

# Machine Learning

## Lecture 6: Ensembles

Harbour.Space University  
February 2020

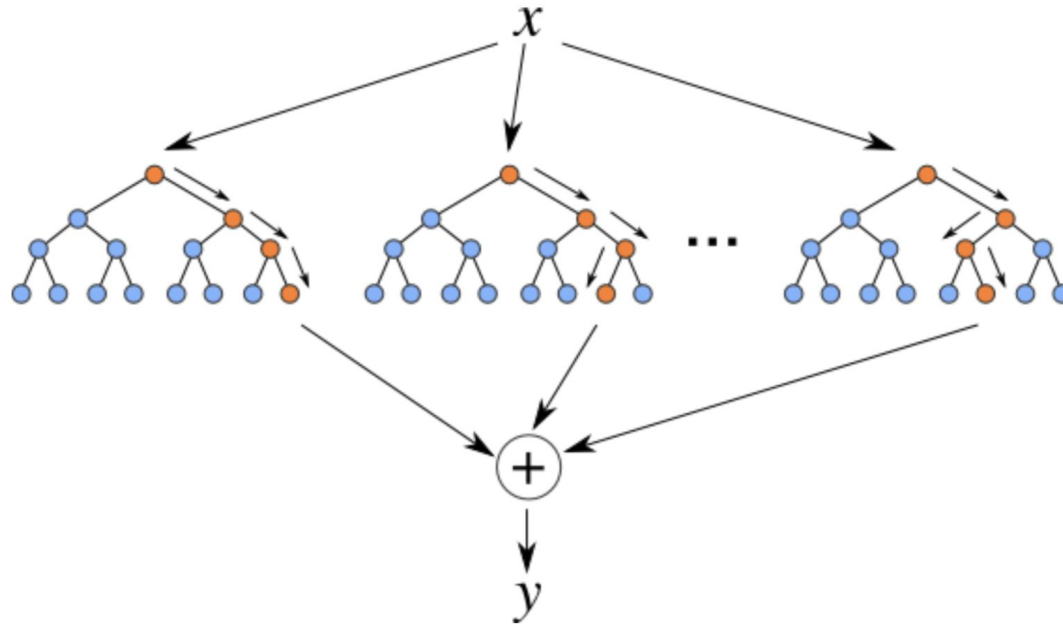
**Radoslav Neychev**

# Outline

1. Bagging & Random Forest recap
2. Bias-variance tradeoff
3. Stacking.
4. Blending.
5. Gradient boosting

# Random Forest

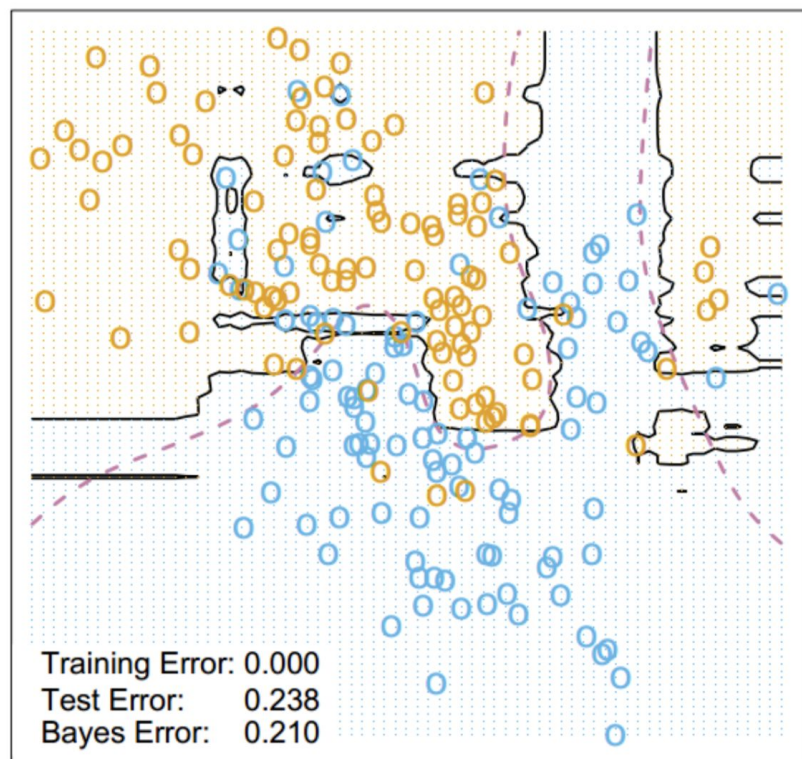
Bagging + RSM = Random Forest



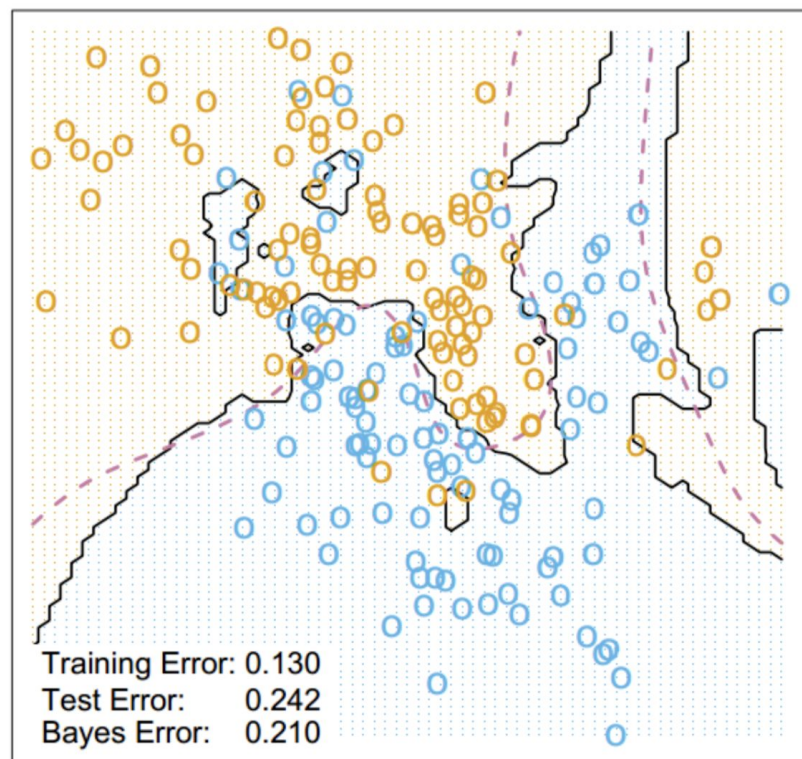
- One of the greatest “universal” models.
- There are some modifications: Extremely Randomized Trees, Isolation Forest, etc.
- Allows to use train data for validation: OOB

