In the Name of God
Project-3
Functional Gradient Descent: GradientBoost Algorithm
Department of Electrical Engineering - Shiraz University

Instructor: Dr. Hamed Masnadi-Shirazi   Fall 2015

1 Introduction

Use functional gradient descent (GradientBoost Algorithm) to minimize the risk functional

$$R(\phi(x_i)) = \sum_{i=1}^{n} L(y_i, \phi(x_i))$$

and compare the Exponential loss $L(y_i, \phi(x_i)) = e^{-y_i \phi(x_i)}$ and the Logit loss $L(y_i, \phi(x_i)) = \frac{1}{\sigma}\log(1 + e^{-\sigma y_i \phi(x_i)})$ on a 2-D Gaussian problem. The object of this project is to use the GradientBoost Algorithm to compare the Exponential loss and the Logit loss on a 2-D Gaussian problem. We have already derived the GradientBoost Algorithm using the Exponential loss in class. At this stage you should follow the same procedure and derive the GradientBoost Algorithm using the Logit loss. This is simply achieved by finding the gradient functional for the Logit loss. In a course on Pattern Recognition, it can be shown that both loss functions are Bayes consistent and suitable for classification problems.

2 General Assignment

In this project you will be given a set of positive training data $(x_i, y_i = 1)$ and negative $(x_i, y_i = -1)$ training data and you are to build a boosted classifier for the data using what are called decision stumps as your weak learners (or as your pool of functions γ as we called it in class). Remember that your pool of functions is in fact a pool of classifiers $g(x_i)$ such that $g(x_i) \geq 0$ for points that are classified as positives and $g(x_i) < 0$ for points that are classified as negatives. the training data is two dimensional so x_i^1 is the first dimension and x_i^2 is the second dimension such that $x_i = [x_i^1 \ x_i^2]$. The positive class is sampled from a two dimensional Gaussian with mean $\mu = [0 \ 0]$ and covariance $\Sigma_{D1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and the negative class is sampled from a two dimensional Gaussian with mean $\mu = [2 \ 2]$ and covariance $\Sigma = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}$.

A decision stump classifier is simply a threshold on one of the dimensions of the data plus a parity which decides which side of the threshold to be the positive class and which side of the threshold is the negative class. So for each dimension of data and for each threshold and for each parity we have a classifier $g(x_i^j)$ that is either equal to $+1$ or -1 depending on the data, threshold and parity. The set of all such classifiers will be our pool of functions γ.

So at each iteration t of the GradientBoost algorithm with exponential loss we compute

$$\sum_{i=1}^{n} y_i e^{-y_i \phi^{(t-1)}(x_i^j)} g(x_i^j) \tag{1}$$

for each $g \in \gamma$ and pick the decision stump g^* that results in the maximum value of the above equation. We then update our classifier

$$\phi^{(t)}(x_i^j) = \phi^{(t-1)}(x_i^j) + \alpha g^*(x_i^j) \tag{2}$$

and repeat the above T times. Our final boosted classifier is

$$sign(\phi^{(T)}(x_i^j))$$

Note that in equation (1), the

$$w^{(t-1)}(x_i^j) = e^{-y_i \phi^{(t-1)}(x_i^j)} \tag{3}$$

is independent of $g(x_i^j)$ and can be computed only once at each iteration. We call the vector that holds this value for each x_i^j the vector of weights. So rather that recompute this vector for every $g \in \gamma$, we simply update it once every iteration.

The classifier that you have trained will then be tested or evaluated on the test data set. The true test data labels will be provided from which you will report test error rates. You will compare the test error rates of the Exponenial and Logit methods to the true error rates found using the Bayes decision rule which is the optimal decision for this problem.

3 Matlab details

STEP 1. Use the matlab code below to load the training data.

```
load('C:\GradBoostTrainDATA.mat','PTrain','NTrain','x1');
```

where PTrain is the positive train data matrix and NTrain is the negative train data matrix and x1 is the matrix of PTrain concatinated with NTrain. Plot the positive data points as blue circles and the negative data as red stars using the code below

```
figure,
axis([-5 10 -5 10])
plot(NTrain(:,1),NTrain(:,2),'r*')
hold on
axis([-5 10 -5 10])
plot(PTrain(:,1),PTrain(:,2),'bo')
x1=[PTrain ; NTrain];
```

Include this plot in your report.

STEP 2. We need to choose our parameters in this step. We will choose $\alpha = 1$ and $T = 4$. We will also further define our decision stumps. You need to find the max and min value for each dimension of the training data. You will then divide this range into 100 parts so that your threshold step size is $\frac{max-min}{100}$.

