

Homework Assignment 2

Due date: Tue 10/18 (in class)

1 Introduction

In this project you will implement the Perceptron algorithm (we refer to it below as GD for its relation to gradient descent) and the EG^{+/-} (exponentiated gradient) algorithm, run experiments with them comparing their performance, and test the effect of parameters on their performance.

2 The algorithms

To avoid confusion regarding versions of these algorithms the following gives pseudo code.

We assume n numerical features. An example $\vec{x} = (x_1, \dots, x_n)$ is a vector of n numbers. The dot product is $\vec{w} \cdot \vec{x} = \sum_{k=1}^n w_k x_k$. The sign function gives a value of $\{-1, +1\}$. Labels are also in $\{-1, +1\}$. Both algorithms use a parameter $\eta \in [0, 1]$ called the learning rate. Both algorithms use a threshold parameter θ . We fix $\theta = 0$ for this assignment and omit it from the code below.

The perceptron algorithm maintains a weight vector of n numbers $\vec{w} = (w_1, \dots, w_n)$. The Perceptron Algorithm (GD) is as follows:

1. Initialize $w_k = 1$ for all k .
2. Repeat **Iterations** times
 - (a) For each example (\vec{x}, L) in training set do:
 - (Classify): $O = \text{sign}(\vec{w} \cdot \vec{x})$.
 - (Update if $O \neq L$):
For all $k = 1 \dots n$, $w_k = w_k + \eta(L - O)x_k$
3. Output the last weight vector \vec{w} as the final hypothesis.
4. (For this assignment): Use the final hypothesis to classify examples in the test set, calculating the number of mistakes and error rate (in %) on the test set.

The EG^{+/-} algorithm maintains two weight vectors of n numbers each, $\vec{w}^P = (w_1^P, \dots, w_n^P)$, $\vec{w}^N = (w_1^N, \dots, w_n^N)$. The superscripts stand for Positive and Negative respectively. The EG^{+/-} algorithm is as follows:

1. Initialize $w_k^P = w_k^N = 1$ for all k .
2. Repeat **Iterations** times
 - (a) For each example (\vec{x}, L) in training set do:
 - (Classify): $O = \text{sign}(w^P \cdot \vec{x} - w^N \cdot \vec{x})$.
 - (Update if $O \neq L$):
For all $k = 1 \dots n$, $w_k^P = w_k^P e^{\eta(L-O)x_k}$
For all $k = 1 \dots n$, $w_k^N = w_k^N e^{-\eta(L-O)x_k}$
3. Output the last weight vector pair \vec{w}^P, \vec{w}^N as the final hypothesis.
4. (For this assignment): Use the final hypothesis to classify examples in the test set, calculating the number of mistakes and error rate (in %) on the test set.

3 Your task

3.1 The Main Program

You need to write a program that expects 5 initial arguments (it would be nice if these are command line arguments but that is not mandatory). These are: (1) name of file with training data (2) name of file with test data (3) the value of η (4) the number of iterations, and (5) which algorithm to run (GD or EG). Both training data and test data are in weka format. However, you may assume that all attributes are numerical and the labels are -1 and 1. The program will read the training data and run the chosen algorithm on it. As described in the code above, the program will fix the final hypothesis from this run and then read the test data counting the number of mistakes on it. Finally the program will report the error rate on the test data. (This is the only output required from the program.)

You can write the program in any language. However, please make sure that your code runs on our sun systems.

3.2 Experimenting

We are providing you with several datasets: (1) sparse-target data. (2) dense-target and sparse-example data. Both of these were discussed in class; you can also see the performance graphs shown in class in the experimental section of the “Relative Loss Bounds . . .” paper on the course web page. For each of these we have several data files; 3 training files with 200, 800, and 2000 examples respectively and a test file with 2000 examples. File names are `ST.200.train, . . . , ST.test`, and `DT.200.train, . . . , DT.test`. All these are artificially created as in the paper for the purpose of our experiments. In addition we have (3) data from the sport domain of assignment 1: here we use some numerical features based on words (notice that examples are sparse). We have two variants of this data, one with 50 features and one with 500 features. Again the data is split into a training set (700 examples) and test set (500 examples) and it is provided in files `sport.50.train, sport.50.test, sport.500.train, sport.500.test`.

After implementing (and debugging) the main program you should experiment with the algorithms and datasets as follows:

- Run both GD and EG on the sport datasets. Observe and report the following: what values of η and iterations give the best results? what error rates do you get? how do the algorithms compare on this task? how does performance vary between 50 and 500 features? how do the error rates compare to the results you got in assignment 1 with the decision trees? (NB we are using a different evaluation scheme but it is still valuable to get some comparison.) Also, trace the number of mistakes per iteration during training. What do you observe?
- Run each of {GD,EG} on datasets {ST,DT}. How does the error rate scale when we increase training set size? How does the error rate scale with the number of iterations? How is the error rate affected by η ? (Try to tabulate these e.g. when fixing some values for other parameters. Otherwise your search space and time may be long.) Which of the algorithms does better on ST? on DT? How many iterations does it take before training error goes to 0 for the algorithms?

Of course you may want to write a program/script to run all these tests but the details are up to you.

4 Extra Credit

You can get up to 20 extra points for the following.

Implement a “voted-version” of GD and EG and evaluate their performance along with the other ones. The voted-version stores all hypotheses generated during training along with the number of examples they got correctly before making a mistake. This number serves as a weight for the corresponding hypothesis. In the test phase, we evaluate all the hypotheses and take a weighted vote on the label.

5 Files

All data files are available at `/comp/150ML/files/hw2/`.

6 What You Need to Submit

1. Source code of program file(s) and any scripts used to run it. Your program should be well documented and in good style. In particular the top of the file should include a comment giving instructions for using the program (this can be brief.)
2. A short report on the experiments you ran and their results. Give an explanation in English, analyzing the results; the questions above suggest things that should be discussed in the report. Add accuracy or error tables where appropriate.

7 Submitting Your Assignment

Please Submit printed versions of all the above in class.