Machine Learning Course
basic track

Lecture 6: Ensembles

Radoslav Neychev

07.10.2019, MIPT
Moscow, Russia

Outline

1. Bagging & Random Forest
2. Stacking.
3. Blending.
4. Gradient boosting

Bootstrap

Consider the errors unbiased and uncorrelated:

$$\mathbb{E}_x \varepsilon_j(x) = 0;$$
$$\mathbb{E}_x \varepsilon_i(x)\varepsilon_j(x) = 0, \quad i \neq j.$$

The final model averages all predictions:

$$a(x) = \frac{1}{N}\sum_{j=1}^{N} b_j(x).$$

$$E_N = \mathbb{E}_x\left(\frac{1}{N}\sum_{j=1}^{n} b_j(x) - y(x)\right)^2 =$$
$$= \mathbb{E}_x\left(\frac{1}{N}\sum_{j=1}^{N} \varepsilon_j(x)\right)^2 =$$
$$= \frac{1}{N^2}\mathbb{E}_x\left(\sum_{j=1}^{N} \varepsilon_j^2(x) + \underbrace{\sum_{i\neq j} \varepsilon_i(x)\varepsilon_j(x)}_{=0}\right) =$$
$$\boxed{= \frac{1}{N}E_1.}$$

Error decreased by N times!

Bootstrap

Consider the errors ~~unbiased and uncorrelated~~:

Because this is a lie

$$\mathbb{E}_x \varepsilon_j(x) = 0;$$
$$\mathbb{E}_x \varepsilon_i(x)\varepsilon_j(x) = 0, \quad i \neq j.$$

$$E_N = \mathbb{E}_x\left(\frac{1}{N}\sum_{j=1}^{n} b_j(x) - y(x)\right)^2 =$$
$$= \mathbb{E}_x\left(\frac{1}{N}\sum_{j=1}^{N} \varepsilon_j(x)\right)^2 =$$
$$= \frac{1}{N^2}\mathbb{E}_x\left(\sum_{j=1}^{N} \varepsilon_j^2(x) + \underbrace{\sum_{i\neq j} \varepsilon_i(x)\varepsilon_j(x)}_{=0}\right) =$$

The final model averages all predictions:

$$a(x) = \frac{1}{N}\sum_{j=1}^{N} b_j(x).$$

Error decreased by N times!

$$\boxed{= \frac{1}{N}E_1.}$$

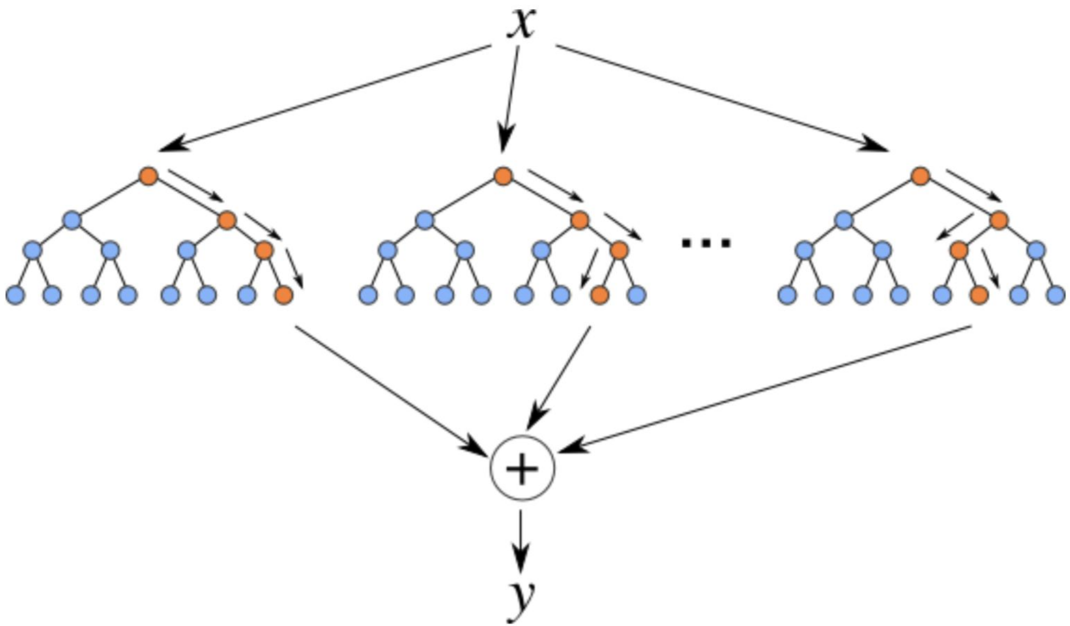
Bagging = Bootstrap aggregating

Decreases the variance if the basic algorithms are not correlated.

RSM - Random Subspace Method

Same approach, but with features.

Bagging + RSM = Random Forest



Random Forest

- One of the greatest “universal” models.

Random Forest

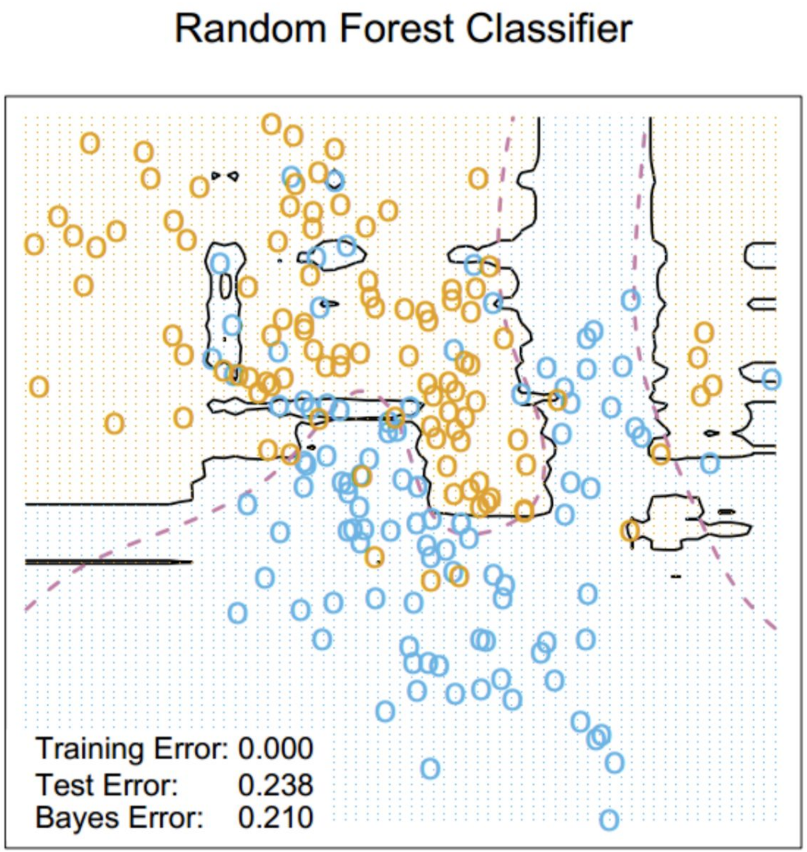
- One of the greatest “universal” models.
- There are some modifications: Extremely Randomized Trees, Isolation Forest, etc.
-

