Ensemble Learning

Lior Sidi & Noa Lubin

Ensemble Learning?

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained by any of the constituent algorithms.

— Wikipedia (2015)

https://www.slideshare.net/TedXiao/winning-kaggle-101-introduction-to-stacking

Condorcet's Jury Theorem (1785) as Motivation

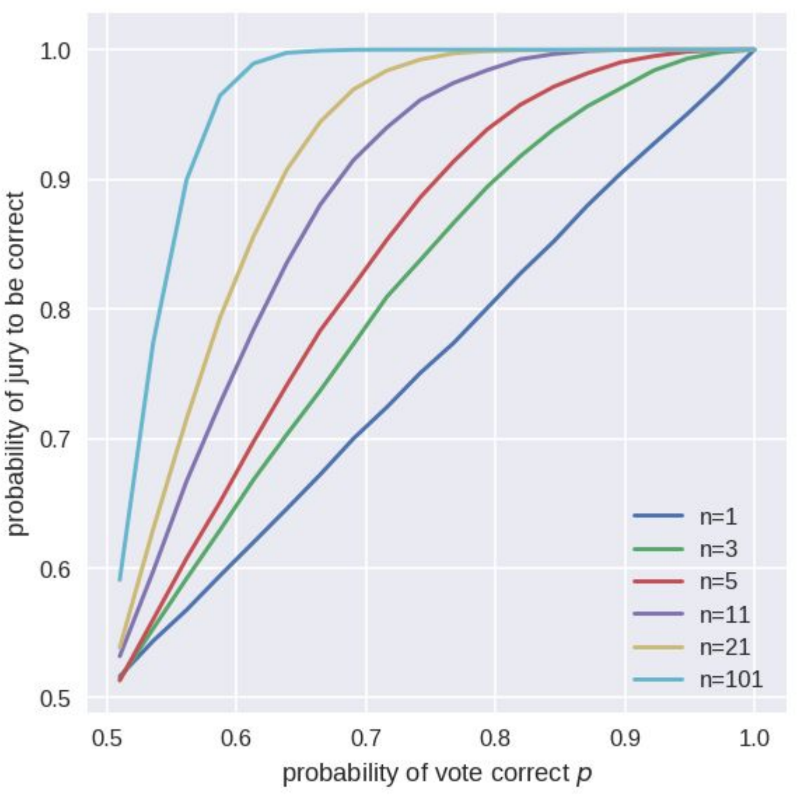
- Given N juries
- Probability of jury to make right decision is $p > 0.5$
- What is the probability q of **majority** decision being correct?

Essay on the Application of Analysis to the Probability of Majority Decisions, Marquis de Condorcet. 1785

Condorcet's Jury Theorem (1785)

- Assume odd independent voters with probability higher than chance probability of being correct.

- The probability jury majority is correct is higher than each voter and is increasing and asymptotically reaches 1.



Y-DATA SCHOOL OF DATA SCIENCE

Yandex

Wisdom of the Crowd

- Francis Galton 1906 livestock fair
- Guess the ox weights (1198 ponds)
- ~800 submitted their guess

- Nobody was correct

- But their average was 1197 pounds!

How many cookies in the Jar?

https://forms.gle/Tj3aar22xTL4jzc38

Back to our models

Robust to Overfitting - Ensemble may not perform like the best train classifier but more likely to be robust and stable for unseen samples

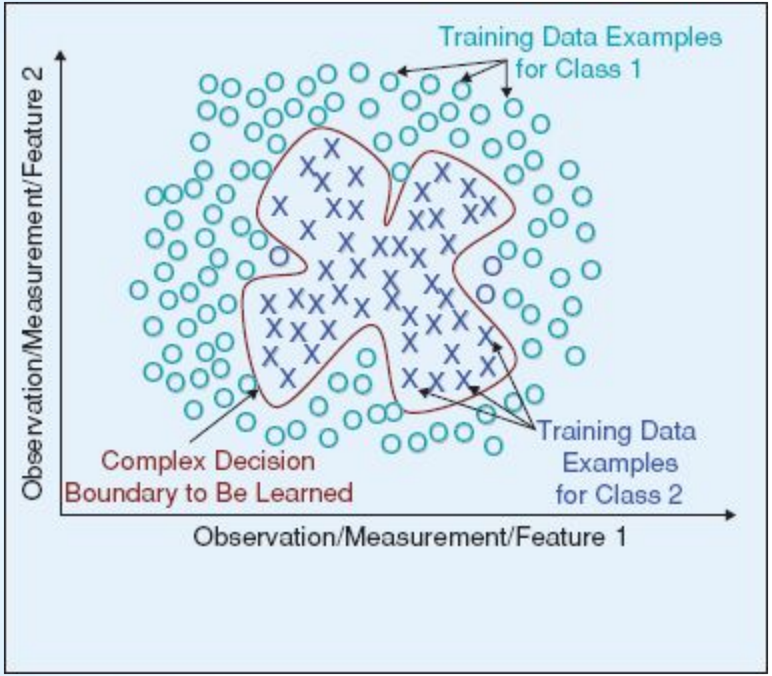
Big data - More suitable for ensembles that can divide the computation

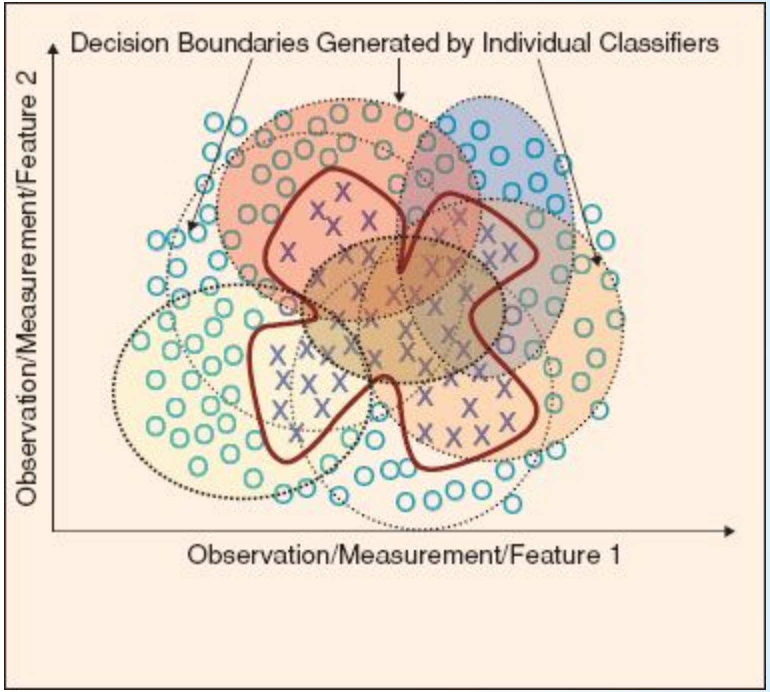
Small data - The sampling technique can uncover and understand better data distribution

Y-DATA
SCHOOL OF DATA SCIENCE

Yandex

Combining several estimators





Key elements in ensemble learning

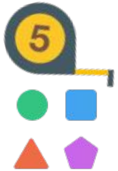
Diversity

- Each person should have private information and interpretations.
- Each estimator should base its prediction on separate data or/and algorithm

Diversity of opinion

- Manipulate the estimator
- Manipulate the training data
- Manipulate the label representation
- Partition the search space - Each estimator is trained on a different search subspace.

Measuring the Diversity



- Pairwise agreement measures between estimators predictions such as kappa-statistic.

- Non-pairwise agreement measures using all estimators predictions such as entropy or correlation of each estimator with the averaged output.

Y-DATA SCHOOL OF DATA SCIENCE

Yandex

Cohen's Kappa Statistics

$$K = \frac{P_{agree} - P_{chance}}{1 - P_{chance}}$$

P_agree - proportion of instances agreed by the classifiers

P_chance - proportion of instances that agreed by chance

Classifier 1

Classifier 2	**no**	**yes**
no	33	20
yes	13	40

P_agree = 73 / 106 = 0.68

P_chance = P(class=yes|classifier=1)*P(yes|2) + P(no|1)*P(no|2)

= (60/106)*(53/106) + (46/106)*(53/106) = 0.5

K = (0.68 - 0.5) / (1- 0.5) = 0.36

Y-DATA
SCHOOL OF DATA SCIENCE

Yandex

K=0.36

Cohen's Kappa	Interpretation
0	No agreement
0.10 - 0.20	Slight agreement
0.21 - 0.40	Fair agreement
0.41 - 0.60	Moderate agreement
0.61 - 0.80	Substantial agreement
0.81 - 0.99	Near perfect agreement
1	Perfect agreement

Key elements in ensemble learning

Diversity

- Each person should have private information and interpretations.
- Each estimator should base its prediction on separate data or/and algorithm

Independence

- People's opinions aren't determined by the opinions of those around them.
- Estimator’s predictions aren't determined by the predictions of other estimators.

Dependency

- Independent Methods
  - Each estimator is train separately
  - Seperate data, features, labels.
- Dependent Methods:
  - **Model-guided Instance Selection**: the estimators from the previous training iterations selects the training set for the next iteration.
  - **Incremental Batch Learning**: the estimators predictions serve as a feature for the next training iteration.

