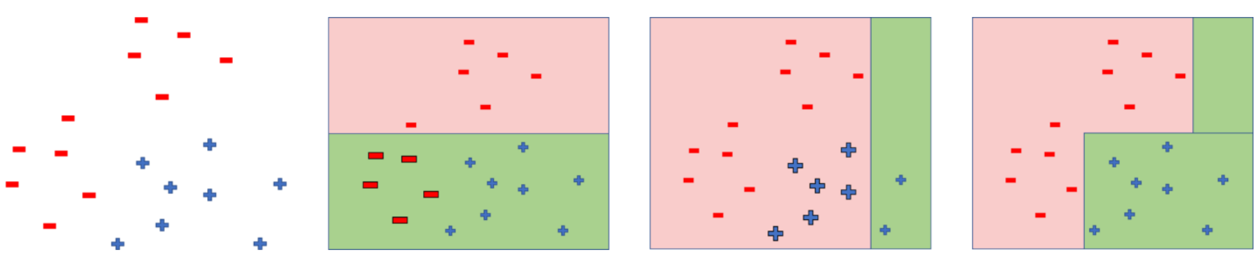
# BOOSTING



## 1. ADABOOST

- Classes should be +1 and -1
- Train N weak learners
- Build $1^{st}$ 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 $2^{nd}$ 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 $N^{th}$ 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:**

$$\alpha_t = \frac{1}{2}\log_e\left(\frac{1 - error_t}{error_t}\right);\ Where, error_t = 1 - accuracy_t$$

***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*** :

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

$Where,$

$\alpha_t = Weight\ of\ the\ model_t$

$h_t(x) = model_t\ Predictions$

$T = number\ of\ learners$

- **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 $accuracy_i = \frac{(TP+FP)}{(TP+FP+TN+FN)}$

Calculate $error_i\ =\ 1\ -\ accuracy_i$

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

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

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

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

## 2. GRADIENT BOOSTING

- **Original:** $Y$
- **Base model:** $Yhat\ (for\ each\ class)$
- $for\ each\ class$
  - As our base model is a weak learner:
    - $Y\ not\ equal\ to\ Yhat$
    - i.e. $Y = Yhat + error$
    - i.e. $error = Y - Yhat$
    - i.e. $residual = Y - Yhat$
  - Now, we try building a regressor model to predict residuals, Say $h_1(x)$
    - $Y = Yhat + h_1(x) +\ residual$
  - Then try building one more model to predict the new residuals, Say $h_2(x)$
    - $Y = Yhat +\ h_1(x) + h_2(x) + residual$
  - Now, for n estimators
    - $Y = Yhat +\ h_1(x) + h_2(x) + \cdots + h_n(x)$
  - Here the loss function we use is Huber Loss. Because, Huber loss handles outliers whereas squared error penalizes the outliers at huge cost
    - $L = \frac{1}{2}(Y - Yhat)^2$
    - $\frac{\partial L}{\partial Yhat} = -\frac{2}{2}\ (Y - Yhat)$
    - $\frac{\partial L}{\partial Yhat} = -\ (Y - Yhat)$
    - $\frac{\partial L}{\partial Yhat} = -residual$
  - **Mapping Gradient Descent to the Boosting Algorithm**
    - $Y = Yhat + h_1(x)$
    - $Y = Yhat + (Y - Yhat)$
    - $Y = Yhat - 1(\frac{\partial L}{\partial Yhat})$
    - where $learning\ rate = 1$

- **Prediction**
  - $for\ each\ class$
    - Apply first model on the new data and predict the gradient
    - $Yhat = -h_1(x)$
    - Apply the second model and predict the next gradient
    - $Yhat = -h_1(x) - h_2(x)$
    - For n estimators
    - $Yhat = -h_1(x) - h_2(x) - \cdots - h_n(x)$
  - Apply Soft max $\frac{e^x}{\sum e^x}$ on each row so that we will have probabilities summed up to '1' for all the classes

- **Link**
  - [https://github.com/kartheekpnsn/machine-learning-from-scratch/blob/master/R/gradient-boosting.R](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 $depth$