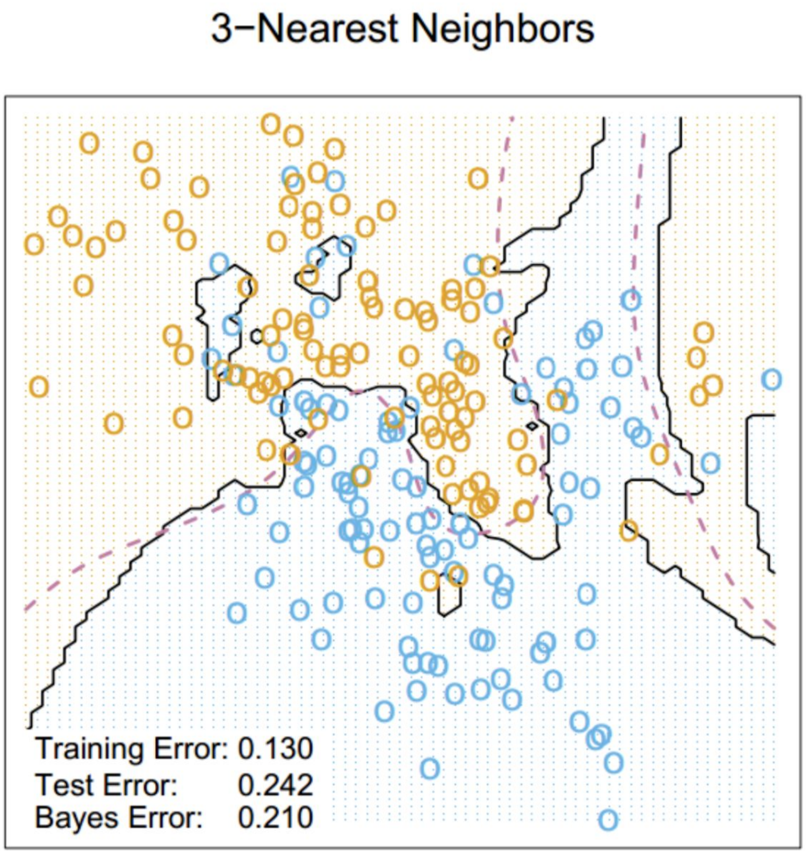
Random Forest

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

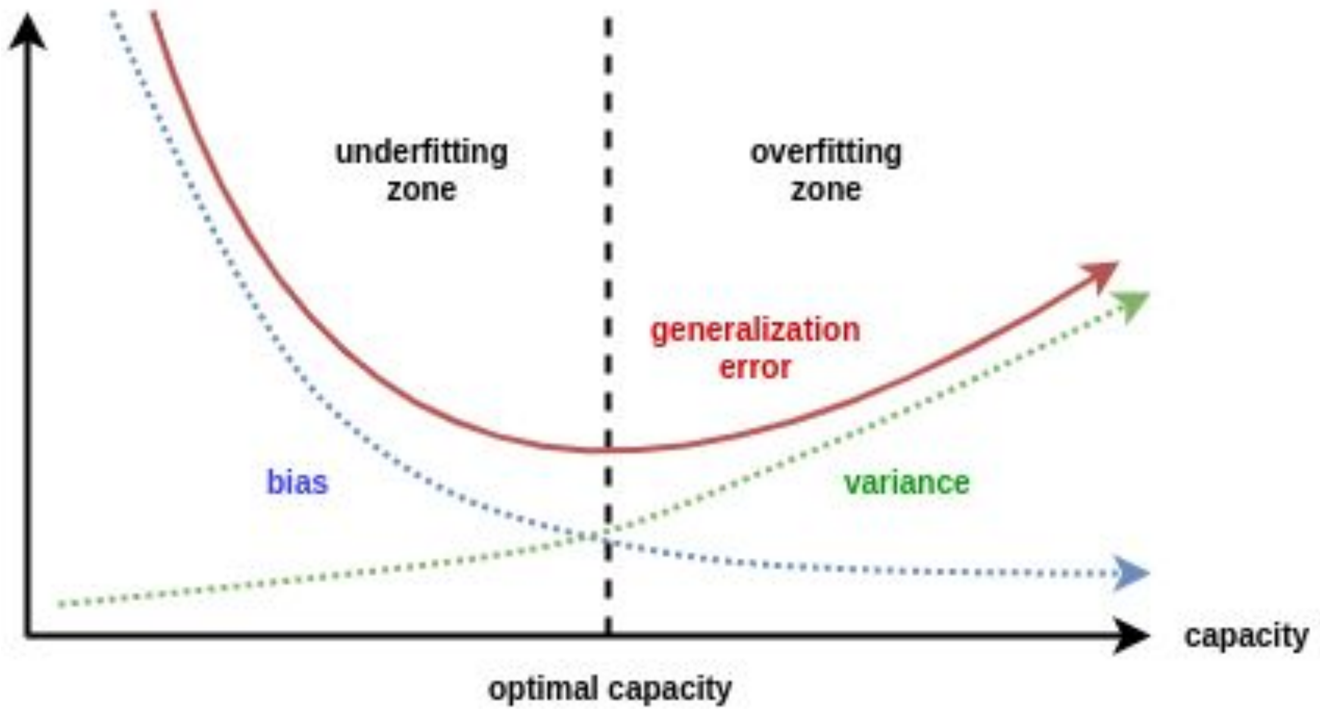
- One of the greatest “universal” models.
- There are some modifications: Extremely Randomized Trees, Isolation Forest, etc.
- Allows to use train data for validation: OOB

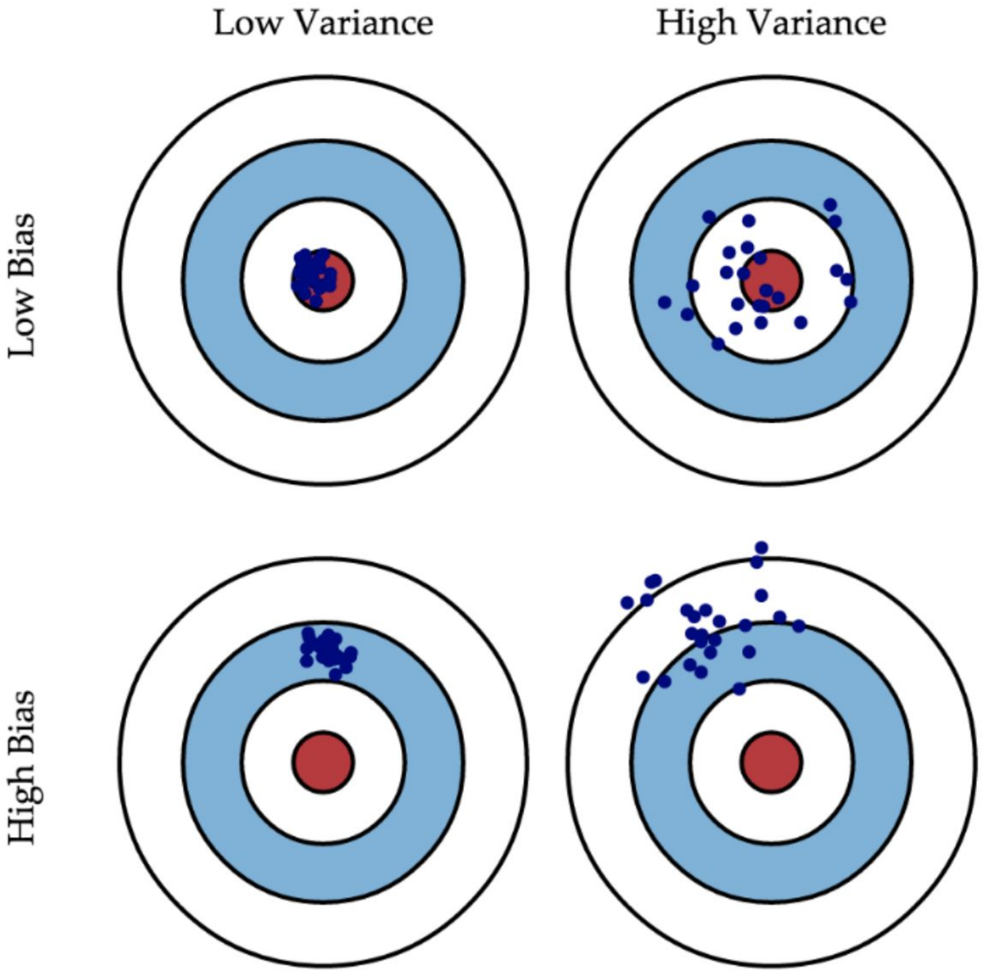
$$\mathrm{OOB} = \sum_{i=1}^{\ell} L\left(y_i, \frac{1}{\sum_{n=1}^{N}[x_i \notin X_n]} \sum_{n=1}^{N}[x_i \notin X_n] b_n(x_i)\right)$$


Random Forest Classifier
Training Error: 0.000
Test Error: 0.238
Bayes Error: 0.210


3−Nearest Neighbors
Training Error: 0.130
Test Error: 0.242
Bayes Error: 0.210

Bias-variance tradeoff




Low Variance
High Variance
Low Bias
High Bias

Bias-variance decomposition

The dataset $X = (x_i, y_i)_{i=1}^{\ell}$ with $y_i \in \mathbb{R}$ for regression problem.

