Introduction to Machine Learning
Lecture 6: Gradient boosting

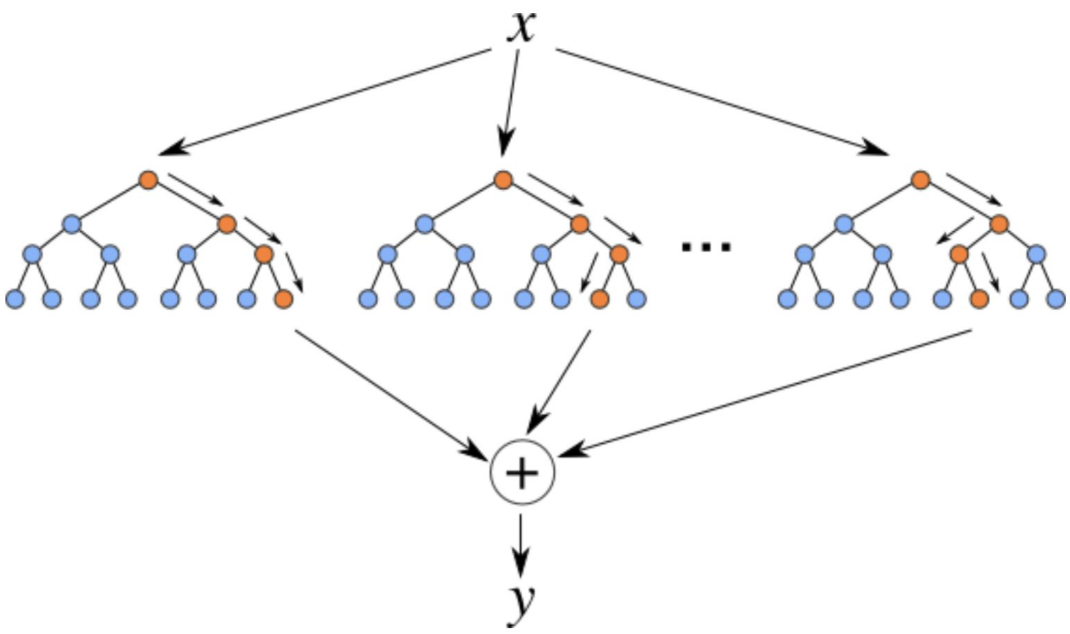
Harbour.Space University
February 2021

Nikolay Karpachev

Outline

1. Random Forest recap. Out-of-bag error.
2. Boosting technique
3. Gradient Boosting

Bagging + RSM = Random Forest

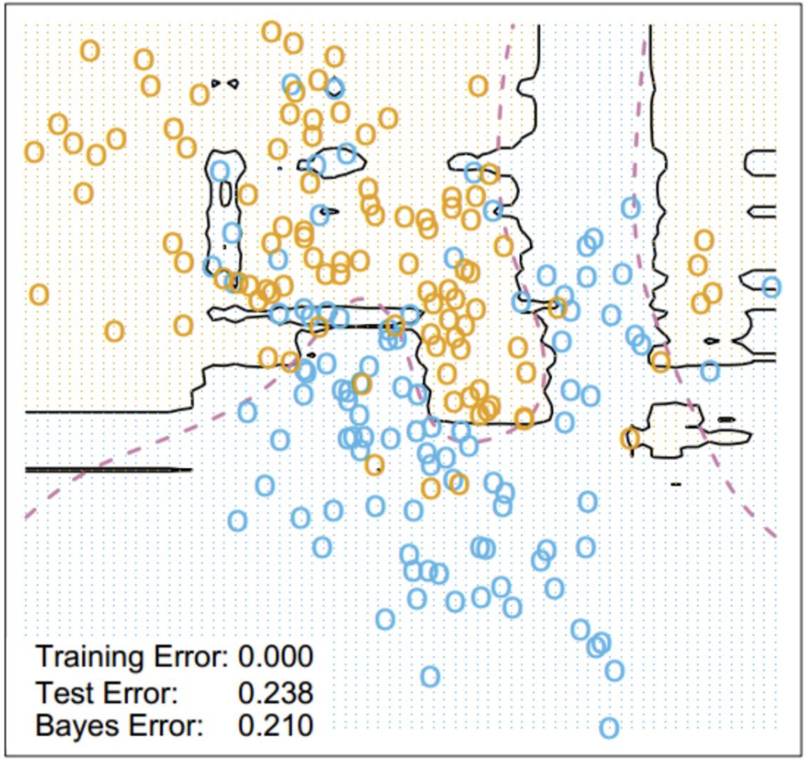


Random Forest

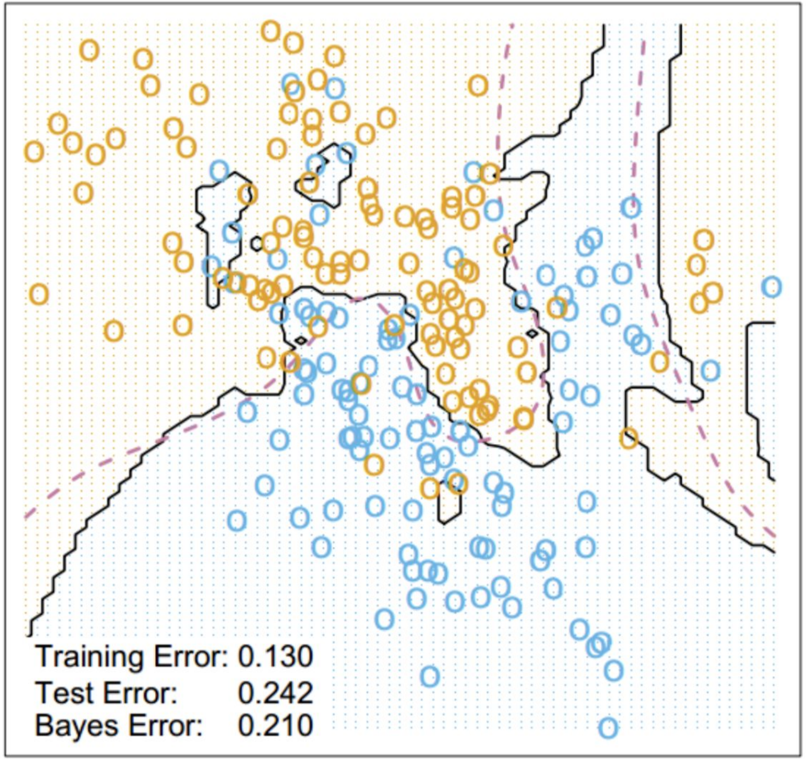
- One of the greatest “universal” models.
- There are some modifications: Extremely Randomized Trees, Isolation Forest, etc.
- Allows to use train data for validation: OOB

$$\mathrm{OOB} = \sum_{i=1}^{\ell} L\left(y_i, \frac{1}{\sum_{n=1}^{N}[x_i \notin X_n]} \sum_{n=1}^{N}[x_i \notin X_n] b_n(x_i)\right)$$

