Machine Learning Course
basic track

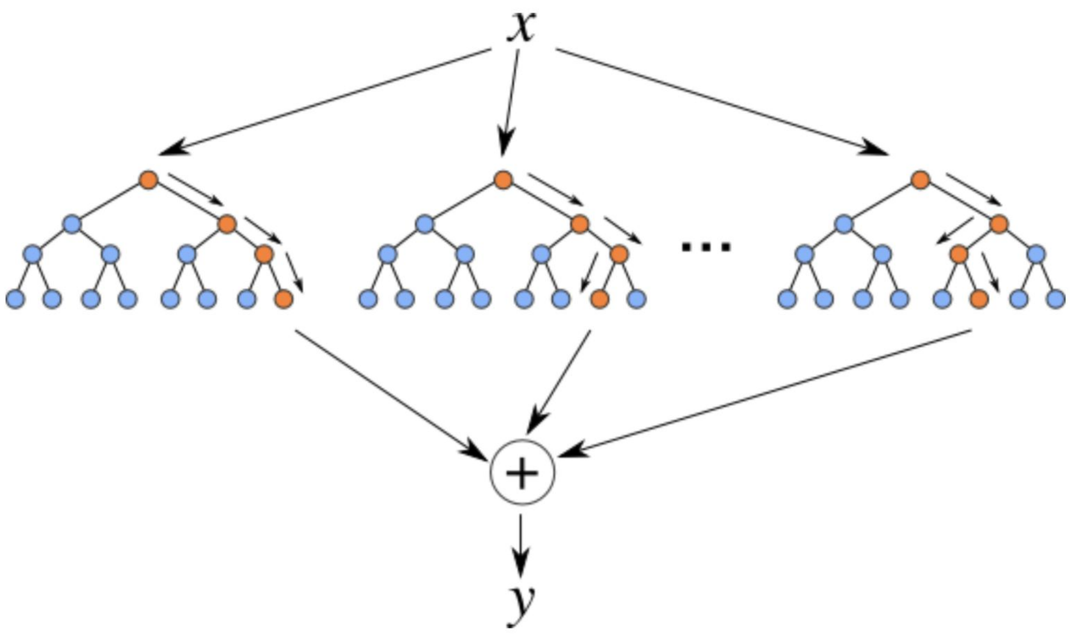
Lecture 7: Gradient boosting

Radoslav Neychev

Outline

1. Boosting intuitions
2. Gradient boosting
3. Blending
4. Stacking

Bagging + RSM = Random Forest

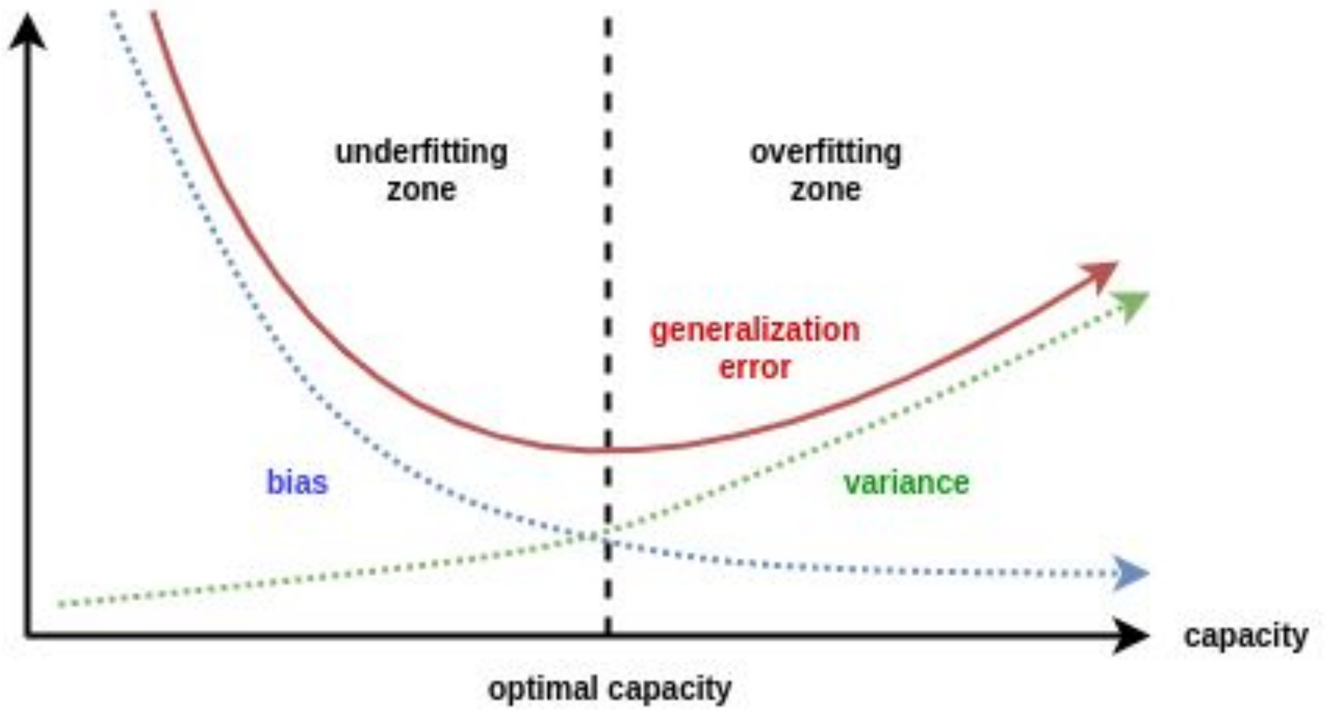


Random Forest

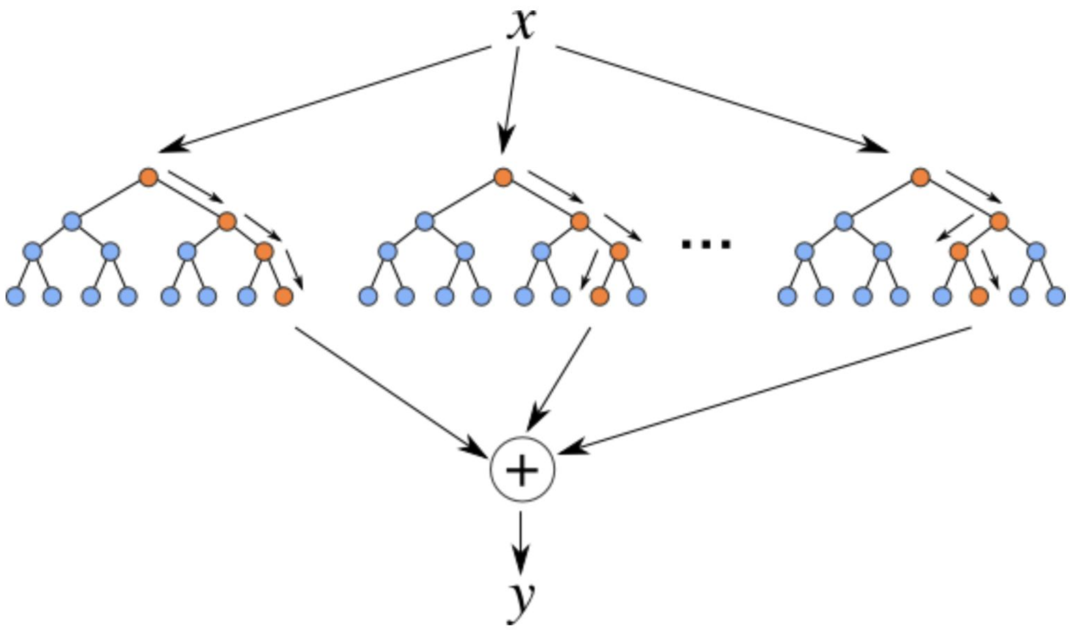
