Boosting

Boosting

Key Ideas

- Sequentially training weak learners
  - Weak learners perform slightly better than random

- Each new weak classifier built must strategically cover for other classifiers weaknesses
  - 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

- Averaging the results of each classier built to get the final result
  - The weight of each classifier's vote is a function of its accuracy

Boosting

Bagging	Boosting
Parallelly training strong learners	Sequentially train weak learners
Samples selected to build each model are randomly selected	Samples selected to build each model are selected based on their weight
Each model is built independently which prevents overfitting	Sequentially trained models can achieve higher accuracy at the risk of overfitting
Each classifier has equal weightage in final result	The weight of each classifier is a function of its accuracy
The aim of combining models is to reduce the variance of each individual model	The aim of combining models is to reduce the bias of each individual model created

Boosting

Models built using boosting

- Adaboost
- Gradient Boosted trees
- Extreme Gradient Boosting (XGBoost)
- Catboost
- Logitboost

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

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

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

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

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,

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

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 $e_m = W_e/W$

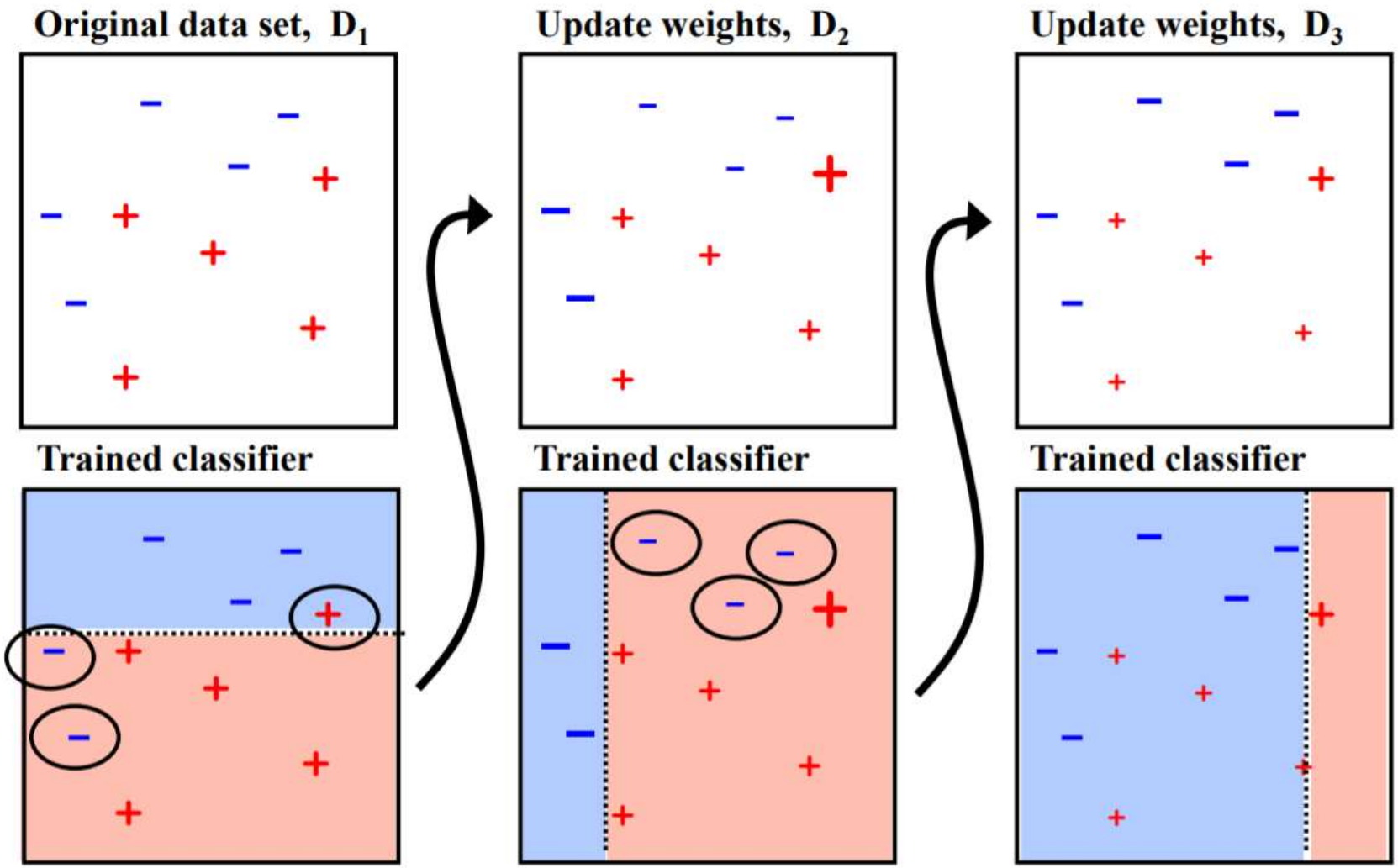
3. Update the weights of the data points for the next iteration. If $k_m(x_i)$ is a miss, set