Denote loss function $L(y, a) = \big(y - a(x)\big)^2$

The corresponding risk estimation is

$$R(a) = \mathbb{E}_{x,y}\Big[\big(y - a(x)\big)^2\Big] = \int_{\mathbb{X}}\int_{\mathbb{Y}} p(x, y)\big(y - a(x)\big)^2 dx dy.$$

Denote $\mu : (\mathbb{X} \times \mathbb{Y})^\ell \to \mathcal{A}$, where $\mathcal{A}$ is some family of algorithms.

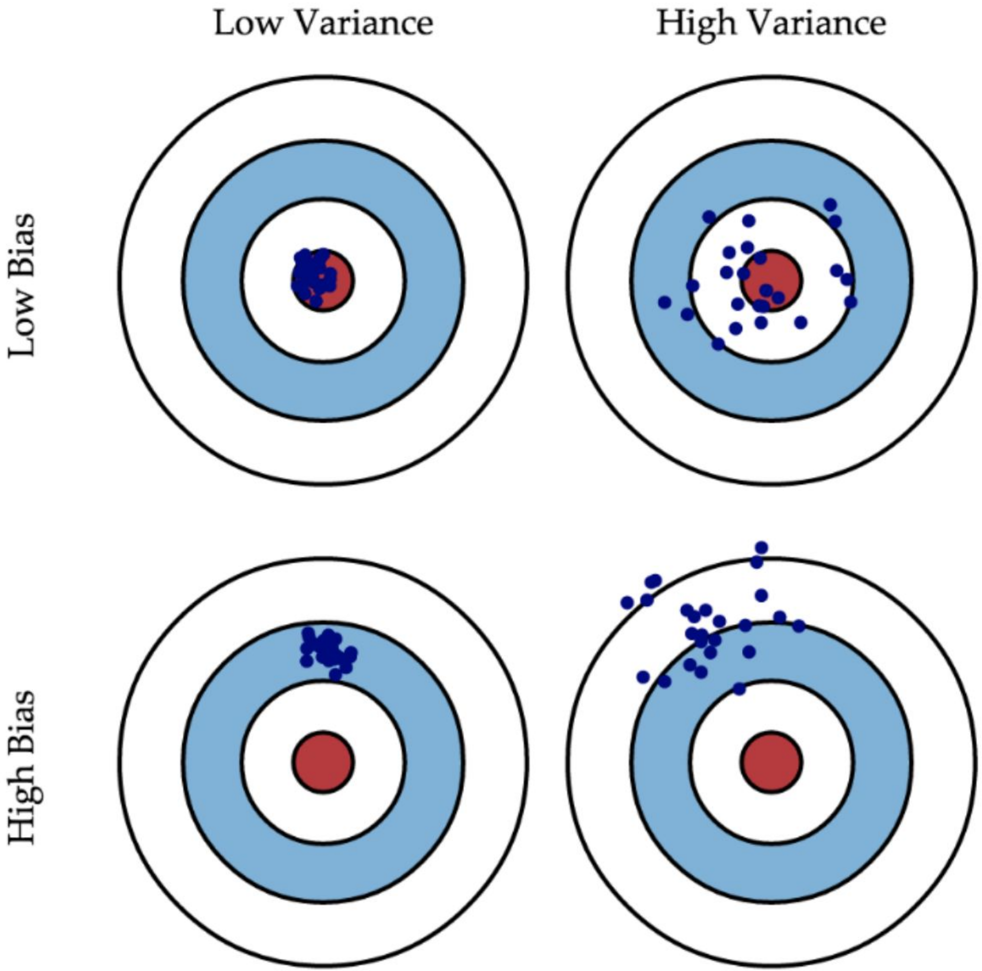
So $L(\mu) = \mathbb{E}_X\left[\mathbb{E}_{x,y}\left[\left(y - \mu(X)(x)\right)^2\right]\right]$, where X dataset.

$$L(\mu) = \underbrace{\mathbb{E}_{x,y}\left[\left(y - \mathbb{E}[y \mid x]\right)^2\right]}_{\text{noise}} +$$

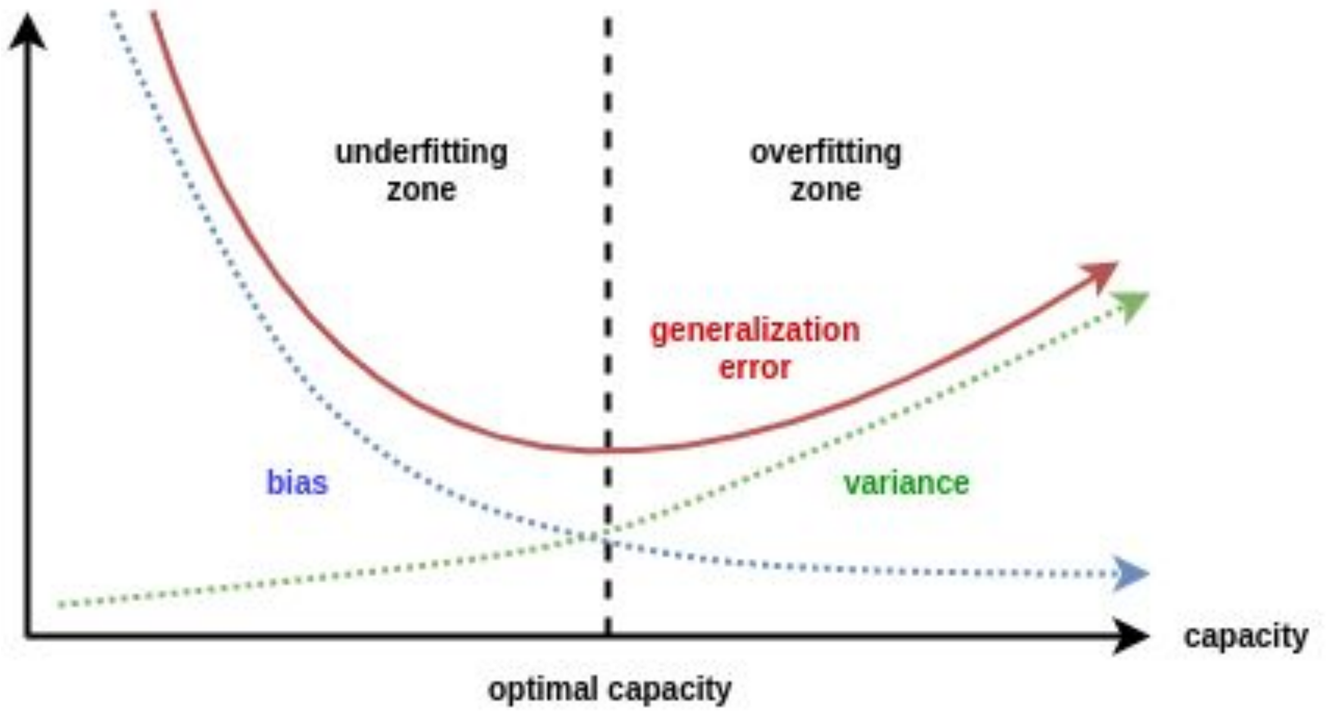
$$+ \underbrace{\mathbb{E}_x\left[\left(\mathbb{E}_X\left[\mu(X)\right] - \mathbb{E}[y \mid x]\right)^2\right]}_{\text{bias}} + \underbrace{\mathbb{E}_x\left[\mathbb{E}_X\left[\left(\mu(X) - \mathbb{E}_X\left[\mu(X)\right]\right)^2\right]\right]}_{\text{variance}}.$$

This exact form of bias-variance decomposition is correct for square loss in regression.