Y-DATA SCHOOL OF DATA SCIENCE

Yandex

Key elements in ensemble learning

Diversity

- Each person should have private information and interpretations.
- Each estimator should base its prediction on separate data or/and algorithm

Independence

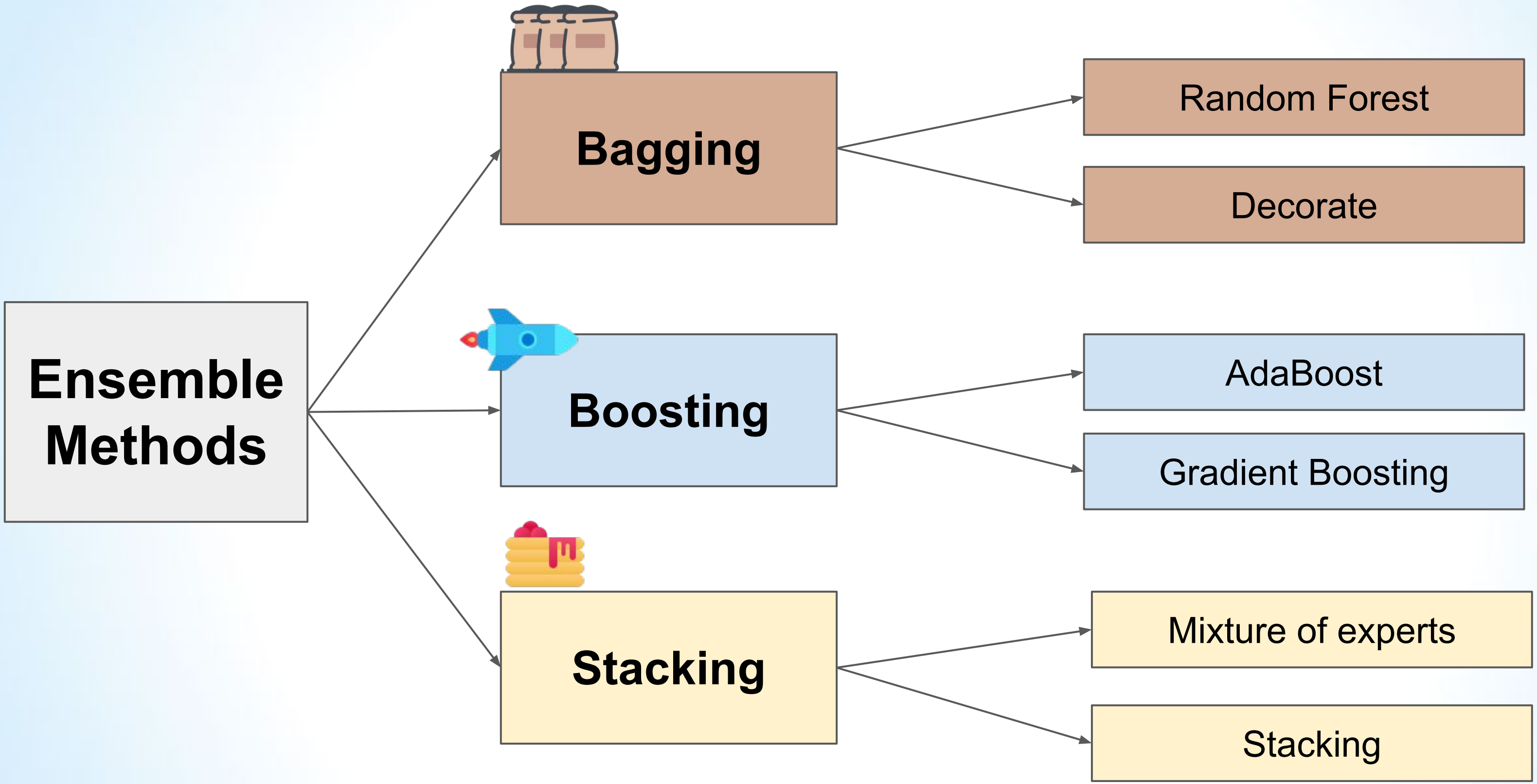
- People's opinions aren't determined by the opinions of those around them.
- Estimator’s predictions aren't determined by the predictions of other estimators.

Aggregation

- Some mechanism exists for turning private judgments into a collective decision.
- Some mechanism exists for turning private prediction into a ensembled prediction.

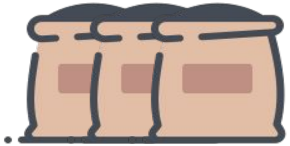
Aggregation - Output combination

- Majority Voting
- Performance weighting
- Learn how to aggregate it


Ensemble Methods
Bagging
Boosting
Stacking
Random Forest
Decorate
AdaBoost
Gradient Boosting
Mixture of experts
Stacking

Y-DATA SCHOOL OF DATA SCIENCE

Yandex



Bagging

Bootstrap **Agg**regat**ing**

Board

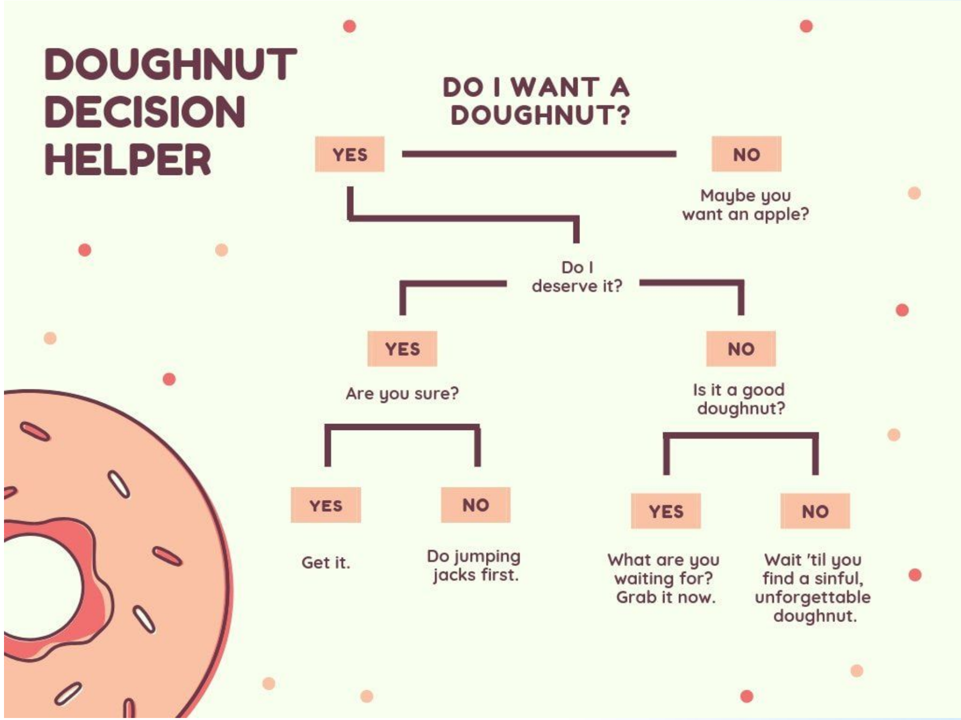
Let’s go back to Decision Trees?

How do they work?

What it the core mechanism?

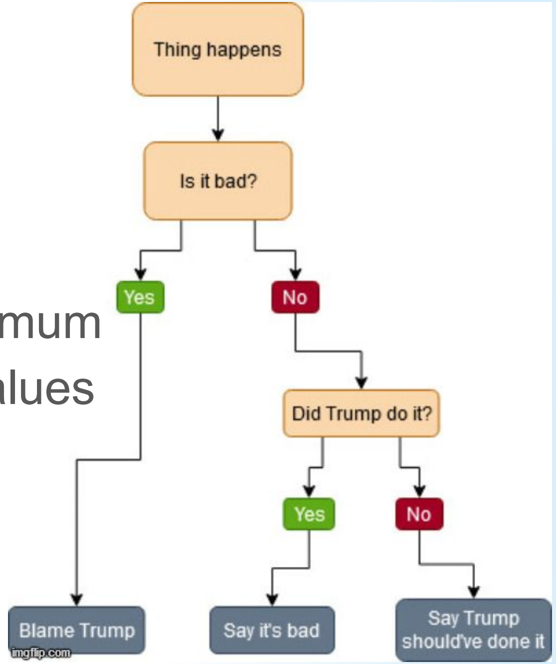
Why Decision Trees?

- Easy to understand
- Non-linear
- Fast train and predict
- Apply feature selection
- Robust to skewed features



Why Not Decision Trees?

- Decision boundaries are rectilinear
- Problem handling unbalanced classes
- Problem handling missing/new data
- Uses a greedy approach and can get stuck in local minimum
- Time consuming splitting on features with continuous values



Leads to

Overfitting


THE BEST WAY TO
EXPLAIN OVERFITTING


A GOOD EXAMPLE OF
OVERFITTING

What is overfitting?

"The production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit to additional data or predict future observations reliably"

OxfordDictionaries.com: this definition is specifically for statistics.

Y-DATA SCHOOL OF DATA SCIENCE

Yandex

What is overfitting?

$$\text{Loss}(h(x), y) = \text{Variance}(\ h(x)\ ) + \text{Bias}(\ h(x), y\ )$$

The model predictions shouldn't change across datasets from same distribution

The error of the model should be low

If model predictions change across datasets it's performance will decrease

Variance and Bias in the loss

E[h(x)] = h’(x) = the expected decision of our model (regardless the dataset)

Loss(h(x) , y) = Variance + Bias

- Variance(h(x)) = E[loss(h(x), h’(x))]
- Bias(h(x),y) = E[loss(h’(x), y)]

The **expected value** of something we are uncertain about is average outcome, weighing each possibility according to its likelihood. It is, in some sense, what we should “expect on average” from an uncertain risk we’re about to take.

Proof:

MSE(h(x) , y) = E[(h(x) - y)^2]

Variance and Bias in the loss

$E[h(x)] = h'(x)$ = the expected decision of our model (regardless the dataset)

$Loss(h(x), y) = Variance + Bias$

$Variance(h(x)) = E[\, loss(h(x), h'(x)) \,]$

$Bias(h(x), y) = E[\, loss(h'(x), y) \,]$

Proof:

$MSE(h(x), y) = E[(h(x) - y)^2]$

$= E[\, ((h(x) - h'(x)) + (h'(x) - y))^2 \,]$, **h’(x) is constant**

$= E[\, (h(x) - h'(x))^2 + 2(h(x) - h'(x))(h'(x) - y) + (h'(x) - y)^2 \,]$, **simple algebra**

$= E[(h(x) - h'(x))^2] + 2E[(h(x) - h'(x)]E[(h'(x) - y)] + E[(h'(x) - y)^2]$, **simple algebra**

$= \underbrace{E[(h(x) - h'(x))^2]}_{\text{Variance}} + \underbrace{E[(h'(x) - y)^2]}_{\text{Bias^2}}$, **because E[(h(x) - h’(x)] is 0**

The **expected value** of something we are uncertain about is average outcome, weighing each possibility according to its likelihood. It is, in some sense, what we should “expect on average” from an uncertain risk we’re about to take.