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

otherwise

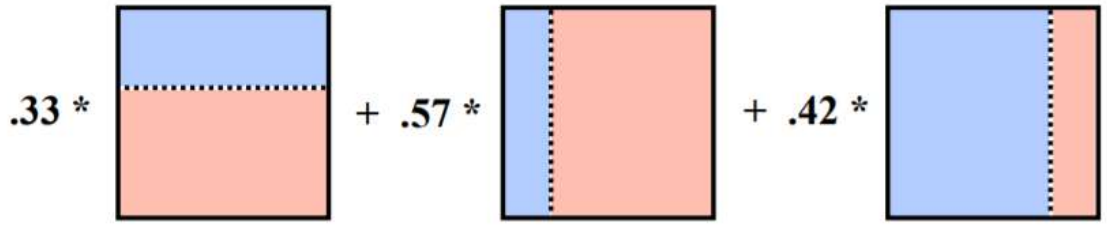
$$w_i^{(m+1)} = w_i^{(m)}\mathrm{e}^{-\alpha_m} = w_i^{(m)}\sqrt{\frac{e_m}{1 - e_m}}$$

ADABOOST - EXAMPLE

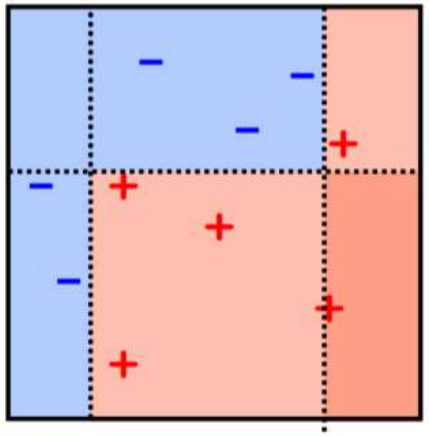


ADABOOST

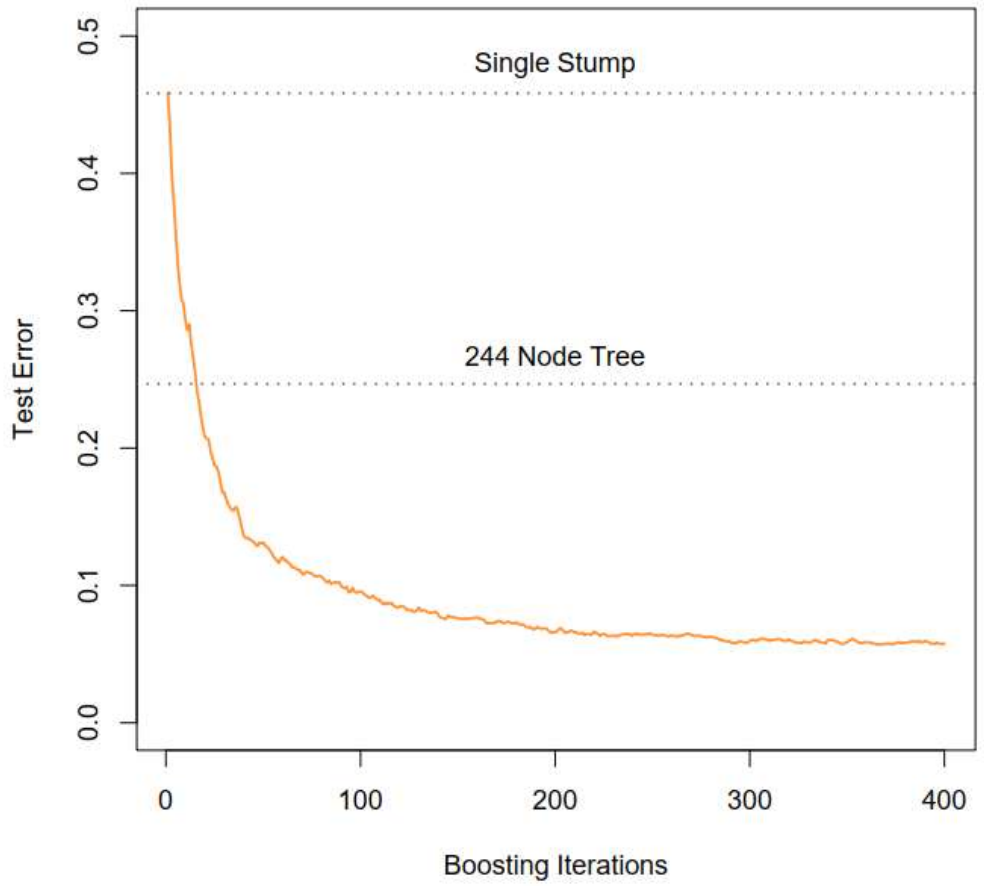
Weight each classifier and combine them:



