets of files. All words in these files are lower case and punctuation has been removed.

1) The training data:

traindata.txt: This is the training data consisting of fortune cookie messages.
trainlabels.txt: This file contains the class labels for the training data.

2) The testing data:

testdata.txt: This is the testing data consisting of fortune cookie messages.
testlabels.txt: This file contains the class labels for the testing data.

3) A list of stopwords: stoplist.txt

There are two steps: the pre-processing step and the classification step. In the pre-processing step, you will convert fortune cookie messages into features to be used by your classifier. You will be using a bag of words representation. The following steps outline the process involved:

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). The vocabulary will now be the features of your training data. Keep the vocabulary in alphabetical order to help you with debugging.

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. Each slot in that feature vector takes the value of 0 or 1. For these M slots, if the ith slot is 1, it means that the ith word in the vocabulary is present in the fortune cookie message; otherwise, if it is 0, then the ith word is not present in the message. Most of these 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.

a. Implement the ID3 decision tree learning algorithm that we discussed in the class. The key step in the decision tree learning is choosing the next feature to split on. Implement the information gain heuristic for selecting the next feature. Please see lecture notes or `https://en.wikipedia.org/wiki/ID3_algorithm` for more details.

b. Run the decision tree construction algorithm on the training examples. Compute the accuracy on validation examples (20 percent of training examples) and testing examples.

c. Implement the decision tree pruning algorithm discussed in the class (via validation data).

d. Run the pruning algorithm on the decision tree constructed using training examples. Compute the accuracy on validation examples and testing examples. List your observations by comparing the performance of decision tree with and without pruning.

To debug and test your implementation, you can employ Weka (weka.classifiers.trees.J48): `http://www.cs.waikato.ac.nz/ml/weka/downloading.html`

e. Run the Naive Bayes classifier from Weka (weka.classifiers.bayes.NaiveBayes) on the training data. Compute the training and testing accuracy.

f. Run the Naive Bayes classifier from Weka (weka.classifiers.functions.Logistic) on the training data. Compute the training and testing accuracy.