- One of the greatest “universal” models.
- There are some modifications: Extremely Randomized Trees, Isolation Forest, etc.
- Allows to use train data for validation: OOB

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

Bias-variance tradeoff



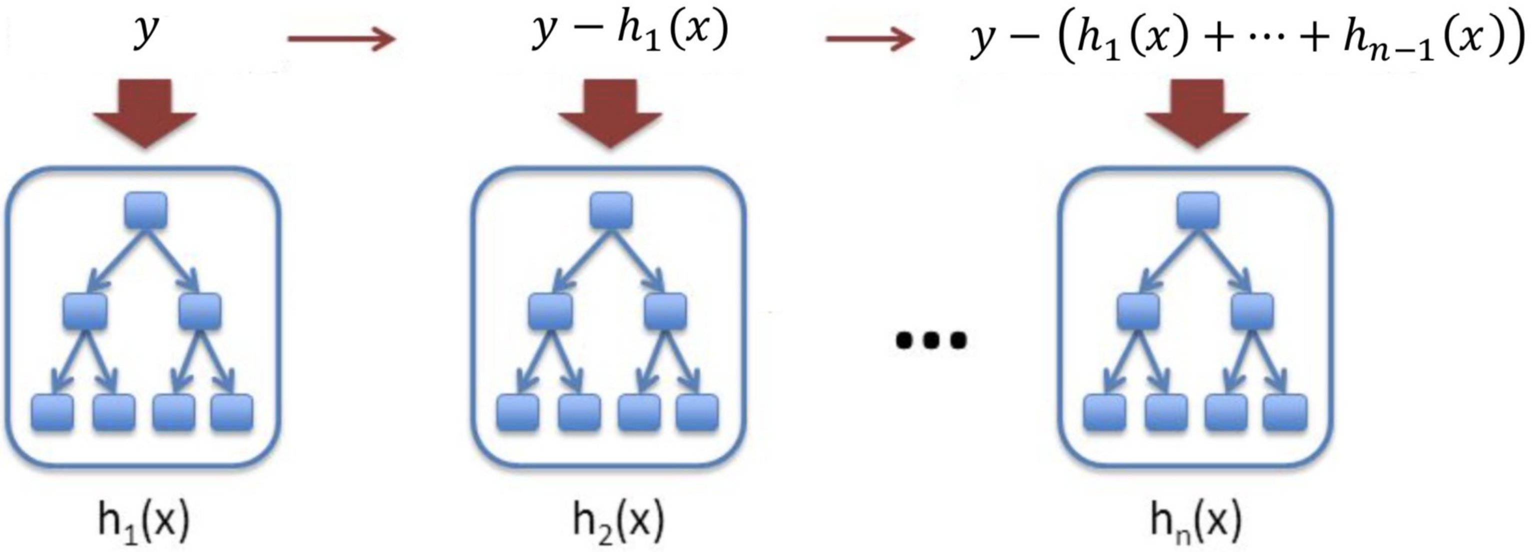
Is Random Forest decreasing bias or variance by building the trees ensemble?