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

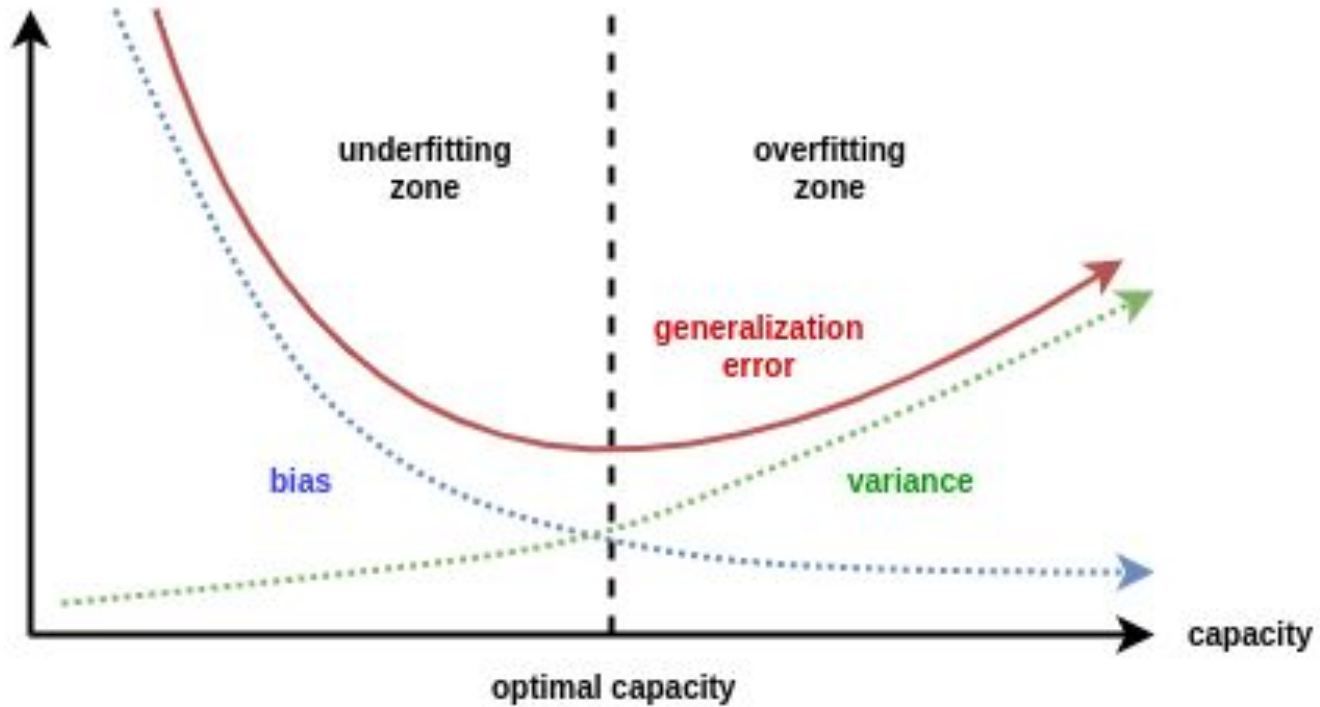
Random Forest Classifier



3-Nearest Neighbors



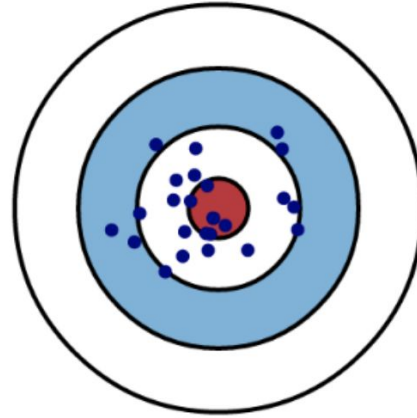
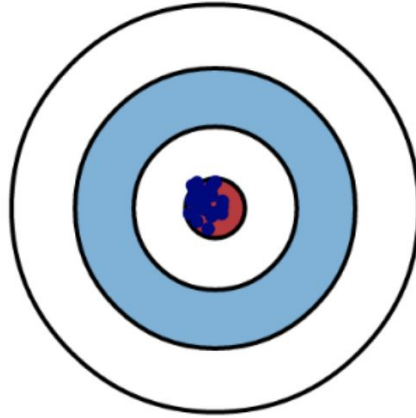
# Bias-variance tradeoff



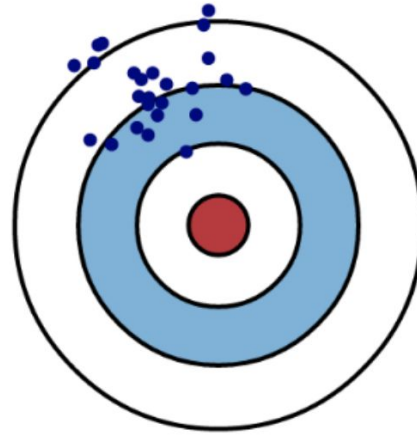
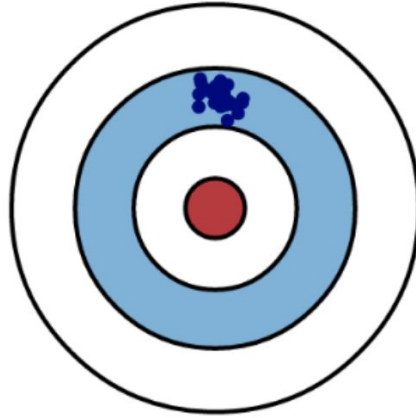
Low Variance