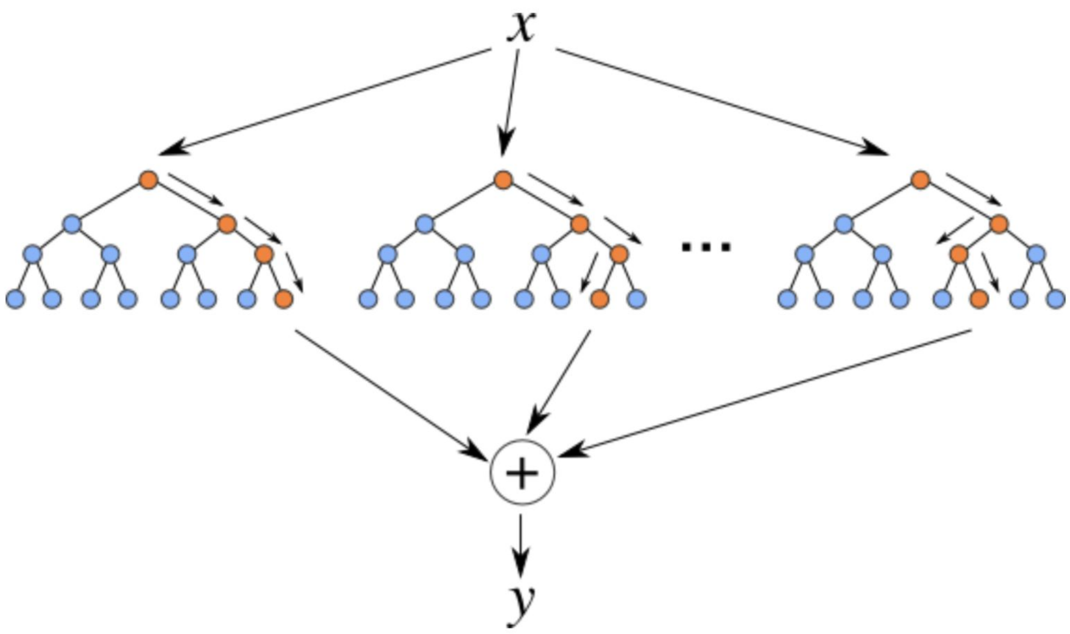
However, it is much more general. See extra materials for more exotic cases.


Low Variance
High Variance
Low Bias
High Bias

Bias-variance tradeoff

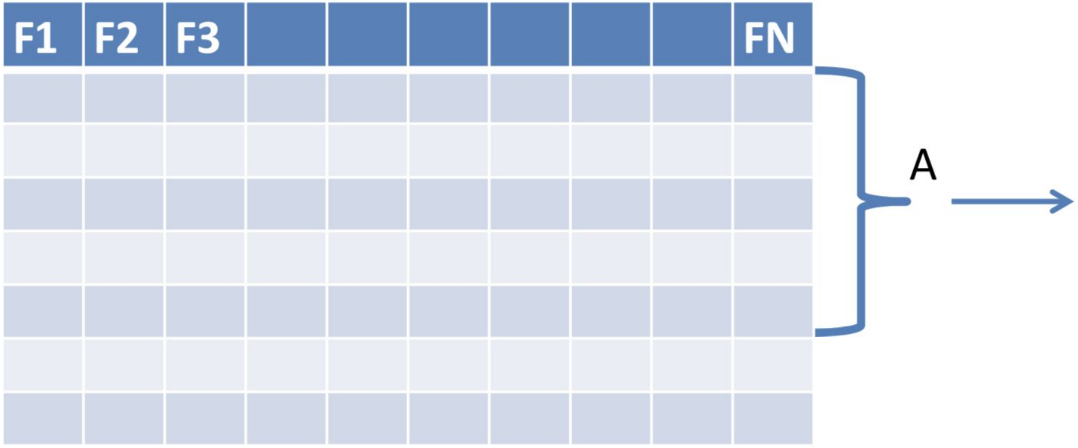


Bagging + RSM = Random Forest



Stacking

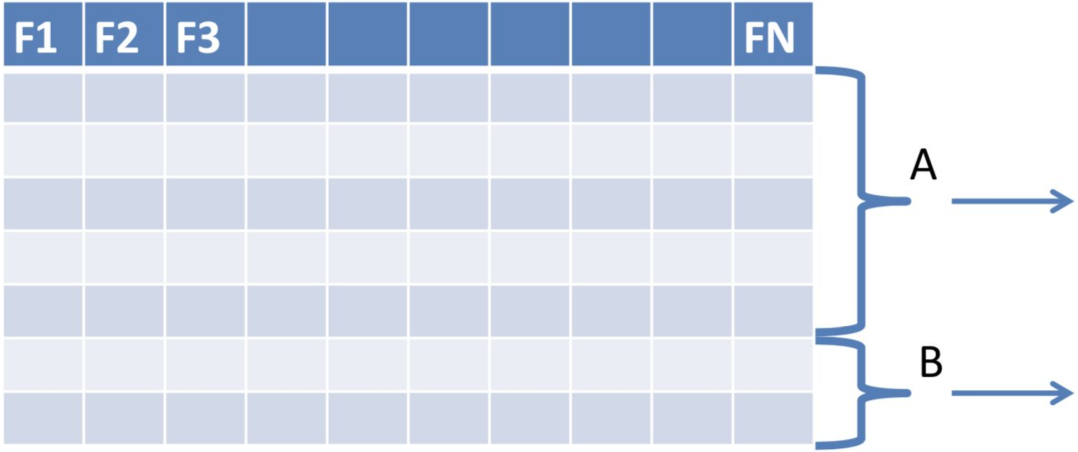
How to build an ensemble from *different* models?



Tune M basic algorithms

Stacking

How to build an ensemble from *different* models?

