CptS 315: Introduction to Data Mining
Homework 3

(Due date: April 4 in class)

Instructions

- Please use a word processing software (e.g., Microsoft word) to write your answers and submit a printed copy to me at the beginning of the class on Feb 8. The rationale is that it is sometimes hard to read and understand the hand-written answers.
- All homeworks should be done individually.

Analytical Part (50 points)

Q1. (5 points) Answer the following with a yes or no along with proper justification.
a. Is the decision boundary of voted perceptron linear?
b. Is the decision boundary of averaged perceptron linear?

Q2. (5 points) Consider the following setting. You are provided with n training examples: $(x_1, y_1, h_1), (x_2, y_2, h_2), \cdots, (x_n, y_n, h_n)$, where x_i is the input example, y_i is the class label (+1 or -1), and $h_i > 0$ is the importance weight of the example. The teacher gave you some additional information by specifying the importance of each training example. How will you modify the perceptron algorithm to be able to leverage this extra information? Please justify your answer.

Q3. (5 points) Consider the following setting. You are provided with n training examples: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$, where x_i is the input example, and y_i is the class label (+1 or -1). However, the training data is highly imbalanced (say 90% of the examples are negative and 10% of the examples are positive) and we care more about the accuracy of positive examples. How will you modify the perceptron algorithm to solve this learning problem? Please justify your answer.

Q4. (10 points)
You were just hired by MetaMind. MetaMind is expanding rapidly, and you decide to use your machine learning skills to assist them in their attempts to hire the best. To do so, you have the following available to you for each candidate i in the pool of candidates $\mathcal{I}$: (i) Their GPA, (ii) Whether they took Data Mining course and achieved an A, (iii) Whether they took Algorithms course and achieved an A, (iv) Whether they have a job offer from Google, (v) Whether they have a job offer from Facebook, (vi) The number of misspelled words on their resume. You decide to represent each candidate $i \in \mathcal{I}$ by a corresponding 6-dimensional feature vector $f(x^{(i)})$. You believe that if you just knew the right weight vector $w \in \Re^6$ you could reliably predict the quality of a candidate i by computing $w \cdot f(x^{(i)})$. To determine w

your boss lets you sample pairs of candidates from the pool. For a pair of candidates (k, l) you can have them face off in a "DataMining-fight." The result is $\texttt{score}\,(k \succ l)$, which tells you that candidate k is at least $\texttt{score}\,(k \succ l)$ better than candidate l. Note that the score will be negative when l is a better candidate than k. Assume you collected scores for a set of pairs of candidates $\mathcal{P}$.

Describe how you could use a perceptron based algorithm to learn the weight vector w. Make sure to describe the basic intuition; how the weight updates will be done; and pseudo-code for the entire algorithm.

Q5. (15 points) Suppose we have n_+ positive training examples and n_- negative training examples. Let C_+ be the center of the positive examples and C_- be the center of the negative examples, i.e., $C_+ = \frac{1}{n_+}\sum_{i:\ y_i=+1} x_i$ and $C_- = \frac{1}{n_-}\sum_{i:\ y_i=-1} x_i$. Consider a simple classifier called CLOSE that classifies a test example x by assigning it to the class whose center is closest.

- Show that the decision boundary of the CLOSE classifier is a linear hyperplane of the form $sign(w \cdot x + b)$. Compute the values of w and b in terms of C_+ and C_-.
- Recall that the weight vector can be written as a linear combination of all the training examples: $w = \sum_{i=1}^{n_+ + n_-} \alpha_i \cdot y_i \cdot x_i$. Compute the dual weights (α's). How many of the training examples are support vectors?

Q6. (10 points) Please read the following paper and write a brief summary of the main points in at most TWO pages.

Pedro M. Domingos: A few useful things to know about machine learning. Communications of ACM 55(10): 78-87 (2012)
`https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf`

Programming and Experimental Part (50 points)

- (**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.

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

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

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.

Implement a binary classifier with perceptron weight update as shown below. Use learning rate η=1.

Algorithm 1 Online Binary-Classifier Learning Algorithm

Input: $\mathcal{D}$ = Training examples, T = maximum number of training iterations
Output: w, the final weight vector

```
1: Initialize the weights $w = 0$
2: for each training iteration $itr \in \{1, 2, \cdots, T\}$ do
3:     for each training example $(x_t, y_t) \in \mathcal{D}$ do
4:         $\hat{y}_t = sign(w \cdot x_t)$ // predict using the current weights
5:         if mistake then
6:             $w = w + \eta \cdot y_t \cdot x_t$ // update the weights
7:         end if
8:     end for
9: end for
10: return final weight vector $w$
```

a) Compute the the number of mistakes made during each iteration (1 to 20).

b) Compute the training accuracy and testing accuracy after each iteration (1 to 20).

c) Compute the training accuracy and testing accuracy after 20 iterations with standard perceptron and averaged perceptron.

