Sane's STATS Academy of Statistics

Boosting

Ada Boost, Gradient Boost, Extreme Gradient Boost

Weak Learners & Strong Learner

- On any data, there can be algorithms which do not give the accuracy which we expect. e.g. Classification ML Algorithms with accuracy better than random guess (0.5) but not high. Such algorithms can neither be used for predicting on the data nor can those be ignored. We term such ML algorithms as **Weak Learners or Weak Hypothesis**.
- The algorithms having accuracy to reasonable extent which are worth deploying on production data are termed as **Strong Learners or Strong Hypothesis**.

Sane's STATS Academy of Statistics

What is Boosting?

- The term “Boosting” refers to a family of algorithms which converts weak learner to strong learners.
- An algorithm which gives less precision is referred as weak learner
- An algorithm which gives one of the highest accuracies is termed as strong learner
- Usually simple classification algorithms like Naïve Bayes, K nearest neighbour etc can be considered as weak learners
- We can iteratively apply the simple classifiers and combine their solution to obtain better prediction result

How does Boosting Work?

1. Apply a weak learner to the whole data. The weak learner will be having less precision.
2. For the observations which have been wrongly predicted, the algorithm pay more attention to them by giving them more weights before it applies the any other classifier (or even same classifier also)
3. Apply the next classifier to the data with weights applied to the wrongly classified observations
4. Repeat steps 2 and 3, until the precision is reached to a desired level

Boosting Algorithms

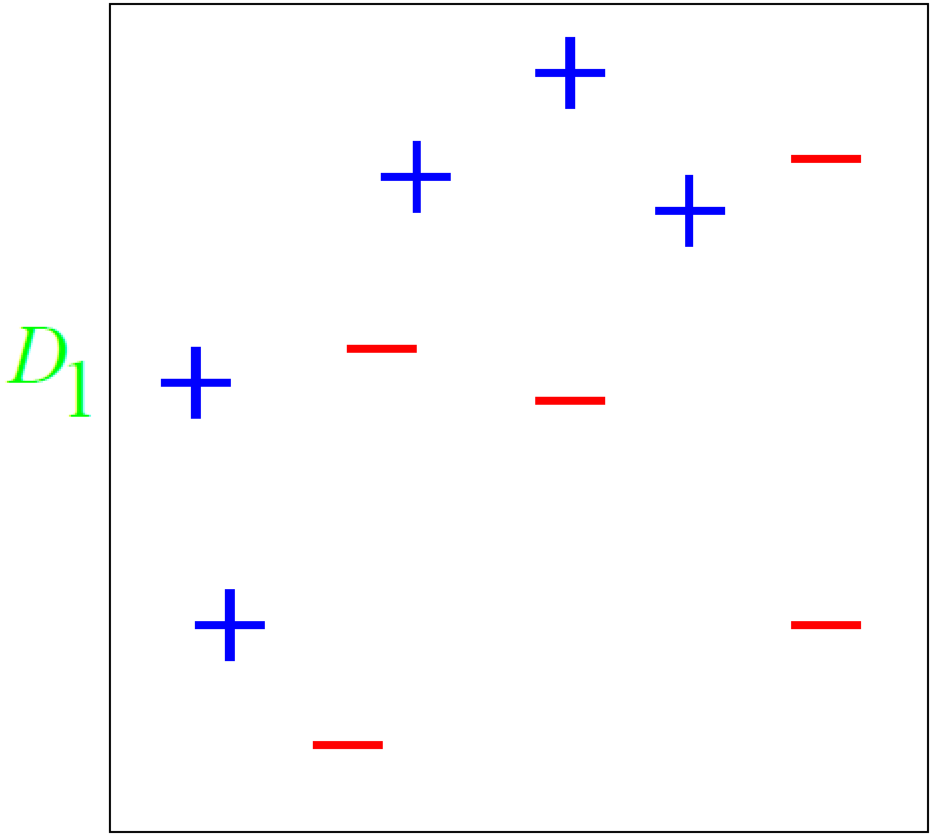
- The boosting algorithms that we are going to cover:
  - Adaptive Boosting (adaboost)
  - Gradient Boosting (GBM)
  - Extreme Gradient Boosting (XGBoost)