High Variance

Low Bias

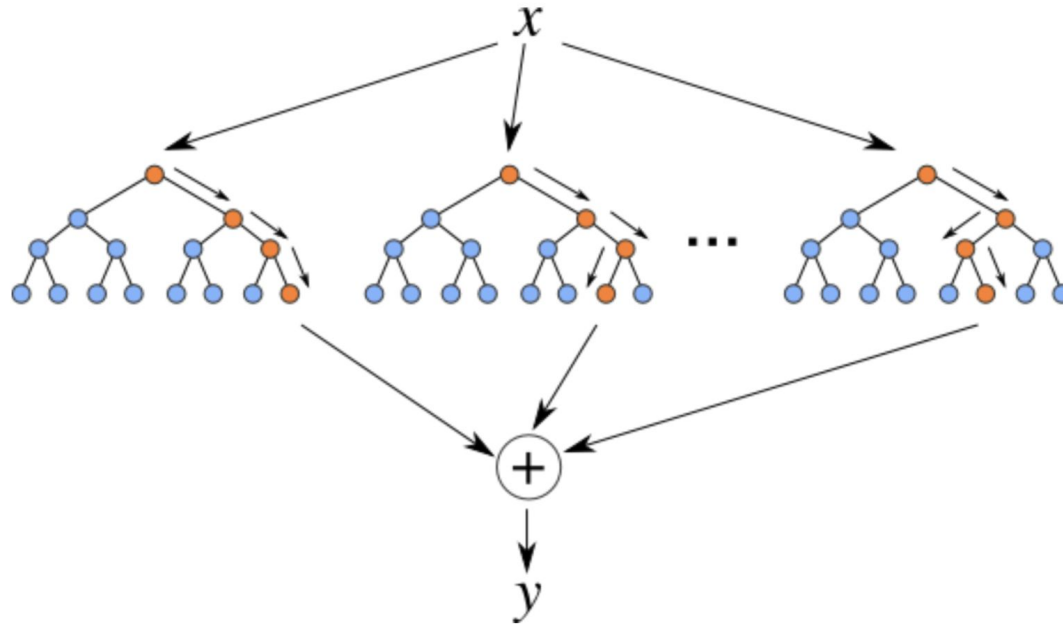


High Bias



# Random Forest

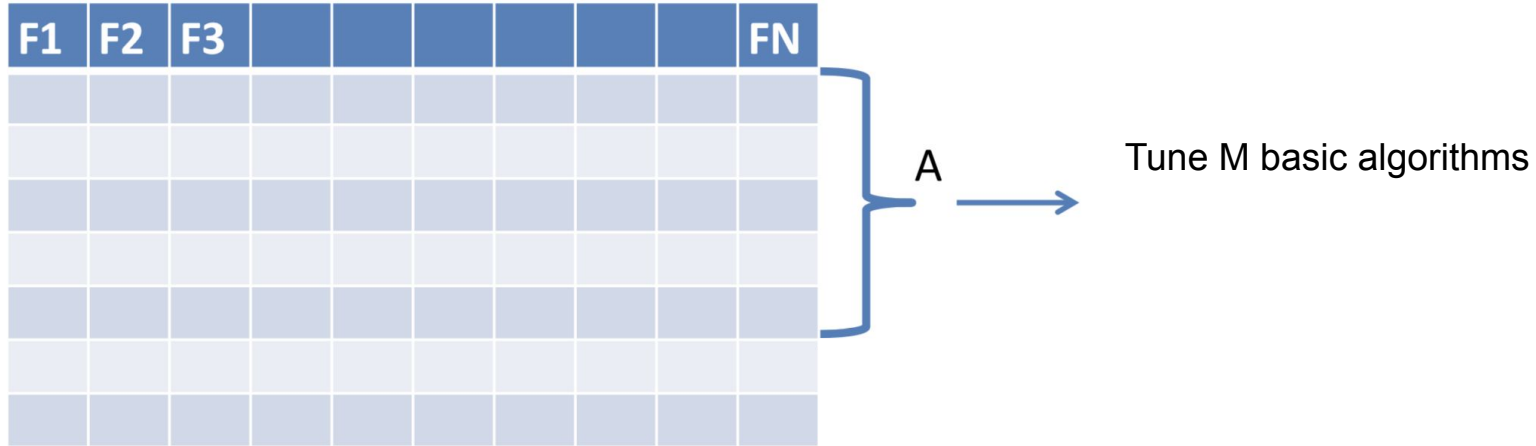
Is Random Forest decreasing bias or variance by building the trees ensemble?