Y-DATA SCHOOL OF DATA SCIENCE

Yandex

How can we reduce the Variance?

Assume we gave D1…Dm, independent datasets from D

h(x) = fit a model per D_i

$$\frac{\sum_{i=1}^{m} h_i(x)}{m} \xrightarrow[\text{Weak law of large numbers}]{m \to \infty} h(x) = E[h(x)] = h'(x)$$

E[(h(x) - h'(x))^2] Variance will be 0!

We prove that training many models on independent datasets reduce the Variance!

*What's the catch? Why not everyone use it?

Bootstrapping

D

D1 Di Dm

h1 hi hm

$$\frac{\sum_{i=1}^{m} h_i(x)}{m}$$

1. Sample datasets with replacements

*Why?

2. Fit a model independently

*What we are getting?

3. Make a decision

*How can we make D more independent?

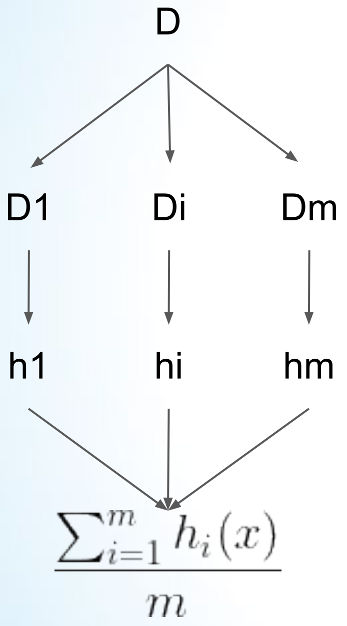
Bagging - main concepts

Dependency - Same type of estimators.
Aggregate - Voting or averaging with equal weight
Diversify - subset of dataset based on features, sampling and/or labels

Model Flow:

a. Sample training sets of size n
b. Fit an estimator for each training set
c. Combine the classifier’s predictions

Random Forest - a bagging variant



1. Sample datasets with replacements

-> From K random dimensions: K << d, log(d)

2. Fit a model independently

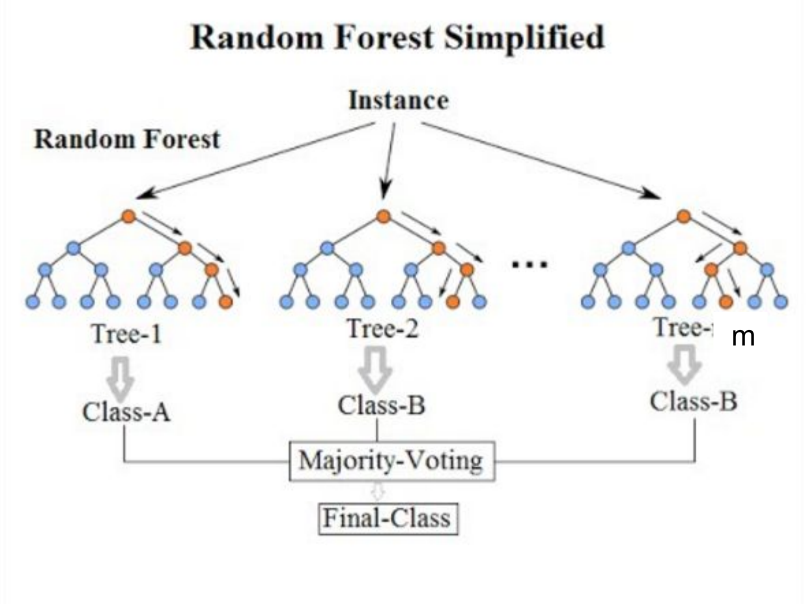
-> Decision Trees (common: unlimited depth)

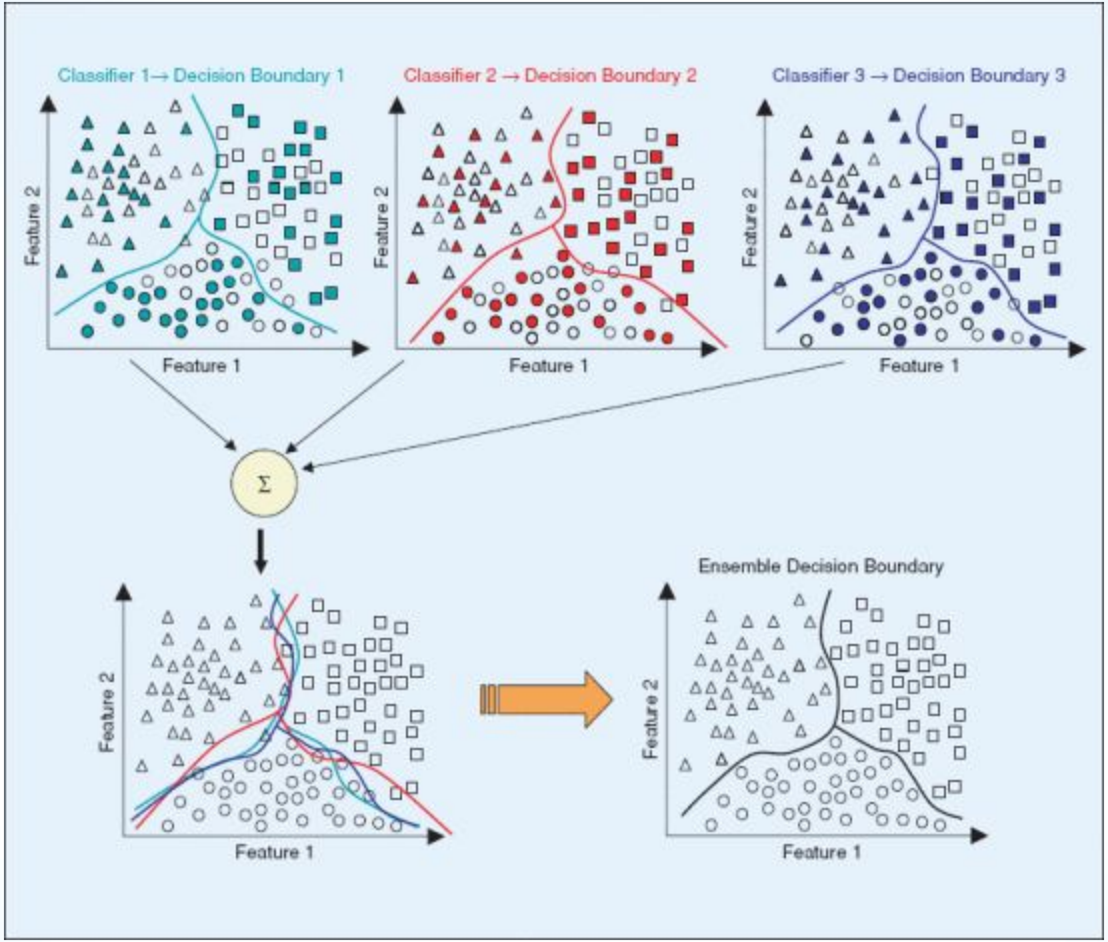
3. Make a decision

-> Majority voting

Model Flow:

a. Sample training sets of size n
b. Fit an estimator for each training set
c. Combine the classifier's predictions

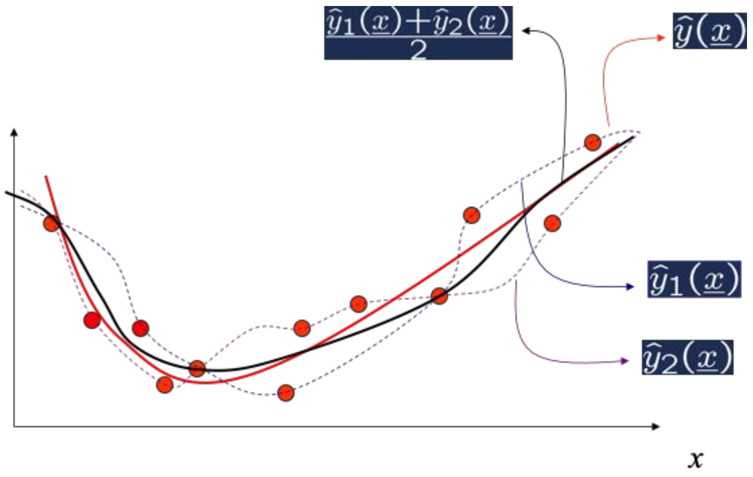



Classifier 1→ Decision Boundary 1
Classifier 2 → Decision Boundary 2
Classifier 3 → Decision Boundary 3
Feature 2
Feature 1
Σ
Ensemble Decision Boundary

Y-DATA SCHOOL OF DATA SCIENCE

Yandex

Bagging behavior simulation



Random Forest - Bootstrap as validation

Out of bag loss:

Evaluate the trained estimator on the samples it wasn't trained on

Aggregate their results into a validation score

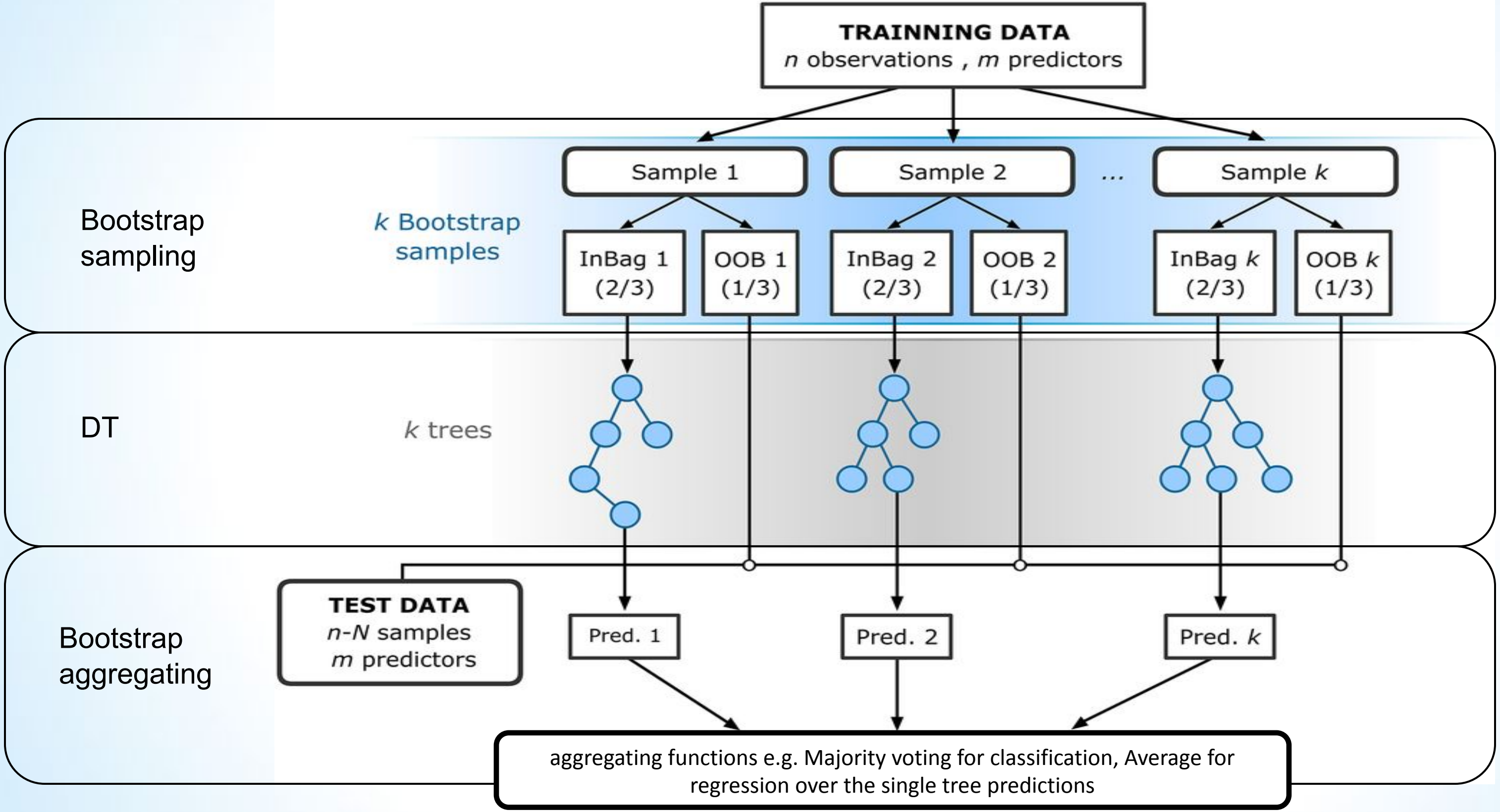
$$out\ of\ bag\ loss = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{z_i}\sum_{j, x_i not\ in\ D_j}^{m} loss(h_j(x_i, y_i))$$

Zi is the number of classifiers not trained on Xi

n is number of samples in dataset D

*How you can use it during training?

Random Forest Trees Bagging



https://www.analyticsvidhya.com/blog/2020/12/out-of-bag-oob-score-in-the-random-forest-algorithm/

Bagging Variants

Pasting

- When random subsets of the dataset are drawn as random subsets of the samples
  - L. Breiman, Pasting small votes for classification in large databases and on-line, 1999

Bagging

- When random samples of the dataset are drawn with replacement
  - L. Breiman, Bagging predictors, 1996

Random Subspaces

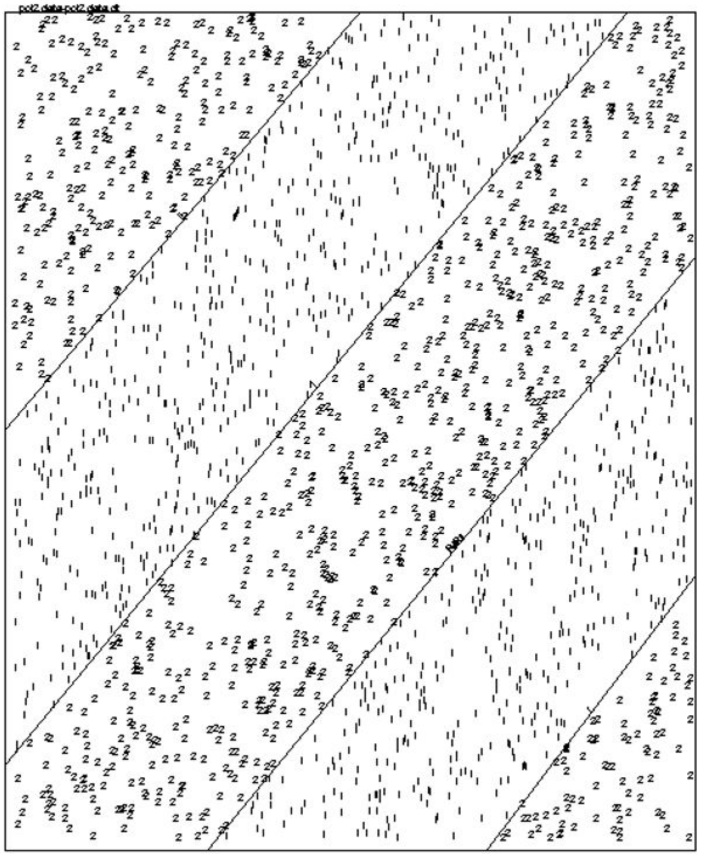
- When random subsets of the dataset are drawn as random subsets of attributes
  - T. Ho, The random subspace method for constructing decision forests, 1998

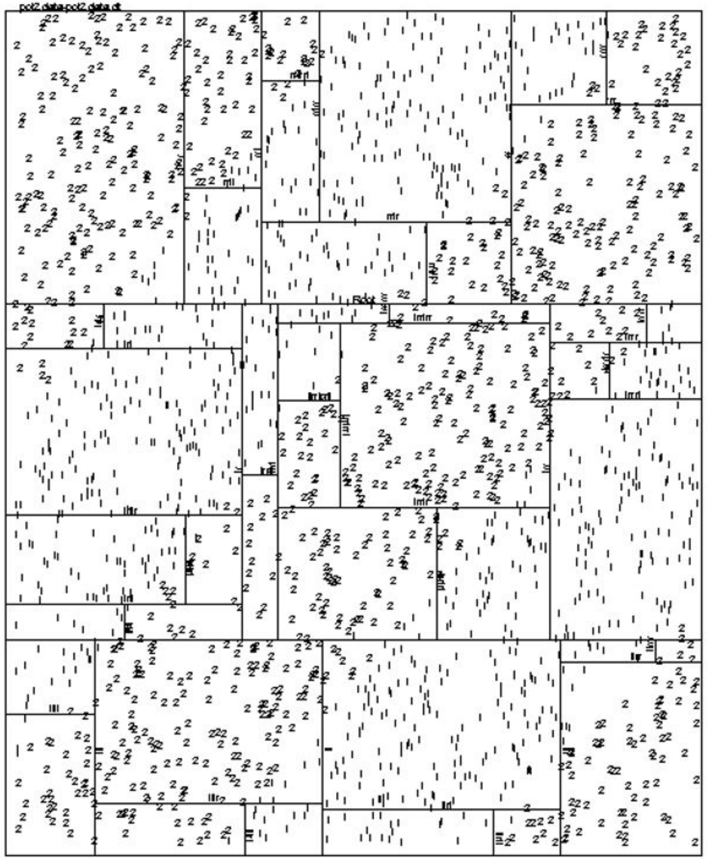
Random Patches

- When base estimators are built on subsets of both samples and attributes
  - G. Louppe and P. Geurts, Ensemble on random patches, 2012

Random Forests

- A hybrid of Bagging and Random Subspaces, uses Decision Trees as the base classifier with random splits
  - L. Breiman, Random Forests, 2001


pol2.data-pol2.data.dt


pol2.data-pol2.data.dt
Root

How to handle Rectilinearity?

Train a linear model in the nodes: M5, GUIDE, FT

Apply transformation on the data -> Rotation Forest

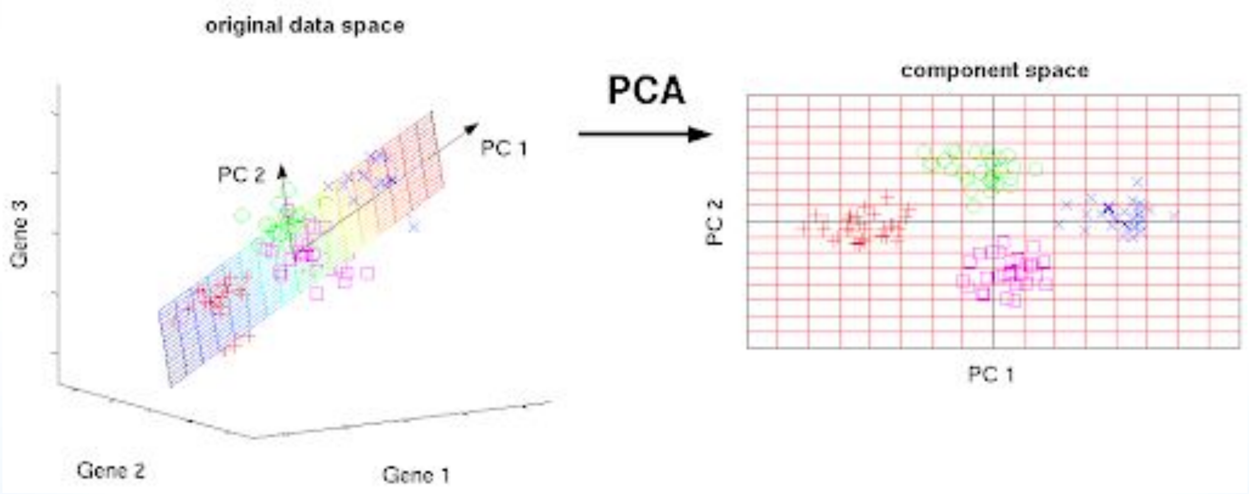
Rotation Forest

- Rotation forest transforms the data set while preserving all information
- PCA is used to transform the data
  - subset of the instances
  - subset of the classes
  - subset of the features: low computation, low storage

Rodriguez, Juan José, Ludmila I. Kuncheva, and Carlos J. Alonso. "Rotation forest: A new classifier ensemble method." IEEE transactions on pattern analysis and machine intelligence 28.10 (2006): 1619-1630.