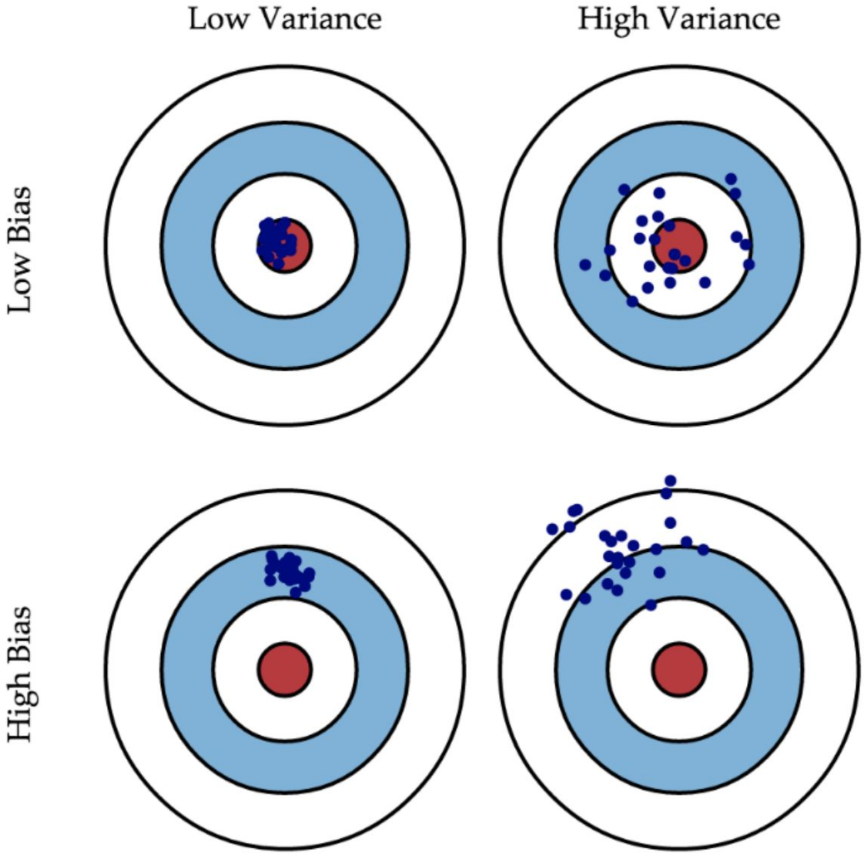
Random Forest Classifier



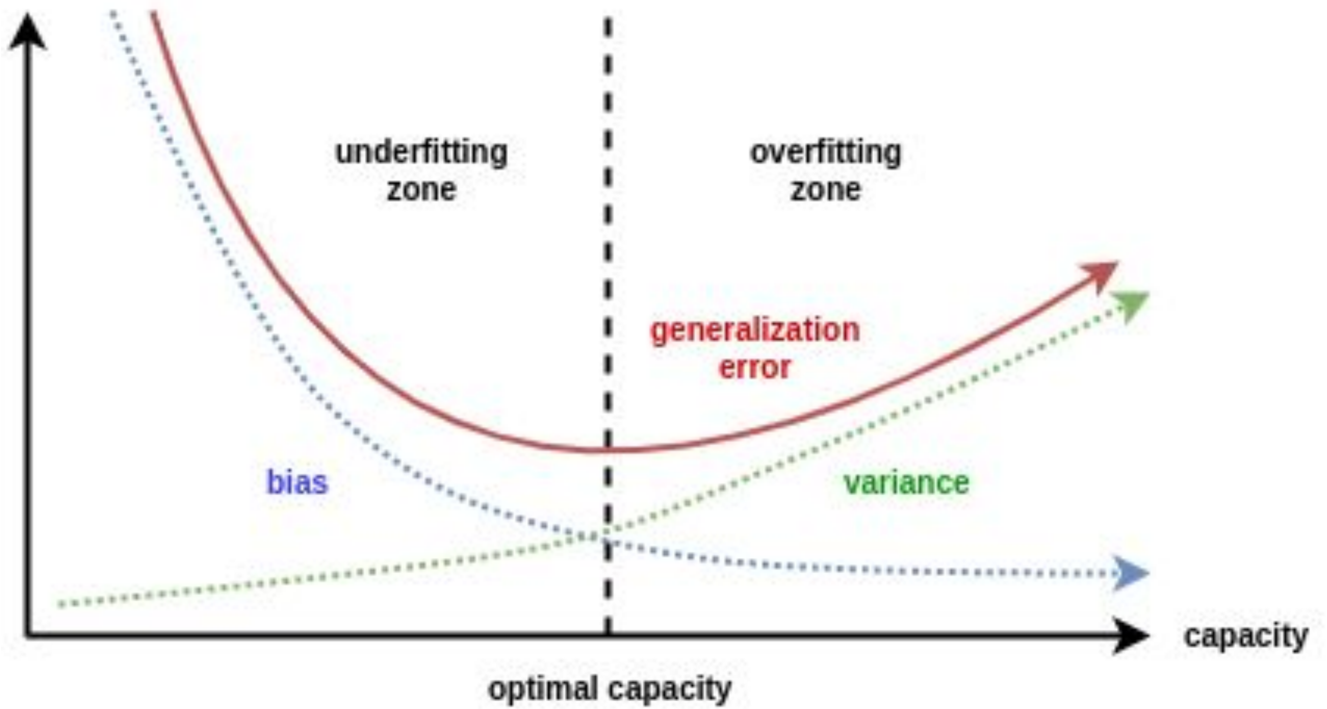
3−Nearest Neighbors



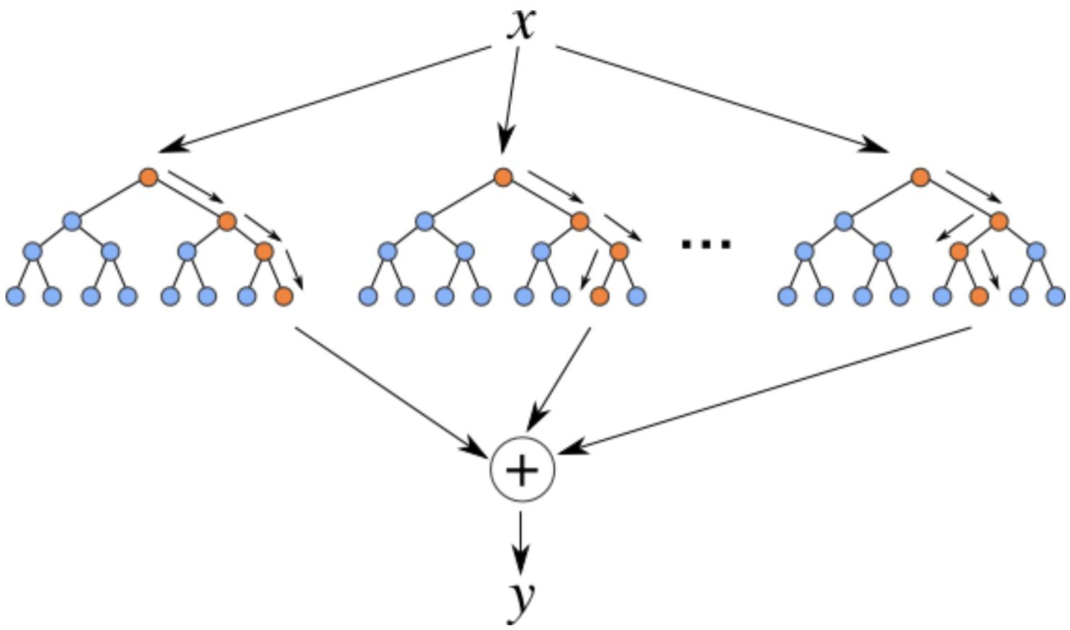
Bias-variance tradeoff



Bias-variance tradeoff



Is Random Forest decreasing bias or variance by building the trees ensemble?



