Machine Learning
Lecture 7: Gradient boosting

Harbour.Space University
February 2020

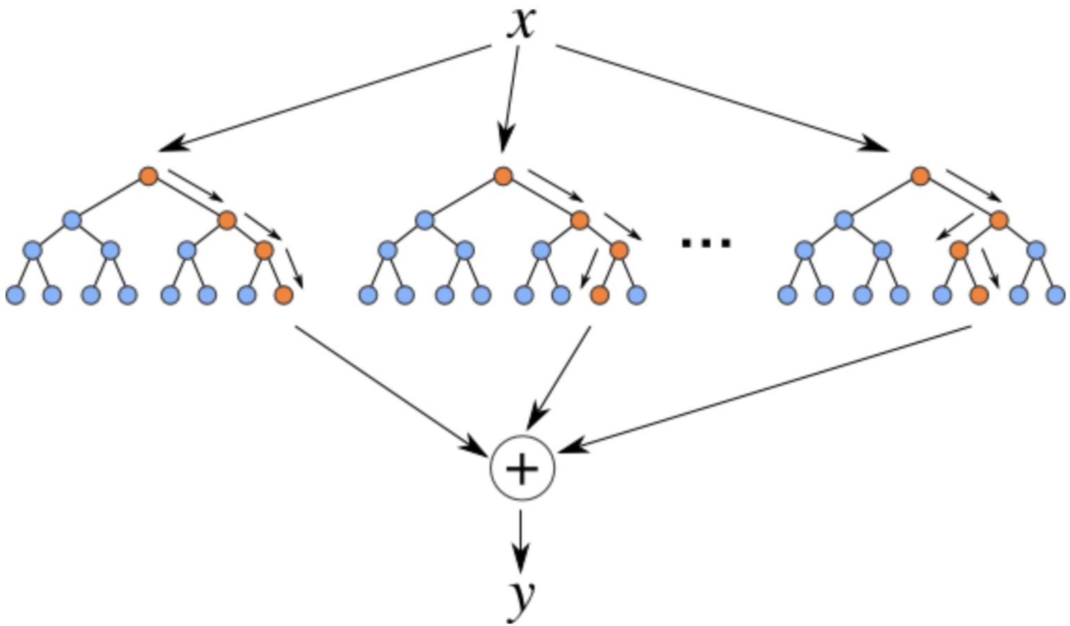
Radoslav Neychev

Outline

1. Boosting
2. Gradient boosting
3. Offtopic: word representations

Based on following materials: ml-mipt 2017 by Victor Kantor, Evgeny Sokolov HSE 2018, mlcourse_open by ODS

Bagging + RSM = Random Forest

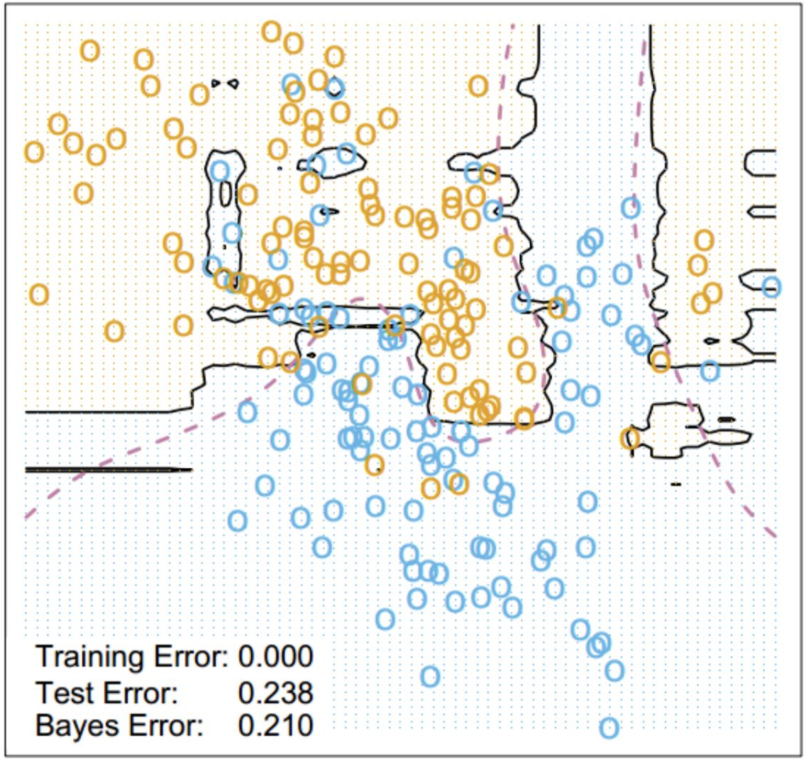


Random Forest

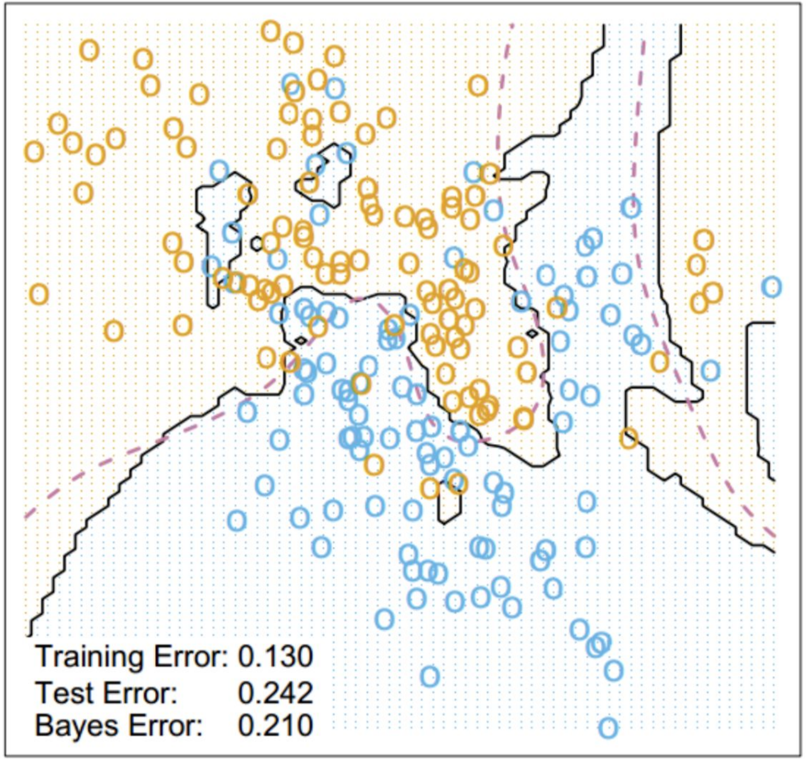
- One of the greatest “universal” models.
- There are some modifications: Extremely Randomized Trees, Isolation Forest, etc.
- Allows to use train data for validation: OOB

