Mining Massive Datasets

B. Ravindran

Reconfigurable and Intelligent Systems Engineering (RISE) Group
Department of Computer Science and Engineering

Robert Bosch Centre for Data Science and Artificial Intelligence (RBC-DSAI)
Indian Institute of Technology Madras

Ensembles

Problem Setting

$$X^{(1)} = \langle 0.15, 0.25 \rangle, Y^{(1)} = -1$$
$$X^{(2)} = \langle 0.4, 0.45 \rangle, Y^{(2)} = +1$$
$$\vdots$$

The input can be thought of as $X = \left(X_1, X_2, \cdots, X_p\right)$

The X_i are the features that describe the input.

The output is either a categorical variable, typically denoted by G or a continuous variable denoted by Y

The i-th instance of the input is denoted as x_i and output is denoted as y_i

We loosely state the learning problem as given a value of input X make a good prediction $\hat{Y}$ of Y or $\hat{G}$ of G

Problem Setting

$\chi \subseteq \Re^p$ is the input space

$X = \left(X_1, X_2, \cdots X_p\right)$ is a random variable describing the input

$\Upsilon \subseteq \Re$ or Γ is the output space

Y is a random variable describing the output

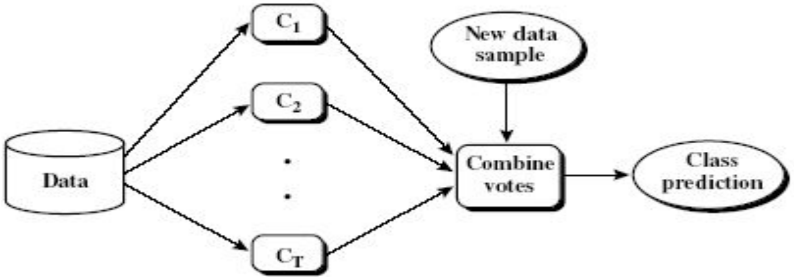
$p(X,Y)$ is the data distribution

$p(X,Y) = p(Y|X)p(X)$

$p(Y|x)$ is the predicted output probabilities given an input x

Ensemble Methods: Increasing the Accuracy

- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, $M_1, M_2, \ldots, M_k$, with the aim of creating an improved model M^*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers



Bagging: Bootstrap Aggregation

- Training
  - Given a set D of d tuples, at each iteration i, a training set D_i of d tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample $\boldsymbol{X}$
  - Each classifier M_i returns its class prediction
  - The bagged classifier M^* counts the votes and assigns the class with the most votes to $\boldsymbol{X}$
- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noisy data: not considerably worse, more robust
  - Proved improved accuracy in prediction
- More *Stable*

Bagging: Example

Random Forests

Bagging Decision Trees

1. If the number of cases in the training set is N, sample N cases at random - but *with replacement*, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number $m<<M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

Why Random Forests Work?

- The *correlation* between any two trees in the forest needs to be low. **Increasing the correlation increases the forest error rate**.
- A tree with a low error rate is a *strong* classifier. **Increasing the strength of the individual trees decreases the forest error rate.**
- Reducing m reduces both the correlation and the strength. Increasing it increases both.
- Somewhere in between is an "optimal" range of m - usually quite wide.

Features of Random Forest

- Excellent accuracy among current algorithms.
- Runs efficiently on large data bases.
- Can handle thousands of input variables without variable deletion.
- Gives estimates of what variables are important in the classification.
- Generates an internal unbiased estimate of the generalization error as the forest building progresses.

Error Estimates

- Out-of-Bag Error Estimates
  - We know that about 27% of the data is not sampled in bootstrap
  - This is true for each tree
  - Hence each data point is not used in about a third of the trees.
- Measure average error on each data point from the trees not using them
  - All these trees vote on the class of this data point
  - If the predicted majority class doesn’t match the true class then error
- Shown to be unbiased sample of the error

Committee Methods

- Takes a simple unweighted average of the predictions from each model
- Assigns equal probability to each model.
- Applicable in cases where the different models arise from the same parametric model, with different parameter values.

