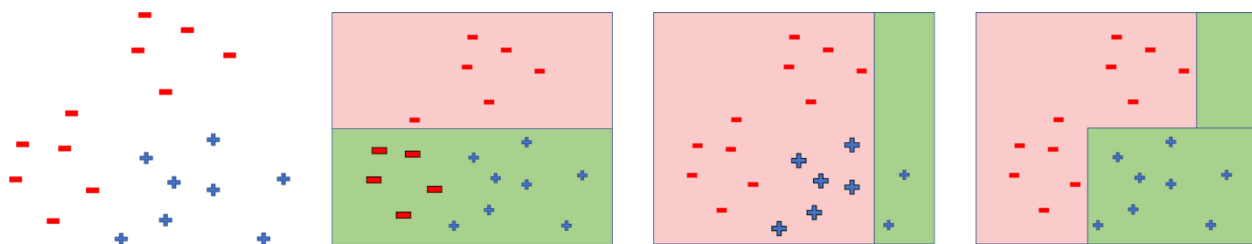


BOOSTING



1. ADABOOST

- Classes should be +1 and -1
- Train N weak learners
- Build 1st weak learner
 - get misclassified points
 - increase the weight of these points
 - weight increasing helps in increasing the probability of getting picked for building the model
- Build 2nd weak learner
 - Hope the above misclassified points get classified correctly
 - get misclassified points
 - increase the weight of these points
 - weight increasing helps in increasing the probability of getting picked for building the model
- and so on
- Build Nth weak learner
 - Hope the above misclassified points get classified correctly
 - get misclassified points
 - increase the weight of these points
 - weight increasing helps in increasing the probability of getting picked for building the model
- Ensemble them.

- **Formulae:**

$$\alpha_t = \frac{1}{2} \log_e \left(\frac{1 - \text{error}_t}{\text{error}_t} \right); \text{ Where, } \text{error}_t = 1 - \text{accuracy}_t$$

Weights Updation:

$$D(t + 1) = D(t) * e^{(-\alpha_t * Y * h_t(x))}$$

$$D(t + 1) = \frac{D(t + 1)}{\sum D(t + 1)}$$

Final model :

$$H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t * h_t(x) \right)$$

Where,

α_t = Weight of the model_t

$h_t(x)$ = model_t Predictions

T = number of learners

- **Algorithm:**

Initialize mean weights to each point

i.e. $\frac{1}{N}$ for all the N points in the sample data

For i in $[T]$ Trees

Build i^{th} Tree on the weighted data points

Calculate $\text{accuracy}_i = \frac{(TP+FP)}{(TP+FP+TN+FN)}$

Calculate $\text{error}_i = 1 - \text{accuracy}_i$

Calculate $\alpha_i = \frac{1}{2} \log_e \left(\frac{(1 - \text{error}_i)}{\text{error}_i} \right)$

Update Weights

$$D(t + 1) = D(t) * e^{(-\alpha_t * Y * h_t(x))}$$

$$D(t + 1) = \frac{D(t + 1)}{\sum D(t + 1)}$$

End For Loop

Final Model

$$H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t * h_t(x)\right)$$

- **Note:**
 - α_t is *+ve* for a good classifier (more the better)
 - α_t is 0 for 50% accurate classifier
 - α_t is *-ve* for bad classifier (less the weaker)
 - If α_t is *-ve* (weak learner):
 - if a point is wrongly classified: $D(t)$ increases a little
 - if a point is correctly classified: $D(t)$ decreases by a lot
 - If α_t is *+ve* (strong learner):
 - if a point is wrongly classified: $D(t)$ increases a lot
 - if a point is correctly classified: $D(t)$ decreases a little

2. GRADIENT BOOSTING

- **STEP: 1 (BUILDING)**

- Build Decision Stump Classifier and Predict or Predict as mean value of train target value
- For each iteration
 - Calculate gradient
 - $gradient = sigmoid(y_{pred}) - Y$
 - $cbind(X, gradient)$ - build Decision Stump Regressor with $maxdepth = 2$ and $minsplit = 2$
 - store the model in a list
 - predict the gradient on X ($predictedGradient$)
 - update the prediction using learning rate:
 - $y_{pred} = y_{pred} - learningRate * predictedGradient$

- **STEP: 2 (PREDICTION)**

- For each tree
 - predict the gradient
 - update the gradient using learning rate:
 - $gradient = gradient * learningRate$
 - update the final prediction as:
 - $y_{pred} - gradient$
- squash it using soft max
 - $\frac{e^x}{\sum e^x}$

- **Link**

- <https://github.com/kartheekpnsn/machine-learning-from-scratch/blob/master/R/gradient-boosting.R>

3. GBM vs ADABOOST

- In GBM, first learner to classify the points then Calculates loss then Builds second to predict the loss after first step then Adjusts predictions, Builds loss after second step... and so on...
- If a learner misclassifies a sample, the weight of the learner is reduced and the weight of the sample point increases. It will repeat such process until converge.

4. GBM vs XGBOOST

- parallelized inside each tree
- handles missing values
- regularization
- tree pruned from maximum *depth*