Boosting

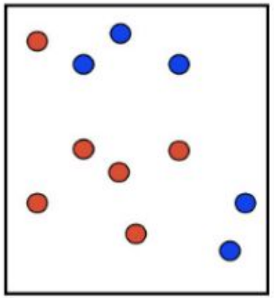
Boosting

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



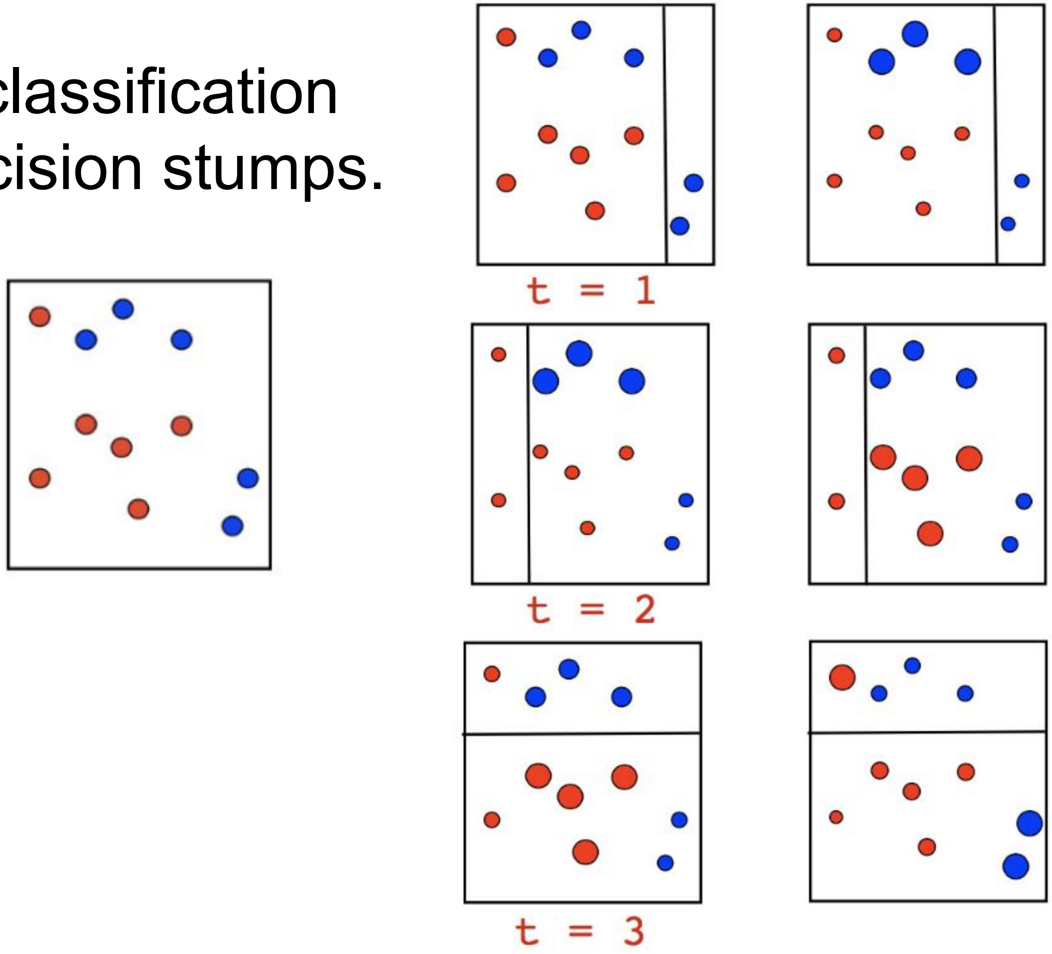
Boosting: intuition

Binary classification
Use decision stumps.



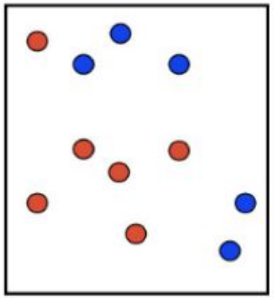
Boosting: intuition

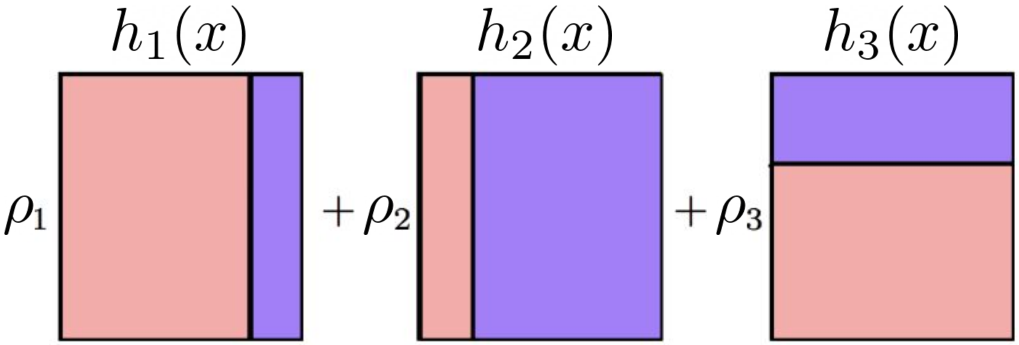
Binary classification
Use decision stumps.



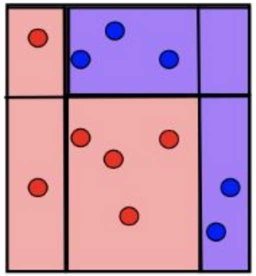
Boosting: intuition

Binary classification
Use decision stumps.

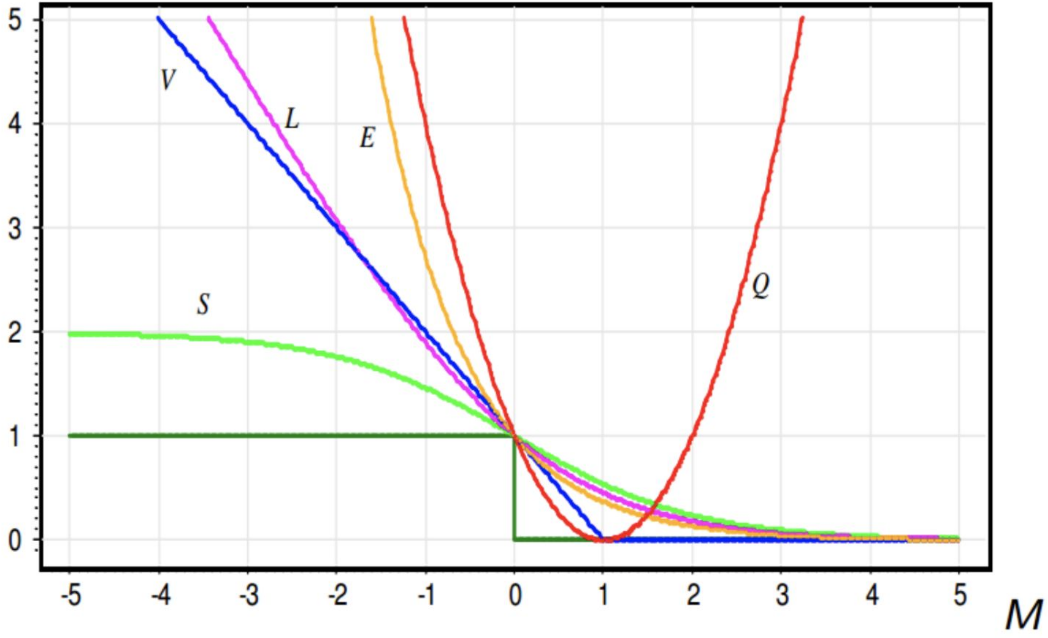




$$\hat{f}_T(x) = \sum_{t=1}^{T} \rho_t h_t(x) \quad =$$

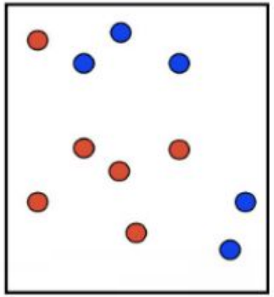


Recap: loss functions for classification



$$Q(M) = (1 - M)^2$$
$$V(M) = (1 - M)_+$$
$$S(M) = 2(1 + e^M)^{-1}$$
$$L(M) = \log_2(1 + e^{-M})$$
$$\boxed{E(M) = e^{-M}}$$