Stacking

- Learning methods are “stacked” on top of one another .
- Combines multiple models’ output with estimated optimal weights.
- Leads to better prediction.
- Train a “predictor of predictors”
  - Treat individual predictors as features
  - Similar to multi-layer perceptron idea
  - Special case: binary, f_e linear => weighted vote

$\hat{y}_1 = f_1(x_1,x_2,\ldots)$

$\hat{y}_2 = f_2(x_1,x_2,\ldots)$ => $\hat{y}_e = f_e(\hat{y}_1, \hat{y}_2, \ldots)$

…

Boosting

- Focus new learners on examples that others get wrong
- Train learners sequentially
- Errors of early predictions indicate the “hard” examples
- Focus later predictions on getting these examples right
- Combine the whole set in the end
- Convert many “weak” learners into a complex predictor

Boosting

- How boosting works?
  - Weights are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to pay more attention to the training tuples that were misclassified by M_i
  - The final M^* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data
  - Can be shown to maximize margin of classifier

AdaBoost (Freund and Schapire, 1997)

- The current linear combination of classifiers is

$$C_{(m-1)}(x_i) = \alpha_1 k_1(x_i) + \alpha_2 k_2(x_i) + \cdots + \alpha_{m-1} k_{m-1}(x_i)$$

- Extend it to,

$$C_m(x_i) = C_{(m-1)}(x_i) + \alpha_m k_m(x_i)$$

- Total cost, or total error, of the extended classifier as the exponential loss

$$E = \sum_{i=1}^{N} \mathrm{e}^{-y_i (C_{(m-1)}(x_i) + \alpha_m k_m(x_i))}$$

Source: AdaBoost and the Super Bowl of Classifiers
A Tutorial Introduction to Adaptive Boosting -Ra ́ul Rojas

AdaBoost

- Since our intention is to draft k_m we rewrite the above expression as,

$$E = \sum_{i=1}^{N} w_i^{(m)} \mathrm{e}^{-y_i \alpha_m k_m(x_i))}$$

where

$$w_i^{(m)} = \mathrm{e}^{-y_i C_{(m-1)}(x_i)}$$

- Split the sum into two sums

$$E = \sum_{y_i = k_m(x_i)} w_i^{(m)} \mathrm{e}^{-\alpha_m} + \sum_{y_i \neq k_m(x_i)} w_i^{(m)} \mathrm{e}^{\alpha_m}$$

- Simplify the notation to

$$E = W_c \mathrm{e}^{-\alpha_m} + W_e \mathrm{e}^{\alpha_m}$$

Source: AdaBoost and the Super Bowl of Classifiers
A Tutorial Introduction to Adaptive Boosting -Ra ́ul Rojas

AdaBoost - Weighting

- To determine weight of m[th] classifier,

$$\frac{\mathrm{d}E}{\mathrm{d}\alpha_m} = -W_c \mathrm{e}^{-\alpha_m} + W_e \mathrm{e}^{\alpha_m}$$

- Equating it to zero,

$$\alpha_m = \frac{1}{2} \ln \left(\frac{\mathrm{W_c}}{\mathrm{W_e}} \right)$$

- Rewriting, with W as the total sum of weights,

$$\alpha_m = \frac{1}{2} \ln \left(\frac{\mathrm{W} - \mathrm{W_e}}{\mathrm{W_e}} \right) = \frac{1}{2} \ln \left(\frac{1 - \mathrm{e_m}}{\mathrm{e_m}} \right) \qquad \text{Where } e_m = W_e / W$$

Source: AdaBoost and the Super Bowl of Classifiers
A Tutorial Introduction to Adaptive Boosting -Ra ́ul Rojas