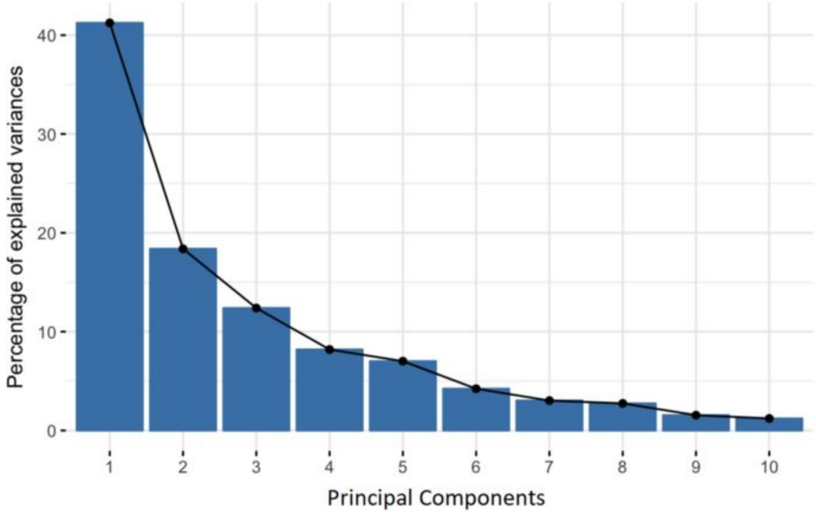
Principal Components Analysis

- PCA projects the data along the directions where the data varies the most.

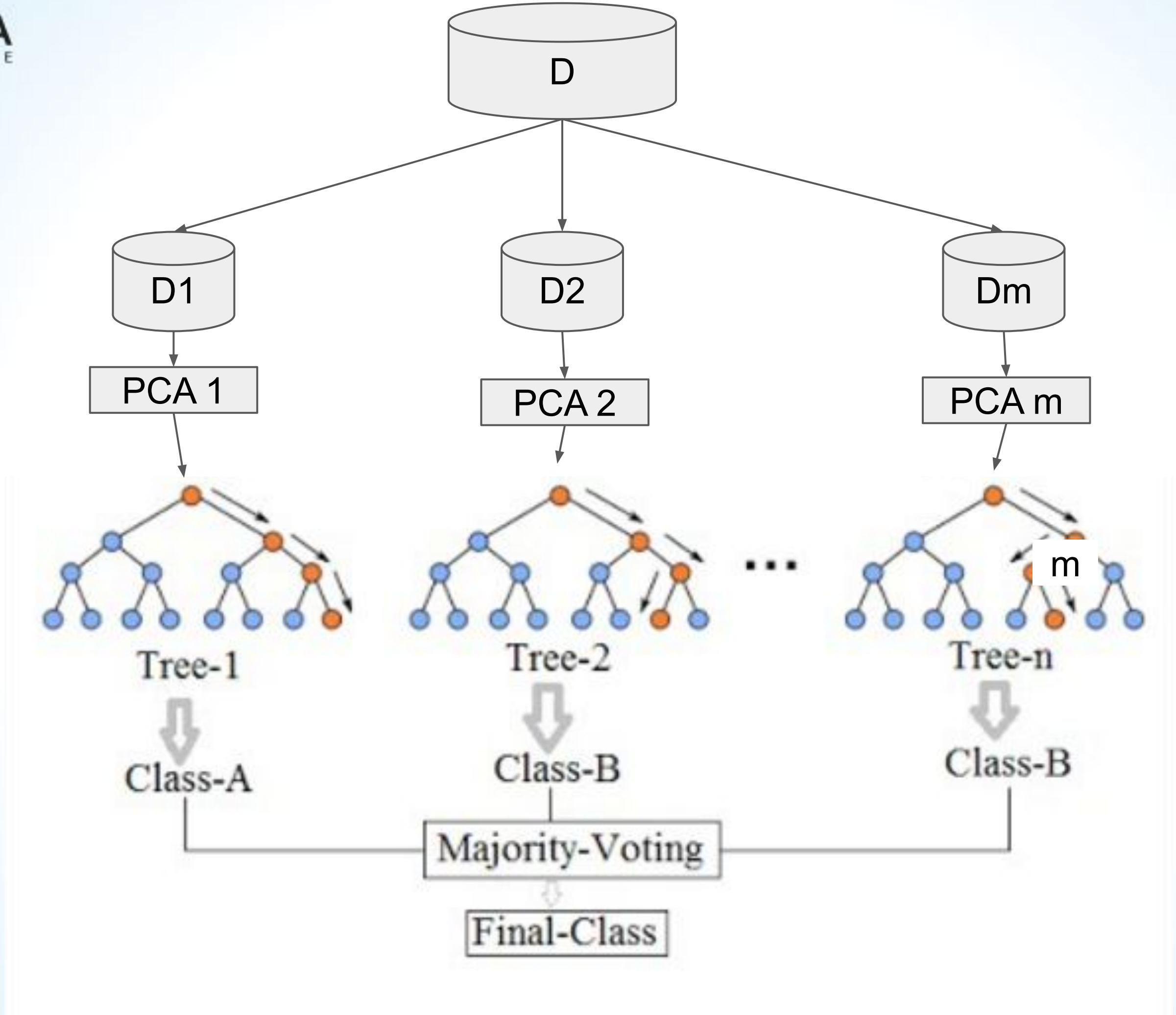


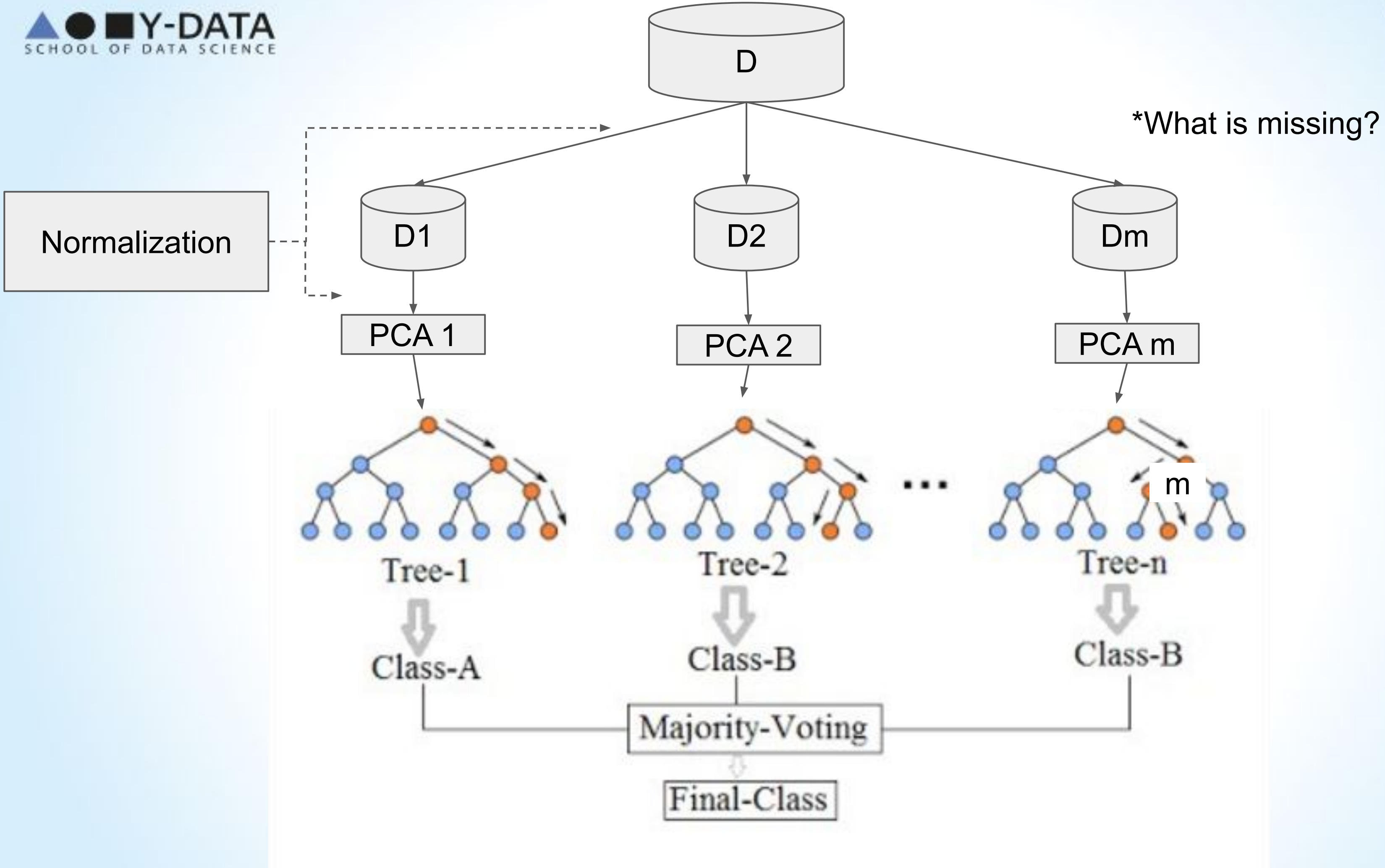
Principal Components Analysis

Principal components are constructed as linear combinations which are uncorrelated and most of the information is compressed into the first components.



https://builtin.com/data-science/step-step-explanation-principal-component-analysis


D
D1
D2
Dm
PCA 1
PCA 2
PCA m
Tree-1
Tree-2
...
m
Tree-n
Class-A
Class-B
Class-B
Majority-Voting
Final-Class



D
*What is missing?
Normalization
D1
D2
Dm
PCA 1
PCA 2
PCA m
...
m
Tree-1
Tree-2
Tree-n
Class-A
Class-B
Class-B
Majority-Voting
Final-Class