Boosting: AdaBoost



$$\hat{f}_T(x) = \sum_{t=1}^{T} \rho_t h_t(x)$$

$$L(y_i, \hat{f}_T(x_i) = \exp(-y_i \hat{f}_T(x_i)) = \exp(-y_i \sum_{t=1}^{T} \rho_t h_t(x_i))$$

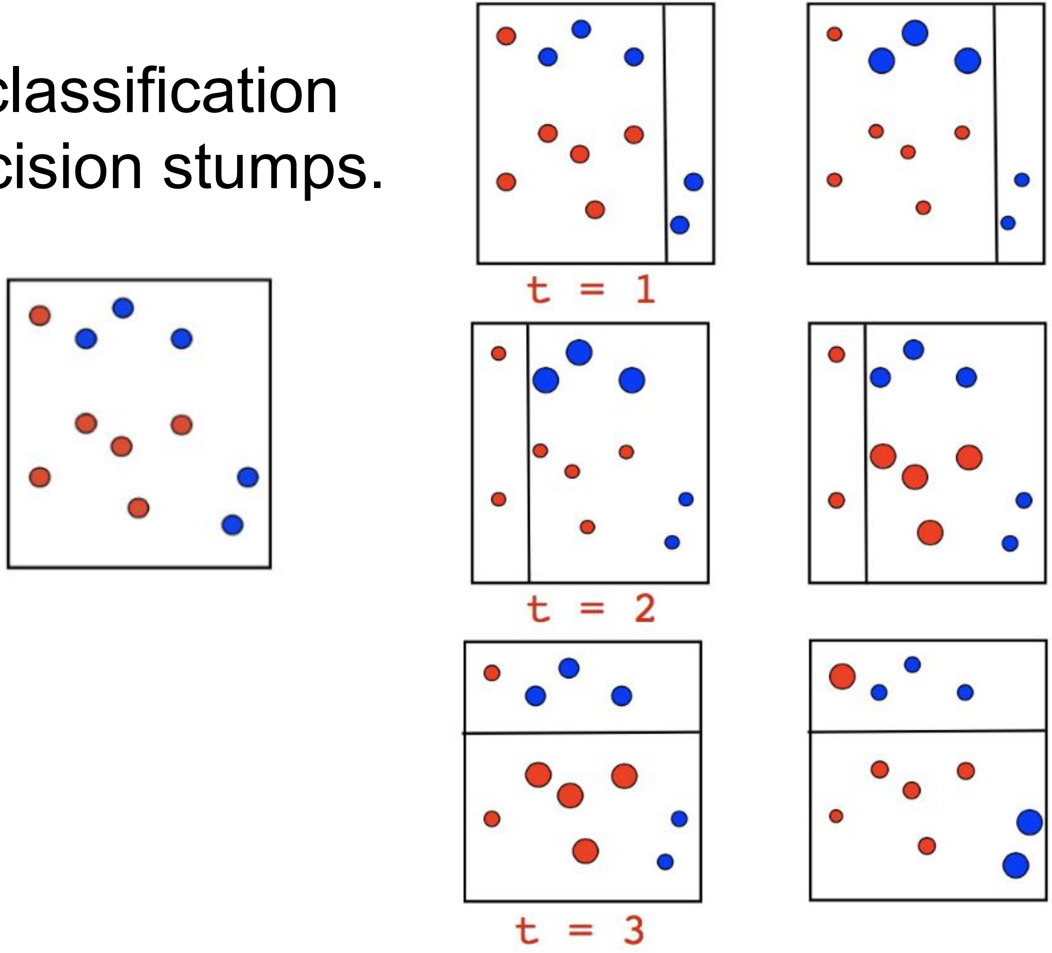
$$= \boxed{\exp\Big(- y_i \sum_{t=1}^{T-1} \rho_t h_t(x_i)\Big)} \cdot \exp(-y_i \rho_T h_T(x_i))$$

const on step T

$$= w_i \cdot \exp(-y_i \rho_T h_T(x_i))$$

Boosting: intuition

Binary classification
Use decision stumps.



Gradient boosting

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

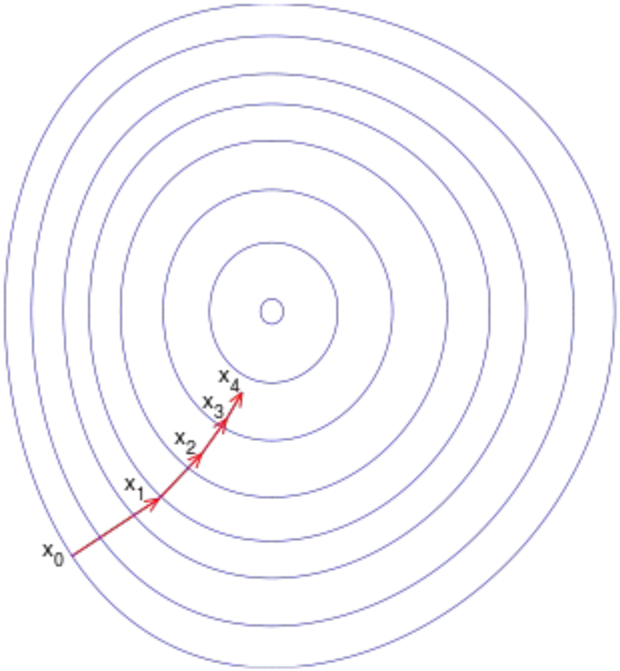
$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x),$$

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

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

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: beautiful demo

Great demo:

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

Gradient boosting

What we need:

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

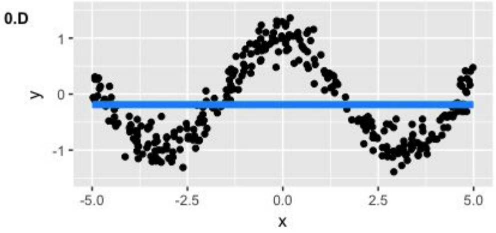
Gradient boosting: example

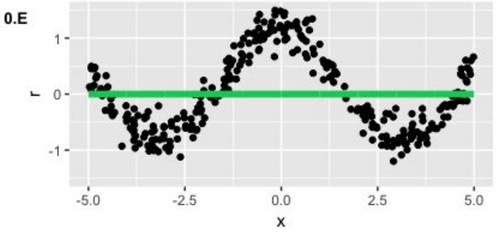
What we need:

- 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



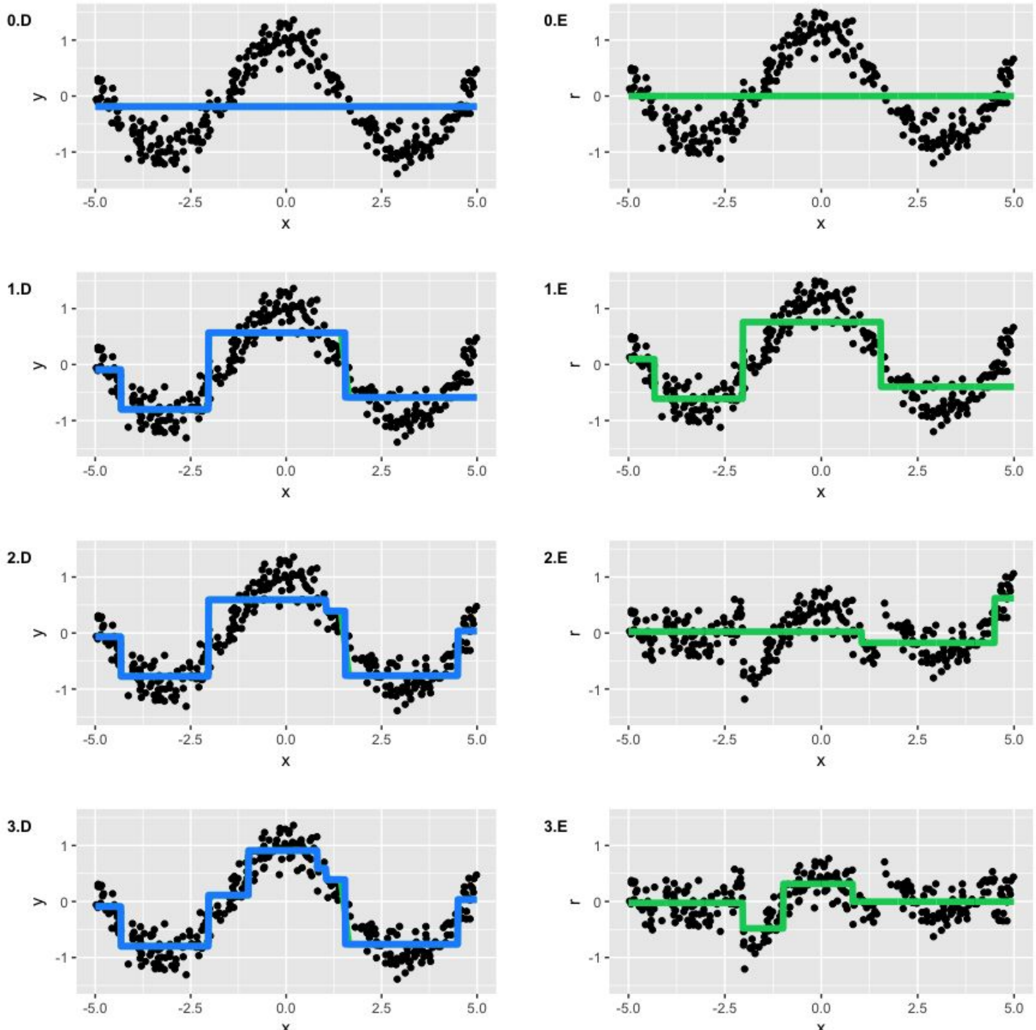


Left: full ensemble on each step.

Right: additional tree decisions.