Adaptive Boosting

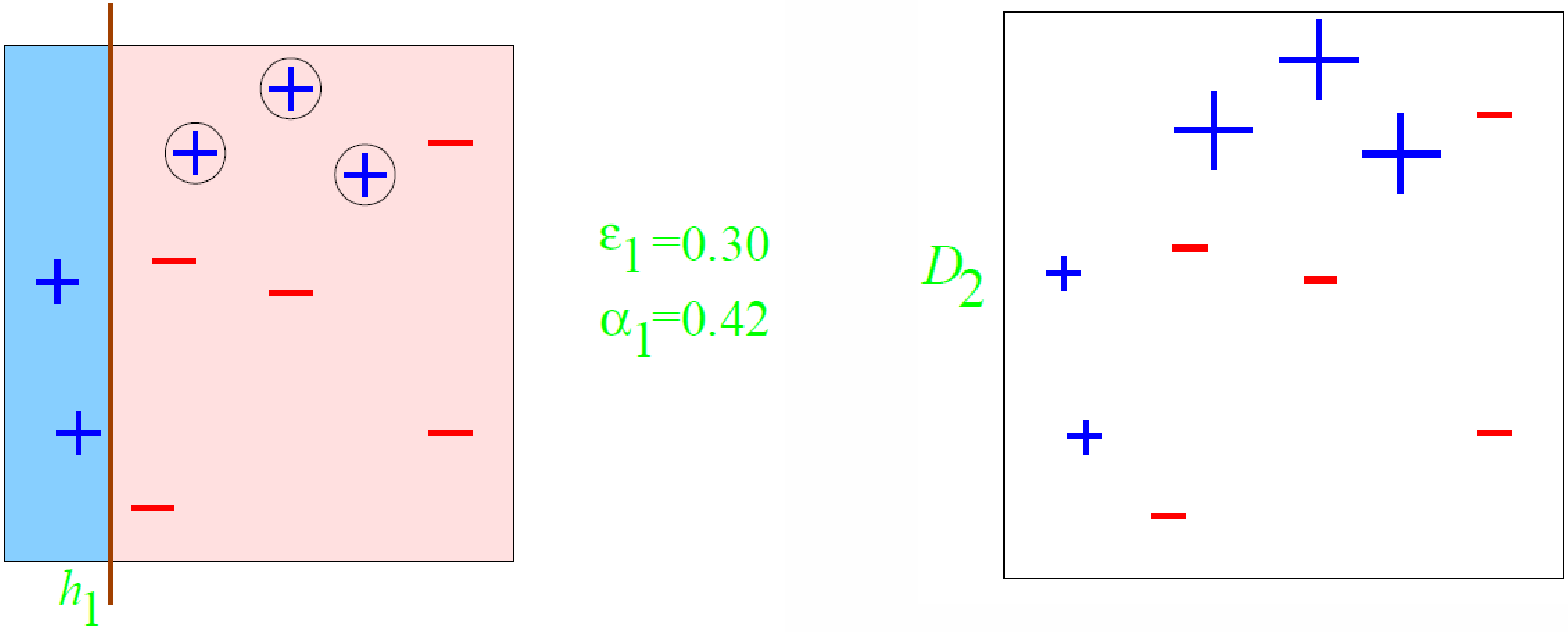
Adaptive Boosting

1. The process starts with a weak learner getting applied to the data
2. The observations in the data which cause errors are given more weight. In the other words, those are upweighted.
3. After upweighting, same the weak learner is again applied.
4. The result will be again having errors, hence the steps 2 and 3 are repeated until we have a sufficient accuracy in the result