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

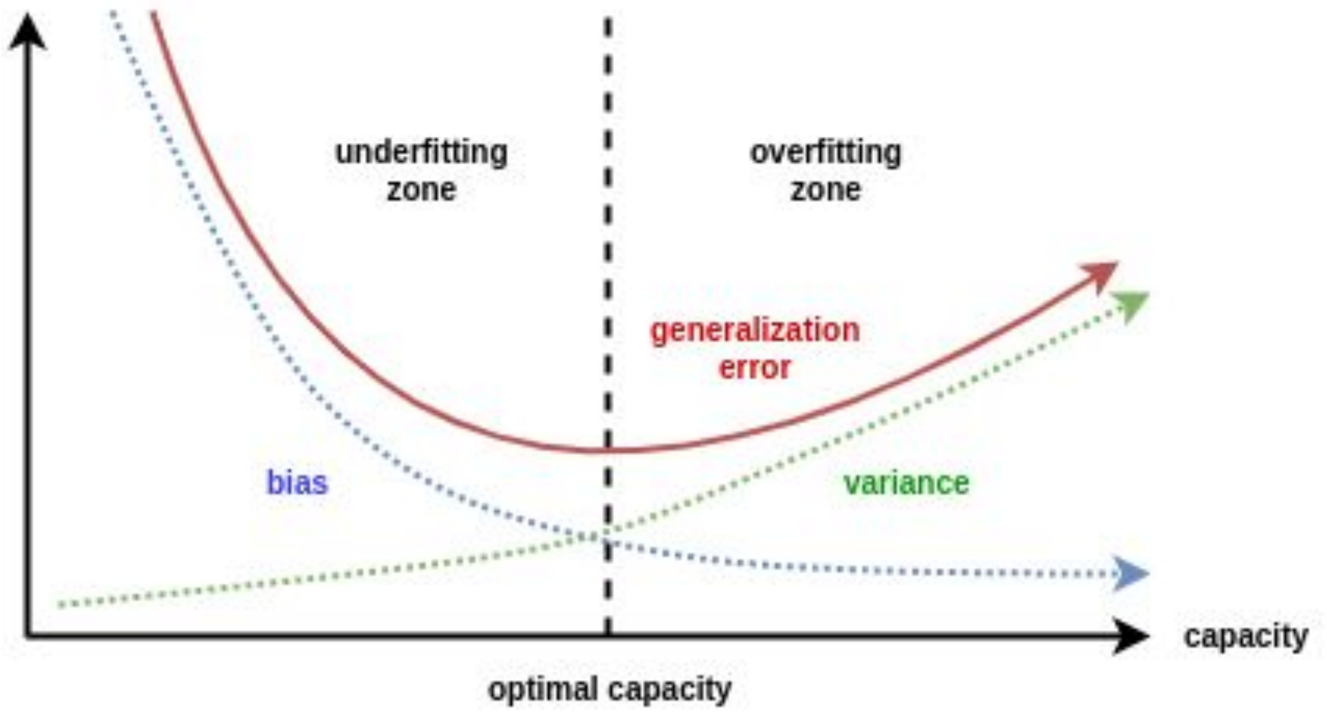
Random Forest Classifier

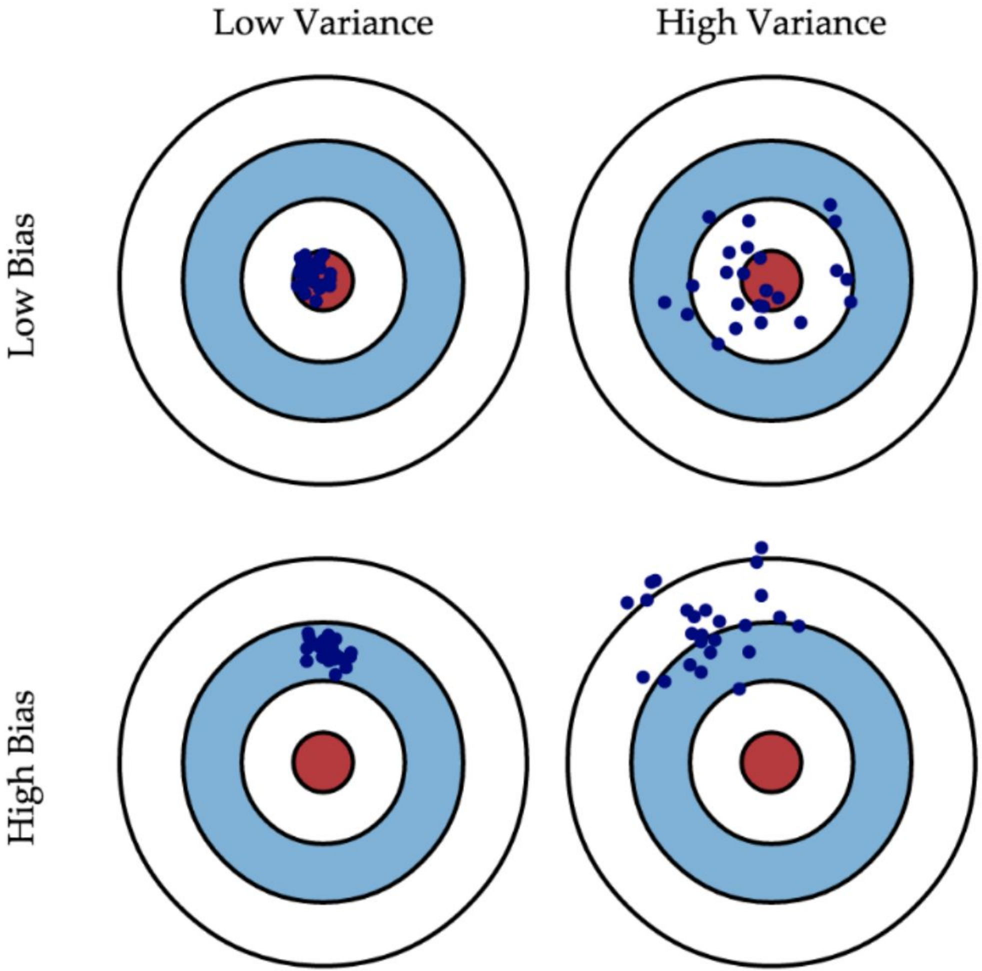


3−Nearest Neighbors

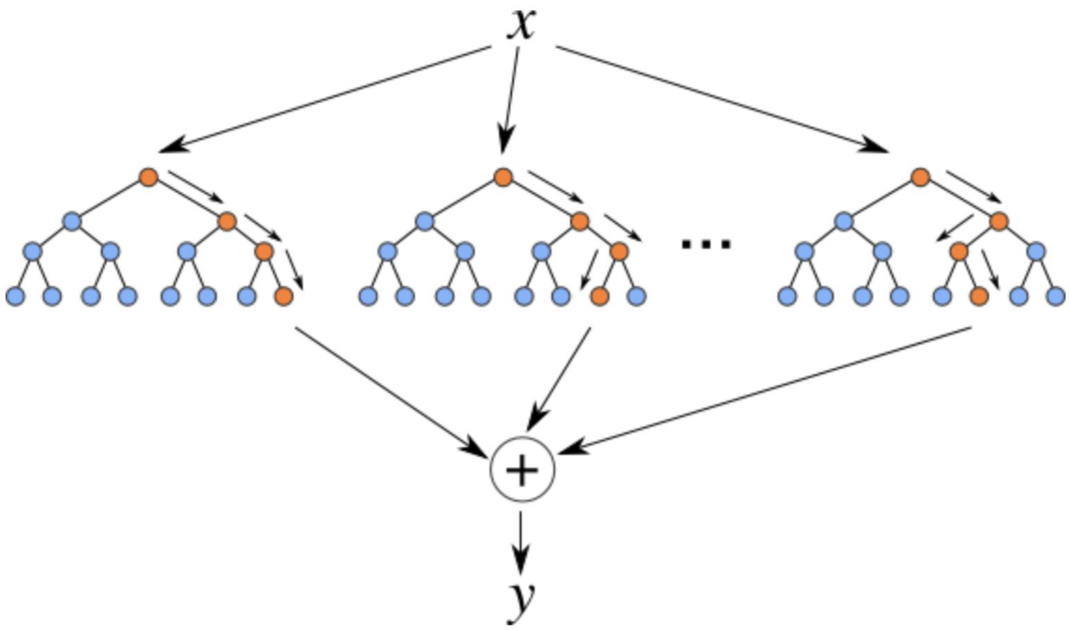


Bias-variance tradeoff



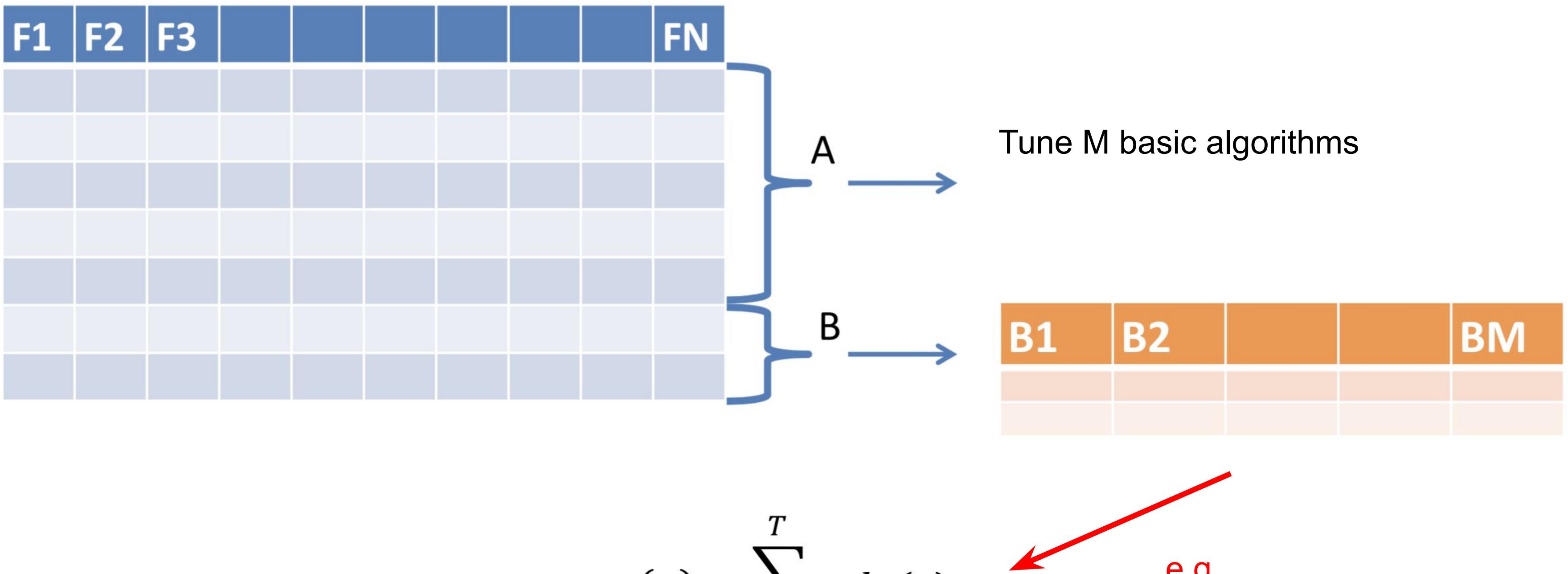

Low Variance
High Variance
Low Bias
High Bias

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



Stacking

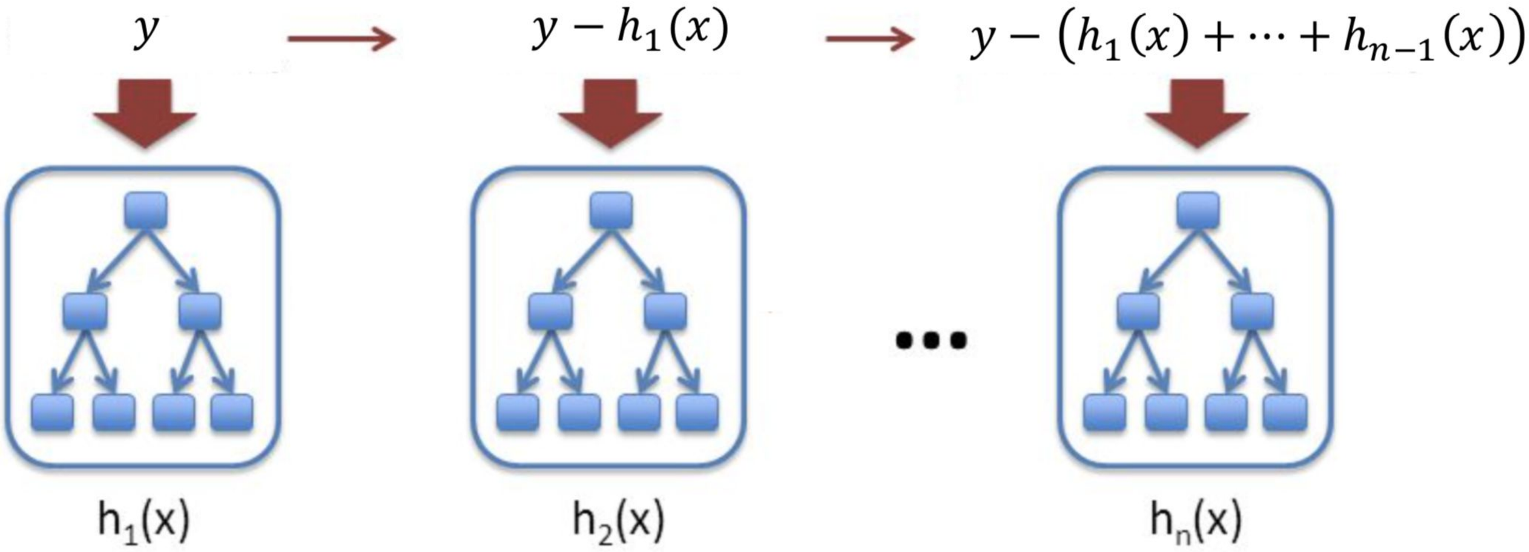
How to build an ensemble from *different* models?



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

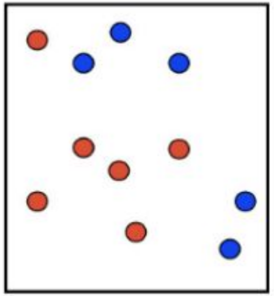
Gradient boosting

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

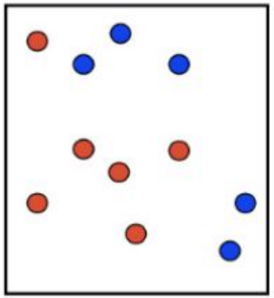


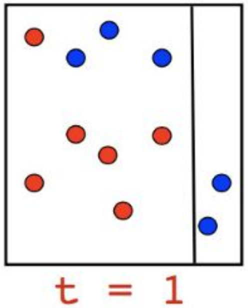
Boosting: intuition

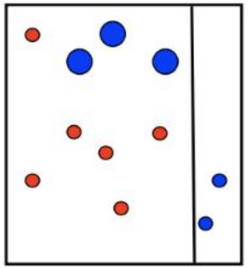
Binary classification problem.
Models - decision stumps.