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

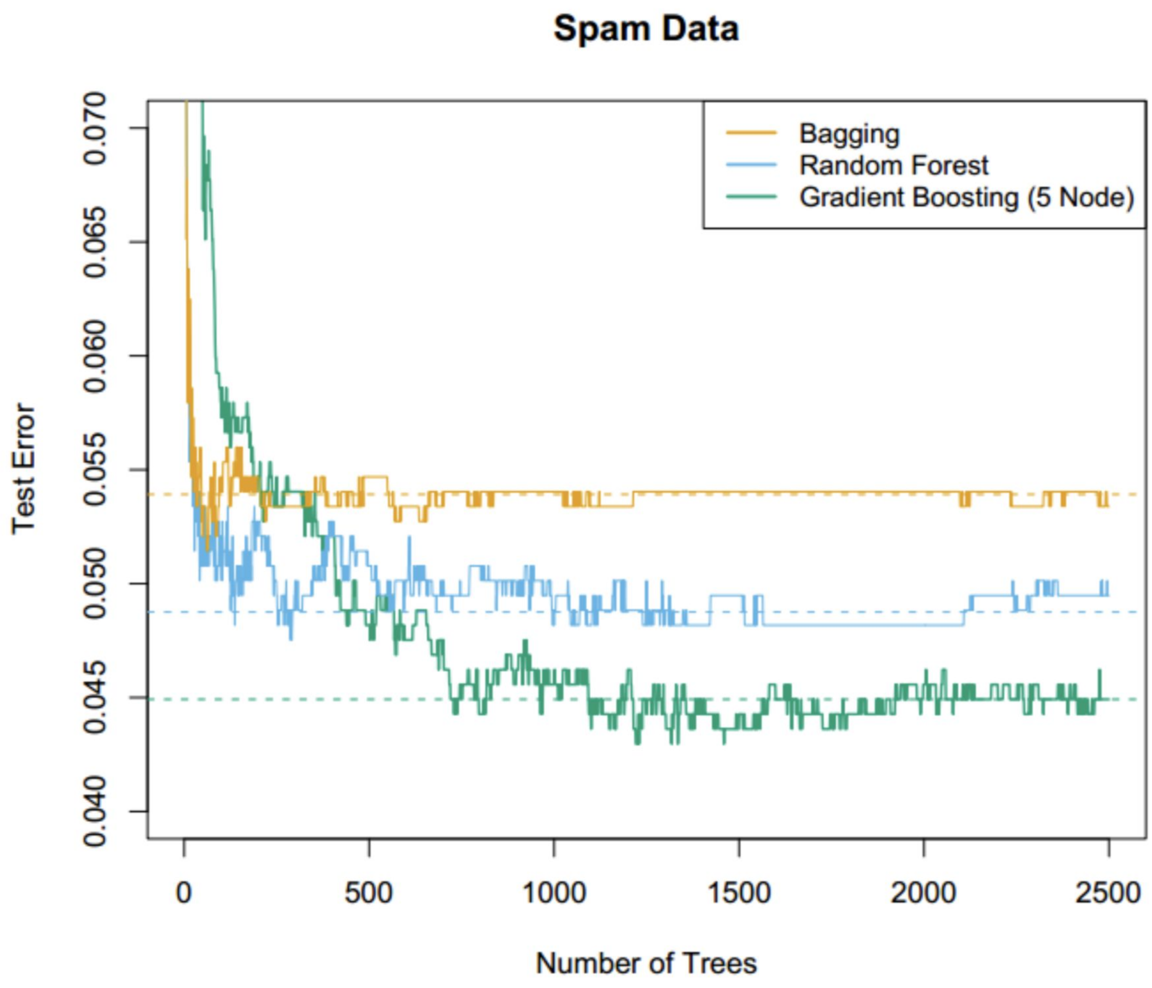
Gradient boosting: example



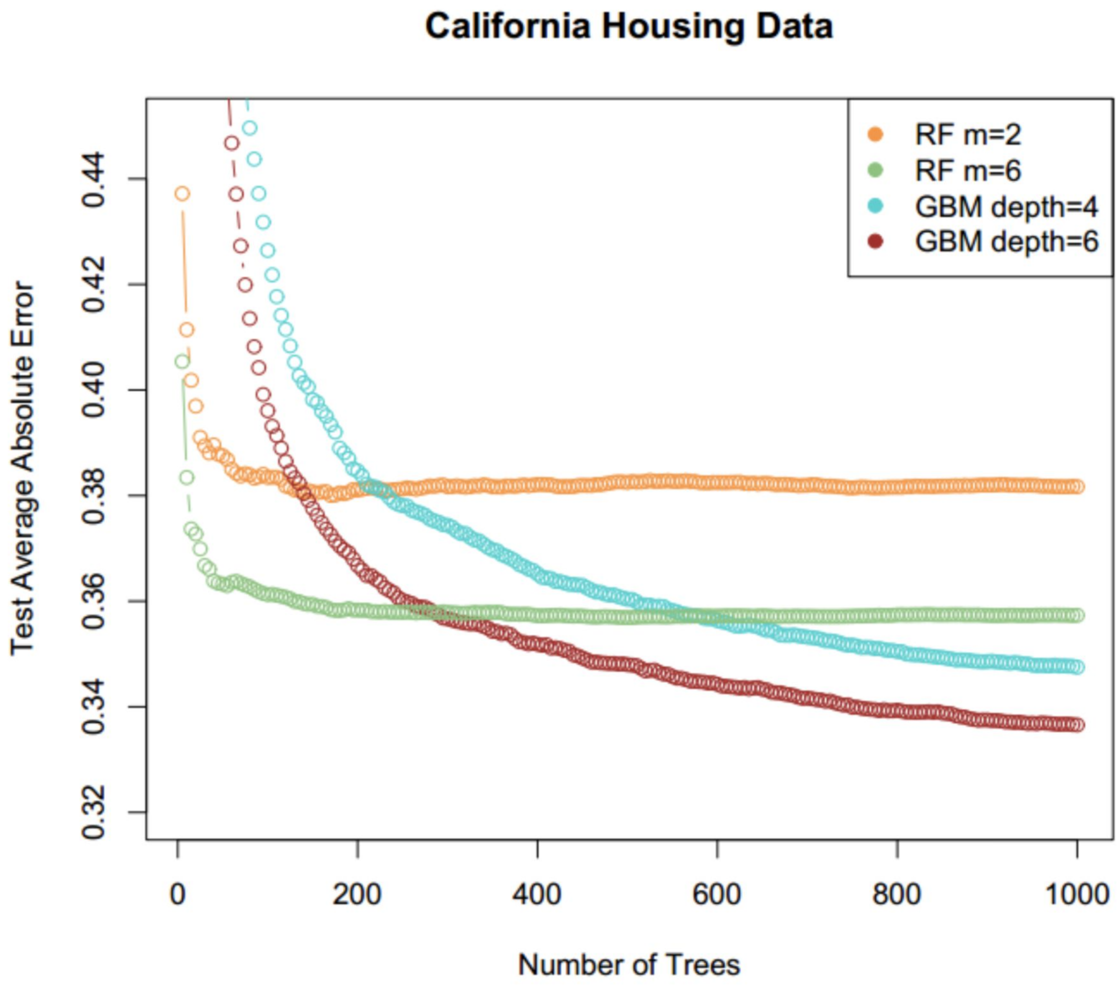
Left: full ensemble on each step.

Right: additional tree decisions.

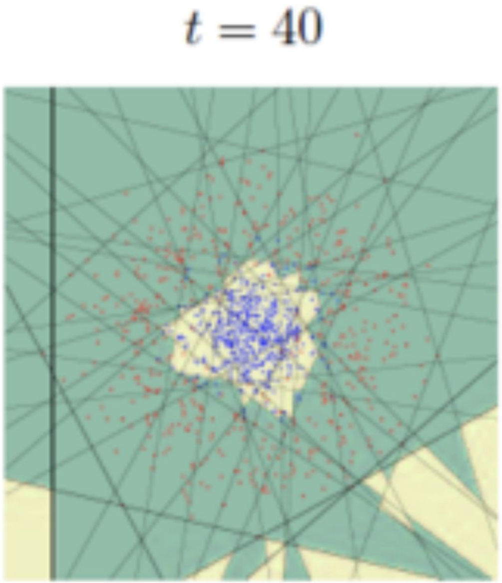
Example by ODS; source: https://habr.com/ru/company/ods/blog/327250/