Toy Example
by Yoav Freund & Rob Schapire

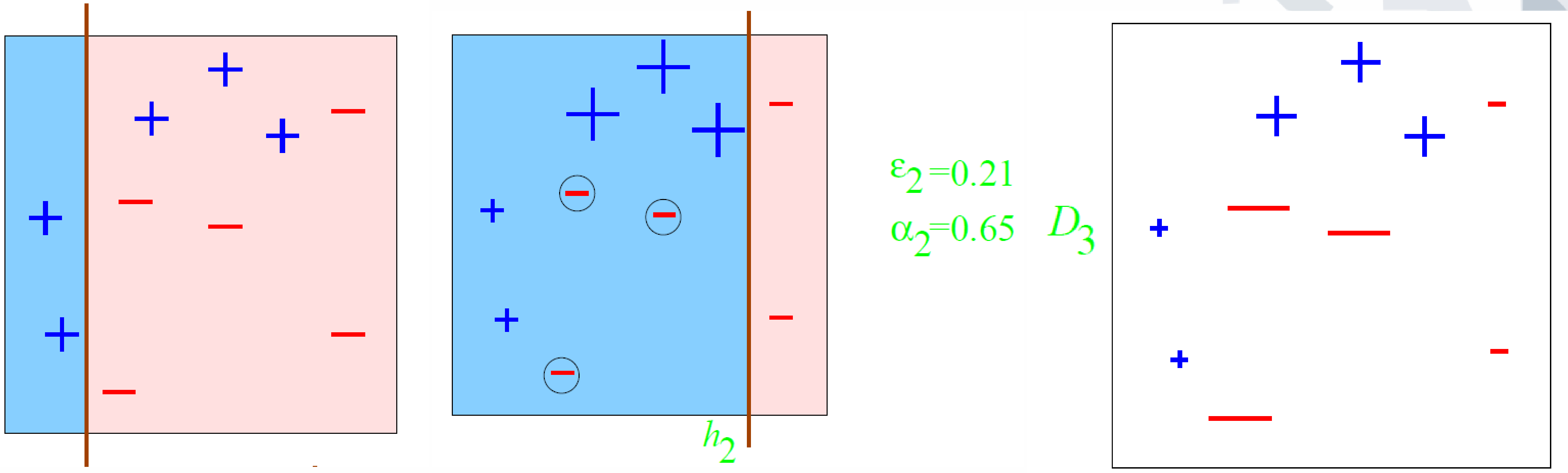


Toy Example : Round 1

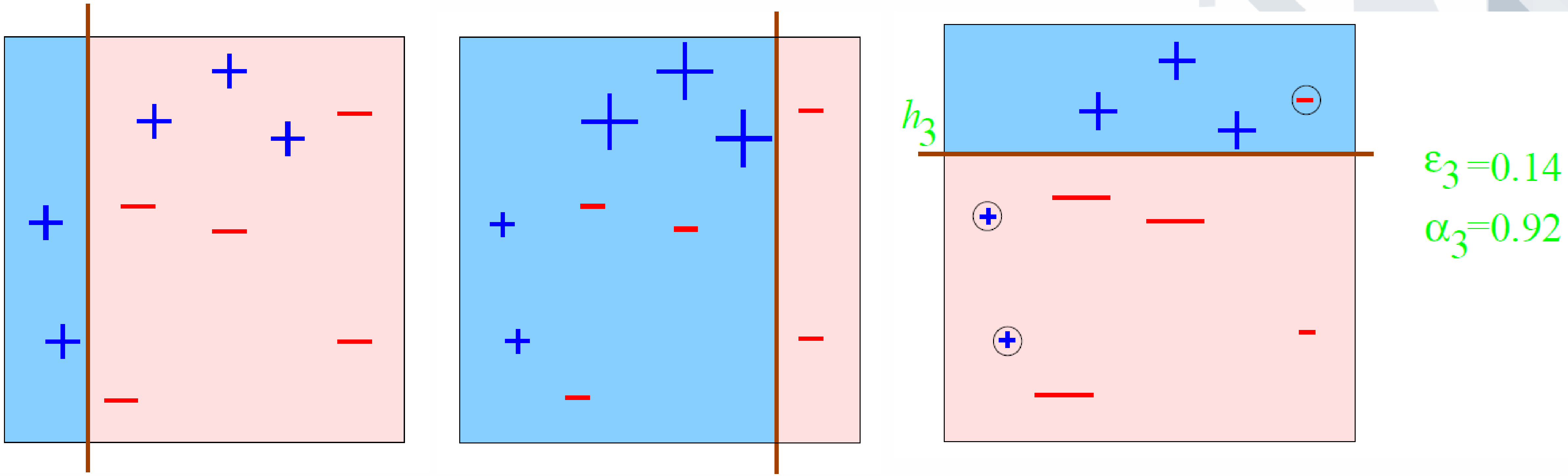


Misclassified observations upweighted

Round 2

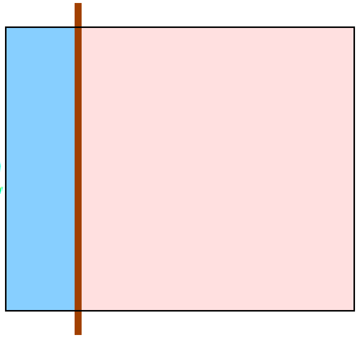
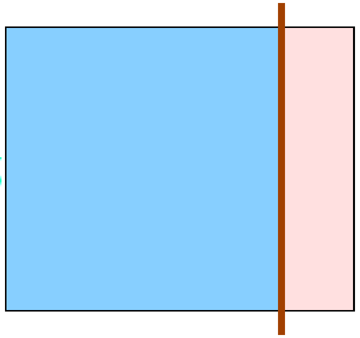
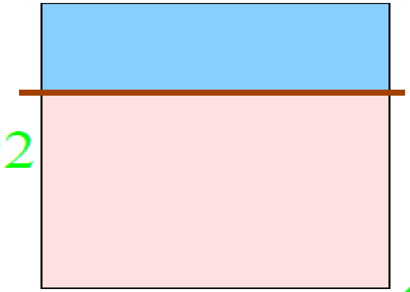


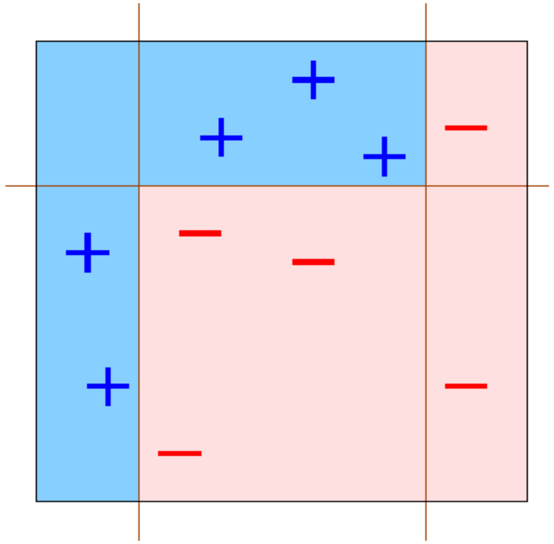
Round 3



Final Hypothesis

H_{final}

$= \text{sign}\Big(0.42$  $+ 0.65$  $+ 0.92$  $\Big)$

$=$ 

Ada Boost in R

- There are several packages implementing adaptive boosting:
  - Function ada from package ***ada*** : Binary Response
  - Function boosting from package ***adabag*** : Response with two and more than two classes.
  - Function glmboost from package ***mboost*** : Continuous and Discrete Numeric Response

Function **ada** from package ***ada***

Syntax :

```
ada(x, y, loss=c("exponential","logistic"), type=c("discrete","real","gentle"),
iter=50,…)

## S3 method for class 'formula'
ada(formula, data, ...)
```

- x : matrix of predictors
- y : vector of response
- There are subtypes of Ada Boost hence the different values of parameters **loss** and **type**.
- The weak learner taken by function ada is rpart.

Example

Dataset Sonar from package ***mlbench***