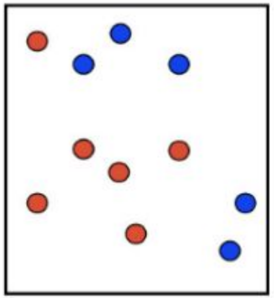
Boosting: intuition

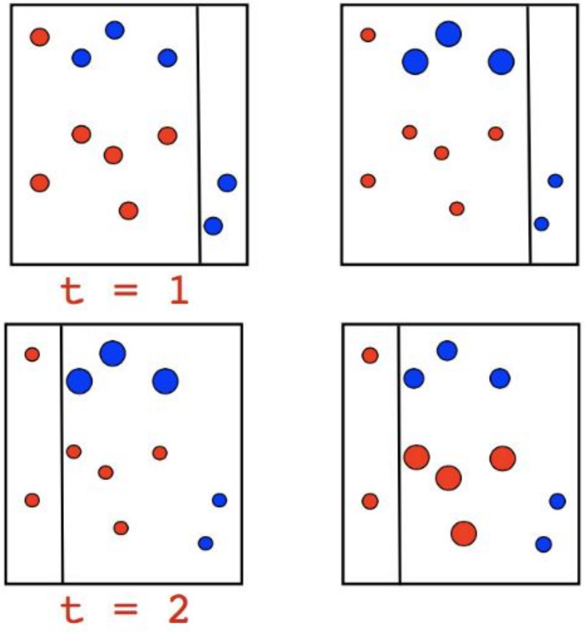




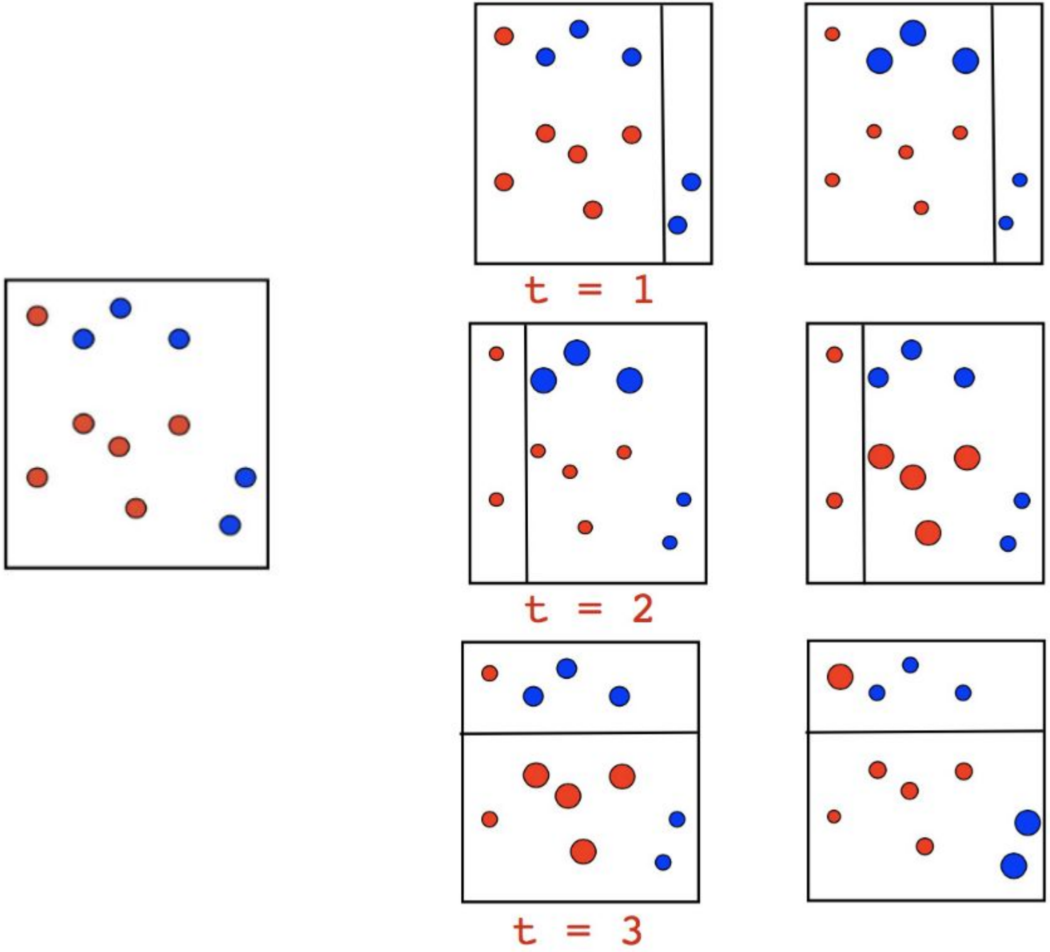


Boosting: intuition



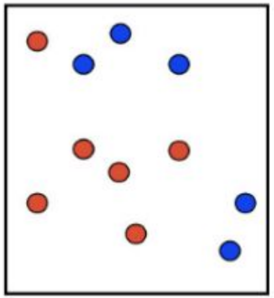


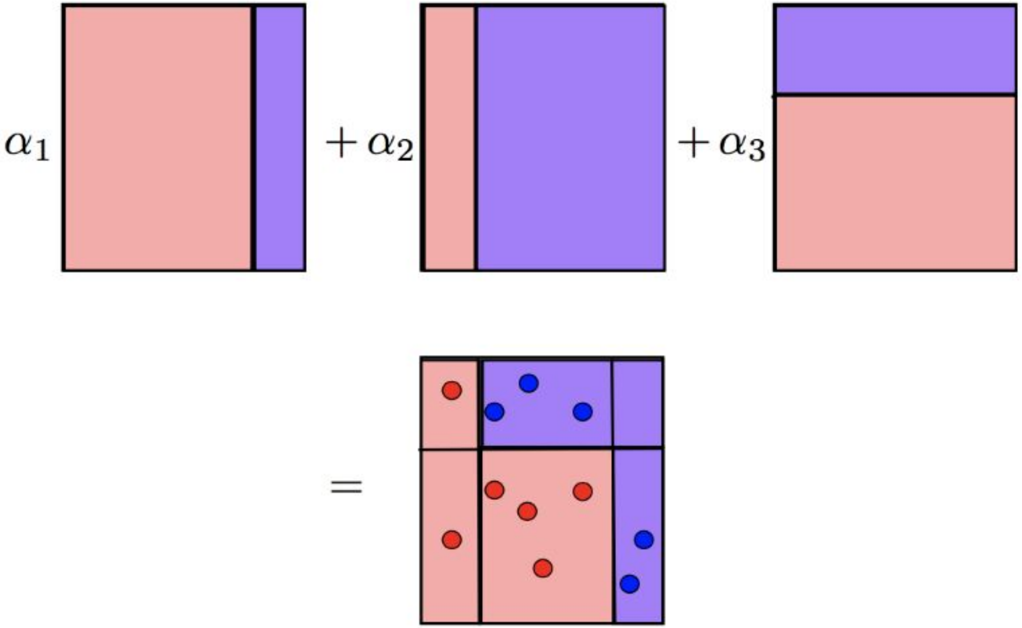
Boosting: intuition



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 $\{(x_i, y_i)\}_{i=1,\dots,n}$, 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