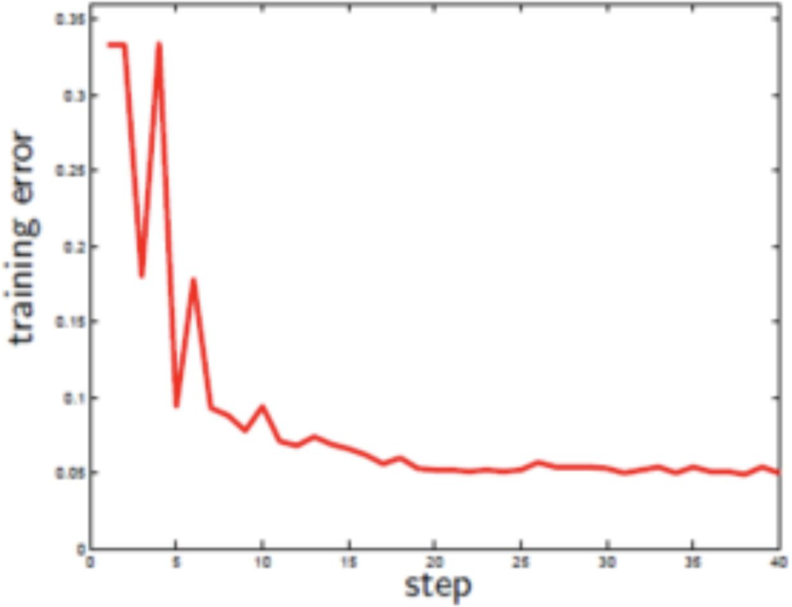


Spam Data
Bagging
Random Forest
Gradient Boosting (5 Node)
Test Error
0.040
0.045
0.050
0.055
0.060
0.065
0.070
0
500
1000
1500
2000
2500
Number of Trees


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





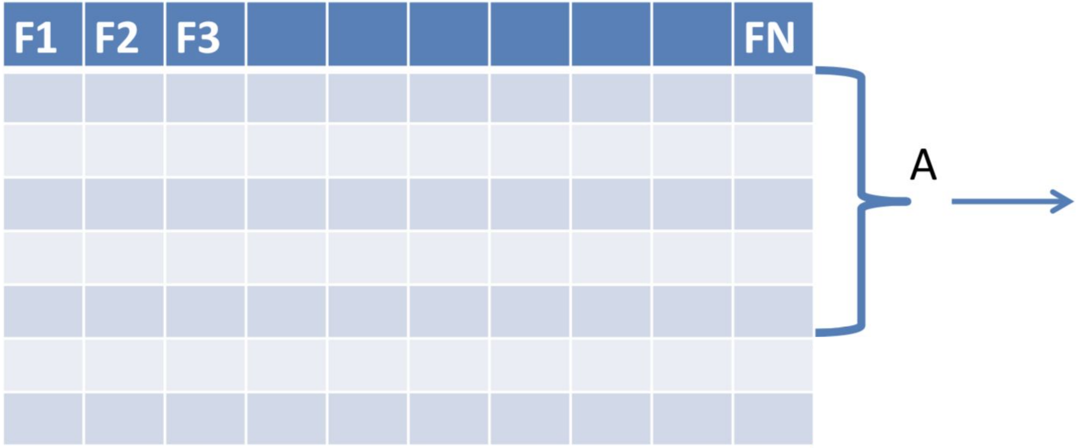
Technical side: training in parallel

Which of the ensembling methods could be parallelized?

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

Stacking and blending

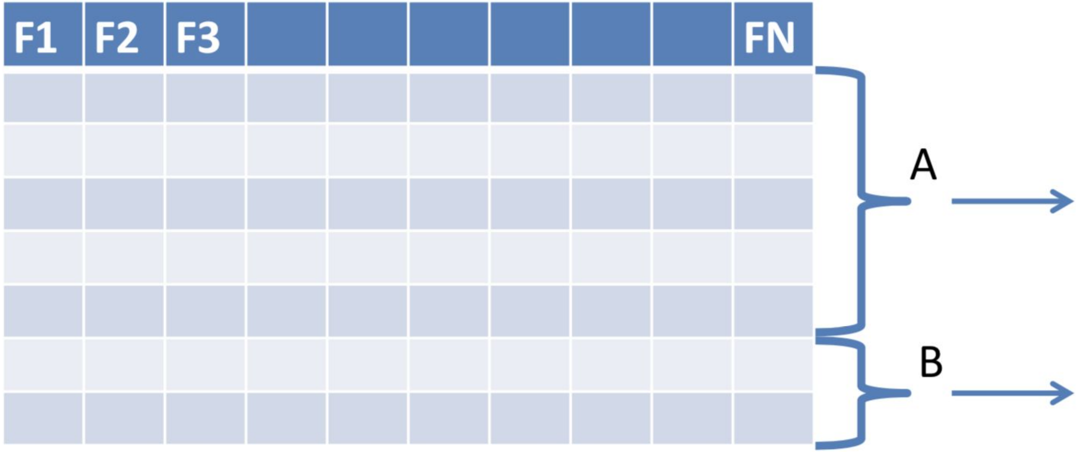
How to build an ensemble from *different* models?



