CS57800 Statistical Machine Learning
Homework 2

Due: Oct 6, 2015 on Tuesday

1 Foundations

1. Given n boolean variables $(x_1, \ldots, x_n)$, we define our target classification function $f(.)$ as $f(.) = 1$ if at least 2 variables are active. For $n = 4$ show how this function can be represented as (1) Boolean function (2) Linear function.

2. Let CON_B be the set of all different monotone conjunctions defined over n boolean variables. What is the size of CON_B ?

3. Suppose there are N points x_i in $\mathcal{R}^p$, with labels $y_i \in \{1, 1\}$. Denote a hyperplane by a function $f(x) = \beta_1' x + \beta_0 = 0$, or $\beta' x^* = 0$, where $x^* = (x, 1)$ and $\beta = (\beta_1, \beta_0)$. Let $u_i = x_i^* / \|x_i^*\|$. Given a current β_o, the perceptron algorithm identifies a point u_i that is misclassified, and produces the update $\beta_n \leftarrow \beta_o + y_i u_i$. Show that

$$\|\beta_n - \beta^*\|^2 \leq \|\beta_o - \beta^*\|^2 - 1,$$

and hence that the perceptron converges to a separating hyperplane (β^* is the final separating parameter vecter).

4. Suggest a mistake bound algorithm for learning Boolean conjunctions (*hint: recall the elimination algorithm for monotone conjunctions*). Show that your algorithm is a mistake bound algorithm for Boolean conjunctions.

5. Given a linearly separable dataset consisting of 50000 positive examples and 50000 negative examples, we train two linear classifier using the perceptron algorithm. We provide the first classifier with a sorted dataset in which all the positive examples appear first, and then the negative examples appear. The second classifier is trained by randomly selecting examples at each training iteration. (1) Will both classifiers converge? (2) what will be the training error of each one of the classifiers?

6. Let N denote the set of iterations at which the Perceptron algorithm makes an update when it sees a sequence of training instances $x_1, ..., x_n \in \mathcal{R}^N$. Then, the following inequality holds:

$$\| \sum_{i \in N} y_i x_i \| \leq \sqrt{\sum_{i \in N} \|x_i\|^2}$$

Try to prove this.

2 Programing: Perceptron and Winnow

General Programing Requirements

You should implement your solution using Python or JAVA. You may **not** use anybody else's code; you **must** implement your own version! We will use **MOSS** (Measure Of Software Similarity) from Stanford University to check for plagiarism. You should submit your source code files along with your typed HW report. Any external packages are **not** allowed (e.g. scikit-learn for python) except numpy and scipy, but feel free to use any internal packages (e.g. sys, os, math in python). The TAs should be able to compile and run your code. (And also keep in mind the JAVA version on data.cs.purdue.edu is 1.8 while python is 2.7)

Task Description

In this section, you are required to implement two online learning algorithms, the Perceptron algorithm and the Winnow algorithm, and observe their performances in practice by running the algorithms over the movie review dataset. In the dataset, each line corresponds to a single snippet which comes from a movie review and is labeled with either a positive or negative sign. Your task is to learn the weights using a language model and then predict whether a given snippet is positive or negative.

The dataset is split into training, validation, and test sets. You need to use the training set for learning weights and the validation set for tuning hyperparameter(s), and then use the test set for evaluation.

Please see more details of the dataset in this paper(Bo Pang and Lillian Lee, Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, *Proceedings of ACL 2005.*).

1. **Feature Representation** You will need to consider two different feature sets as described below

   (1) unigram
   (2) bigram
   (3) use both

2. **Implementation** You will need to complete three functions *perceptron(maxIterations, featureSet)*, *winnow(maxIterations, featureSet)*, and *predict_one(weights, input_snippet)* in the *template.py* file. *maxIterations* is a hyperparameter which determines the number of iterations each algorithm will perform over the dataset in training phase. *featureSet* takes an integer value from 1, 2, or 3 each of which corresponds to a feature set described above. The function *predict_one* takes a set of weights learned from the training set and a single example, and then returns the sign of the input example.

   The best assignment to *maxIterations* needs to be obtained by tuning its value over the validation set. You can run your implementation several times with different values of the hyperparameter for learning the weights and then choose the best assignment to *maxIterations* by evaluating the learned weights over the validation set.

3. **Report** As you did in the previous assignment, you should write a report describing the experimental results of your algorithms. For each algorithm and each feature set, report its performance with accuracy, precision, recall, and F-score over the training, validation, and testing sets. Your report also should include the optimal value of *maxIterations* and explain how the value is picked for each feature representation.

4. **Extra Credits** Some extra credits will be given if you implement the Perceptron in a dual form(in addition to a primal form) and show how the dual representation affects the performance of the algorithm in your report.

Submission Instructions:

You are required to use LaTeX to type your solutions to questions, and report of your programing as well. Other formats of submission will **not** be accepted. A template named "homework.tex" is also provided for your convenience.

After logging into data.cs.purdue.edu (physically go to the lab or use ssh remotely, you are all granted the accounts to CS data machines during this class), please follow these steps to submit your assignment:

1. Make a directory named *'your Name_your Surname'* and copy all of your files there.

2. While in the upper level directory (if the files are in /homes/dan/dan_goldwasser, go to/homes/dan), execute the following command:

   ```
   turnin -c cs578 -p HW2 *your_folder_name*
   ```

   (e.g. your prof would use: `turnin -c cs578 -p HW2 dan_goldwasser` to submit his work)

   Keep in mind that old submissions are overwritten with new ones whenever you execute this command.

3. You can verify the contents of your submission by executing the following command:

   ```
   turnin -v -c cs578 -p HW2
   ```

   Do **not** forget the -v flag here, as otherwise your submission would be replaced with an empty one.

Your submission should include the following files:

1. The source code in python or java (and the executable .jar file for java).
2. Your evaluation & analysis in .pdf format (together with your HW solutions in it).
3. A README file containing your name, instructions to run your code and anything if you would like us to know about your program (like errors, special conditions, etc).