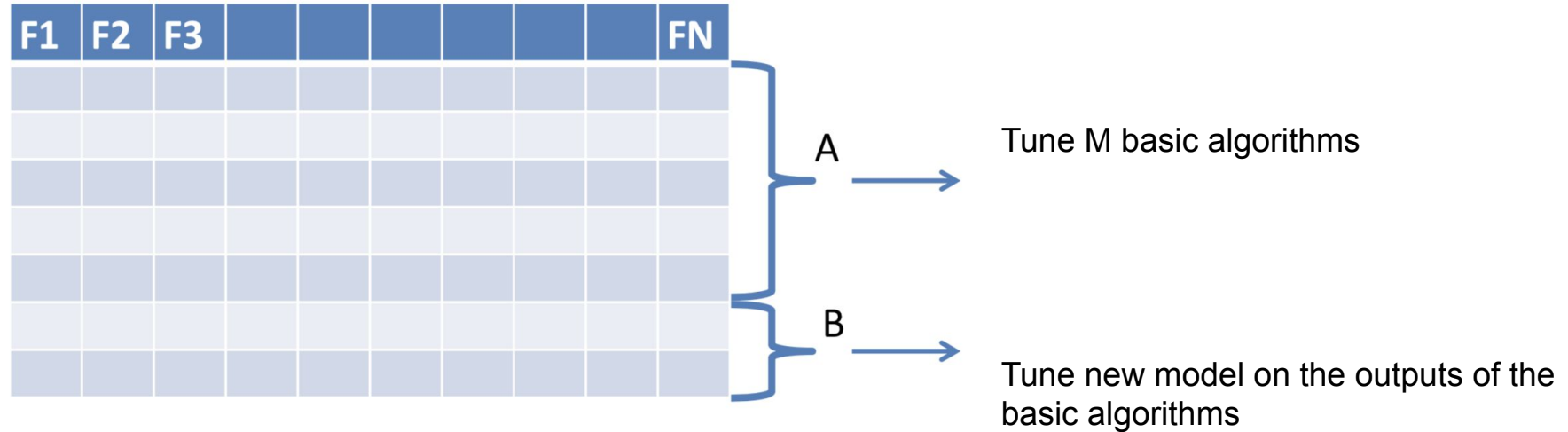
# Stacking

How to build an ensemble from *different* models?



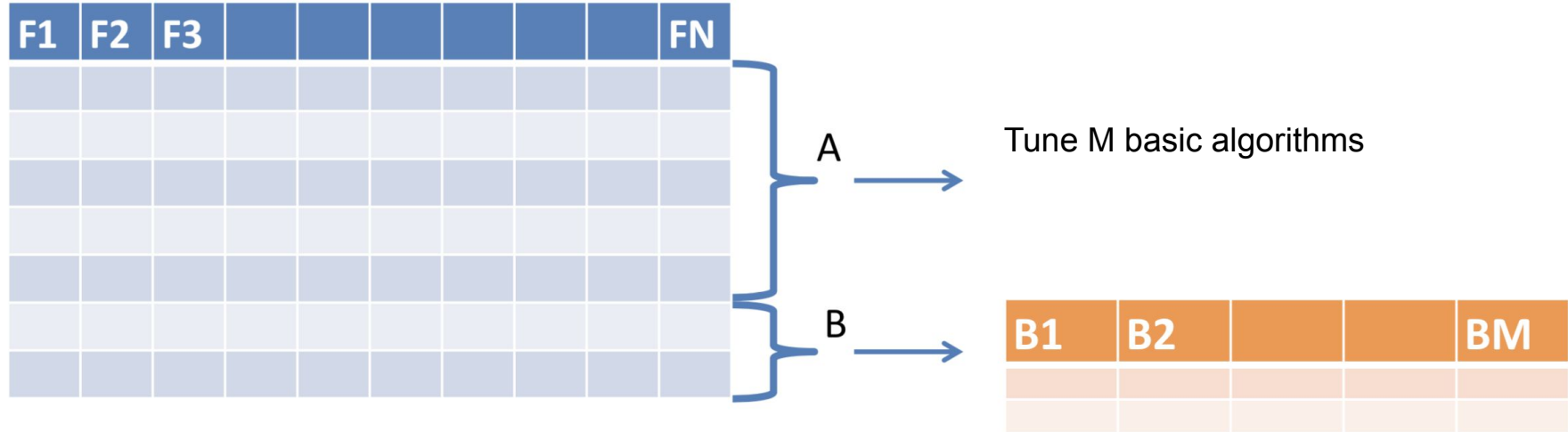
# Stacking

How to build an ensemble from *different* models?



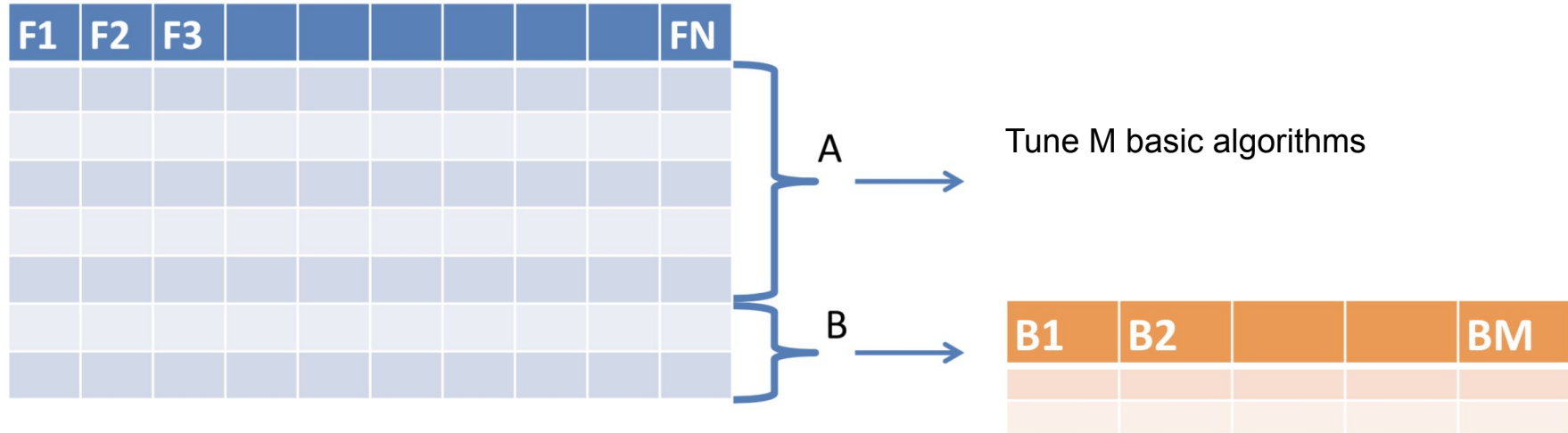
# Stacking

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.

How to build an ensemble from *different* models?

How to build an ensemble from *different* models?

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

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.)

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 :)



Just combine several *strong/complex* models.

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

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

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

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.

Just combine several *strong/complex* models.