Boosting

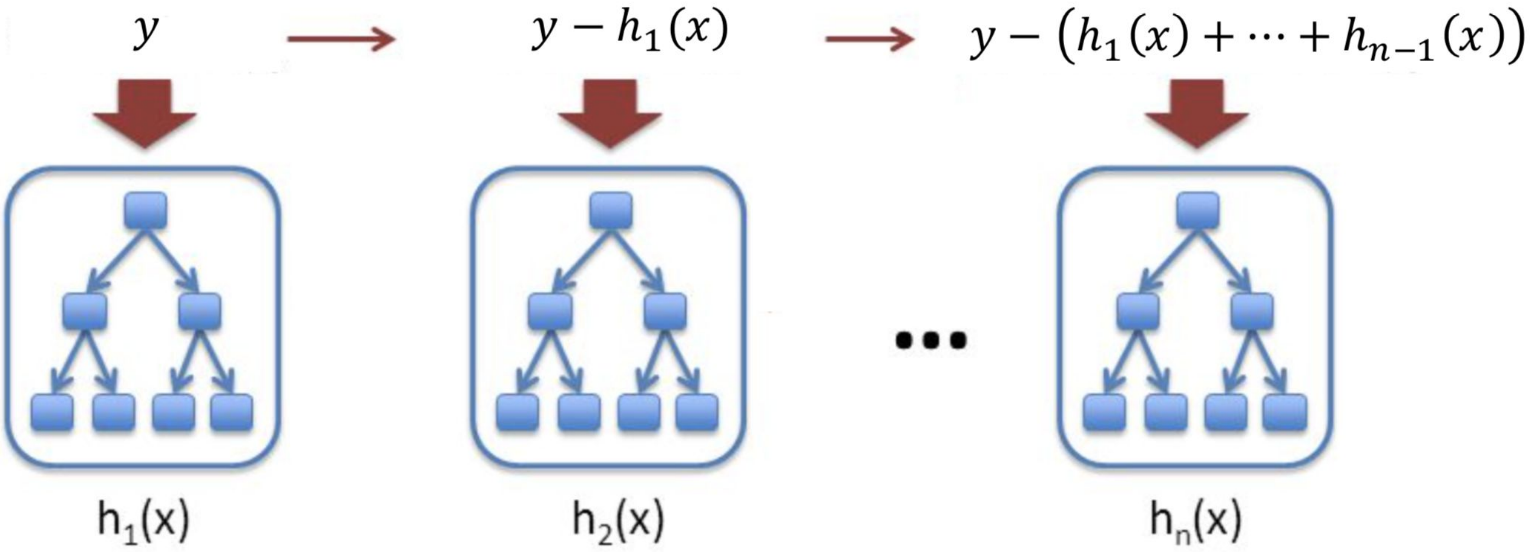
Boosting

Boosting is a technique to combine multiple weak algorithms into one strong ensemble.

- Boosting adds algorithms incrementally
- At the current step, ensemble consists of N algorithms
- Algorithm N+1 is trained with respect to already built composition to minimize the total error

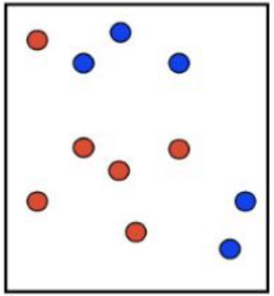
Example: regression

$$a_n(x) = h_1(x) + \cdots + h_n(x)$$

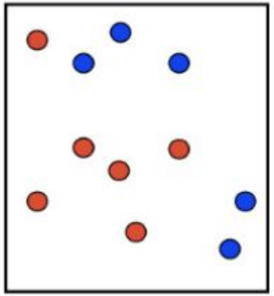


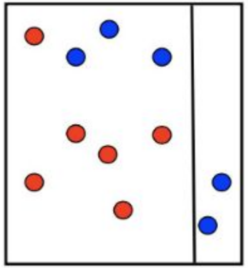
Boosting

Binary classification problem.
Models - decision stumps.