```r
library(ada)
model.ada <- ada(Class ~ . , data = training)
pred.ada <- predict(model.ada, newdata = validation, type = "vector")
confusionMatrix(pred.ada, validation$Class,
                dnn=c("Predicted", "Actual"))
```

```
Confusion Matrix and Statistics

         Actual
Predicted  M  R
        M 25  7
        R  8 22

               Accuracy : 0.7581
                 95% CI : (0.6326, 0.8578)
    No Information Rate : 0.5323
    P-Value [Acc > NIR] : 0.0002126

                  Kappa : 0.5151
 Mcnemar's Test P-Value : 1.0000000

            Sensitivity : 0.7576
            Specificity : 0.7586
         Pos Pred Value : 0.7812
         Neg Pred Value : 0.7333
             Prevalence : 0.5323
         Detection Rate : 0.4032
   Detection Prevalence : 0.5161
      Balanced Accuracy : 0.7581

       'Positive' Class : M
```

Ada Boost References

- http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Eric-Boosting304FinalRpdf.pdf
- http://rob.schapire.net/papers/explaining-adaboost.pdf
- https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/
- http://www.inf.fu-berlin.de/inst/ag-ki/adaboost4.pdf

Sane's STATS Academy of Statistics

Gradient Boosting

What is Gradient Boosting?