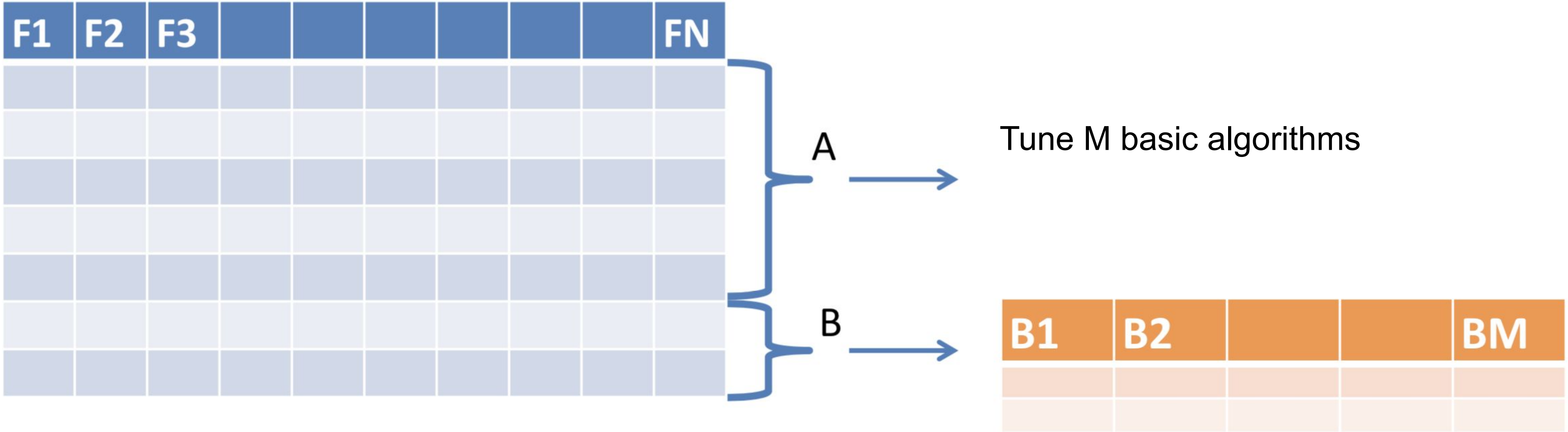


Tune M basic algorithms

Tune new model on the outputs of the basic algorithms

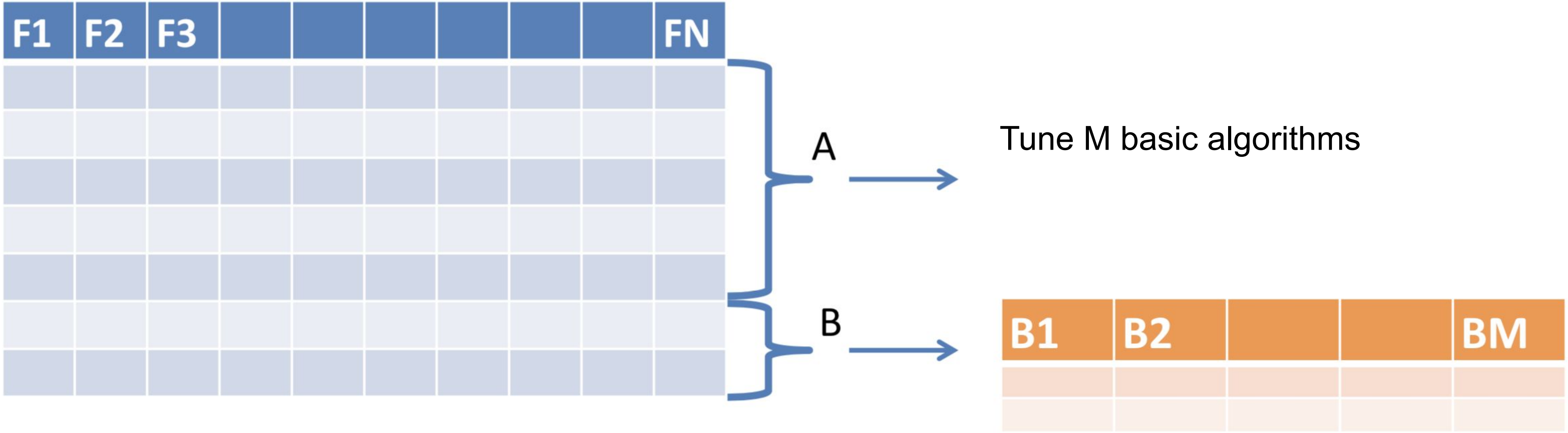
Stacking

How to build an ensemble from *different* models?



Stacking

How to build an ensemble from *different* models?



$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x)$$

e.g.

Stacking

How to build an ensemble from *different* models?

Stacking

How to build an ensemble from *different* models?

- Use different datasets (or datasets parts) for different level models.

Stacking

How to build an ensemble from *different* models?

- Use different datasets (or datasets parts) for different level models.
- Experiment with different models (linear, trees ensembles, simple networks, etc.)

Stacking

How to build an ensemble from *different* models?

- Use different datasets (or datasets parts) for different level models.
- Experiment with different models (linear, trees ensembles, simple networks, etc.)
- Or just different GBT ensembles (hola, kaggle :)

Blending

Just combine several *strong/complex* models.

Weights should sum up to 1
and come from [0; 1]

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x)$$

Blending

Just combine several *strong/complex* models.

Weights should sum up to 1
and come from [0; 1]

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x)$$

- Simple and intuitive ensembling method

Blending

Just combine several *strong/complex* models.

Weights should sum up to 1
and come from [0; 1]

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x)$$

- Simple and intuitive ensembling method.
- Finding optimal weights could be tricky.

Blending

Just combine several *strong/complex* models.

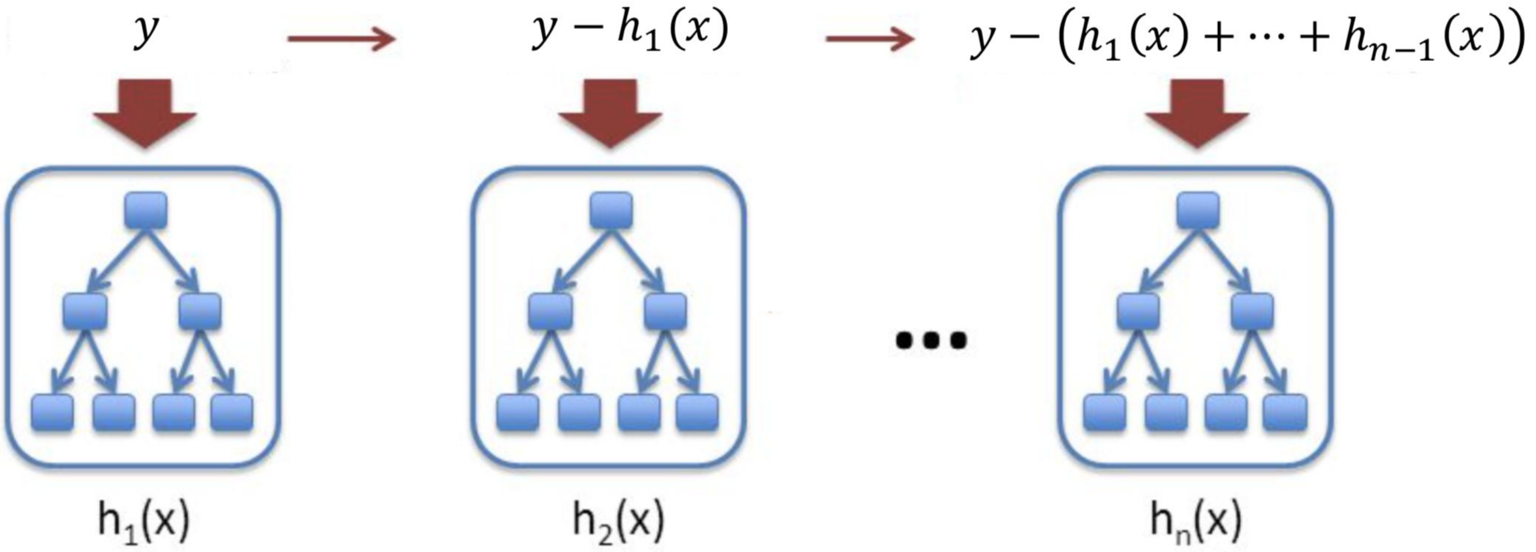
Weights should sum up to 1
and come from [0; 1]

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x)$$

- Simple and intuitive ensembling method.
- Finding optimal weights could be tricky.
- Linear composition is not always enough.

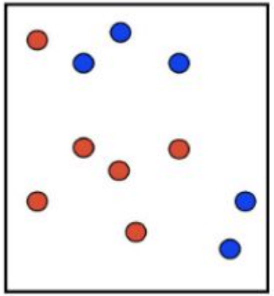
Gradient boosting

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

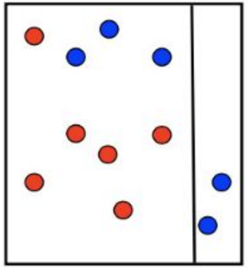


Boosting: intuition

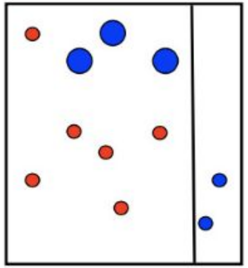
Binary classification problem.
Models - decision stumps.