AdaBoost - Algorithm

AdaBoost
For $m = 1$ to M

1. Select and extract from the pool of classifiers the classifier k_m which minimizes

$$W_e = \sum_{y_i \neq k_m(x_i)} w_i^{(m)}$$

2. Set the weight α_m of the classifier to

$$\alpha_m = \frac{1}{2}\ln\left(\frac{1-\mathrm{e_m}}{\mathrm{e_m}}\right)$$

where $e_m = W_e/W$

3. Update the weights of the data points for the next iteration. If $k_m(x_i)$ is a miss, set

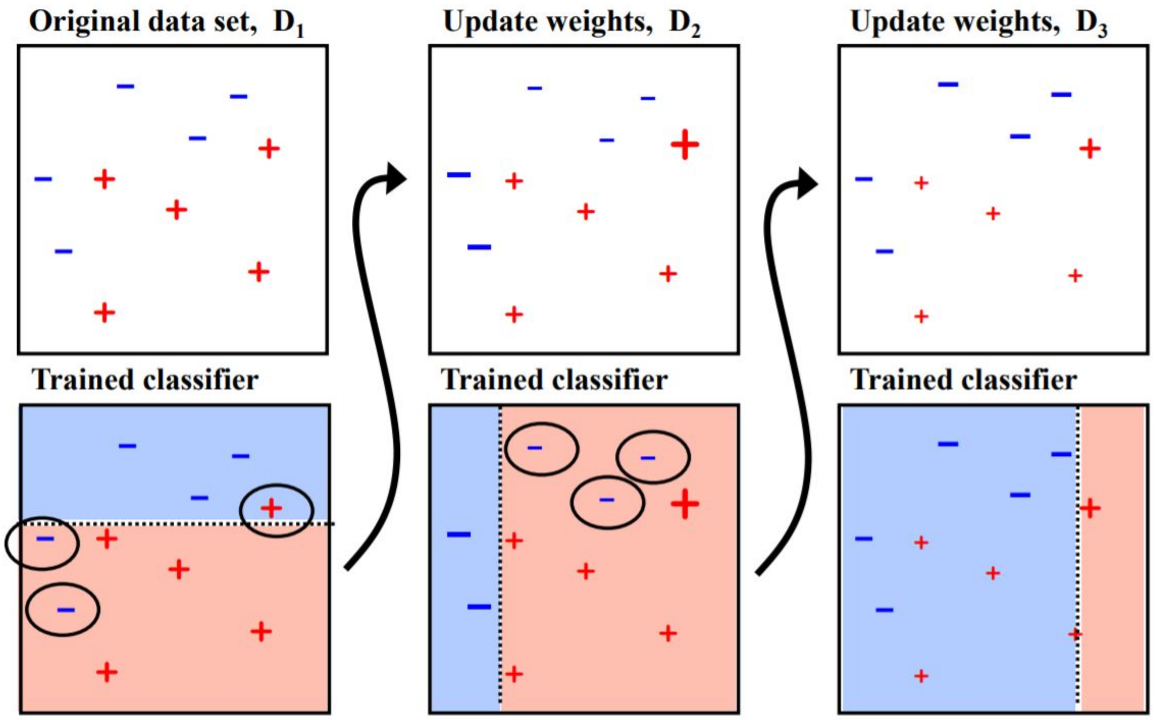
$$w_i^{(m+1)} = w_i^{(m)}\mathrm{e}^{\alpha_m} = w_i^{(m)}\sqrt{\frac{1-e_m}{e_m}}$$

otherwise

$$w_i^{(m+1)} = w_i^{(m)}\mathrm{e}^{-\alpha_m} = w_i^{(m)}\sqrt{\frac{e_m}{1-e_m}}$$

Credits : ESL

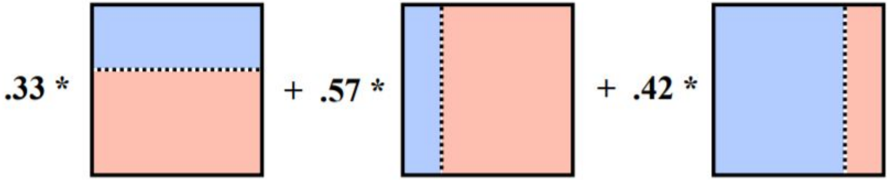
AdaBoost - Example



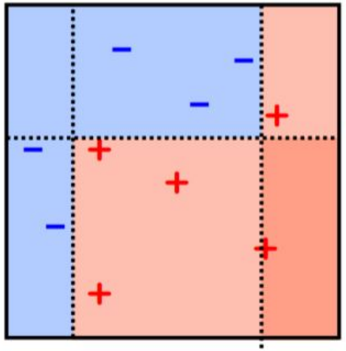
Source : Slides from Dr.Kalev Kask's (UCI) lecture on Ensembles of Learners