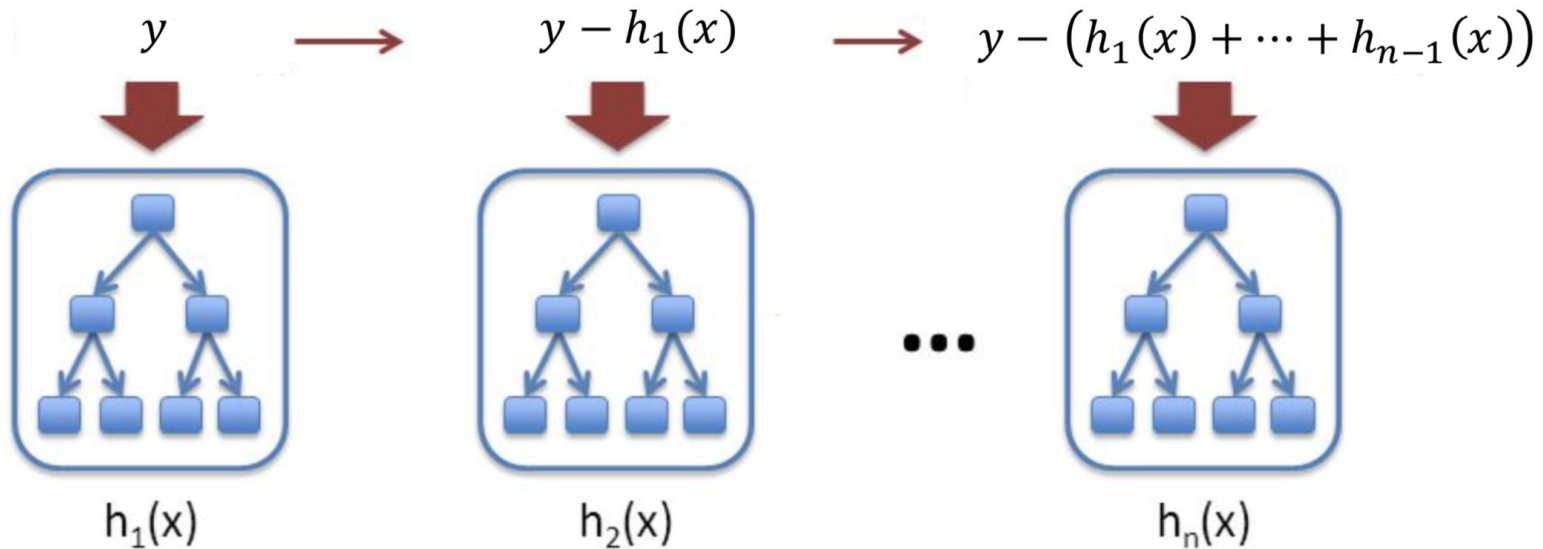
Weights should sum up to 1  
and come from  $[0; 1]$

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

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

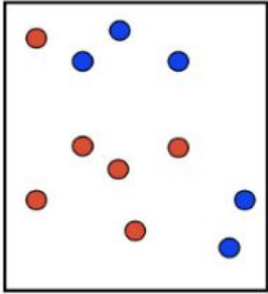
# Gradient boosting

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

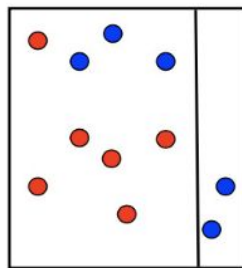
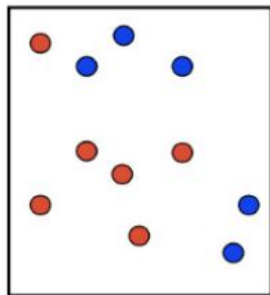


# Boosting: intuition

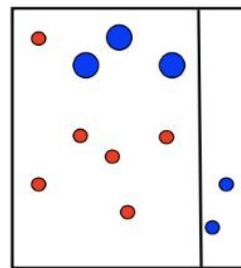
Binary classification problem.  
Models - decision stumps.



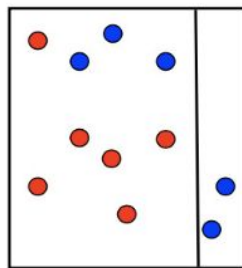
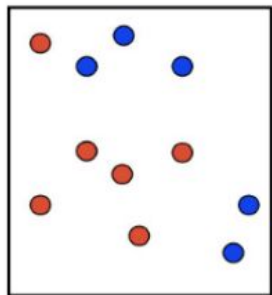
# Boosting: intuition



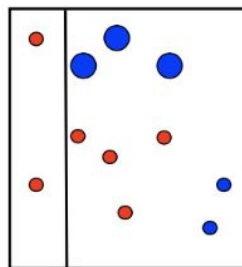
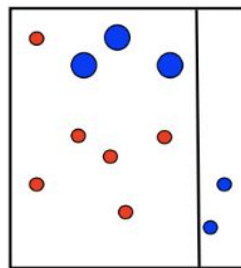
$t = 1$



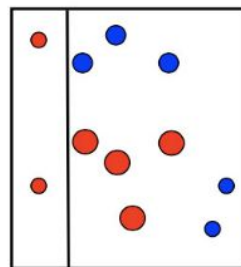
# Boosting: intuition



$t = 1$

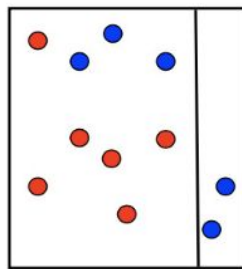
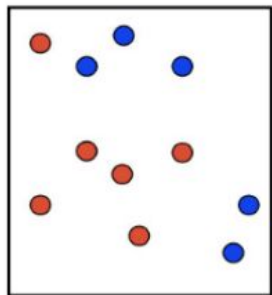


$t = 2$

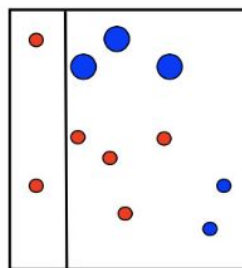
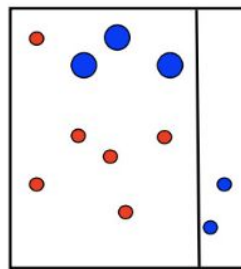