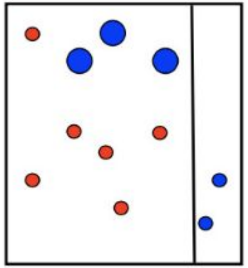


Boosting: intuition

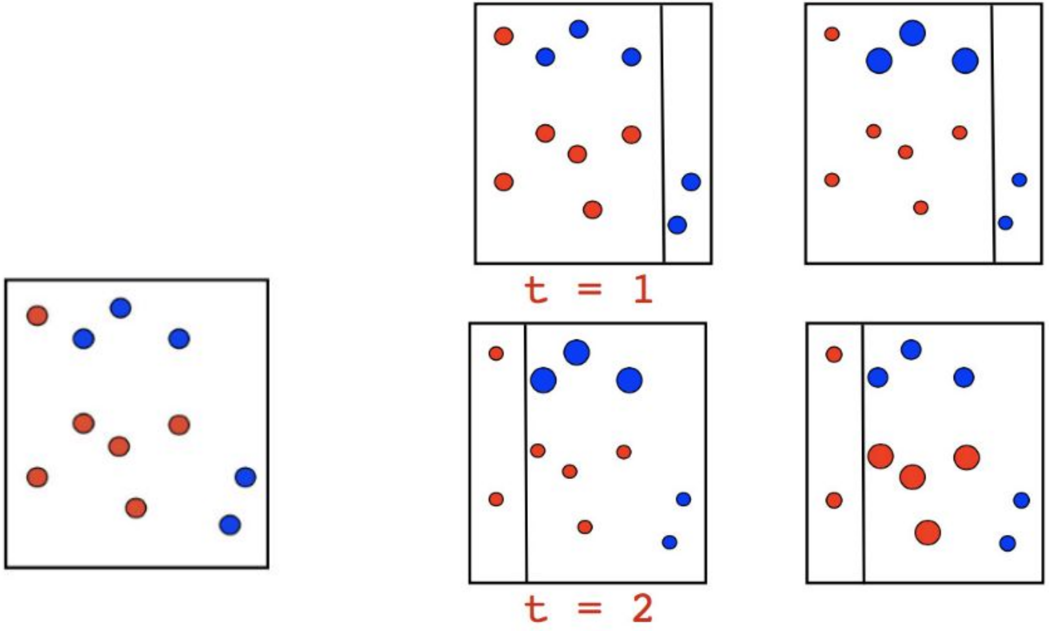




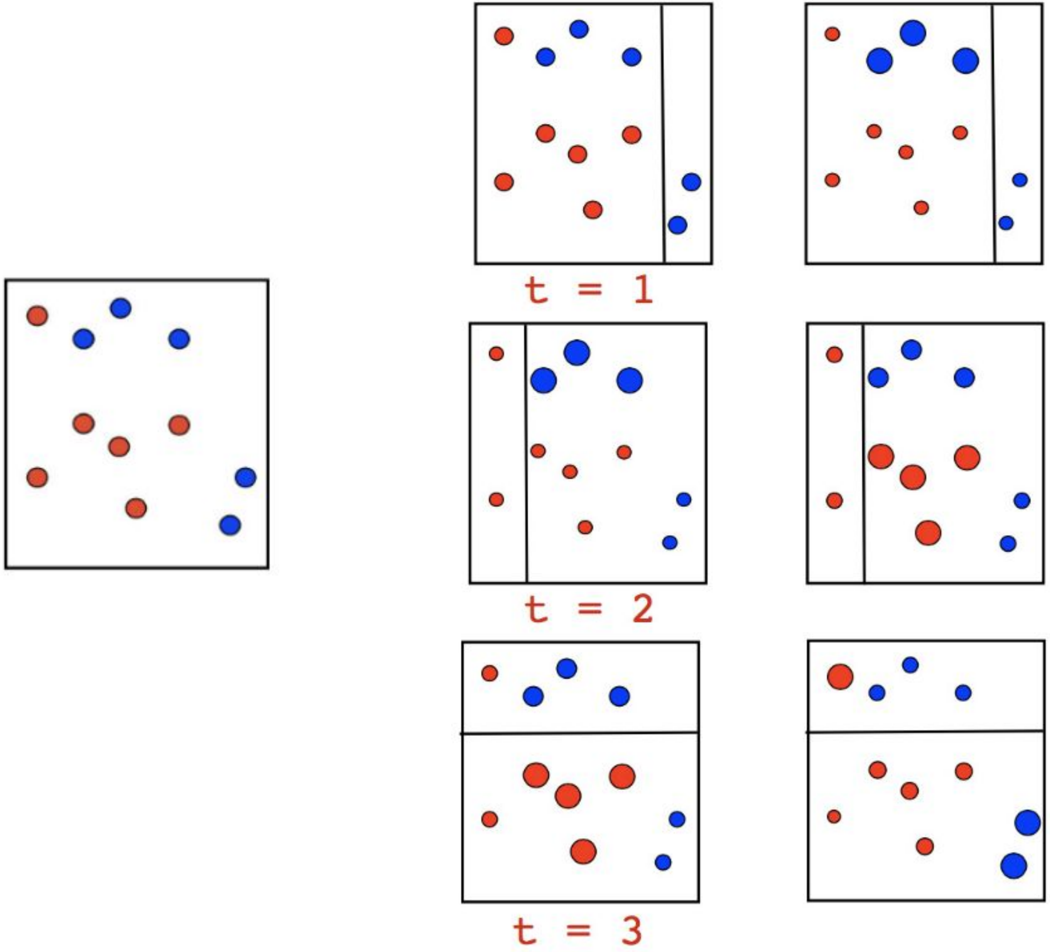
t = 1



Boosting: intuition

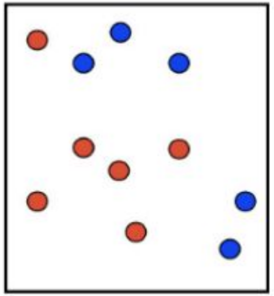


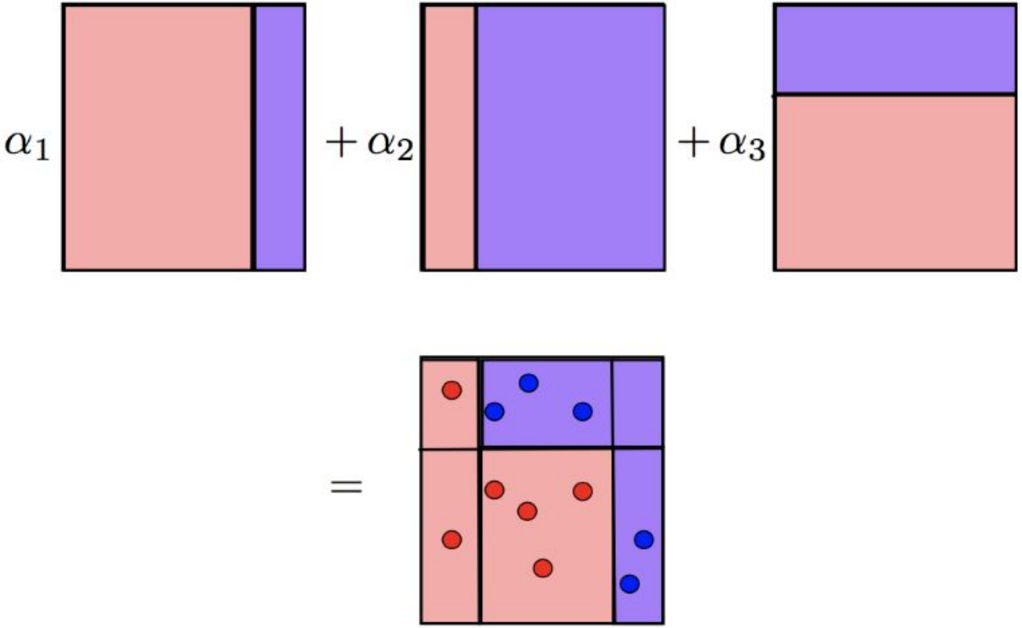
Boosting: intuition



Boosting: intuition

Binary classification problem.
Models - decision stumps.





Gradient boosting: theory

Denote dataset $\{(x_i, y_i)\}_{i=1,\ldots,n}$, loss function $L(y, f)$.

Gradient boosting: theory

Denote dataset $\{(x_i, y_i)\}_{i=1,\ldots,n}$, loss function $L(y, f)$.

Optimal model:

$$\hat{f}(x) = \underset{f(x)}{\arg\min}\ L(y, f(x)) = \underset{f(x)}{\arg\min}\ \mathbb{E}_{x,y}[L(y, f(x))]$$

Gradient boosting: theory

Denote dataset $\{(x_i, y_i)\}_{i=1,\dots,n}$, loss function $L(y, f)$.

Optimal model:

$$\hat{f}(x) = \arg\min_{f(x)} L(y, f(x)) = \arg\min_{f(x)} \mathbb{E}_{x,y}[L(y, f(x))]$$

Let it be from parametric family: $\hat{f}(x) = f(x, \hat{\theta})$,

$$\hat{\theta} = \arg\min_{\theta} \mathbb{E}_{x,y}[L(y, f(x, \theta))]$$

Gradient boosting: theory

- Let's assume real-valued y
- Gradient boosting builds a single model as a weighted sum of *"base learners"*

$$\hat{F}(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + \text{const.}$$

Gradient boosting: theory

- The goal is to minimize the loss function on the training set

Boosting procedure:

1. Start with the constant function

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma),$$

2. Incrementally expand the composition by minimizing the **residual error**

$$F_m(x) = F_{m-1}(x) + \arg\min_{h_m \in \mathcal{H}} \left[\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h_m(x_i)) \right],$$

We select the optimal *h_m*

Loss of the new composition

Time for your questions and a coffee break

Gradient boosting: theory

$$F_m(x) = F_{m-1}(x) + \underset{h_m \in \mathcal{H}}{\arg\min} \left[\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h_m(x_i)) \right],$$

We select the optimal *h_m*

Loss of the new composition

How to find the optimal *h_m*?

Gradient boosting: theory

Key idea: apply gradient descent in the space of model predictions

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i)),$$

Loss gradient at current prediction point

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) - \gamma \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))\right),$$

Find the optimal “learning rate”

“Learning rate”

- We fit the next base learner to the anti-gradient of the error
- Thus, the composition resembles gradient descent
- Each base learner addition = gradient step in functional space

Gradient boosting: theory

Complete algorithm:

- Initialize the model with a constant value

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

Incrementally:

- Compute *pseudo-residuals* for each data point

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

- Fit new base learner to pseudo-residuals
- Find the optimal “learning rate” coefficient
- Update the model

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

Gradient boosting: beautiful demo

Great demo:

http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html

Gradient boosting: theory

What we need:

- Data.
- Loss function and its gradient.
- Family of algorithms (with constraints on hyperparameters if necessary).
- Number of iterations M.
- Initial value (GBM by Friedman): constant.

