Vinay Gaba
UNI: vhg2105

COMS 4771: Machine Learning Assignment 3

1. **Pseudo code for training the Decision Stumps when training examples are given weights**

- The first step of the process is cleaning the data and converting all the 0 labels in the training data to -1.

```
[rs,cs] = size(train_data);
for i = 1:rs;
 if train_data(i,58) == 0
        train_data(i,58) = -1;
end
```

- Next, we pass this data to the *hw_train_adaboost(train data, num_rounds)* function. Here, we start by assigning equal weights to all the data points in the training data.

```
[rowSize,columnSize] = size(train_data);
weights = ones(rowSize,1) * (1/rowSize);
```

- *The hw_train_adaboost method consists of a parameter called num_rounds which drives the number of rounds this algorithm needs to run for. So the program iterates num_round times and makes a call to the train_decision_tree(train_data,decisionStumps,current_num_round,weights)* function.
- *Here is the logic for the train_decision_tree(train_data,decisionStumps,current_num_round,weights)* function.

- Iterate through every feature of the training data
- Consider every unique value in the feature as a possible threshold

```
uniqueValues = sort(unique(labelMatrix(:,1)));
```

- Divide the training data into a leftMatrix and rightMatrix based on whether the value of the training data for that feature is less than or greater than the threshold.

```
left = labelMatrix(labelMatrix(:,1)<=tempThreshold, 2:end);
right = labelMatrix(labelMatrix(:,1)>tempThreshold, 2:end);
```

- Compute the classification error for this split and repeat the above process for all unique values for all the features.

```
classError = classificationError(left,right);
if classError < minErrorForFeature;
  minErrorForFeature = classError;
  minThresholdForFeature = tempThreshold;
end
```

- Choose the threshold and feature which produces the least classification error. We will represent this value as globalMinimumThreshold.

```
[M,I] = min(minErrorForEveryFeature(:,1));
globalMinThreshold =  minErrorForEveryFeature(I,2);
```

- Now split the train data based on the globalMinimumThreshold in two matrices: splitMatrixLeft and splitMatrixRight.

```
splitMatrixLeft = train_data_with_weights(train_data(:,I) <= globalMinThreshold , : );
splitMatrixRight  =  train_data_with_weights(train_data(:,I)  >  globalMinThreshold  ,  :);
```

- Next, we are going to use the weights that were generated and count the total weight of label “1” and label ”-1” in both the matrices: splitMatrixLeft and splitMatrixRight. These weights are going to be stored in four variables, namely: leftPosLabelWeightCount, rightPosLabelWeightCount, leftNegLabelWeightCount, rightNegLabelWeightCount
- The cost of choosing a label a is the total weight of all examples labeled b.

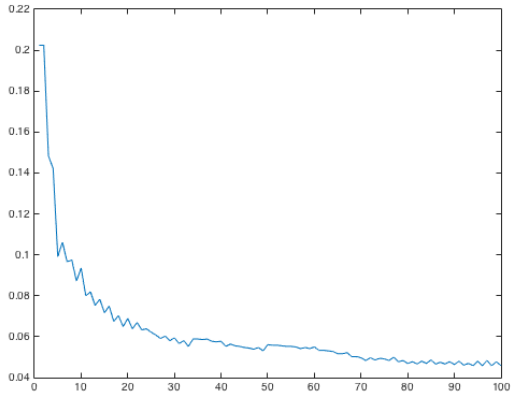
```
if(leftposLabelWeightCount <=  leftnegLabelWeightCount)
      mal = -1;
else
      mal = 1;
end

if(rightposLabelWeightCount <= rightnegLabelWeightCount)
      mar = -1;
else
      mar = 1;
end
```

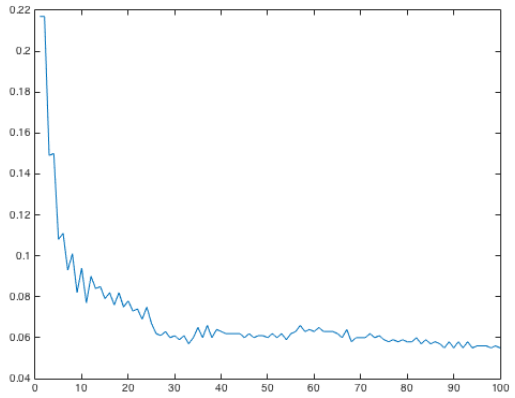
- Thus, all the examples which would have a value less than the threshold would be assigned the label of mal and all the examples that have a value greater than the threshold would be assigned the label of mar.