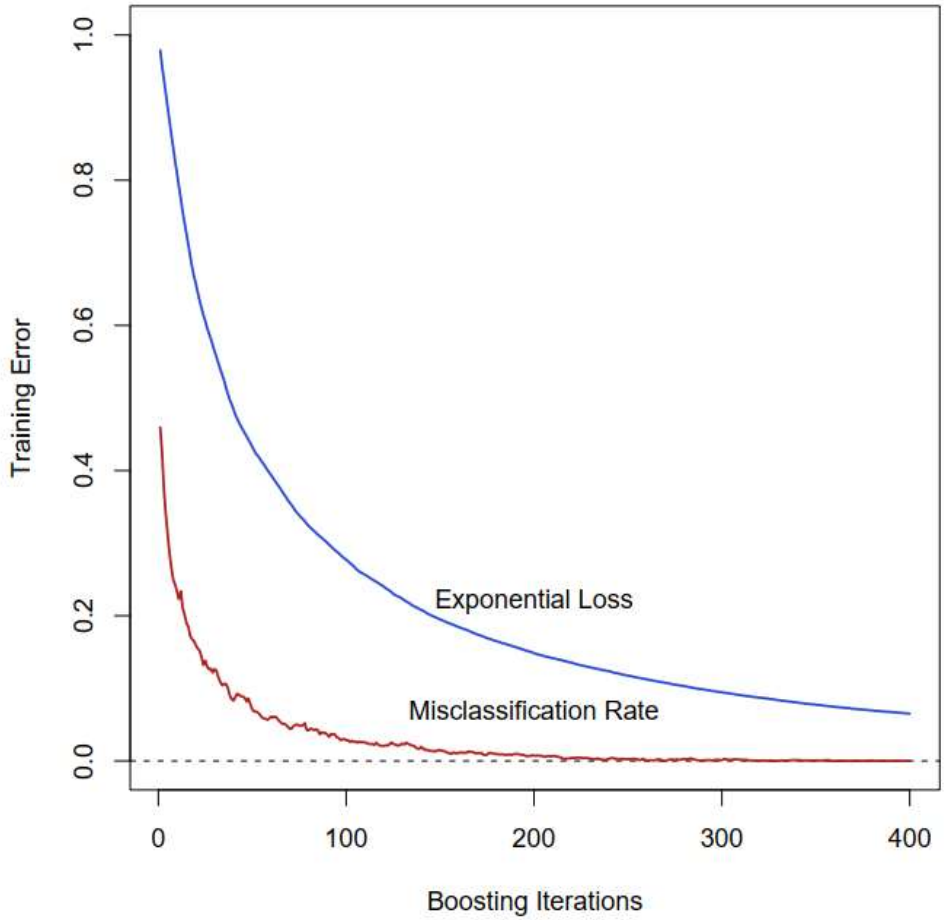
Combined classifier



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

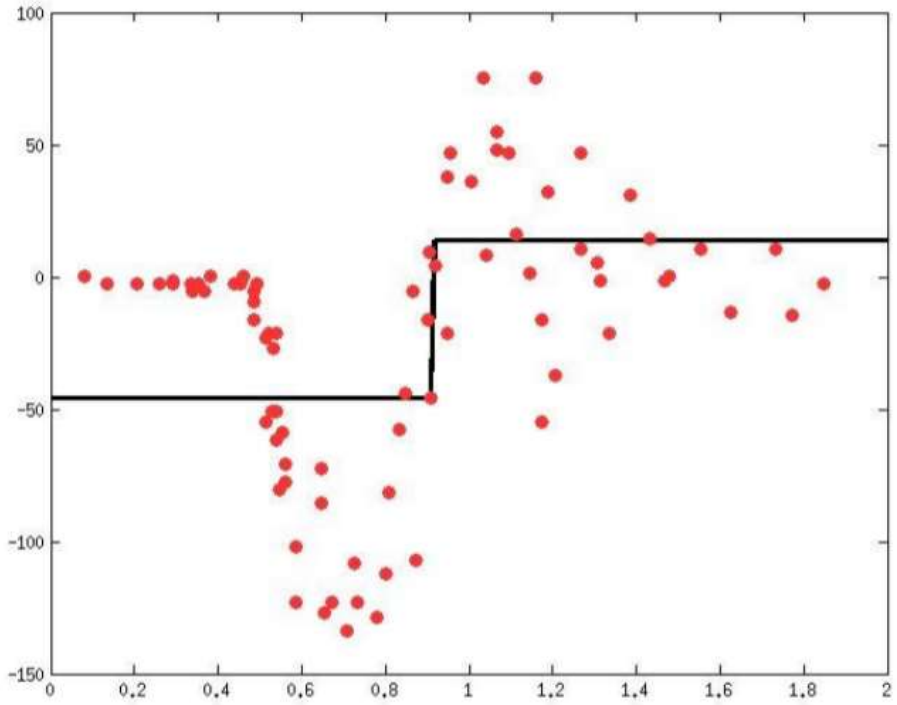
Gradient Boosting

- 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

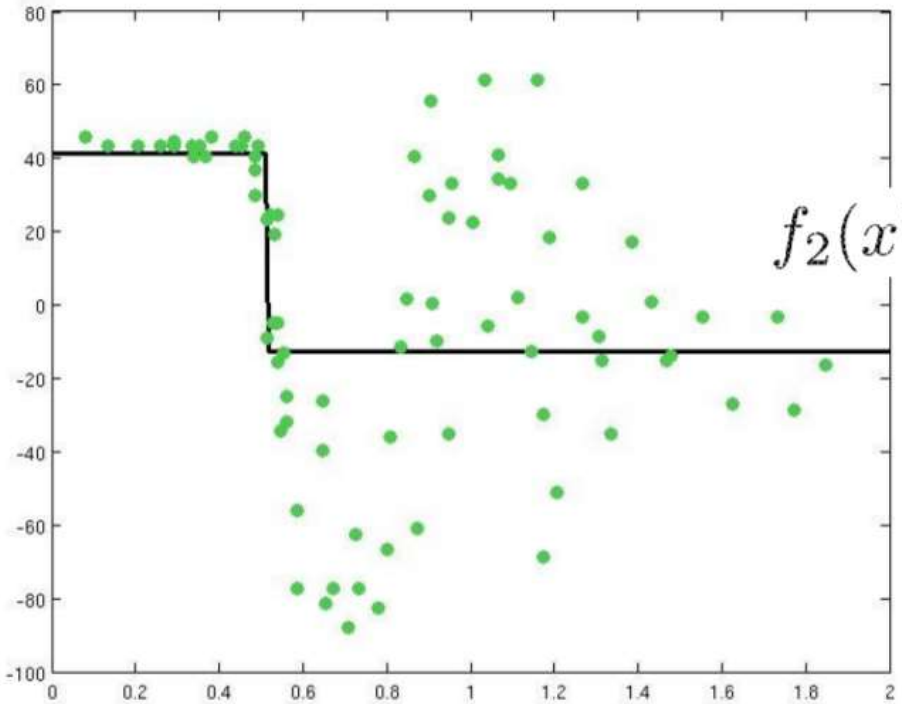
Learn a simple predictor…