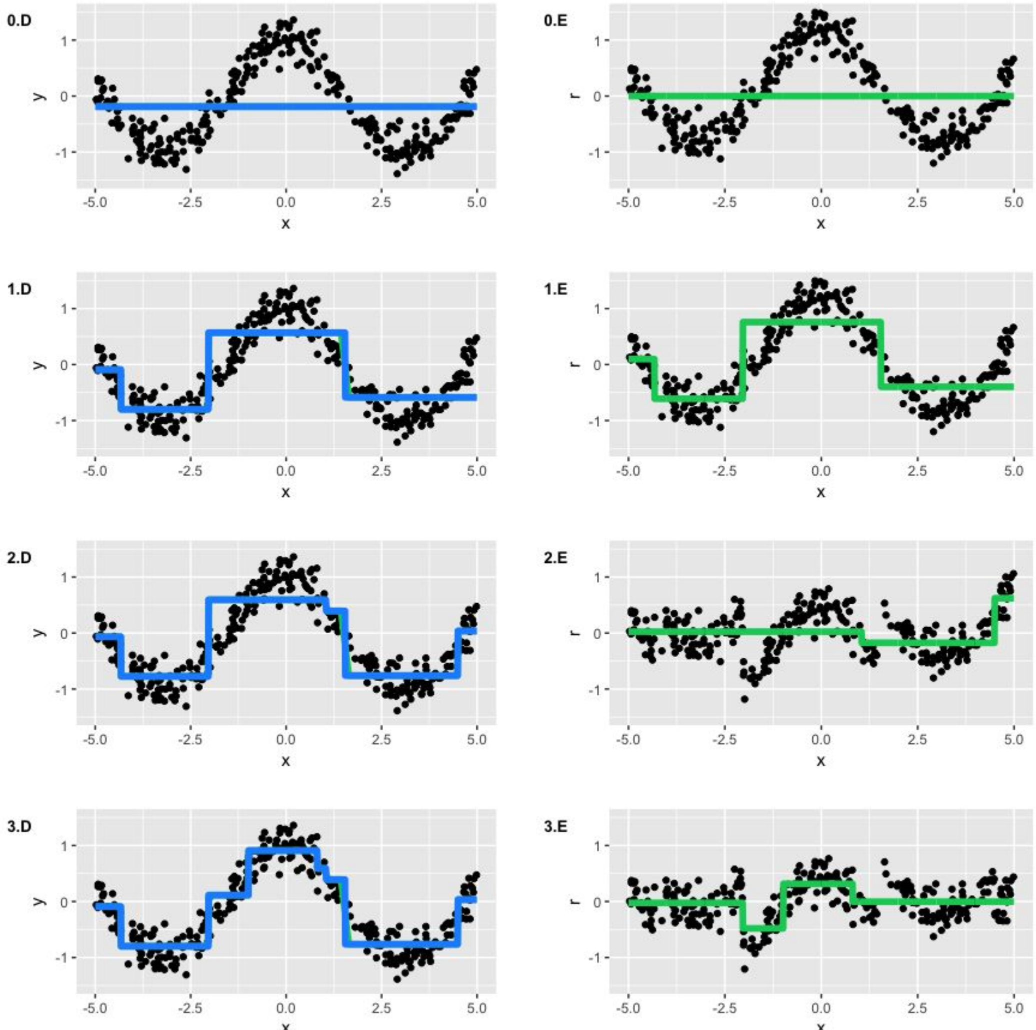
Gradient boosting: example

What we need:

- Data: toy dataset $y = cos(x) + \epsilon, \epsilon \sim \mathcal{N}(0, \frac{1}{5}), x \in [-5, 5]$
- Loss function: MSE
- Family of algorithms: decision trees with depth 2
- Number of iterations M = 3
- Initial value: just mean value

