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 accuaracy 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**.

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)
   - Cat Boost
    - Light GBM

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
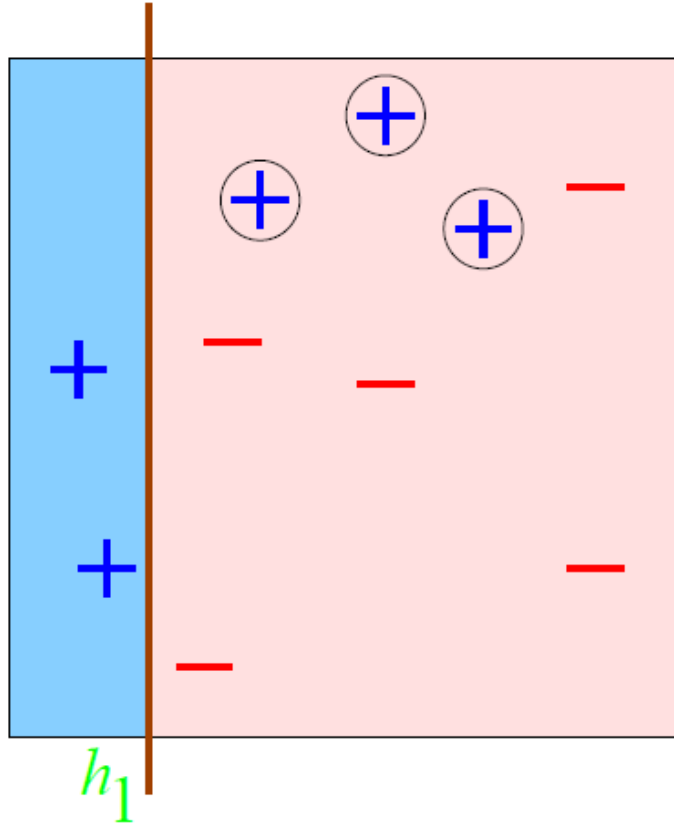
Toy Example
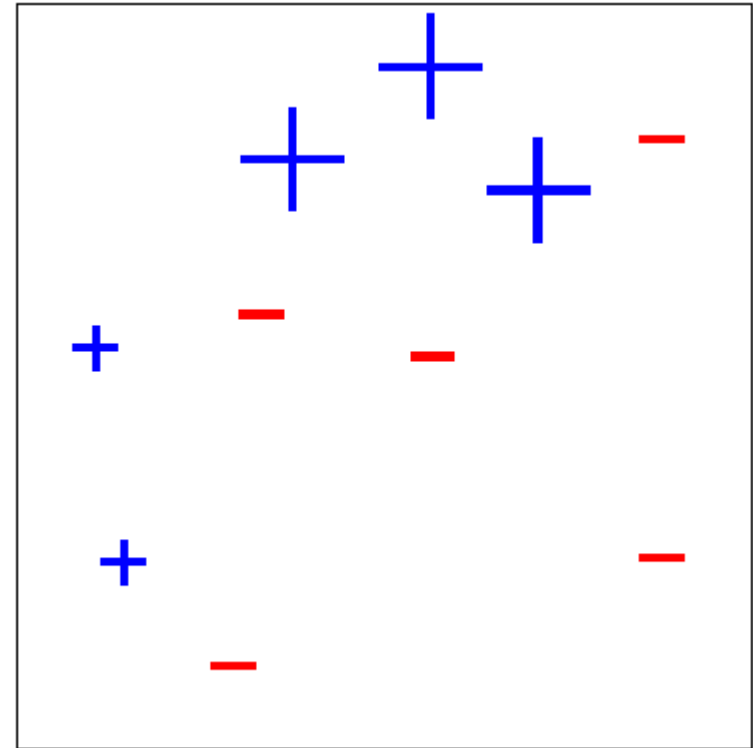## by Yoav Freund & Rob Schapire
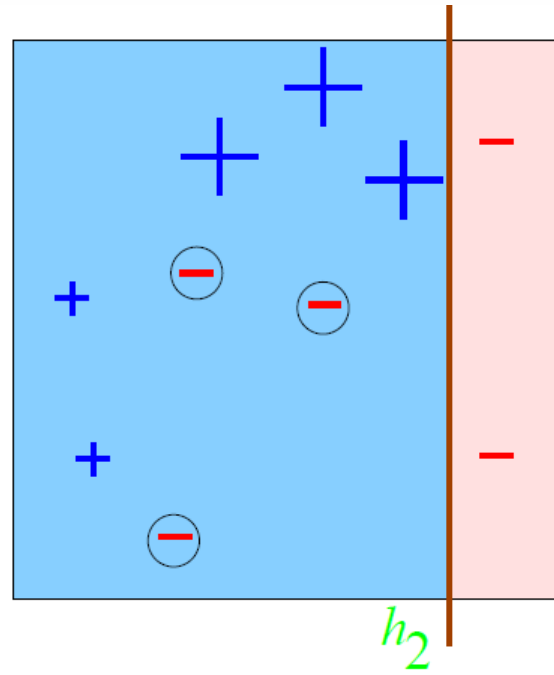
D_1
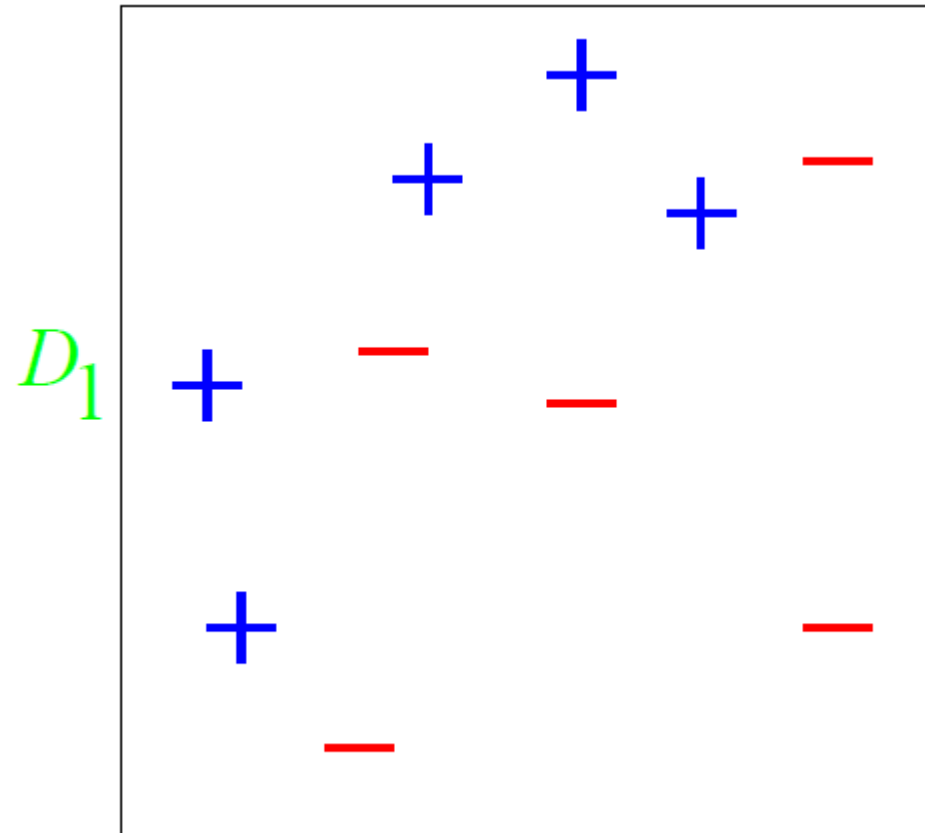
Toy Example : Round 1