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



Then try to correct its errors

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

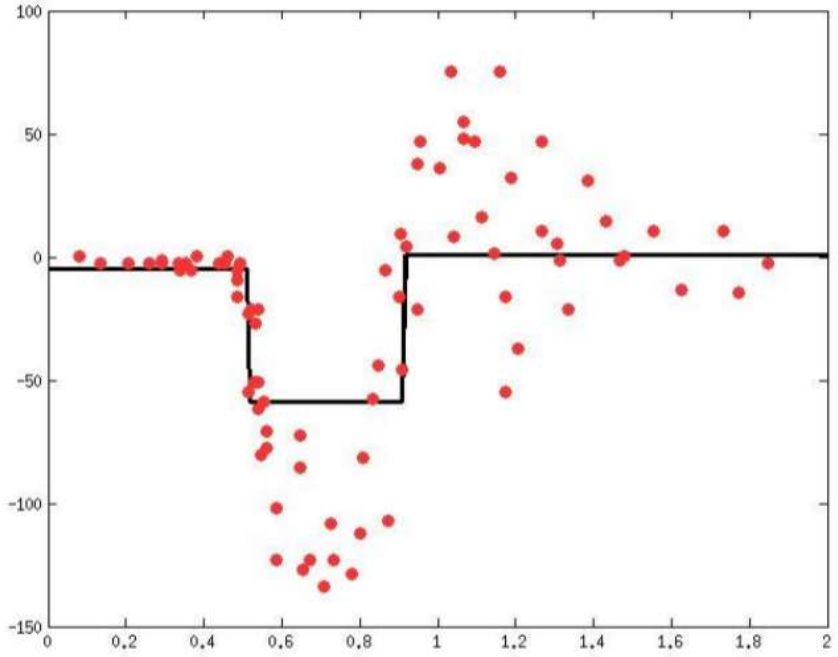


$$f_2(x^{(i)}) \approx \epsilon^{(i)}$$

Gradient Boosting - Example

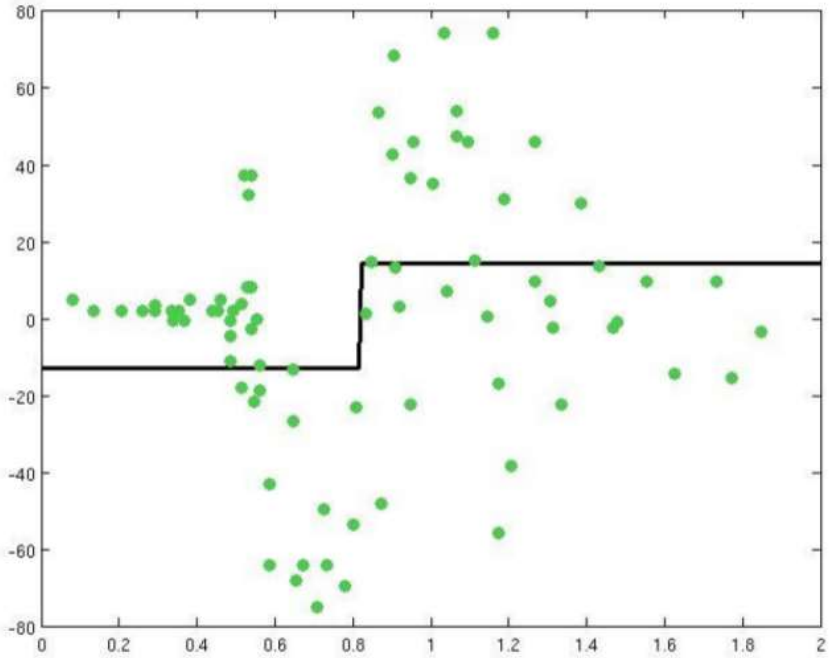