AdaBoost - Example

Weight each classifier and combine them:

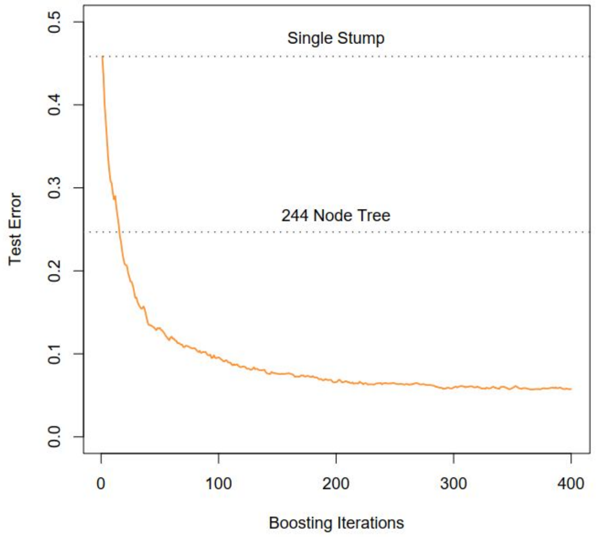


Combined classifier

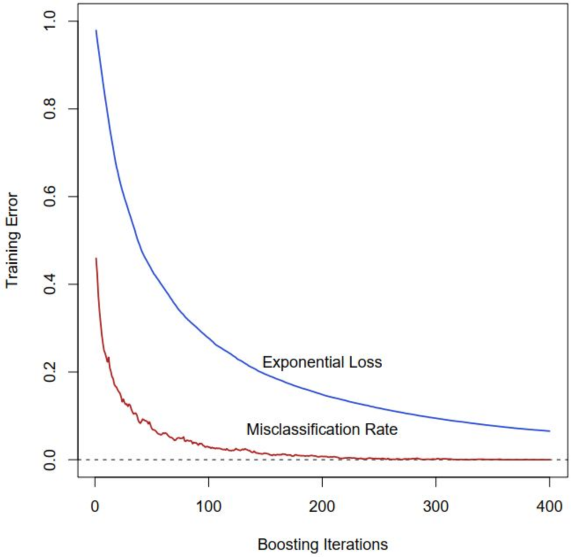


Source : Slides from Dr.Kalev Kask's (UCI) lecture on Ensembles of Learners

Adaboost



Simulated data test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.



Simulated data, boosting with stumps

Credits : ESL

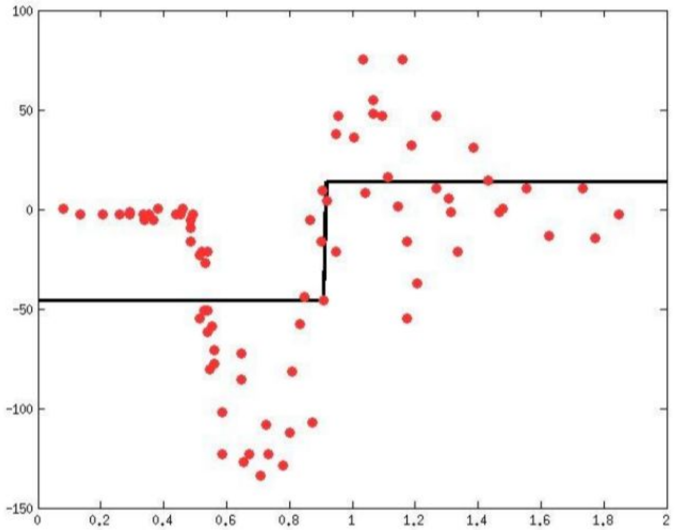
Gradient Boosting

- Learn sequence of predictors
- Subsequent models predict the error residual of the previous predictions
- Sum of predictions is increasingly accurate
- Predictive function is increasingly complex