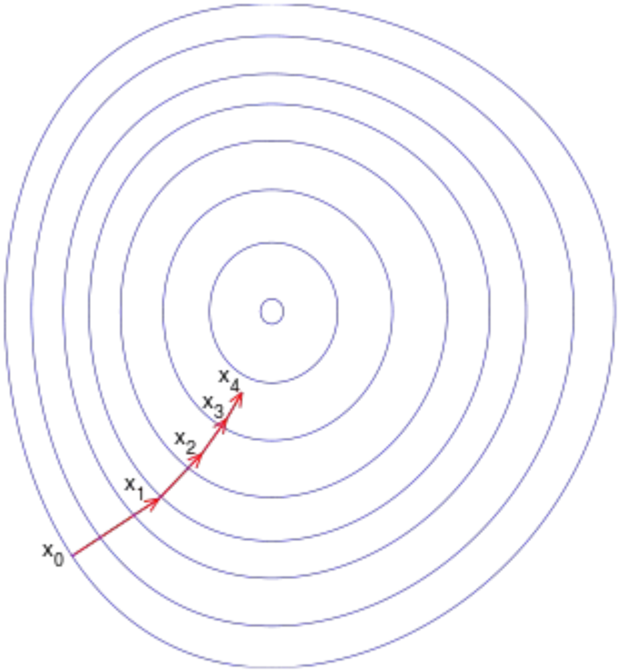
$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x),$$

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

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

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: 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.