STEP 3. In this step we set our initial classifier $\phi^{(0)}(x_i^j) = 0$ and define the vector of weights and initialize it

```
winit=exp(0);
w=winit*ones(1,size(x1,1));
```

In other words all points have an initial weight equal to 1.

STEP 4. At this step we will find the first best $g \in \gamma$. So for each dimension of the data and for each threshold on the data and each parity on the threshold you should compute the value of equation (1) and keep the g that has the maximum value. In other words you should keep the dimension, threshold and parity that resulted in the maximum. What is the answer? Include the dimension number, threshold and parity of this first best g^* in your report. Does this dimension, threshold and parity make sense? Explain your answer.

STEP 5. In this step we will update our classifier and weight vector. We first update the classifier using

$$\phi^{(1)}(x_i^j) = \phi^{(0)}(x_i^j) + \alpha g^*(x_i^j) = 0 + g^*(x_i^j) = g^*(x_i^j)$$

and then update the weights vector using

$$w^{(1)}(x_i^j) = e^{-y_i \phi^{(1)}(x_i^j)}.$$

Note that the only thing you need to change in your code for the Logit loss is to change the weight update formula here if you write your code correctly.

How have the weights changed? You should see that the weights have decreased to $e^{-1} = 0.3679$ for data points that have been classified correctly so far and increased to $e^{1} = 2.7183$ for data points that have not been classified correctly so far.

STEP 6. In this step we will first fix any numerical problems on the weights using the code below

```
wPLC=find(w==0);
w(wPLC)=realmin;
wPLC=find(w==inf);
w(wPLC)=realmax;
```

Explain what this code does in your report. Next we will compute the risk since the risk is our objective function. We can simply use the code below

```
CurrentRISK=sum(w);
```

Explain why this code computes the current risk in your report and include the value of the risk. (Hint: If you did things right the risk should be 736.9.)

STEP 7. In this step we will simply place a for loop on top of our code and repeat steps 4-6. You should save the dimension, threshld and parity of the g^* that you find in each iteration. You should also save the risk at each iteration and plot the value of the risk over all 4 iterations. Include this plot in your report. Does the Risk decrease at each iteration in your plot? Explain your answer. (Hint: If you did things right your final risk at iteration 4 should be 631.)

STEP 8. In this step we will load the test data using the code below

```
load('C:\GradBoostTestDATA.mat','PTest','NTest','x1');
```

Plot the test data like you did in step 1 and include this plot in your report.

STEP 9. In this step we will use the classifier we learned in step 7 to classify all our test data. For example, the first test data point $x_1 = [x_1^1 \; x_1^2]$ is fed to the first g_1^*. If for example the first g_1^* has dimension 2 and threshold 3.1 and parity 1 then we look at the value of x_1^2 and compare it to threshold 3.1 and if it is larger than the threshold we assign $g_1^*(x_1^2) = 1$. We repeat this for the other g_k^* and compute

$$sign(\phi^{(4)}(x_i^j)) = sign(\sum_{k=1}^{(4)} g_k(x_i^j)) \tag{4}$$

If this value is 1 then x_i is classified as a positive and if it is -1 or 0 then x_i is classified as a negative. We know the true class of each x_i. We need to simply count how many times our classifier made a mistake and report this value as the error of our classifier. What is the error of your classifier? What is the percentage of error? The best percentage of error for this data set is 11.10% which is found from Bayes decision theory. How does your classifiers percentage of error compare? (Hit: your percentage of error should be around 12%.)

Plot the test data points that have been classified as 1 by equation (4) using blue circles and the test data that have been classified as -1 as red stars. Explain your results.

STEP 10. Repeat all previous steps for the Logit loss with $\sigma = 2$ and 4 iterations and Logit loss with $\sigma = 1$ and 7 iterations. Note again that you only need to change one line of code that is the weight update part of your code from step 5.

4 What to turn in

You should turn in a CD which includes (1) A report on your project (2) and a Matlab .m file of your program. Your report should include your results for each STEP. You should clearly divide your report into STEPs. Failing to do so will get you a zero grade! You Matlab program should have a comment for each line and each segment of code explaining what that line or segment of code does. If you fail to comment your code, you will get a zero grade! I should be able to run your code directly from the CD and see the plots and results. If your code has errors when I run it or does not work when I run it from the CD, you will get a zero grade!

5 CHEATING :-(

Copying a portion of someones report or Matlab code is considered cheating. If even a portion of your report or Matlab code is similar to someone else you will both get a zero on the project, be introduced to the university disciplinary committee (Komite Enzebaty) and possibly fail the course. SO DO NOT EVEN SHOW YOUR CODE OR REPORT TO ANYONE ELSE!