Tune M basic algorithms

Blending

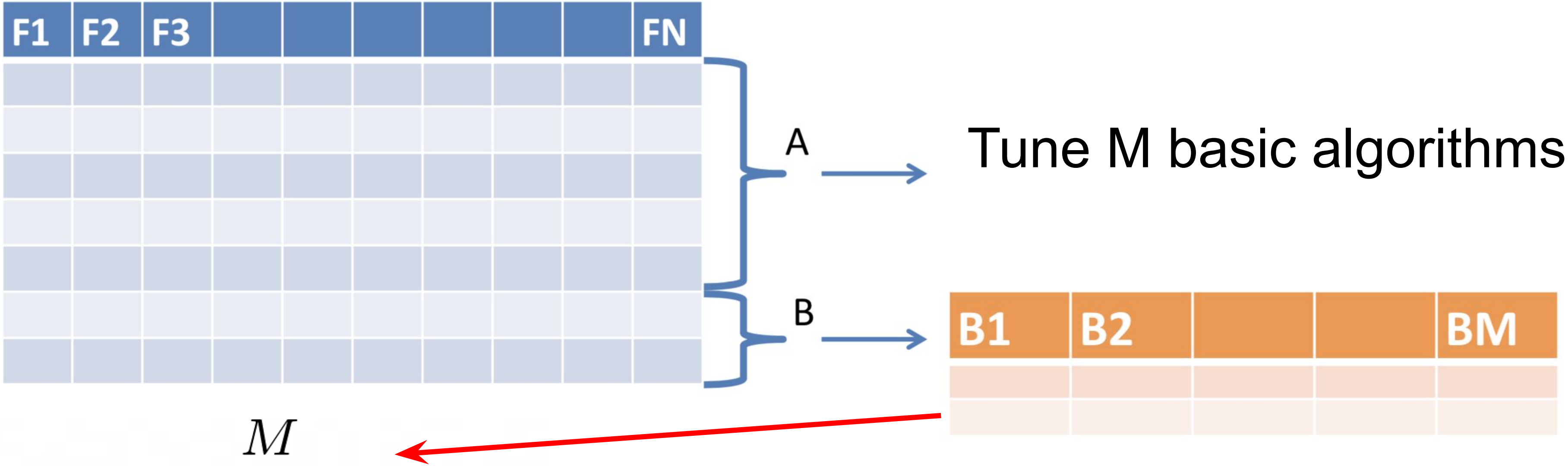
How to build an ensemble from *different* models?



Tune M basic algorithms

Tune new model on the outputs of the basic algorithms

How to build an ensemble from *different* models?



$$\hat{f}(x) = \sum_{i=1}^{M} \rho_i f_i(x)$$

$$\sum_{i=1}^{M} \rho_i = 1, \quad \rho_i \in [0; 1] \ \ \forall i$$

Blending

Just combine several *strong/complex* models.

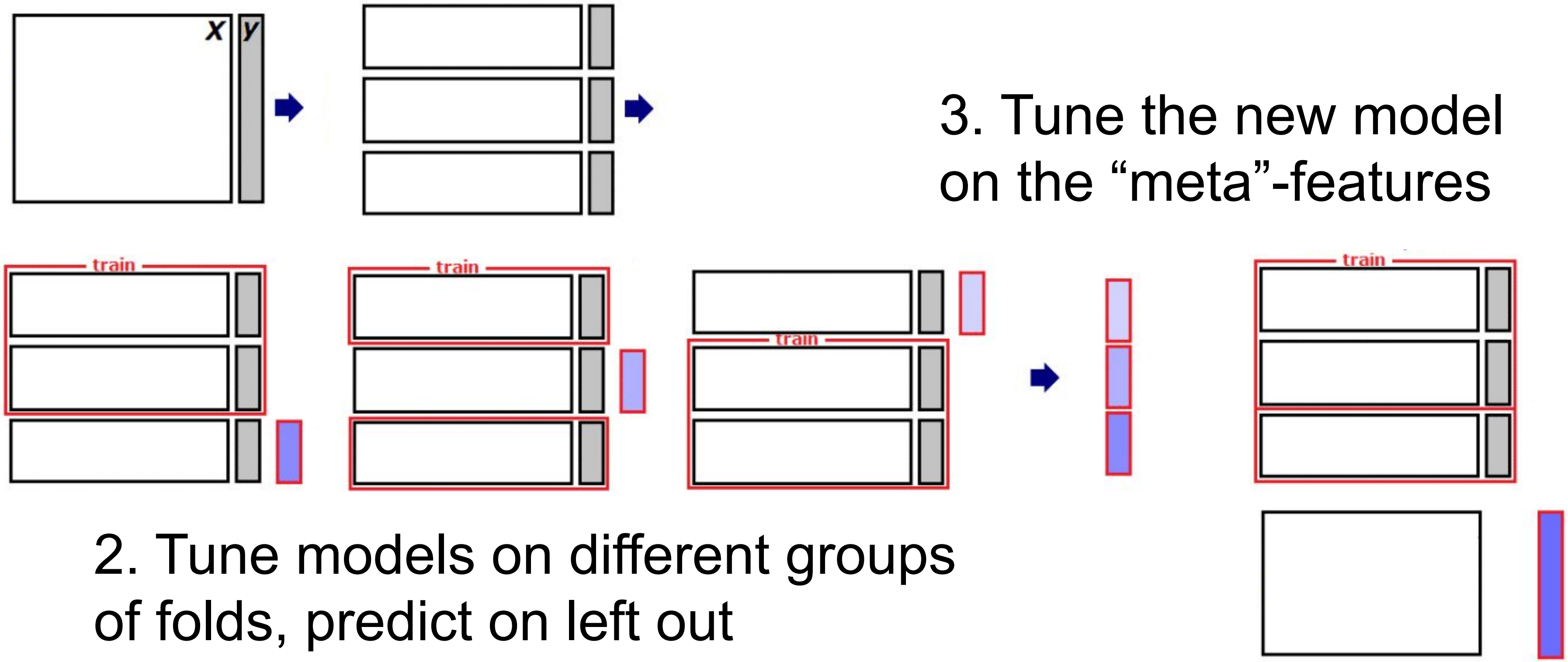
$$\hat{f}(x) = \sum_{i=1}^{M} \rho_i f_i(x), \qquad \sum_{i=1}^{M} \rho_i = 1, \quad \rho_i \in [0; 1] \;\; \forall i$$

- Pros:
  - Simple and intuitive ensembling method.
  - Average several blendings to achieve better results.
- Cons:
  - Linear composition is not always enough.
  - Need to split the data. How to fix it?

Stacking

1. Split data into folds

3. Tune the new model on the "meta"-features



2. Tune models on different groups of folds, predict on left out

Picture source: https://dyakonov.org/2017/03/10/стекинг-stacking-и-блендинг-blending/

Stacking

- Train base algorithm(s) on different groups of folds leaving one fold out.
- Predict the meta-features on the left-out fold and test data.
- Train the meta-algorithm on the meta-features representation of the train data.
- Use it on the meta-features representation of the test data.

Stacking

- Pros:
  - Powerful ensembling method, if you know how to use it
  - Quite popular in ML-competitions
  - One might perform stacking on the meta-features dataset as well
- Cons:
  - Meta-features on each fold are actually predicted by different models
    - However, regularization usually helps
  - Hard to explain your model behaviour

Bonus:

Now you know how to stack XGBoost (or CatBoost/LightGBM)

Recap: ensembling methods

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

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