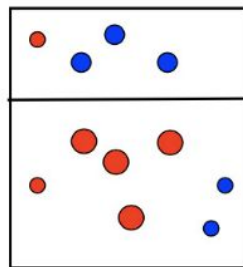
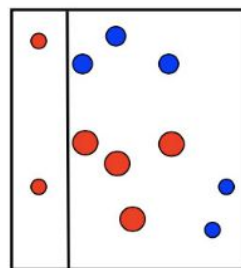
# Boosting: intuition



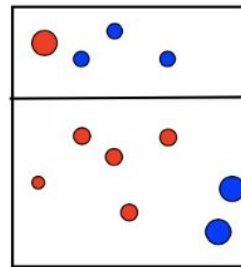
$t = 1$



$t = 2$

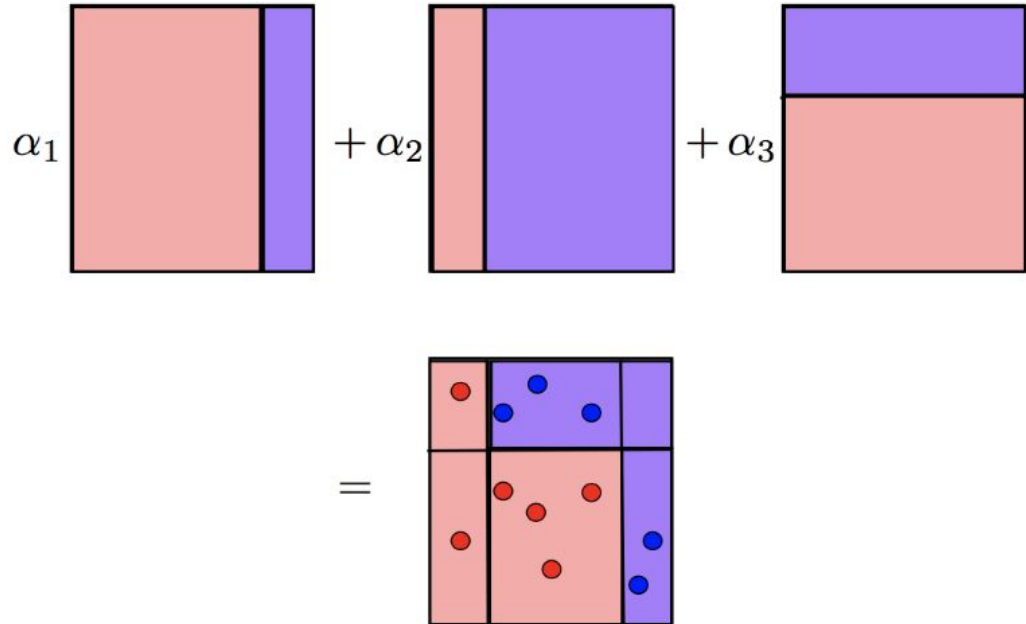
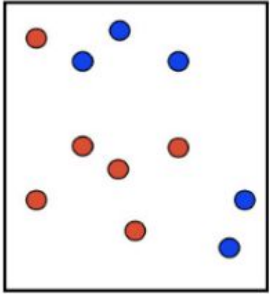


$t = 3$



# Boosting: intuition

Binary classification problem.  
Models - decision stumps.



# Gradient boosting: theory

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

# Gradient boosting: theory

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

Optimal model:

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

# 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

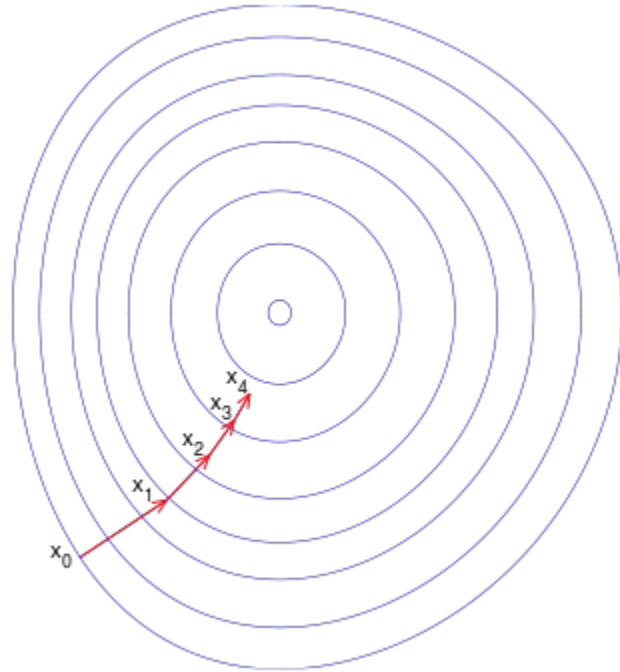
$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x),$$

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

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

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

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

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

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

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

# 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: beautiful demo

Great demo:

[http://arogozhnikov.github.io/2016/06/24/gradient\\_boosting\\_explained.html](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