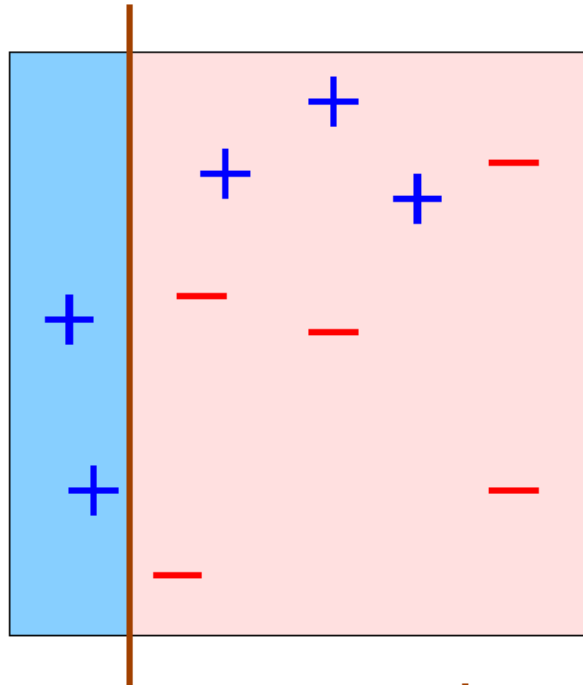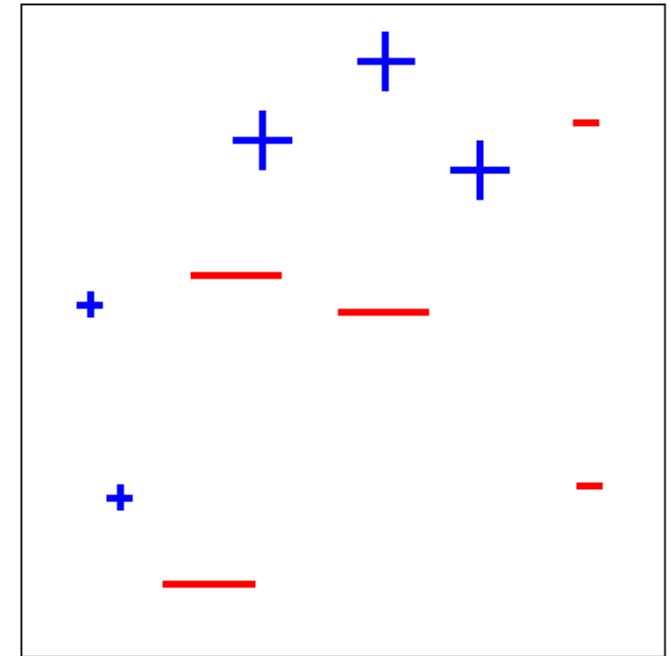
$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$
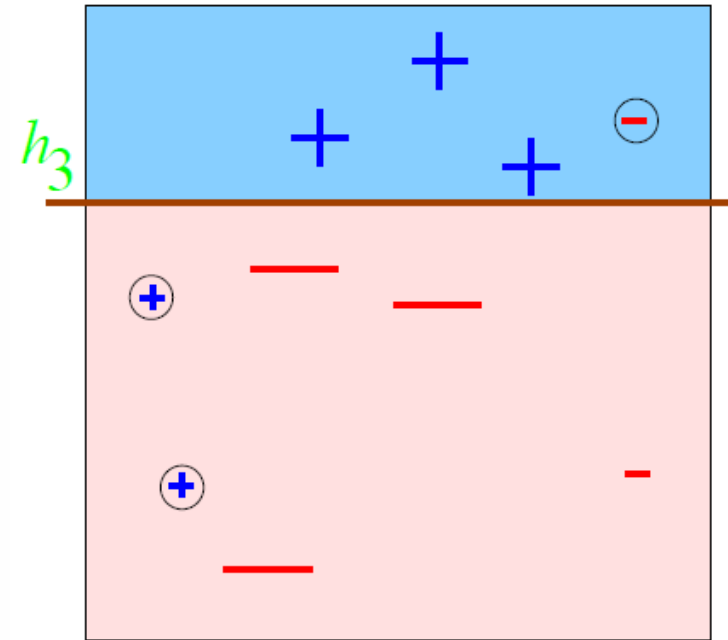
D_2