Example by ODS; source: https://habr.com/ru/company/ods/blog/327250/

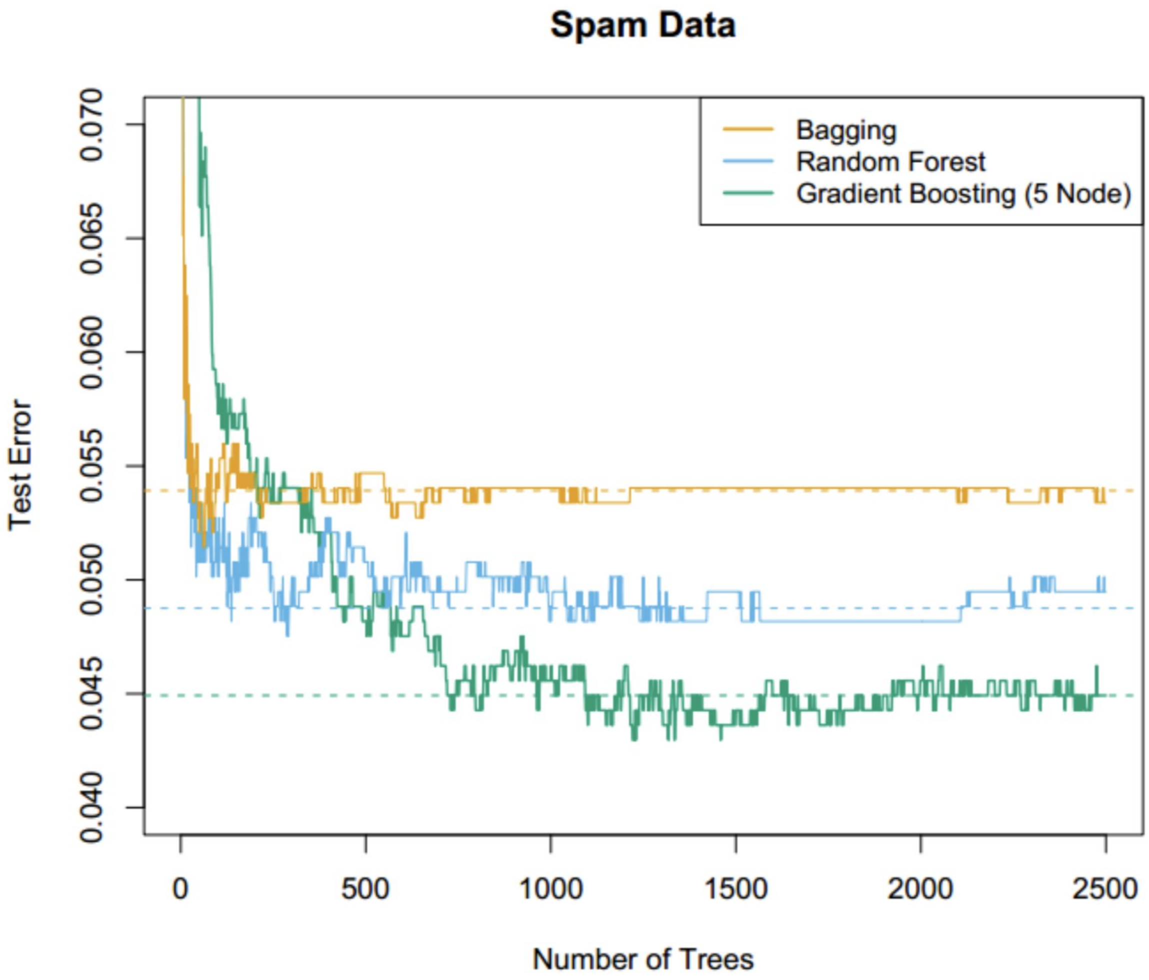
Gradient boosting: example

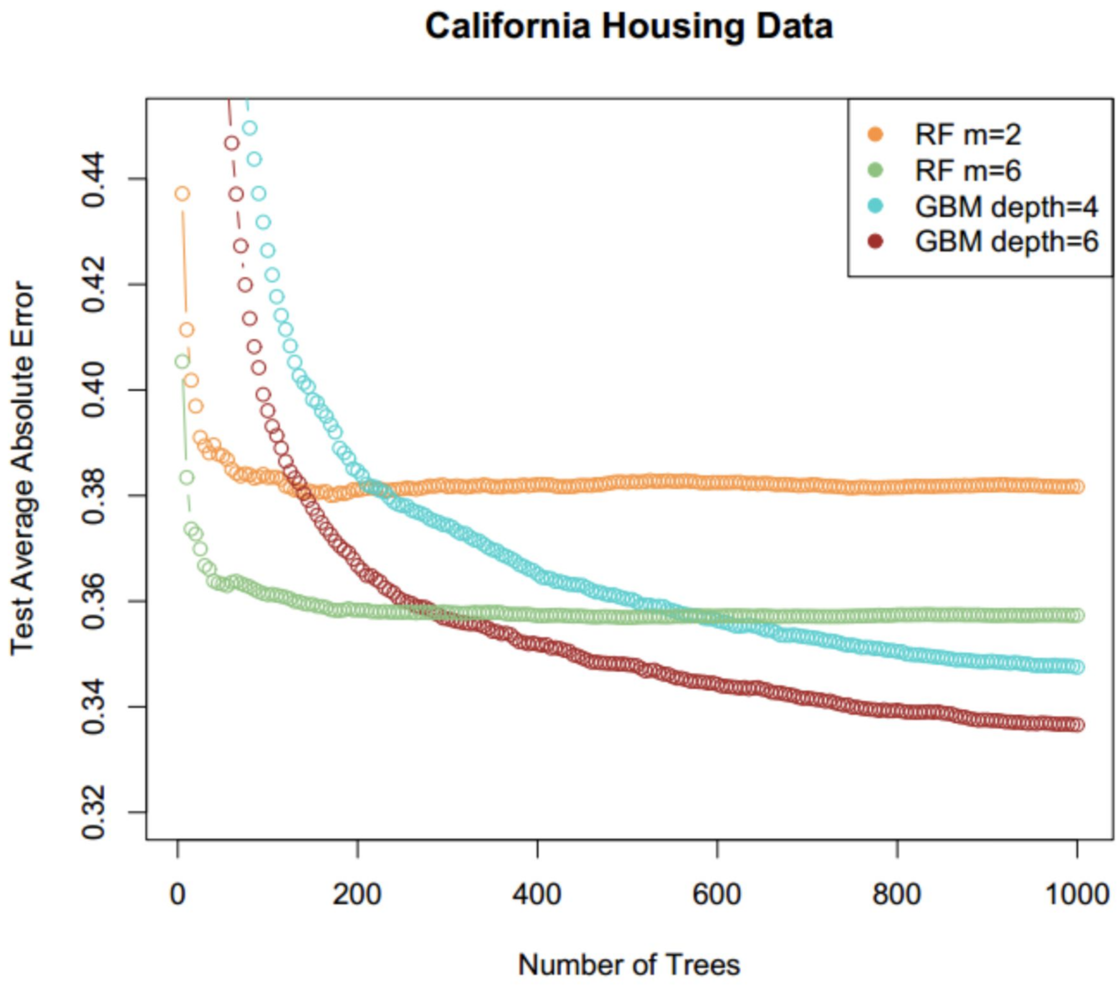


Left: full ensemble on each step.

Right: additional tree decisions.

Example by ODS; source: https://habr.com/ru/company/ods/blog/327250/


Spam Data
Bagging
Random Forest
Gradient Boosting (5 Node)
Test Error
0.040
0.045
0.050
0.055
0.060
0.065
0.070
0
500
1000
1500
2000
2500
Number of Trees


California Housing Data
RF m=2
RF m=6
GBM depth=4
GBM depth=6
Test Average Absolute Error
0.32
0.34
0.36
0.38
0.40
0.42
0.44
0
200
400
600
800
1000
Number of Trees

Gradient boosting: regularization

Boosting is a very powerful algorithm

But it can overfit the training data and lose the generalization ability
We need to regularize the model

Regularization options:

- Constrain number of trees
- Set max. depth of the trees

Thanks for the attention!