Combining gives a better predictor…

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



Can try to correct its errors also, & repeat

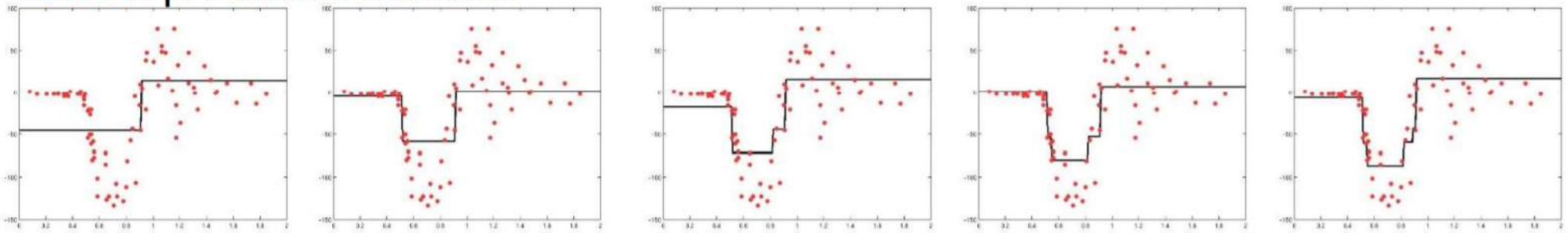
$$\epsilon_2^{(i)} = y^{(i)} - f_1(x^{(i)} - f_2(x^{(i)}) \quad \ldots$$