Training Phase

Given

- X: the objects in the training data set (an $N \times n$ matrix)
- Y: the labels of the training set (an $N \times 1$ matrix)
- L: the number of classifiers in the ensemble
- K: the number of subsets
- $\{\omega_1, \ldots, \omega_c\}$: the set of class labels

For $i = 1 \ldots L$

- Prepare the rotation matrix R_i^a:
  - Split $\mathbf{F}$ (the feature set) into K subsets: $\mathbf{F}_{i,j}$ (for $j = 1 \ldots K$)
  - For $j = 1 \ldots K$
    - Let $X_{i,j}$ be the data set X for the features in $\mathbf{F}_{i,j}$
    - Eliminate from $X_{i,j}$ a random subset of classes
    - Select a bootstrap sample from $X_{i,j}$ of size 75% of the number of objects in $X_{i,j}$. Denote the new set by $X'_{i,j}$
    - Apply PCA on $X'_{i,j}$ to obtain the coefficients in a matrix $C_{i,j}$
  - Arrange the $C_{i,j}$, for $j = 1 \ldots K$ in a rotation matrix R_i as in equation (1)
  - Construct R_i^a by rearranging the the columns of R_i so as to match the order of features in $\mathbf{F}$.
- Build classifier D_i using (XR_i^a, Y) as the training set

Classification Phase

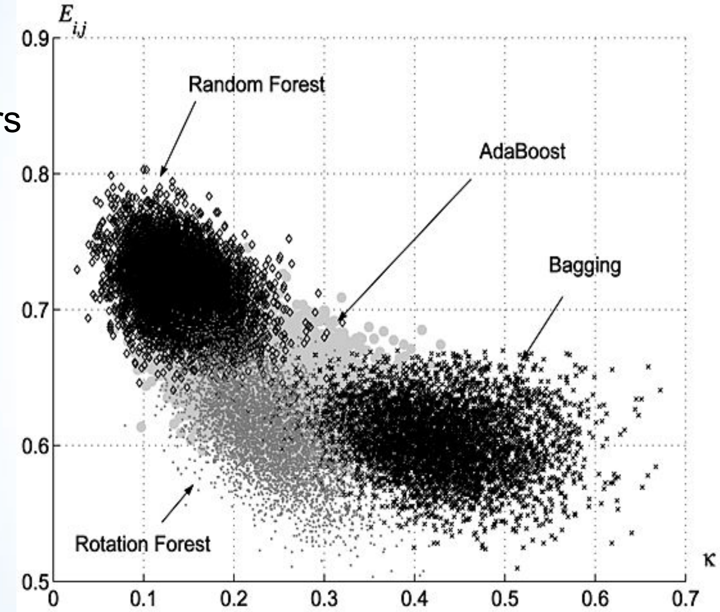
- For a given $\mathbf{x}$, let $d_{i,j}(\mathbf{x}R_i^a)$ be the probability assigned by the classifier D_i to the hypothesis that $\mathbf{x}$ comes from class ω_j. Calculate the confidence for each class, ω_j, by the average combination method:

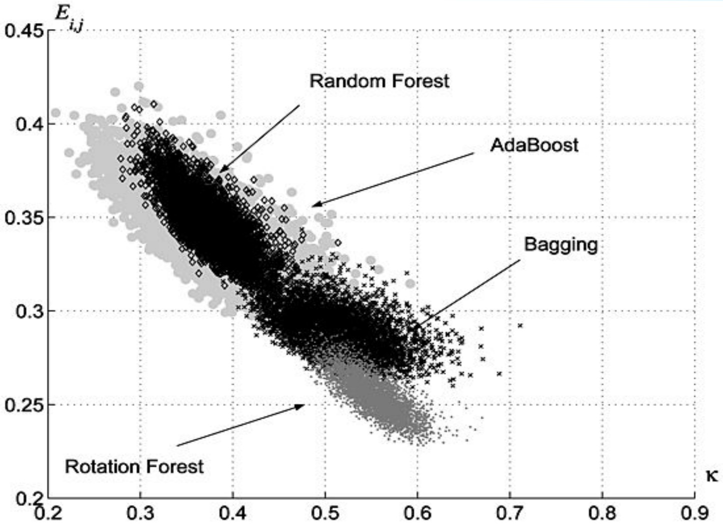
$$\mu_j(\mathbf{x}) = \frac{1}{L}\sum_{i=1}^{L} d_{i,j}(\mathbf{x}R_i^a), \quad j = 1, \ldots, c.$$

- Assign $\mathbf{x}$ to the class with the largest confidence.

Comparing between ensembles

Average
Estimators
Error



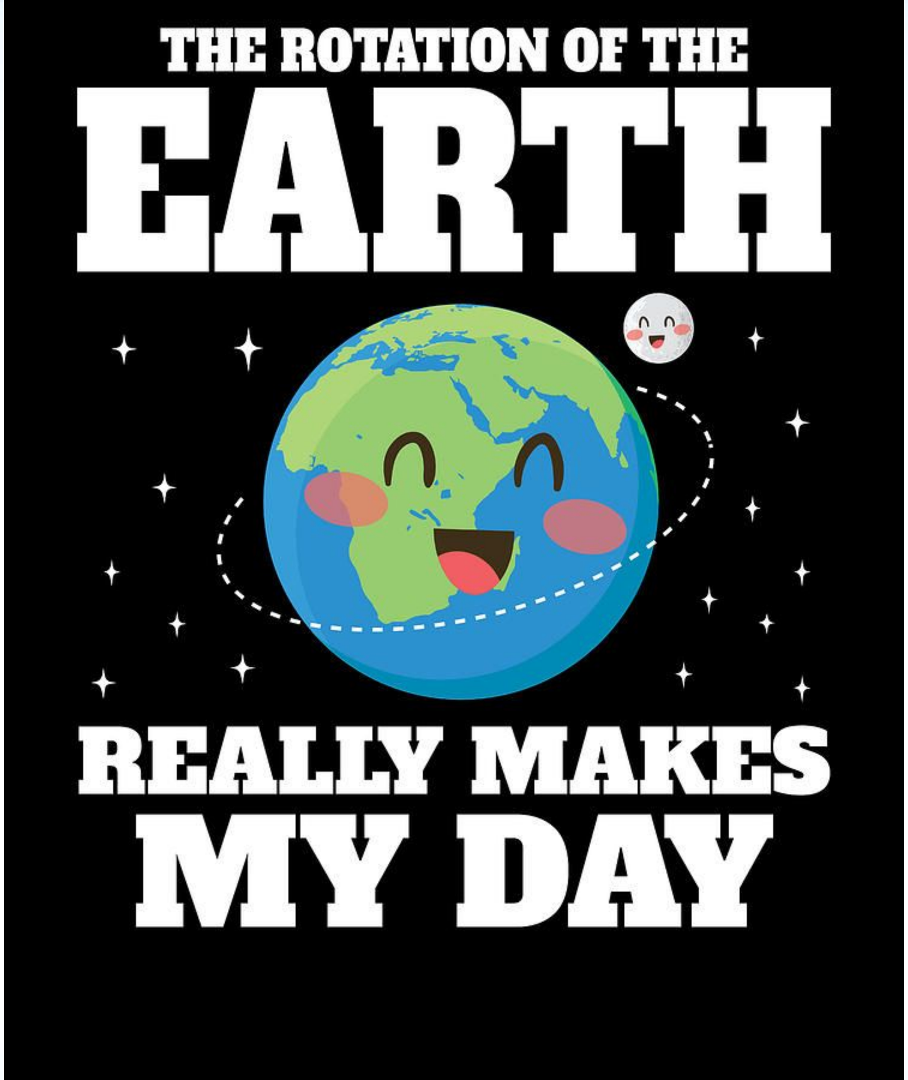


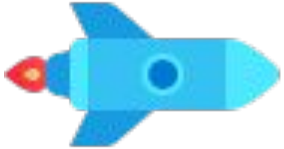
Estimators Agreement Kappa (k) = Diversification

Rodriguez, Juan José, Ludmila I. Kuncheva, and Carlos J. Alonso. "Rotation forest: A new classifier ensemble method." IEEE transactions on pattern analysis and machine intelligence 28.10 (2006): 1619-1630.

Y-DATA
SCHOOL OF DATA SCIENCE

Yandex





Boosting

What are **weak learner** and **strong learner**?

- **Weak learner**
  a classifier that is only slightly correlated with the true classification (it can label examples better than random guessing)

- **Strong learner**
  a classifier that is arbitrarily well-correlated with the true classification.
  Hard to train

How Bias & Variance are effected?

Bias

The tendency to consistently learn the same wrong thing because the hypothesis space considered by the learning algorithm does not include sufficient hypotheses

Variance

The tendency to learn random things irrespective of the real signal due to the particular training set used

How Bias & Variance are effected?

Estimator with many parameters (Strong)

- Generally low bias
- Fits data well
- Yields high variance

Estimator with few parameters (Weak)

How Bias & Variance are effected?

Estimator with many parameters (Strong)

- Generally low bias
- Fits data well
- Yields high variance

Estimator with few parameters (Weak)

- Generally high bias
- May not fit data well
- The fit does not change much for different data sets (low variance)

Can we turn a **weak learner** into a **strong learner**?