- Implement a multi-class online learning algorithm with perceptron weight update as shown below. Use learning rate η=1.

Algorithm 2 Online Multi-Class Classifier Learning Algorithm

Input: $\mathcal{D}$ = Training examples, k = number of classes, T = maximum number of training iterations
Output: $w_1, w_2, \cdots, w_k$ the final weight vectors for k classes

```
1: Initialize the weights $w_1 = 0, w_2 = 0, \cdots, w_k = 0$
2: for each training iteration $itr \in \{1, 2, \cdots, T\}$ do
3:   for each training example $(x_t, y_t) \in \mathcal{D}$ do
4:     $\hat{y}_t = \arg max_{y \in \{1,2,\cdots,k\}} w_y \cdot x_t$ // predict using the current weights
5:     if mistake then
6:       $w_{y_t} = w_{y_t} + \eta \cdot x_t$ // update the weights
7:       $w_{\hat{y}_t} = w_{\hat{y}_t} - \eta \cdot x_t$ // update the weights
8:     end if
9:   end for
10: end for
11: return final weight vectors $w_1, w_2, \cdots, w_k$
```

Task. You will build a optical character recognition (OCR) classifier: given an image of handwritten character, we need to predict the corresponding letter. You are provided with a training and testing set.

Data format. Each non-empty line corresponds to one input-output pair. 128 binary values after "im" correspond to the input features (pixel values of a binary image). The letter immediately afterwards is the corresponding output label.

a) Compute the the number of mistakes made during each iteration (1 to 20).

b) Compute the training accuracy and testing accuracy after each iteration (1 to 20).

c) Compute the training accuracy and testing accuracy after 20 iterations with standard perceptron and averaged perceptron.

Instructions for Code Submission and Output Format.
Please follow the below instructions. It will help us in grading your programming part of the homework. We will provide a dropbox folder link for code submission.

- Supported programming languages: Python, Java, C++
- Store all the relevant files in a folder and submit the corresponding zipfile named after your student-id, e.g., 114513209.zip
- This folder should have a script file named

  ```
  run_code.sh
  ```

  Executing this script should do all the necessary steps required for executing the code including compiling, linking, and execution
- Assume relative file paths in your code. Some examples:

  ```
  ‘‘./filename.txt’’ or ‘‘../hw1/filename.txt’’
  ```

- The output of your program should be dumped in a file named “output.txt” in the following format. One block for binary classifier and another similar block for the multi-class classifier.

  iteration-1 no-of-mistakes
  . . .
  . . .
  iteration-20 no-of-mistakes

  iteration-1 training-accuracy testing-accuracy
  . . .
  . . .
  iteration-20 training-accuracy testing-accuracy

  training-accuracy-standard-perceptron testing-accuracy-averaged-perceptron

  Explanation. Self-explanatory
- Make sure the output.txt file is dumped when you execute the script

  ```
  run_code.sh
  ```

- Zip the entire folder and submit it as

  ```
  <student_id>.zip
  ```

Grading Rubric

Each question in the students work will be assigned a letter grade of either A,B,C,D, or F by the Instructor and TAs. This five-point (discrete) scale is described as follows:

- **A) Exemplary (=100%)**.
  Solution presented solves the problem stated correctly and meets all requirements of the problem.
  Solution is clearly presented.
  Assumptions made are reasonable and are explicitly stated in the solution.
  Solution represents an elegant and effective way to solve the problem and is not overly complicated than is necessary.

- **B) Capable (=75%)**.
  Solution is mostly correct, satisfying most of the above criteria under the exemplary category, but contains some minor pitfalls, errors/flaws or limitations.

- **C) Needs Improvement (=50%)**.
  Solution demonstrates a viable approach toward solving the problem but contains some major pitfalls, errors/flaws or limitations.

- **D) Unsatisfactory (=25%)**
  Critical elements of the solution are missing or significantly flawed.
  Solution does not demonstrate sufficient understanding of the problem and/or any reasonable directions to solve the problem.

- **F) Not attempted (=0%)**
  No solution provided.

The points on a given homework question will be equal to the percentage assigned (given by the letter grades shown above) multiplied by the maximum number of possible points worth for that question. For example, if a question is worth 6 points and the answer is awarded a *B* grade, then that implies 4.5 points out of 6.