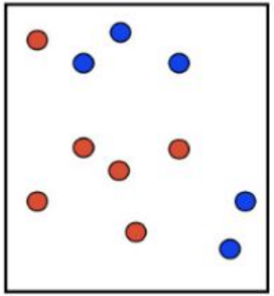


Boosting: intuition

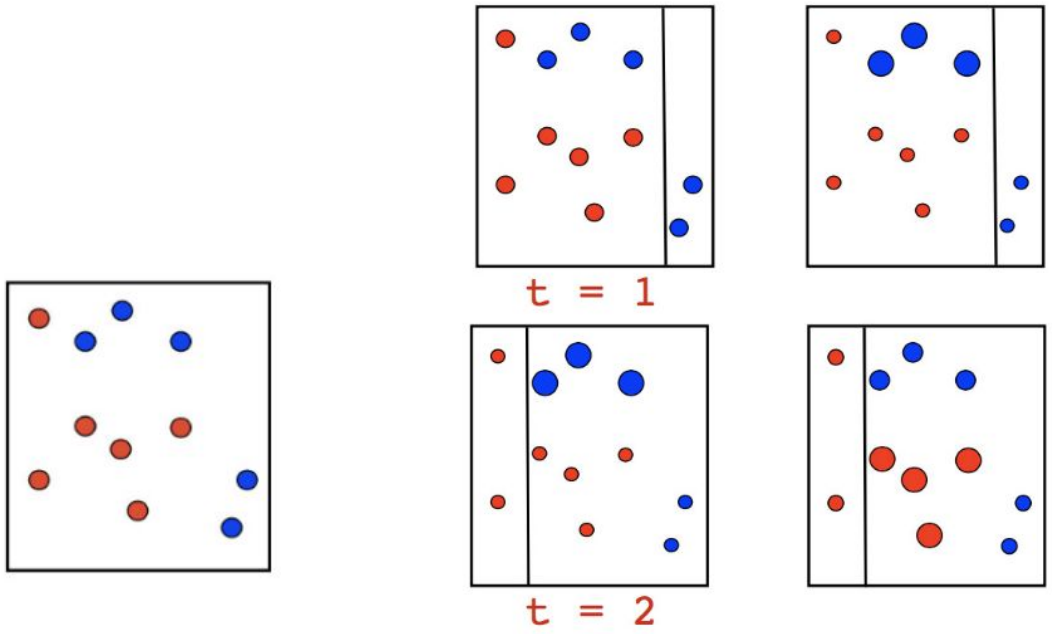


t = 1

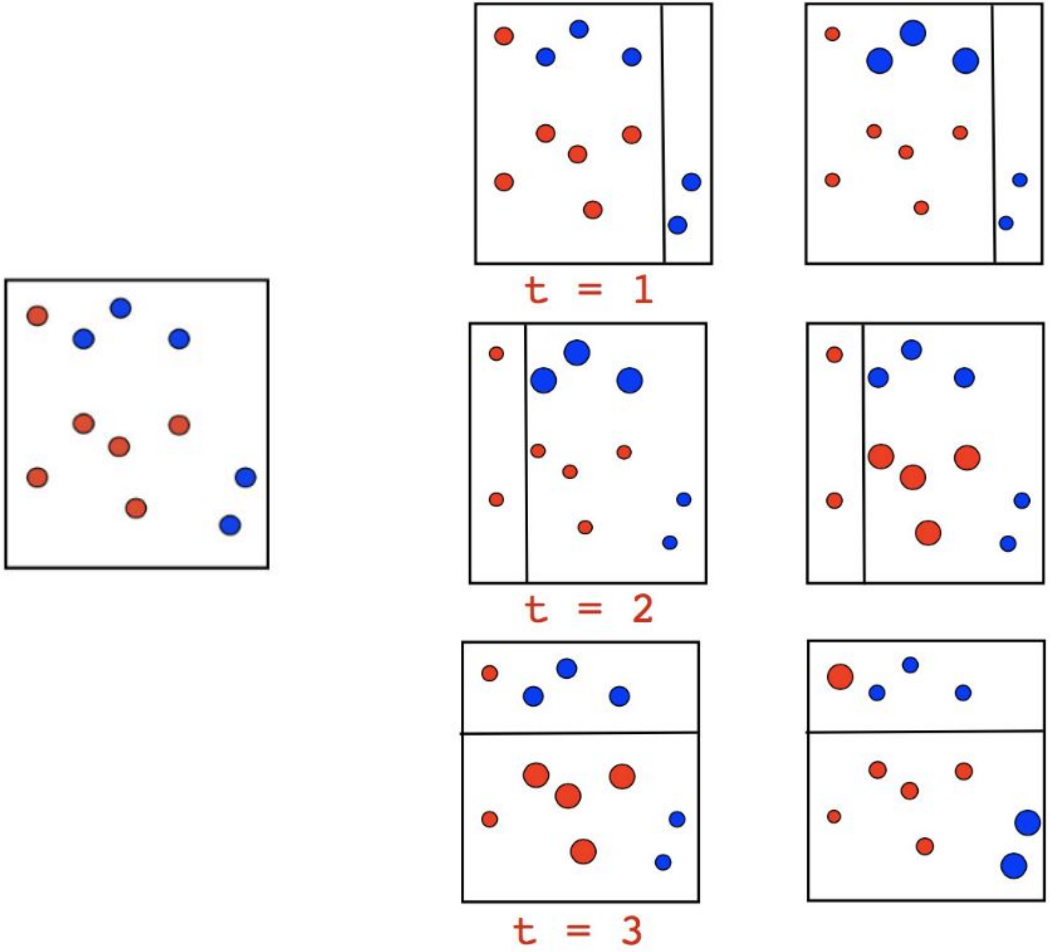




Boosting: intuition

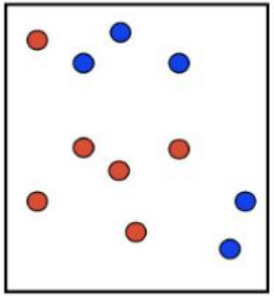


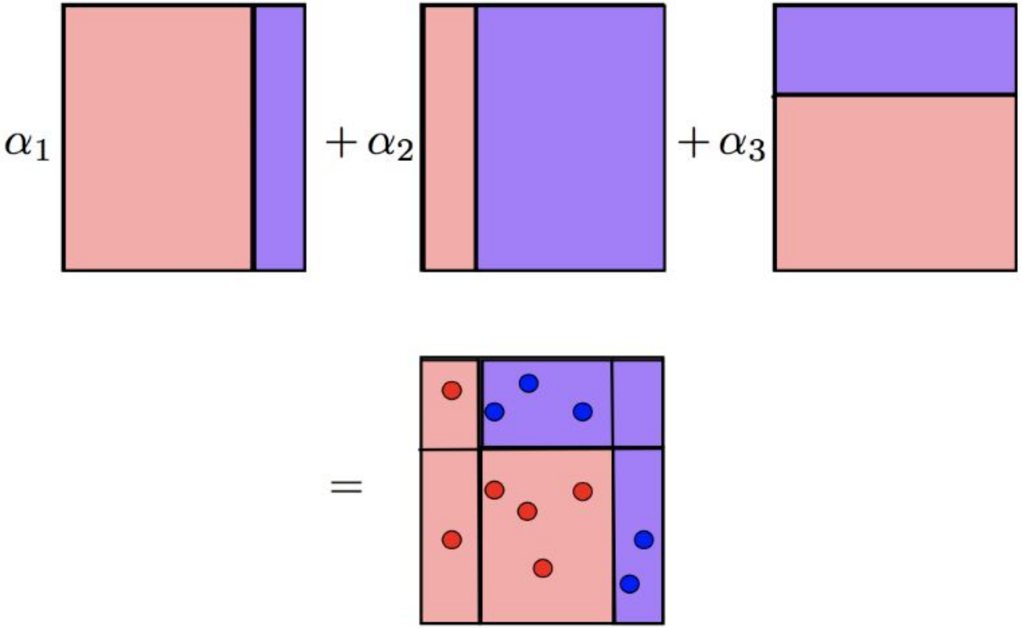
Boosting: intuition



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

$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x),$$

$$(\rho_t, \theta_t) = \arg\min_{\rho,\theta} \mathbb{E}_{x,y}[L(y, \hat{f}(x) + \rho \cdot h(x, \theta))],$$

