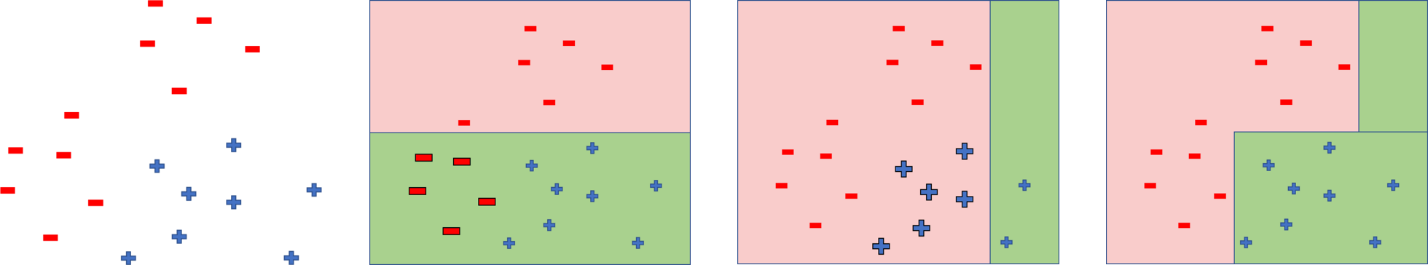
**BOOSTING**



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:**
* **Algorithm:**

Initialize mean weights to each point

i.e. for all the N points in the sample data

Build Tree on the weighted data points

Calculate

Calculate

Calculate

Update Weights

End For Loop

Final Model

* **Note:**
* is for a good classifier (more the better)
* is for accurate classifier
* is for bad classifier (less the weaker)
* If is (weak learner):
  + if a point is wrongly classified: decreases by little
  + if a point is correctly classified: increases a lot
* If is (strong learner):
  + if a point is wrongly classified: increases a lot
  + if a point is correctly classified: decreases by little

1. **GRADIENT BOOSTING**

* **Original:**
* **Base model:** (
  + As our base model is a weak learner:

    - i.e.
    - i.e.
    - i.e.
  + Now, we try building a regressor model to predict residuals, Say
  + Then try building one more model to predict the new residuals, Say
  + Now, for n estimators
  + Here the loss function we use is Huber Loss. Because, Huber loss handles outliers whereas squared error penalizes the outliers at huge cost
  + **Mapping Gradient Descent to the Boosting Algorithm**
    - where
* **Prediction**
  + - Apply first model on the new data and predict the gradient
    - Apply the second model and predict the next gradient
    - For n estimators
  + Apply Soft max 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*

1. **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.

1. **GBM vs XGBOOST**

* parallelized inside each tree
* handles missing values
* regularization
* tree pruned from