2. These are the Training Errors and Test Errors that I received when I ran the Adaboost program.

No of Rounds	Training Error	Test Error
1	20.24%	21.70%
2	20.24%	21.70%
4	14.22%	15%
8	9.75%	10.10%
16	7.50%	8.20%
32	5.80%	6.10%
64	5.28%	6.30%
100	4.58%	5.50%



Training Error



Test Error

3.

Polynomial regression is a form of linear regression in which the relationship between the independent variable *x* and the dependent variable *y* is modelled as an *n*th degree polynomial. Polynomial regression models are usually fit using the method of least squares.

The class of regression functions, Fd, is the set of all polynomial functions of x with degree bounded by d, where d is a parameter. Thus, the class can be defined as

$$\mathcal{F}_d = \{c_0 + c_1 x + c_2 x^2 \ldots + c_d x^d\}$$

The closed form for the objective function to solve the polynomial regression formula is given by the formula–

$$\arg \min_{f \in F_d} \frac{1}{n} \sum_{i=1}^{n} \big(y - f(x)\big)^2$$

1) The label vector y is defined by the vector Y
2) The function class f(x) represented by the vector XC

$$\begin{aligned} R(C) &= \frac{1}{n} \sum_{i=1}^{n} \big(y - f(x)\big)^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} (Y_i - X_i C_i)^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} \left(\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} 1 & x_1 \cdots x_1^k \\ \vdots & \vdots \quad \vdots \\ 1 & x_n \ \ldots x_n^k \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} \right)^2 \\ &= \frac{1}{n} \left\| \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} 1 & x_1 \cdots x_1^k \\ \vdots & \vdots \quad \vdots \\ 1 & x_n \ \ldots x_n^k \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} \right\|^2 \\ &= \frac{1}{n} \|Y - XC\|^2 \end{aligned}$$

To find the closed form we minimize the equation wrt C ie solving gradient = 0 –

$$\nabla_C R = 0$$

$$\nabla_C \left(\frac{1}{n} \|Y - XC\|^2 \right) = 0$$

$$\frac{1}{n} \nabla_C ((Y - XC)^T (Y - XC)) = 0$$

$$\frac{1}{n}\nabla_C(Y^TY - 2Y^TXC + C^TX^TXC) = 0$$
$$\frac{1}{n}(-2Y^TX + 2C^TX^TX) = 0$$
$$(\ldots.\frac{\partial \overrightarrow{Y}^T\overrightarrow{X}}{\partial \overrightarrow{X}} = \overrightarrow{Y}^T, \frac{\partial \overrightarrow{X}^T\overrightarrow{X}}{\partial \overrightarrow{X}} = 2\overrightarrow{X}, \frac{\partial \overrightarrow{X}^T C\overrightarrow{X}}{\partial \overrightarrow{X}}$$
$$= 2\overrightarrow{Y}^T(C + C^T))$$
$$X^TXC = X^TY$$
$$C = (X^TX)^{-1}X^TY$$

Based on the derivation above we can observe that the order – k polynomial regression looks the same as k-dimensional linear regression. I believe the polynomial regression closed form can be represented as –

$$C_i = ({X_i}^T X_i)^{-1}{X_i}^T Y \qquad \ldots.(where\ i = \{0 \ldots k\})$$

4.

- We know that D_{t+1} is given by the formula:

$$D_{t+1}(x, y) = D_t(x, y)e^{-\alpha\, y f_t(x)}$$

- AdaBoost ensures that the weights focus on the hard examples. It tries to assign more wait to misclassified examples and less weight to examples which were classified correctly.
- AdaBoost chooses a new distribution D_{t+1} on which the last base classifier h_t is sure to do extremely poorly.
- It can be shown by a simple computation that the error of h_t with respect to distribution D_{t+1} **is exactly $\frac{1}{2}$**, that is, exactly the trivial error rate achievable through simple random guessing.
- In this way, AdaBoost tries on each round to force the base learner to learn something new about the data.

$\sum D_{t+1}(x, y).\, yf_t(x)$

$= \sum D_t(x, y)e^{-\alpha\, yf_t(x)}.\, yf_t(x)$ [Substituting the value of D_{t+1}]

$= z\sum e^{-\alpha\, yf_t(x)}$ [Substituting the value of z]

5. These are the error rates that I got when I ran my experiments for training error and test error for 1 and 2 rounds.

No of Rounds	Training Error	Test Error
1	20.24%	21.70%
2	20.24%	21.70%

As you might notice, the error rates for both test and training error do not see any difference across both the rounds. In other cases, as we increment the number of rounds, we see a decrease in the error rates. There was only one exception to this rule when the Test error slightly increased for 64 rounds.

Citation

Referred this article - MIT Press

Collaborators

Dikha Vanvari dhv2108
Varun Shetty vs2567