$$\hat{f}_t(x) = \rho_t \cdot h(x, \theta_t)$$

Gradient boosting: theory

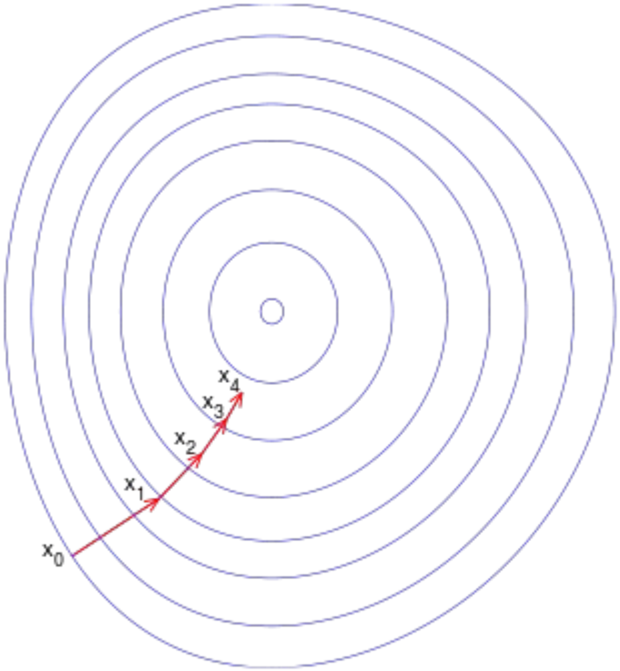
$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x),$$

$$(\rho_t, \theta_t) = \arg\min_{\rho,\theta} \mathbb{E}_{x,y}[L(y, \hat{f}(x) + \rho \cdot h(x, \theta))],$$

$$\hat{f}_t(x) = \rho_t \cdot h(x, \theta_t)$$

What if we could use gradient descent in *space of our models*?

Gradient boosting: theory



What if we could use gradient descent in *space of our models*?

Gradient boosting: theory

$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x),$$

$$r_{it} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}(x)}, \quad \text{for } i = 1, \ldots, n,$$

$$\theta_t = \arg\min_{\theta} \sum_{i=1}^{n} (r_{it} - h(x_i, \theta))^2,$$

$$\rho_t = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i) + \rho \cdot h(x_i, \theta_t))$$

Gradient boosting: theory

In linear regression case with MSE loss:

$$r_{it} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}(x)} = -2(\hat{y}_i - y_i) \propto \hat{y}_i - y_i$$

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.