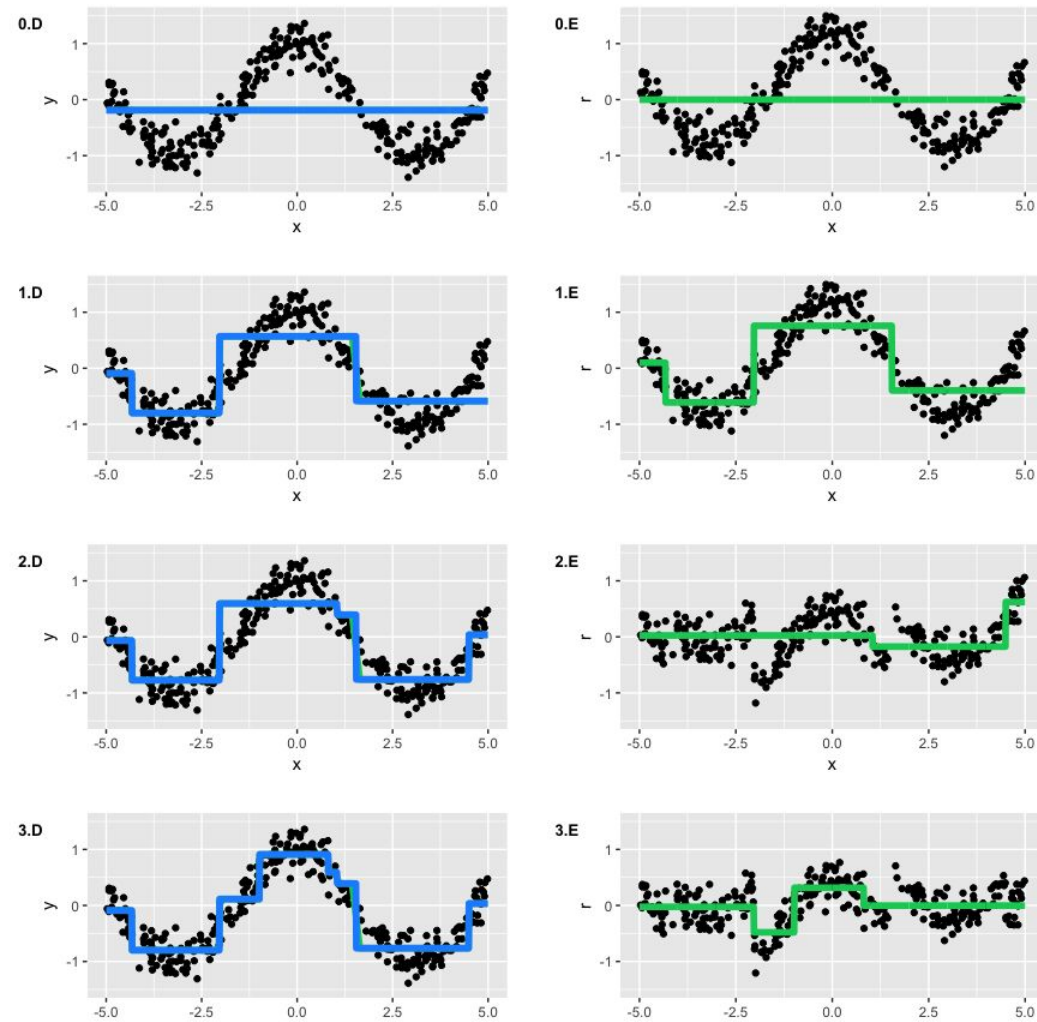
What we need:

- 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

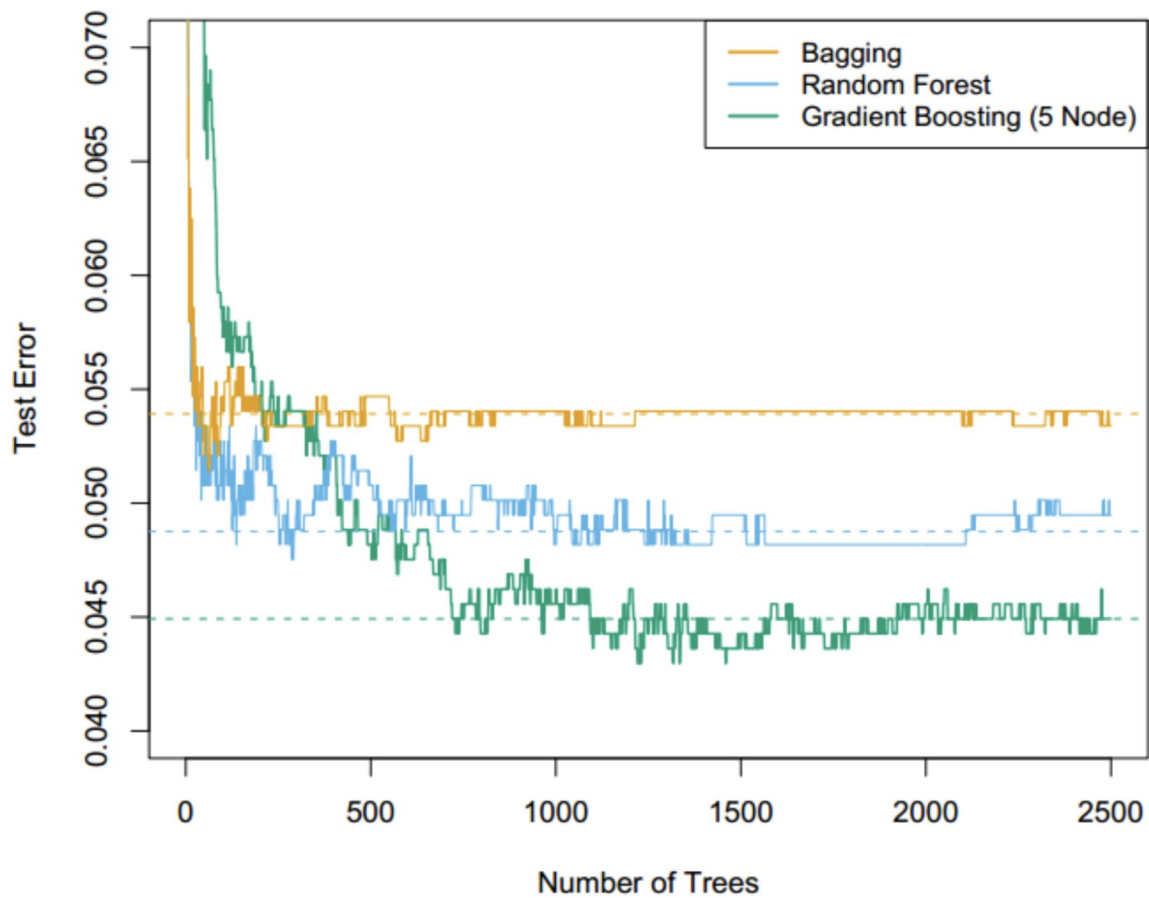
# Gradient boosting: example

Left: full ensemble on each step.