- This question was posed by Kearns and Valiant in 1988

- Solved in 1990 by Robert Schapire, then a graduate student at MIT
  "The Strength of Weak Learnability"
  (details are beyond the scope of this course)

- In 1995, Schapire and Freund proposed AdaBoost:
  Turn a set of weak learners (e.g. simple rule of thumbs) into a strong learner

Board

Y-DATA SCHOOL OF DATA SCIENCE

Yandex

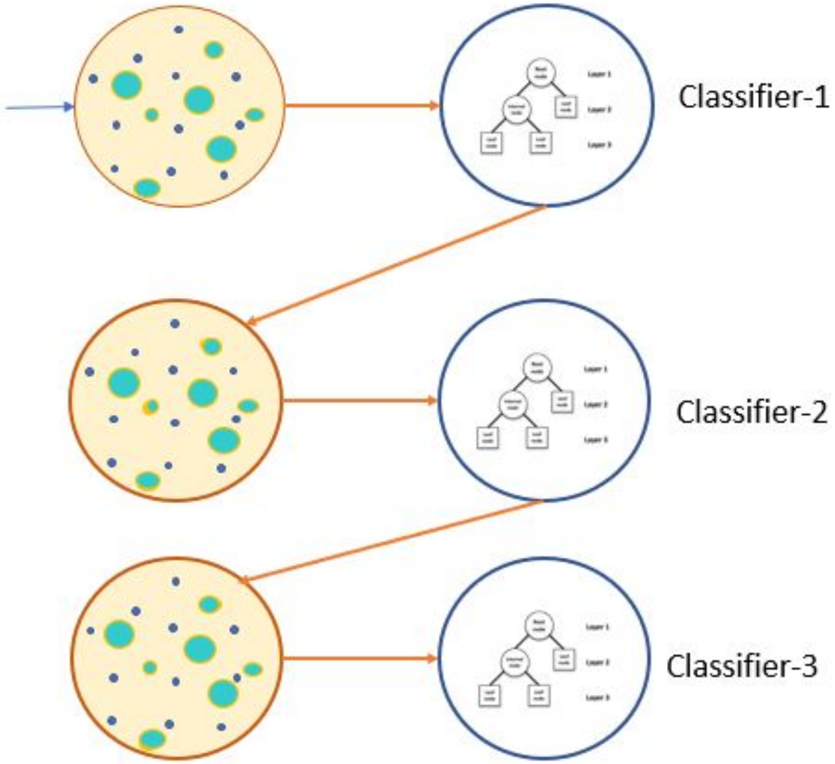
Main Concept

1. Focus on difficult instances with high error

2. Iteratively increase weights for high error examples

3. Combine the results with weighted voting

Boosting



Sequential

andex

Algorithm 1: AdaBoost Sketch

Input: Training Data $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of rounds M.

Training:

Define a weight distribution over the examples $D_i^1 = \frac{1}{N}$, for $i = 1, 2, \ldots, N$.

for round $j = 1$ to M **do**

  Build a model h_j from the training set using distribution D^j.

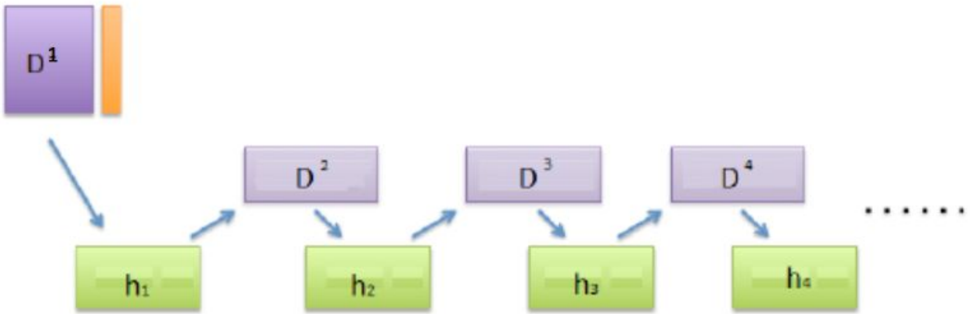
  Update D^{j+1} from D^j:

    Increase weights of examples misclasified by h_j.

    Decrease weights of examples correctly classified by h_j.

end for

Prediction: For a new example $\boldsymbol{x}'$, output the weighted (confidence-rated) majority vote of the models $\{h_1, h_2, \ldots, h_M\}$.



Yandex

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathscr{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Fig. 1 The boosting algorithm AdaBoost.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathscr{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\boxed{\varepsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].}$$

Weighted error of the j'th model

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Fig. 1 The boosting algorithm AdaBoost.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathscr{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\boxed{\varepsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].}$$

Weighted error of the j’th model

- Choose $\boxed{\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)}$.

“Confidence” of the j’th model

- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Fig. 1 The boosting algorithm AdaBoost.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathscr{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\boxed{\varepsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].}$$

Weighted error of the j'th model

- Choose $\boxed{\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)}$.

"Confidence" of the j'th model

- Update, for $i = 1, \ldots, m$:

$$\boxed{D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}}$$

Update weights - after normalization, the probabilities sum to 1

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

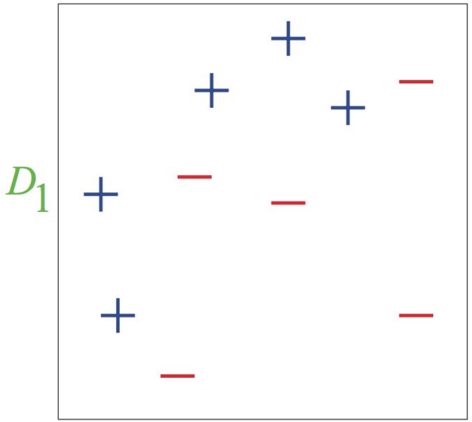
Fig. 1 The boosting algorithm AdaBoost.

AdaBoost: Toy Example

Train data

x1	x2	y	D1
1	5	+	0.10
2	3	+	0.10
3	2	–	0.10
4	6	–	0.10
4	7	+	0.10
5	9	+	0.10
6	5	–	0.10
6	7	+	0.10
8	5	–	0.10
8	8	–	0.10
			1.00

Initialization



Y-DATA SCHOOL OF DATA SCIENCE

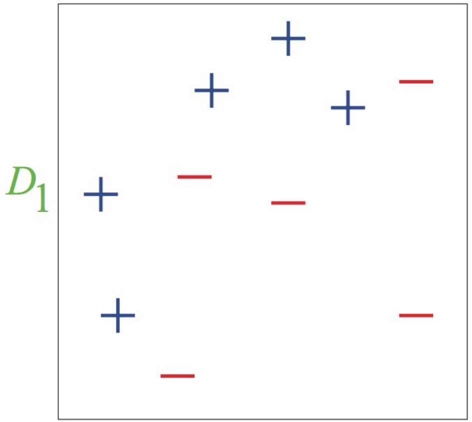
Yandex

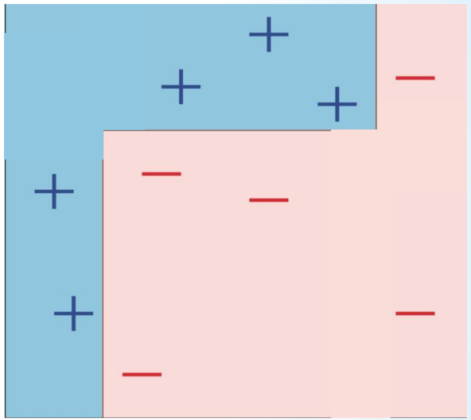
AdaBoost: Toy Example

Train data

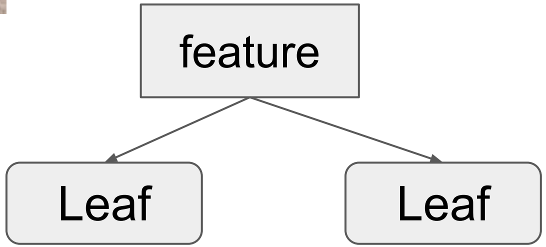
x1	x2	y	D1
1	5	+	0.10
2	3	+	0.10
3	2	–	0.10
4	6	–	0.10
4	7	+	0.10
5	9	+	0.10
6	5	–	0.10
6	7	+	0.10
8	5	–	0.10
8	8	–	0.10
			1.00

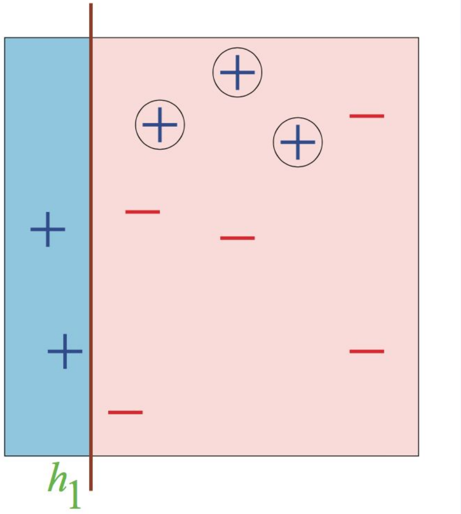
Initialization





Stump tree





Y-DATA SCHOOL OF DATA SCIENCE

Yandex

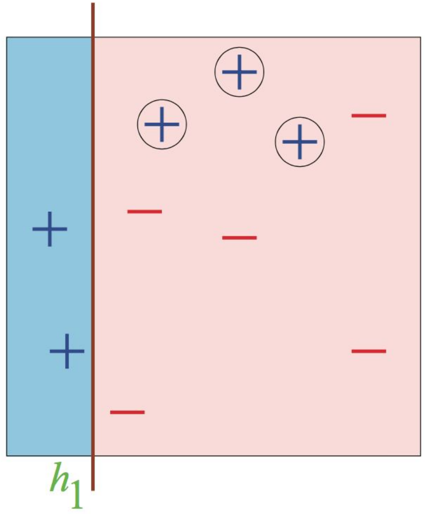
AdaBoost: Toy Example

Train data / Round

x1	x2	y	D1	h1e	ε
1	5	+	0.10	0	0.00
2	3	+	0.10	0	0.00
3	2	–	0.10	0	0.00
4	6	–	0.10	0	0.00
4	7	+	0.10	1	0.10
5	9	+	0.10	1	0.10
6	5	–	0.10	0	0.00
6	7	+	0.10	1	0.10
8	5	–	0.10	0	0.00
8	8	–	0.10	0	0.00
			1.00	ε1	0.30