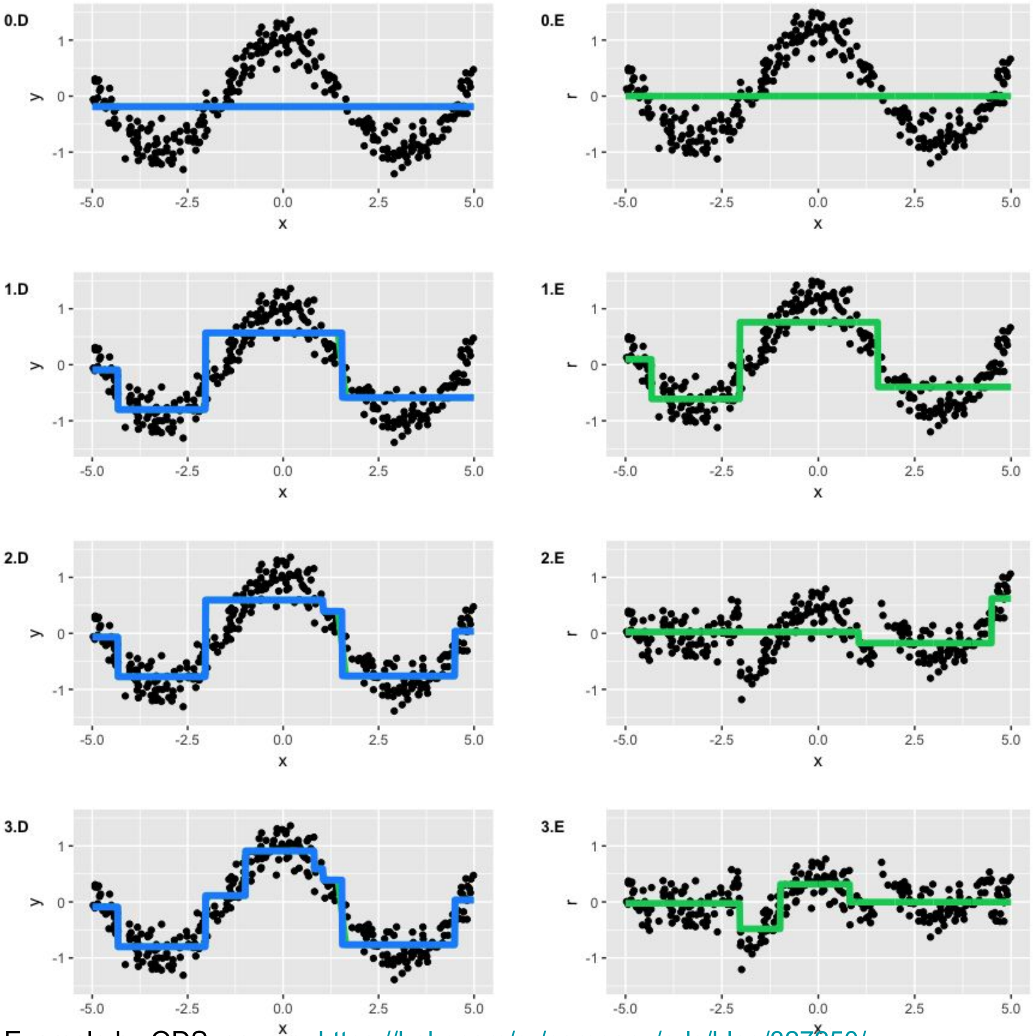
Gradient boosting: example

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/

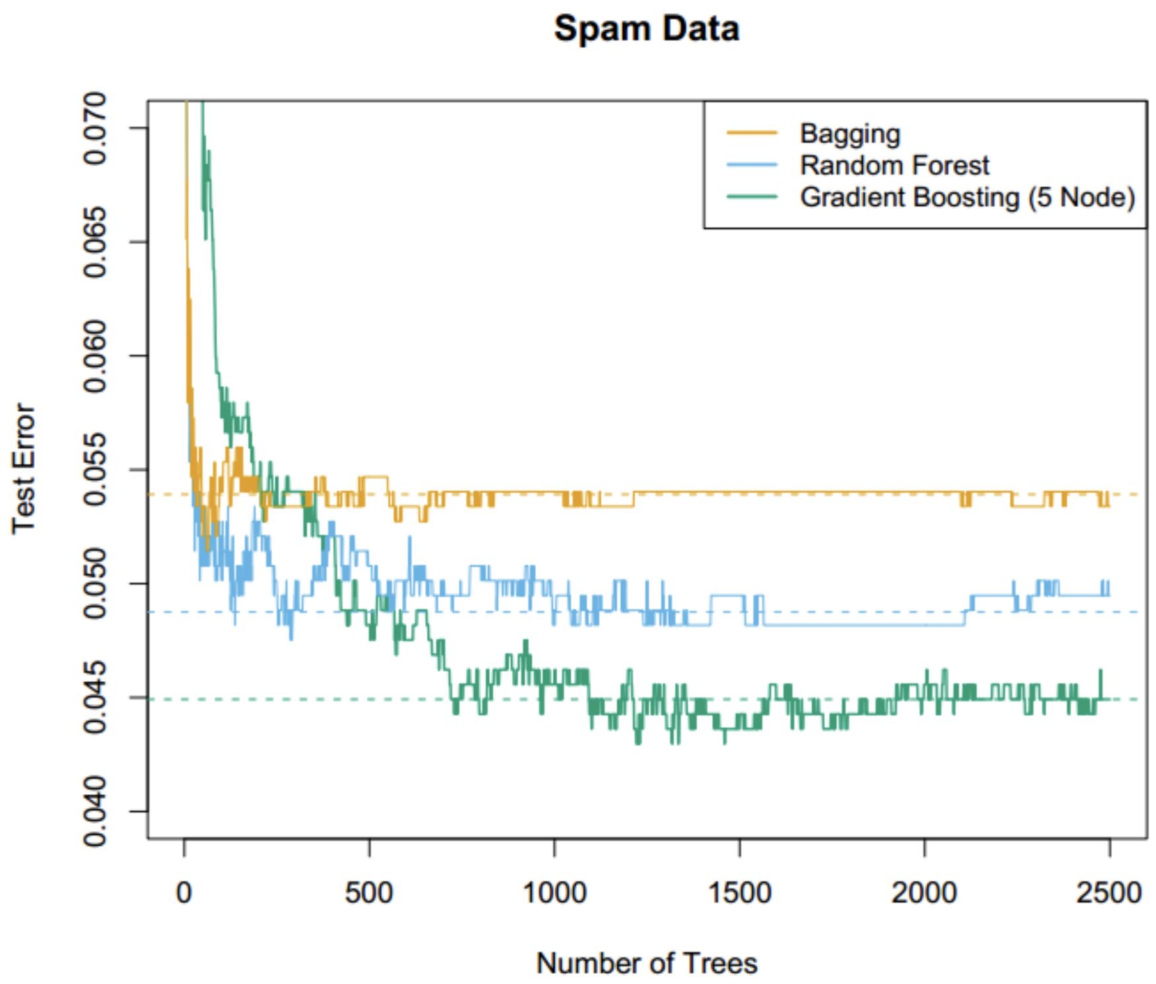
Gradient boosting: example

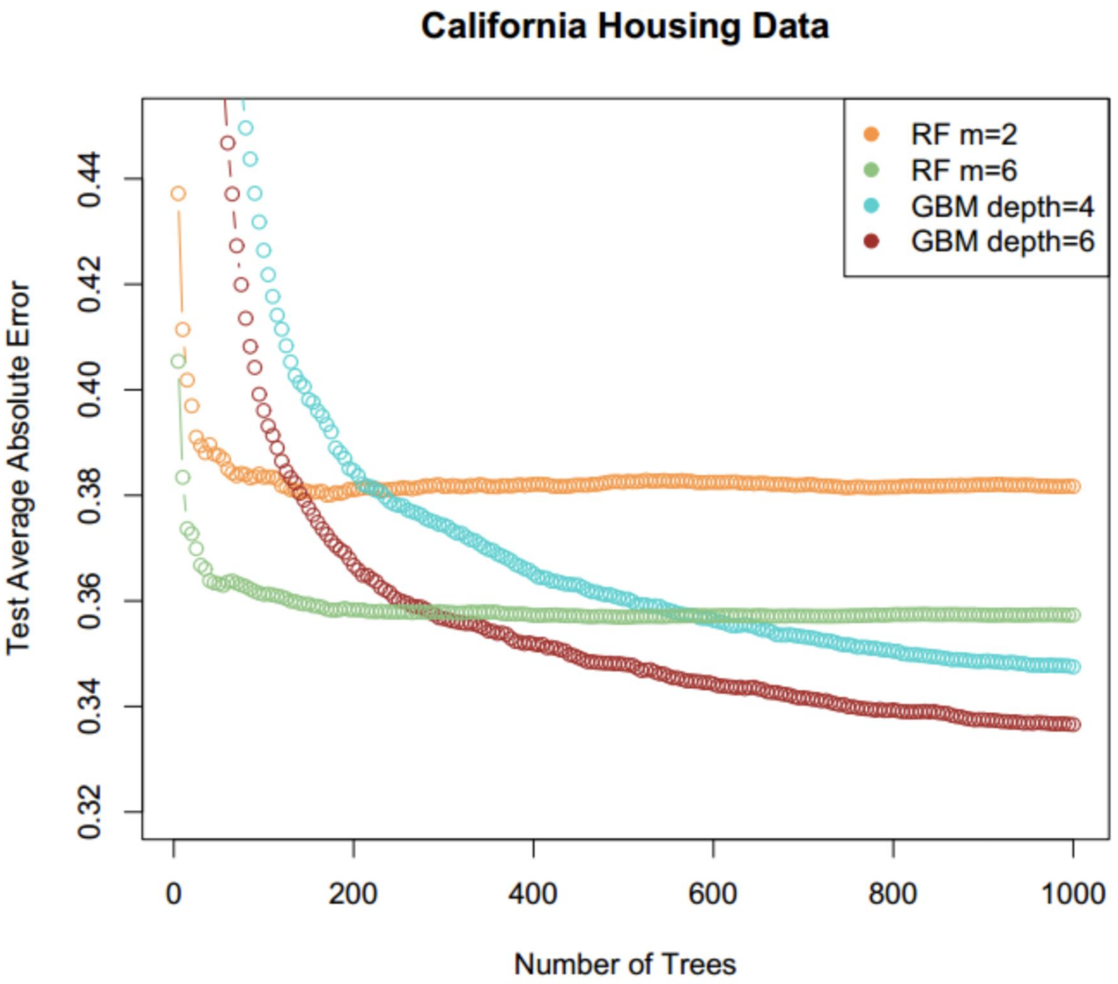


Left: full ensemble on each step.

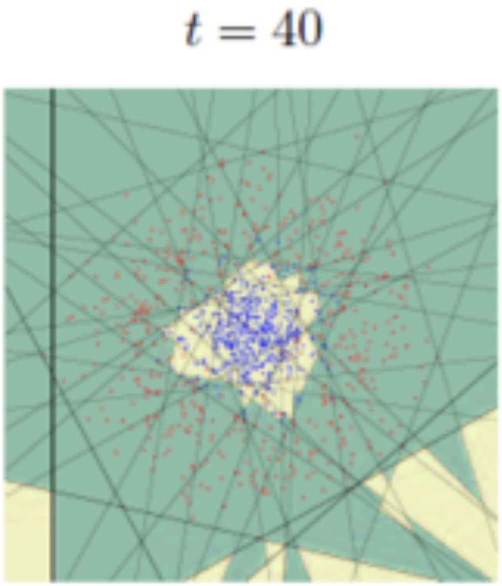
Right: additional tree decisions.

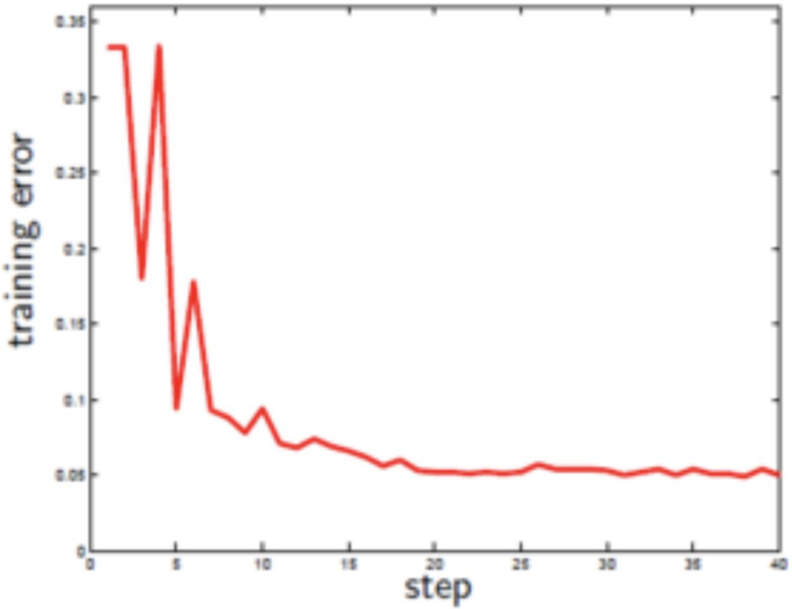
Example by ODS; source: https://habr.com/ru/company/ods/blog/327250/


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

Boosting with linear classification methods





Technical side: training in parallel

Which of the ensembling methods could be parallelized?

Technical side: training in parallel

Which of the ensembling methods could be parallelized?

- Random Forest: parallel on the forest level (all trees are independent)

Technical side: training in parallel

Which of the ensembling methods could be parallelized?

- Random Forest: parallel on the forest level (all trees are independent)
- Gradient boosting: parallel on one tree level

Recap: ensembling methods

1. Bagging.
2. Random subspace method (RSM).
3. Bagging + RSM + Decision trees = Random Forest.
4. Gradient boosting.
5. Stacking.
6. Blending.

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

Extra lecture about feature engineering and ML techniques is coming next week. Stay tuned.