- It is an generalization of Ada Boost
- It makes use of gradient descent method

Gradient Boosting = Gradient Descent + Boosting

How GBM works?

Gradient boosting involves three elements:

1. A loss function which is to be optimized.
2. A weak learner used to make predictions.
3. An additive model to add weak learners to minimize the loss function.

Loss Function

- Depends on the problem being solved. Can be "laplace”, “tdist”, “poisson” etc.
- It must differentiable i.e. if its is y then its $\frac{dy}{dx}$ must exist

Weak Learner

- Decision Trees are used as weak learner in GBM
- Trees are constructed in a greedy manner, which choose the best split based on impurity index (classification) or based deviance(regression).
- Weak Learners should be constraint on parameters such as a maximum number of layers, nodes, splits or leaf nodes to ensure that the learners remain weak.

Additive Model

- While the trees are added to the model, the existing trees are not changed.
- Gradient Descent Method is used to minimize the loss when trees get added.
- Gradient Descent Method minimizes a cost function $J(\bar{\theta})$ (equivalent of squared error) and finds those optimal values of parameters $\bar{\theta}$ which minimize the cost function.
- Instead of parameters, here we have weak learners. On calculating the loss, a tree is added to the model which reduces the loss.
- The output for the new tree is then added to the output of the existing sequence of trees in order to correct or improve the final output of the model.
- A fixed number of trees are added or training may stop once loss reaches an acceptable level or when there is no scope for improvement on an external validation dataset.

GBM in R

- Some of the ways that GBM can be implemented in R :
  - Package ***mboost***
  - Package ***gbm***
- We will cover package ***gbm*** implementation

Syntax :

gbm(formula, data, …)

Example of GBM in R

```r
training$Class <- ifelse(training$Class == "M", 1, 0)
library(gbm)
model.gbm <- gbm(Class ~ . , data = training,
                 distribution = "bernoulli")
pred.gbm <- predict(model.gbm , newdata=validation,
                    n.trees = 100,
                    type = "response")
pred.gbm.class <- factor(ifelse(pred.gbm > 0.5 , "M","R"))
confusionMatrix(pred.gbm.class, validation$Class,
                dnn=c("Predicted", "Actual"))
```

```
Confusion Matrix and Statistics

         Actual
Predicted  M  R
        M 32 21
        R  1  8

               Accuracy : 0.6452
                 95% CI : (0.5134, 0.7626)
    No Information Rate : 0.5323
    P-Value [Acc > NIR] : 0.04813

                  Kappa : 0.2563
 Mcnemar's Test P-Value : 5.104e-05

            Sensitivity : 0.9697
            Specificity : 0.2759
         Pos Pred Value : 0.6038
         Neg Pred Value : 0.8889
             Prevalence : 0.5323
         Detection Rate : 0.5161
   Detection Prevalence : 0.8548
      Balanced Accuracy : 0.6228

       'Positive' Class : M
```

Tuning Parameters for GBM

- Number of trees (n.trees):
- Shrinkage parameter (shrinkage):
- Number of splits (interaction.depth):

Gradient Boost

- http://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/
- http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html
- https://www.r-bloggers.com/free-gradient-boosting-lecture/
- https://en.wikipedia.org/wiki/Gradient_boosting

Extreme Gradient Boosting

- Extreme Gradient Boosting (xgboost) is similar to gradient boosting framework but more efficient.
- It has both methods, linear model solving as well as tree learning algorithms.
- It has the capacity to do parallel computation on a single machine.
- It is that’s why, at least **10 times faster** than existing gradient boosting implementations.
- It can be used for classification as well as regression

Data Preparation for XG Boost

- XG Boost works for only numeric predictors
- The categorical variables need to be converted into dummy variables of 0 and 1
- Consider here an example of our Telecom customers

XG Boost

- https://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/
- https://www.r-bloggers.com/an-introduction-to-xgboost-r-package/
- http://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
- http://xgboost.readthedocs.io/en/latest/model.html