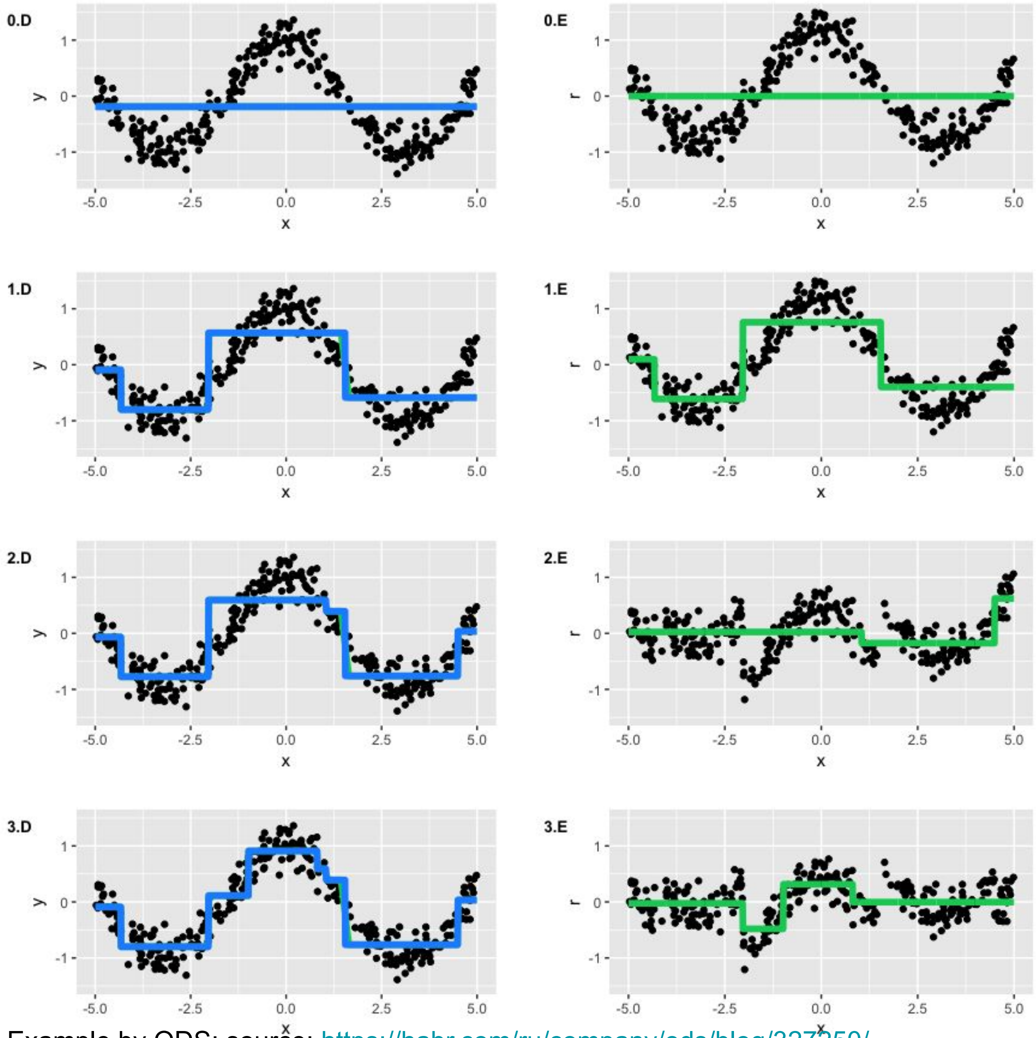
Gradient boosting: example

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/

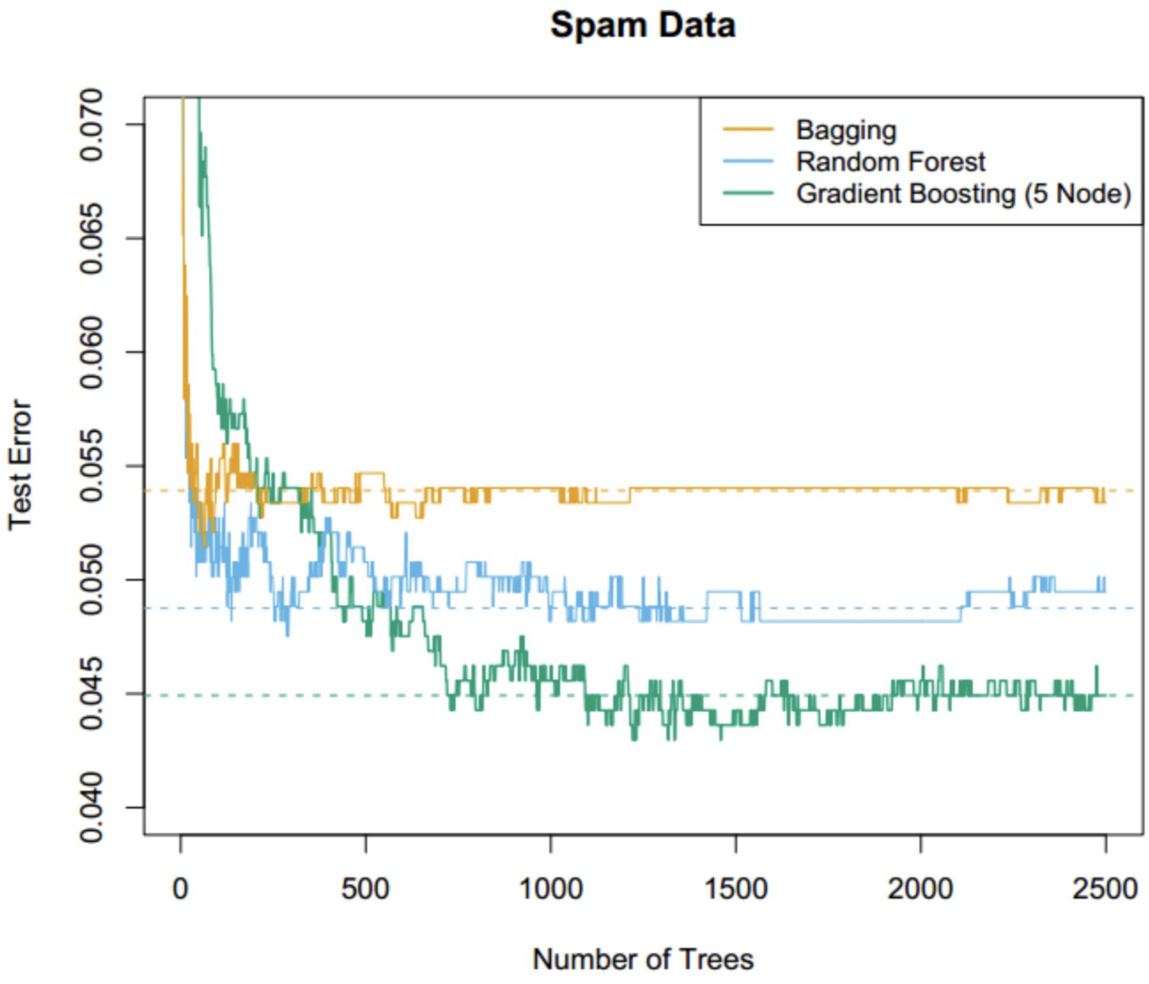
Gradient boosting: example

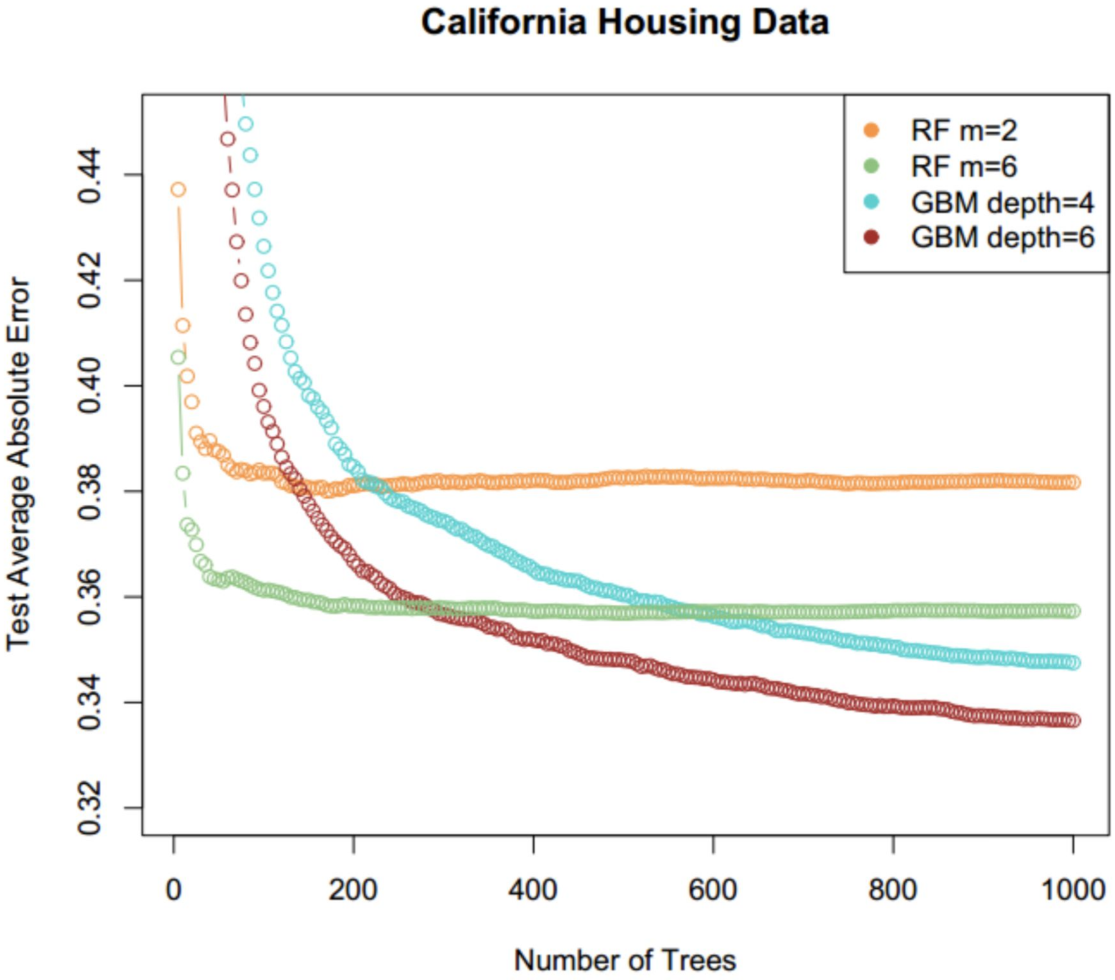


Left: full ensemble on each step.

Right: additional tree decisions.

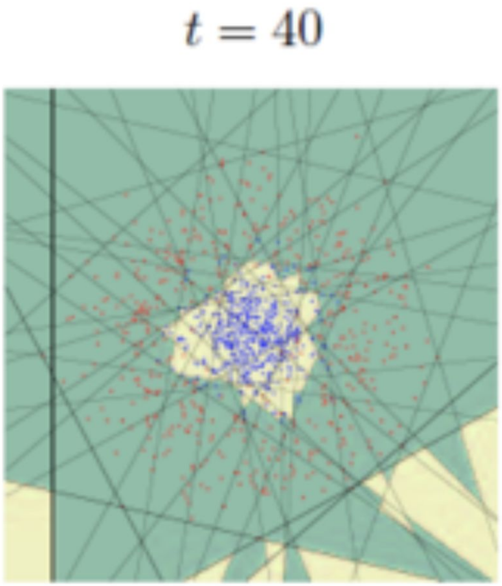
Example by ODS; source: https://habr.com/ru/company/ods/blog/327250/