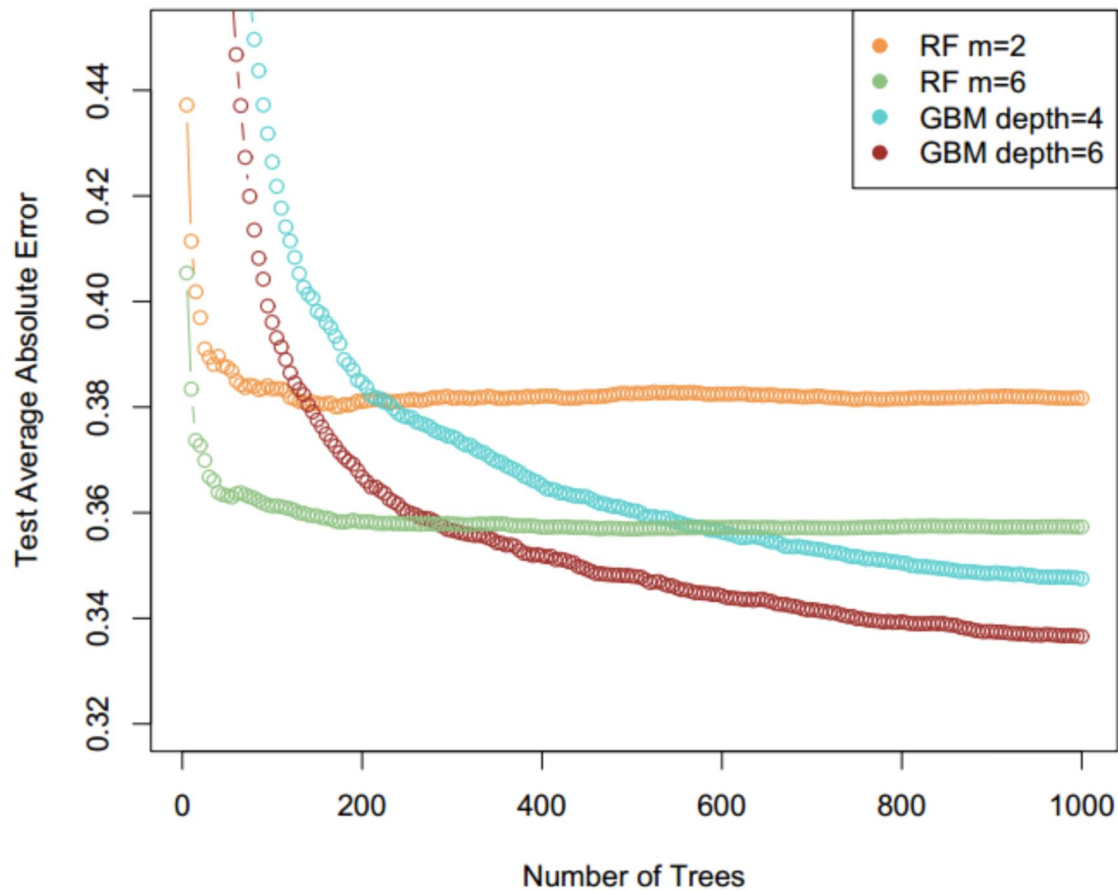
Right: additional tree decisions.



## Spam Data



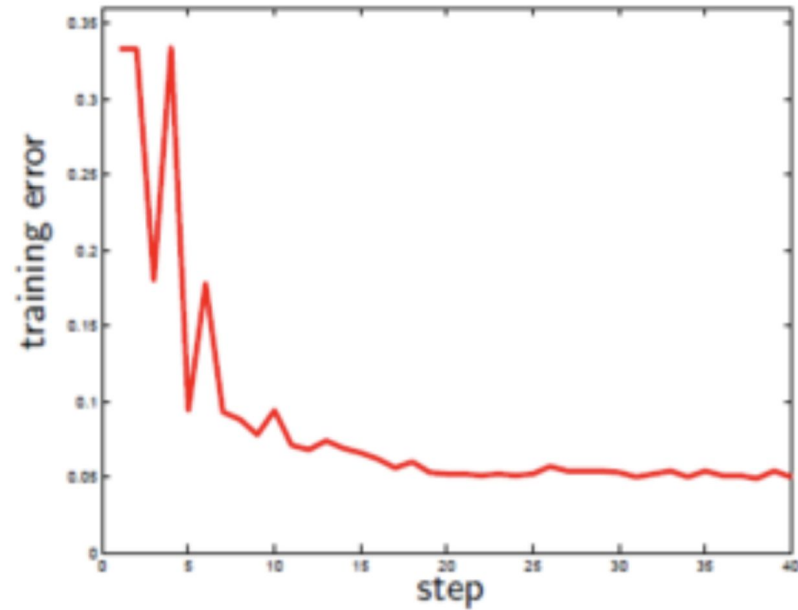
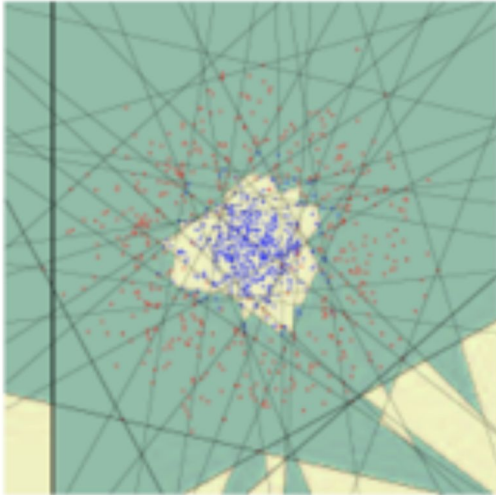
## California Housing Data





# Boosting with linear classification methods

$t = 40$



# 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](http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html)