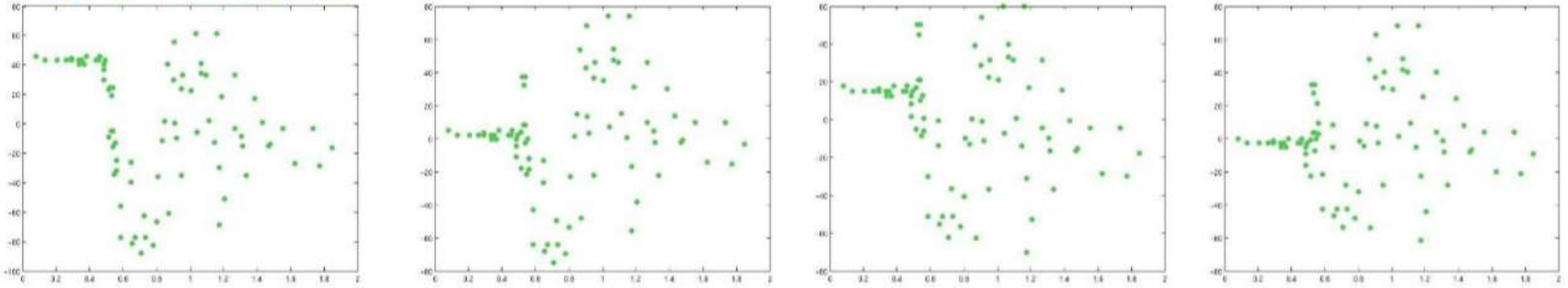
Gradient Boosting - Example

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

Data & prediction function



Error residual



• • •

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 $\hat{f}(x) = f_M(x)$.

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