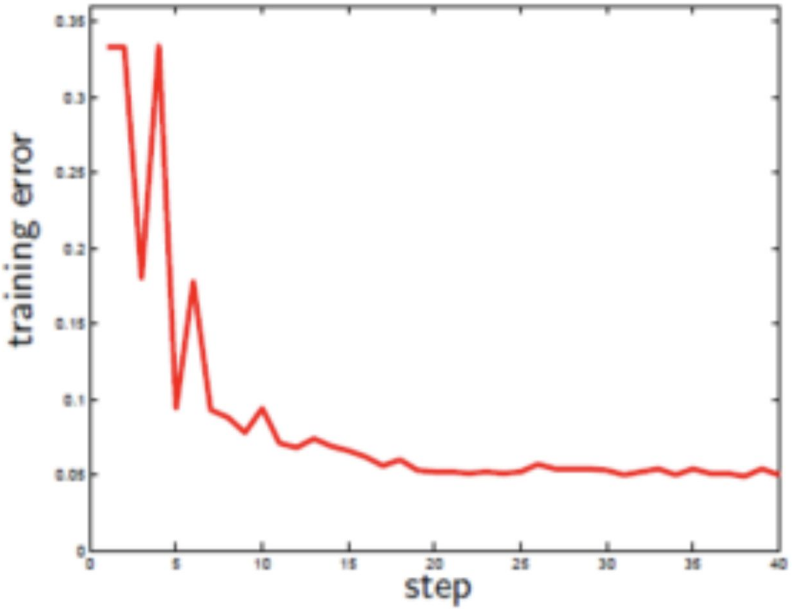

Spam Data
Bagging
Random Forest
Gradient Boosting (5 Node)
Test Error
0.040
0.045
0.050
0.055
0.060
0.065
0.070
0
500
1000
1500
2000
2500
Number of Trees



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

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

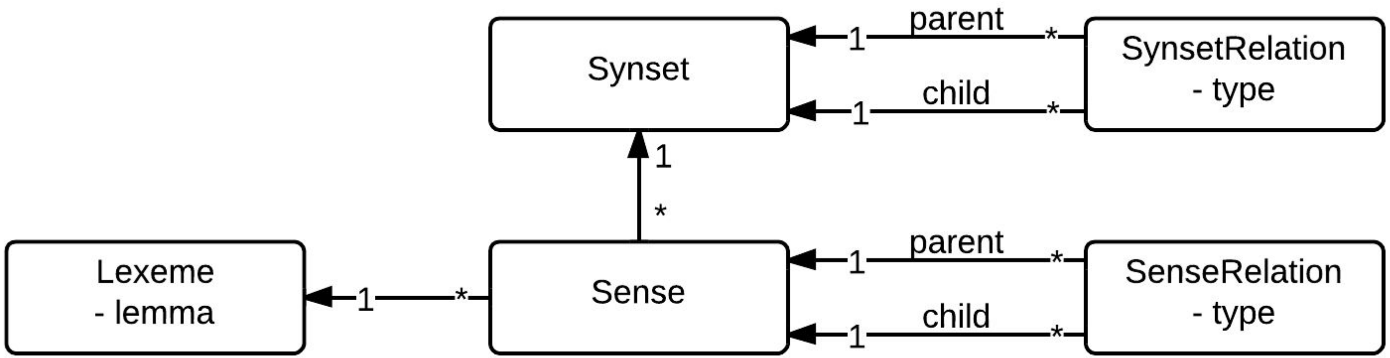
1. Bagging.
2. Random subspace method (RSM).
3. Bagging + RSM + Decision trees = Random Forest.
4. Gradient boosting.
5. Stacking.
6. Blending.

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

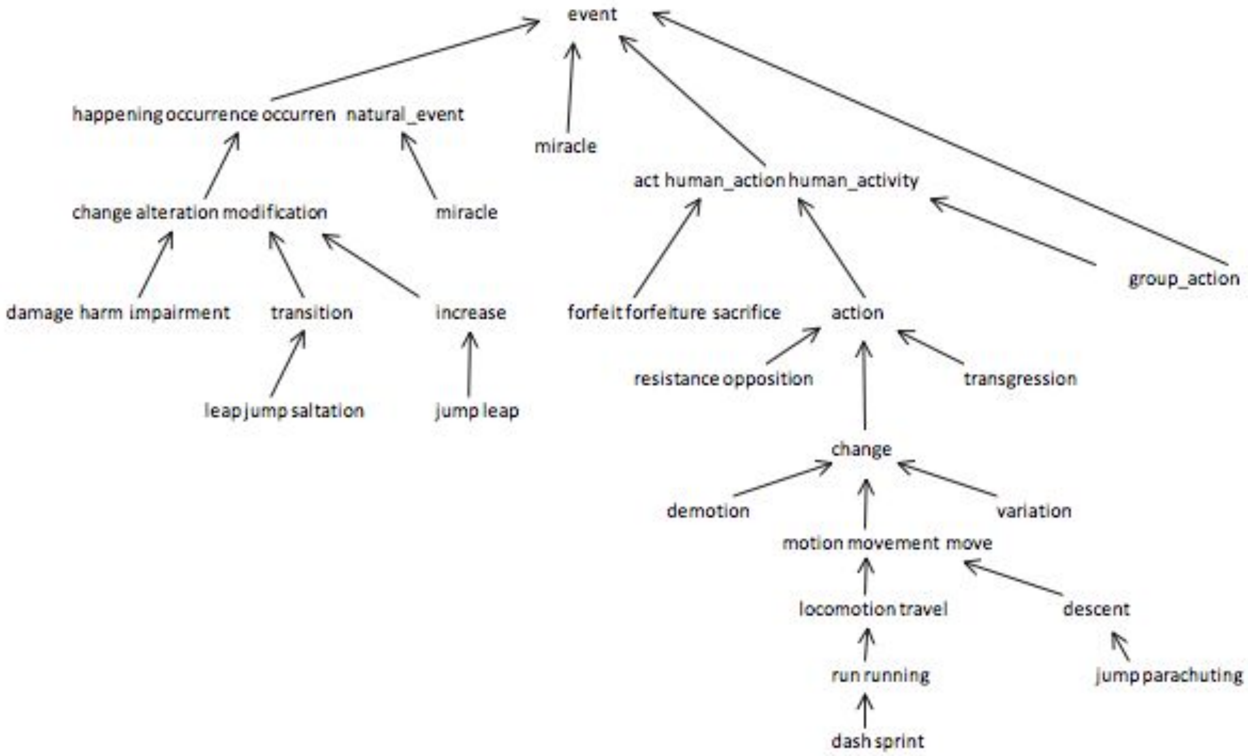
Offtop: words representations

How to represent text in a computer?

Use a taxonomy like WordNet that has hypernyms (is-a) relationships and synonym sets



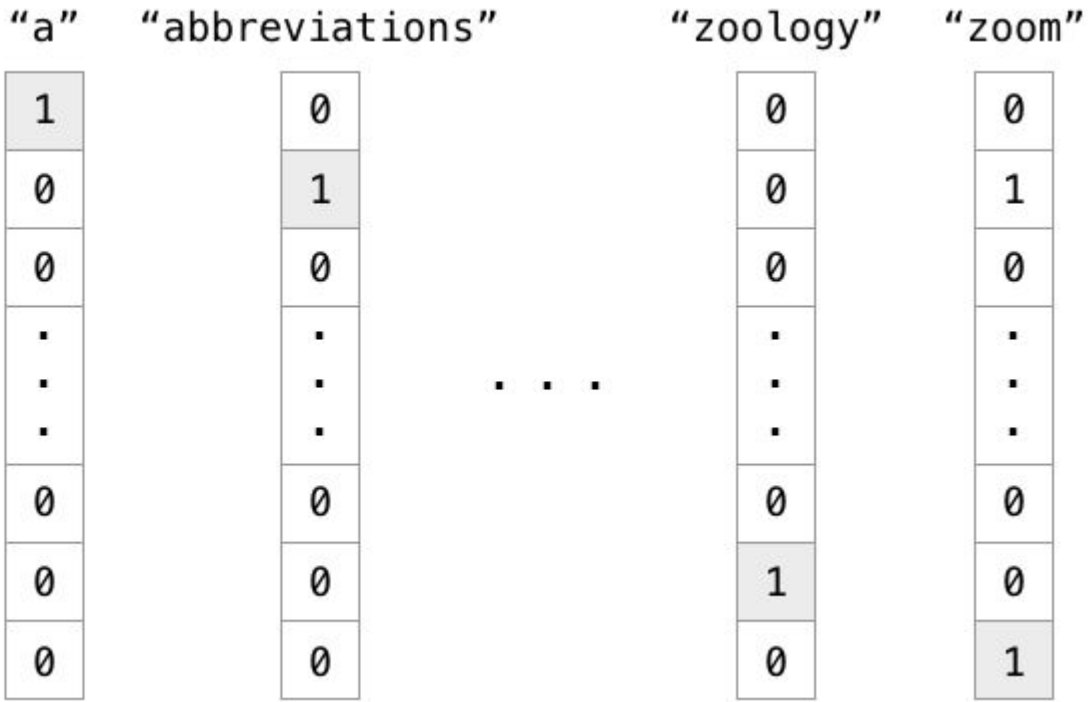
How to represent text in a computer: WordNet