Gradient Boosting - Example

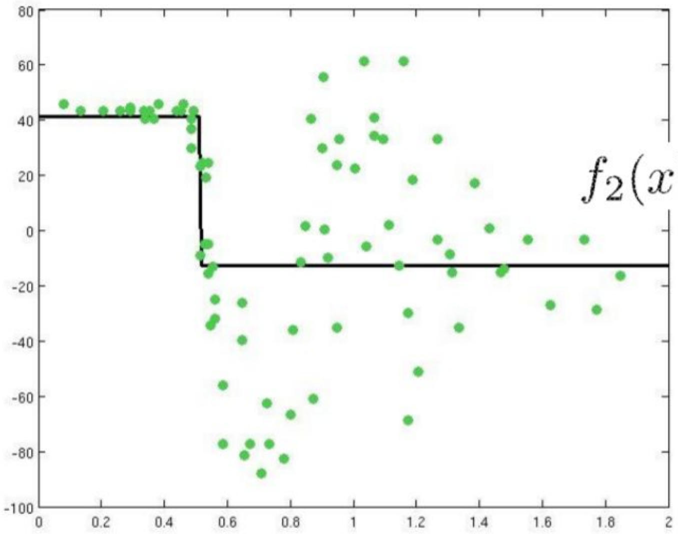
Learn a simple predictor…

$$f_1(x^{(i)}) \approx y^{(i)}$$



Then try to correct its errors

$$\epsilon^{(i)} = y^{(i)} - f_1(x^{(i)})$$



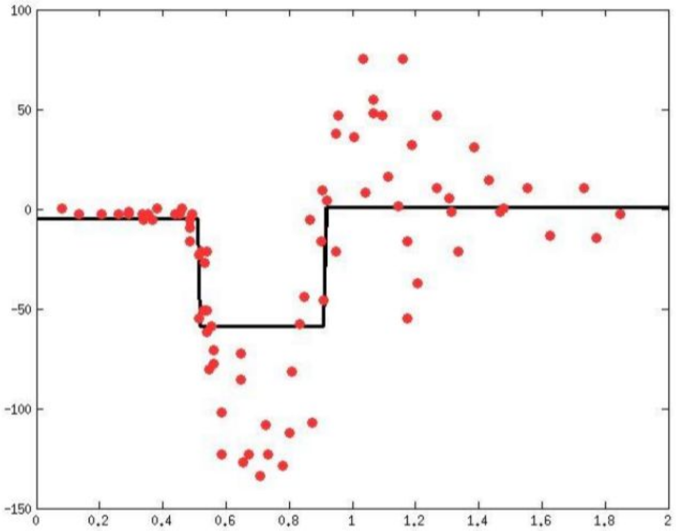
$$f_2(x^{(i)}) \approx \epsilon^{(i)}$$

Source : Slides from Dr.Kalev Kask's (UCI) lecture on Ensembles of Learners

Gradient Boosting - Example

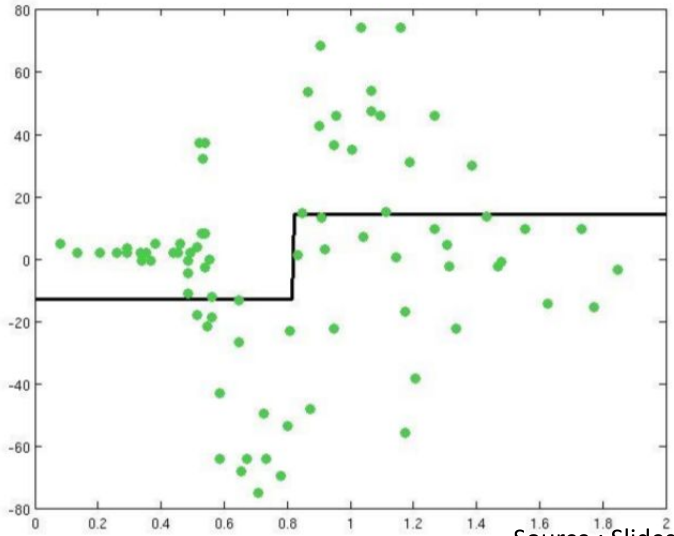
Combining gives a better predictor…

$$\Rightarrow \quad f_1(x^{(i)}) + f_2(x^{(i)}) \approx y^{(i)}$$



Can try to correct its errors also, & repeat

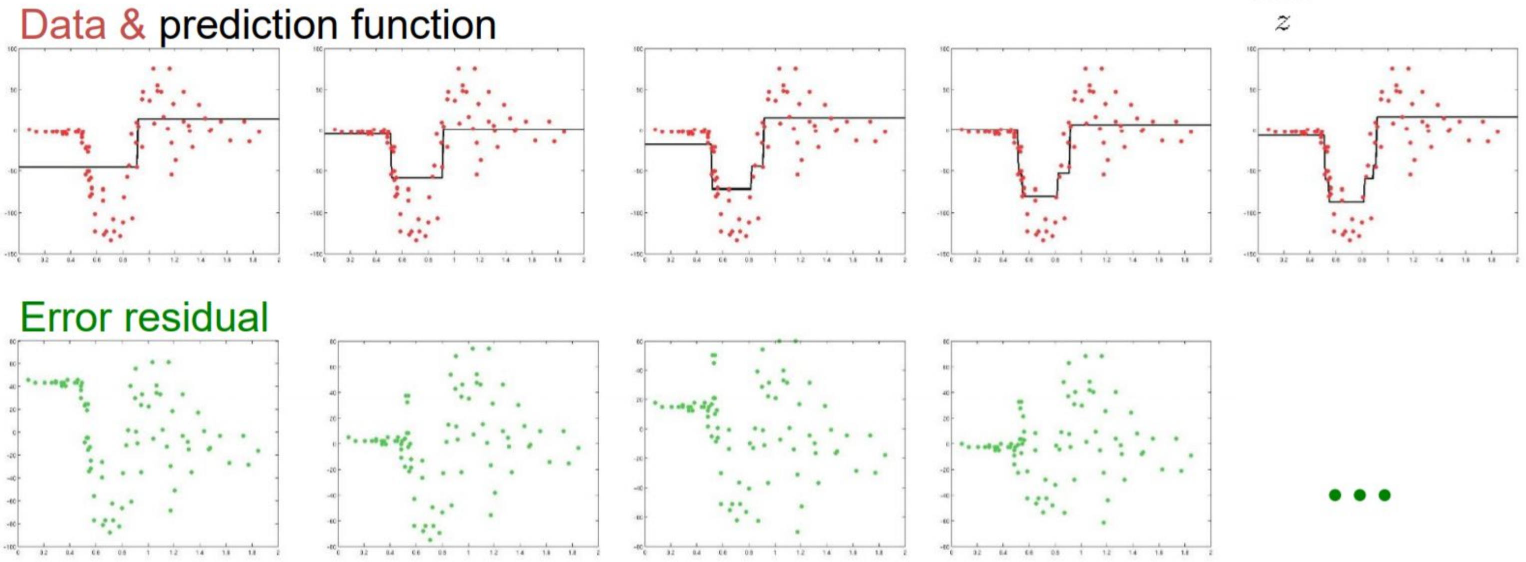
$$\epsilon_2^{(i)} = y^{(i)} - f_1(x^{(i)} - f_2(x^{(i)}) \quad \ldots$$



Source : Slides from Dr.Kalev Kask's (UCI) lecture on Ensembles of Learners

Gradient Boosting - Example

$$y^{(i)} \approx \sum_z f_z(x^{(i)})$$



Source : Slides from Dr.Kalev Kask's (UCI) lecture on Ensembles of Learners

Gradient Boosted Trees

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(y_i, \gamma)$.
2. For $m = 1$ to M:

   (a) For $i = 1, 2, \ldots, N$ compute

   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \ldots, J_m$.

   (c) For $j = 1, 2, \ldots, J_m$ compute

   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

Credits : ESL

Summary

- Ensembles yield powerful classifiers
- Random forest is one of the best performing classifiers
- Bagging results in more stable classifiers
  - Can be parallelized
- Boosting gives better performance
  - Can overfit
- Gradient Boosted Decision Trees are very powerful