h_1

Misclassified observations upweighted
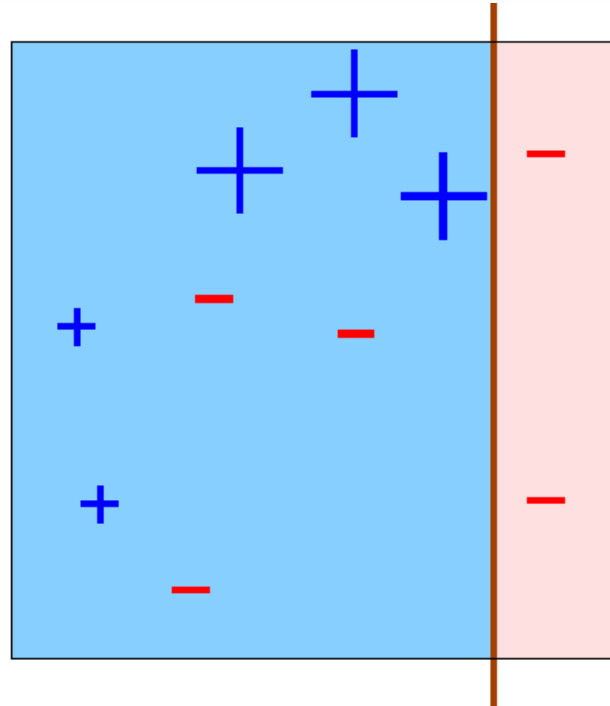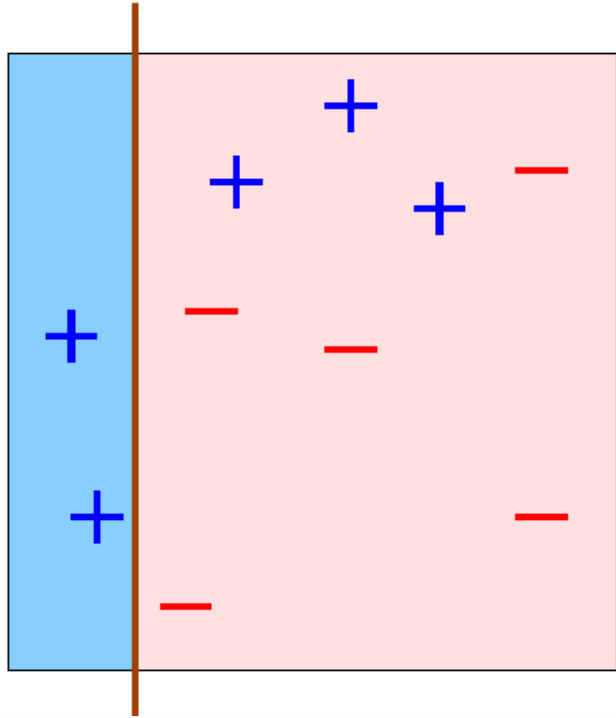
Round 2

$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$ D_3

h_2

Round 3

h_3

$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

Final Hypothesis

$$H_{final} = \text{sign}\left(0.42 \quad \square \quad + 0.65 \quad \square \quad + 0.92 \quad \square \right)$$

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
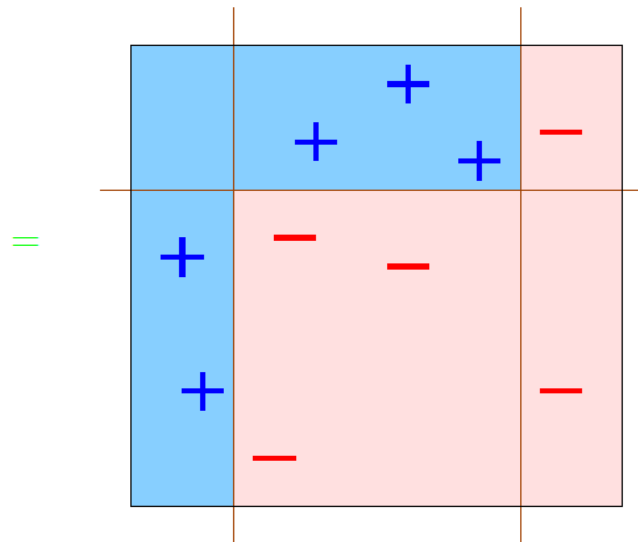
Gradient Boosting

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 θ 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.

Tuning Parameters for GBM

- Number of trees (*n_estimators)*
- Maximum Depth(*max_depth*)
- Learning Rate (*learning_rate)*

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

- https://www.youtube.com/watch?v=zblsrxc7XpM

XGBoost

Extreme Gradient Boosting

Extreme Gradient Boosting

- Extreme Gradient Boosting (xgboost) is similar to gradient boosting framework but more efficient.

- Originally written in C++

- 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, **faster** than existing gradient boosting implementations.

- It can be used for classification as well as regression

Data Preparation for XG Boost

- This algorithm requires that we change the class type of the data to DMatrix which is supported by XGBoost

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