Discrete representations: problems

- Missing new words
- Subjective
- Requires human labor to create and adapt
- Hard to compute accurate word similarity

Discrete representations: one-hot encoding



TF-IDF

If word is repeated in the query, it's probably important

Repetitions of query words in the document ➔ good

Rare words more important

$$s(Q,D) = \sum_{w} tf_{w,Q} \cdot \underbrace{\frac{tf_{w,D}}{tf_{w,D} + \frac{k|D|}{avg|D|}}}_{\text{Repetitions of same word less important than different words. Except in very long documents}} \cdot \log \frac{|C|}{df_w}$$

The more query words we match, the better. Σ over the vocabulary

TF - term frequency

IDF - Inversed Document Frequency

TF-IDF: make it simple

$$\mathrm{tf}(''\mathsf{this}'', d_1) = \frac{1}{5} = 0.2$$

$$\mathrm{tf}(''\mathsf{this}'', d_2) = \frac{1}{7} \approx 0.14$$

$$\mathrm{idf}(''\mathsf{this}'', D) = \log\left(\frac{2}{2}\right) = 0$$

$$\mathrm{tfidf}(''\mathsf{this}'', d_1, D) = 0.2 \times 0 = 0$$

$$\mathrm{tfidf}(''\mathsf{this}'', d_2, D) = 0.14 \times 0 = 0$$

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Word ‘this’ is not very informative

One of the most successful ideas of statistical NLP:

"You shall know a word by the company it keeps"

(J. R. Firth 1957: 11)

Words cooccurrences

Finding N-grams in a text

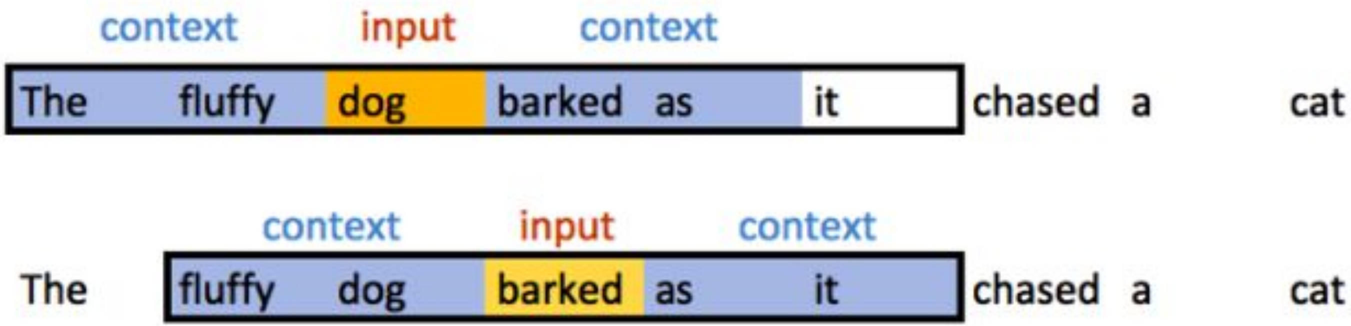
Word-document cooccurrence matrix

Window around each word

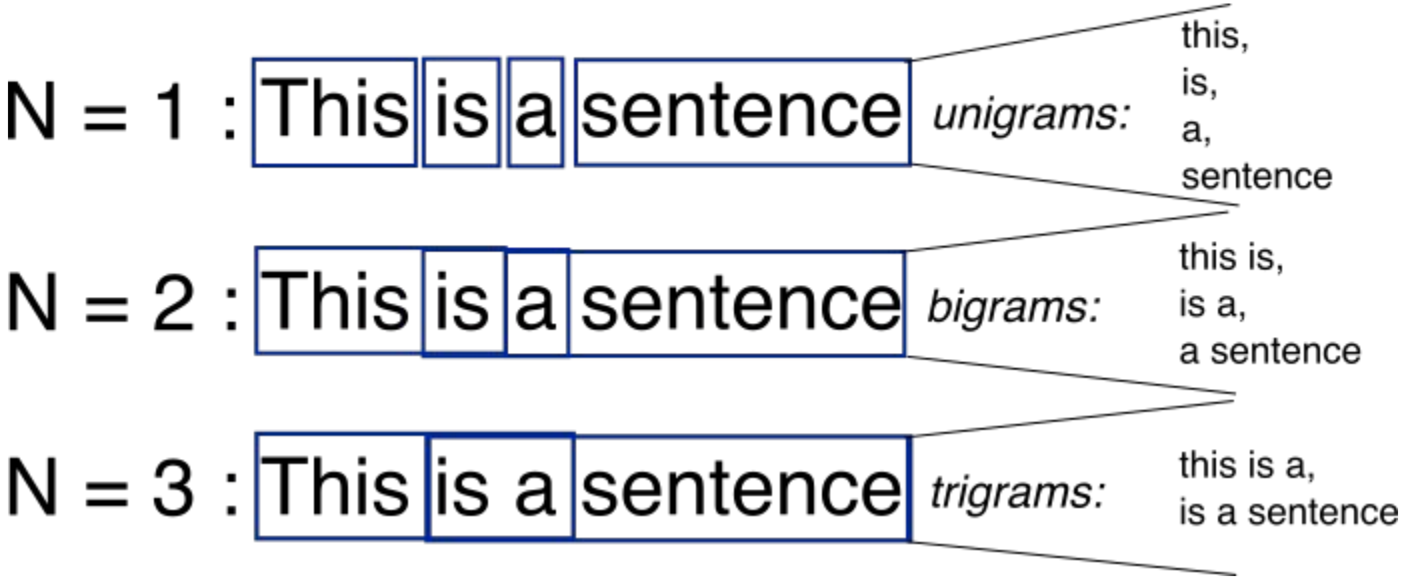
Word-document cooccurrence matrix

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Words cooccurrences: sliding window



Words cooccurrences: n-grams



Cooccurrence vectors: problems

- Increase in size with vocabulary
- Very high dimensional: require a lot of storage
- Subsequent classification models have sparsity issues

↓

Models are less robust

More interesting approaches coming in next classes.
Stay tuned!