Initialization

$$\varepsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$



Y-DATA SCHOOL OF DATA SCIENCE

Yandex

AdaBoost: Toy Example

Train data

x1	x2	y
1	5	+
2	3	+
3	2	–
4	6	–
4	7	+
5	9	+
6	5	–
6	7	+
8	5	–
8	8	–

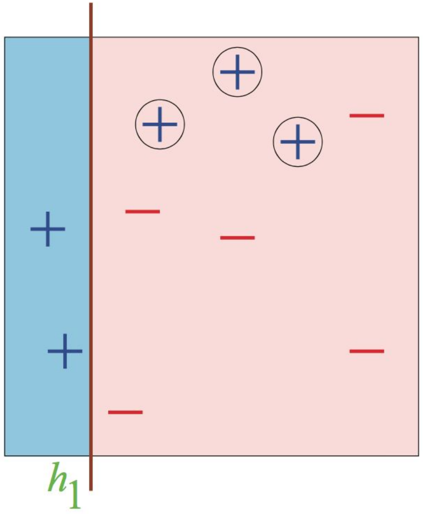
D1
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
1.00

Initialization

Round

h1e	ε
0	0.00
0	0.00
0	0.00
0	0.00
1	0.10
1	0.10
0	0.00
1	0.10
0	0.00
0	0.00
ε1	0.30
α1	0.42

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$$



AdaBoost: Toy Example

Train data

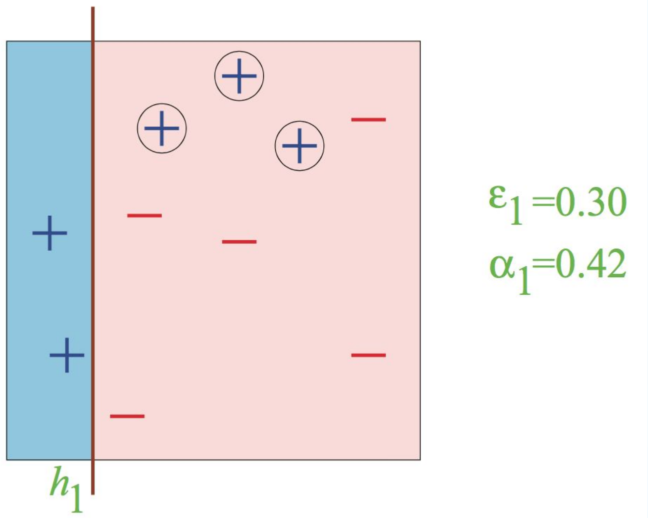
x1	x2	y
1	5	+
2	3	+
3	2	–
4	6	–
4	7	+
5	9	+
6	5	–
6	7	+
8	5	–
8	8	–

D1
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
1.00

Initialization

Round 1

h1e	ε	D2
0	0.00	0.07
0	0.00	0.07
0	0.00	0.07
0	0.00	0.07
1	0.10	0.17
1	0.10	0.17
0	0.00	0.07
1	0.10	0.17
0	0.00	0.07
0	0.00	0.07
ε1	0.30	1.00
α1	0.42	↕
	Zt	0.92



$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

AdaBoost: Toy Example

Train data

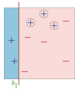
x1	x2	y
1	5	+
2	3	+
3	2	–
4	6	–
4	7	+
5	9	+
6	5	–
6	7	+
8	5	–
8	8	–

D1
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
1.00

Initialization

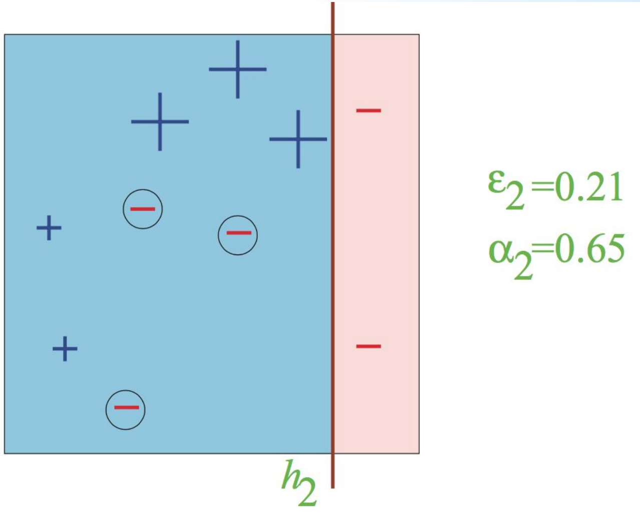
Round 1

h1e	ε	D2
0	0.00	0.07
0	0.00	0.07
0	0.00	0.07
0	0.00	0.07
1	0.10	0.17
1	0.10	0.17
0	0.00	0.07
1	0.10	0.17
0	0.00	0.07
0	0.00	0.07
ε1	0.30	1.00
α1	0.42	⇕
	Zt	0.92



Round 2

h2e	ε	D3
0	0.00	0.05
0	0.00	0.05
1	0.07	0.17
1	0.07	0.17
0	0.00	0.11
0	0.00	0.11
1	0.07	0.17
0	0.00	0.11
0	0.00	0.05
0	0.00	0.05
ε2	0.21	1.00
α2	0.65	⇕
	Zt	0.82



AdaBoost: Toy Example

Train data

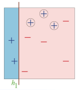
x1	x2	y
1	5	+
2	3	+
3	2	–
4	6	–
4	7	+
5	9	+
6	5	–
6	7	+
8	5	–
8	8	–

D1
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
1.00

Initialization

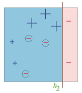
Round 1

h1e	ε	D2
0	0.00	0.07
0	0.00	0.07
0	0.00	0.07
0	0.00	0.07
1	0.10	0.17
1	0.10	0.17
0	0.00	0.07
1	0.10	0.17
0	0.00	0.07
0	0.00	0.07
ε1	0.30	1.00
α1	0.42	⇕
	Zt	0.92



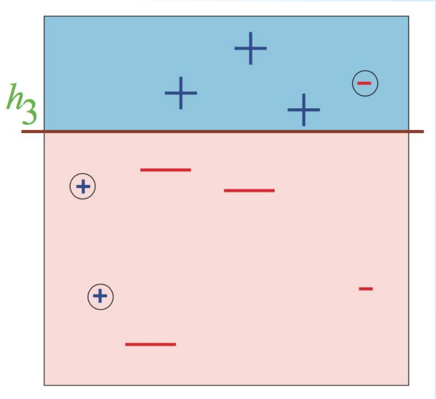
Round 2

h2e	ε	D3
0	0.00	0.05
0	0.00	0.05
1	0.07	0.17
1	0.07	0.17
0	0.00	0.11
0	0.00	0.11
1	0.07	0.17
0	0.00	0.11
0	0.00	0.05
0	0.00	0.05
ε2	0.21	1.00
α2	0.65	⇕
	Zt	0.82

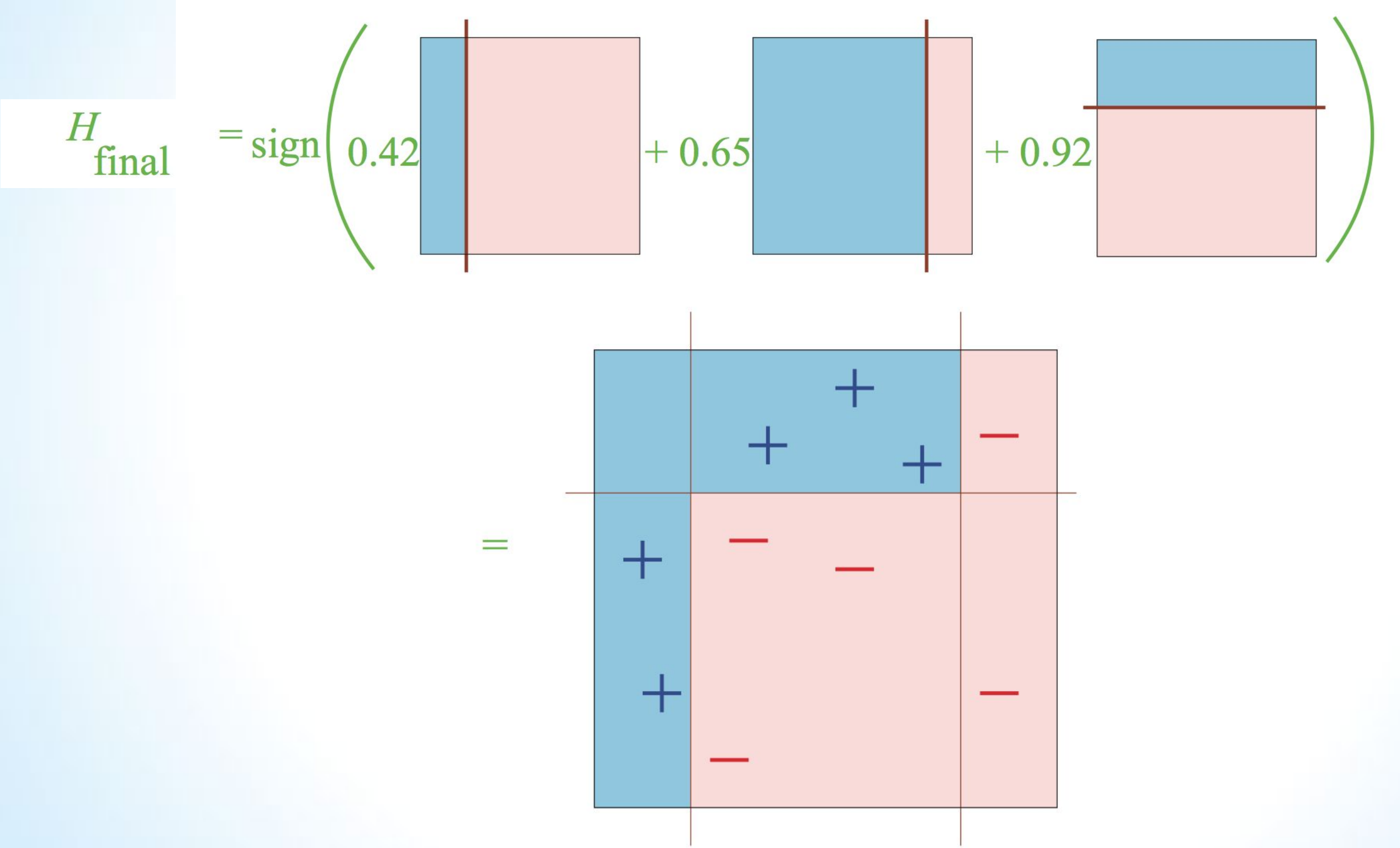


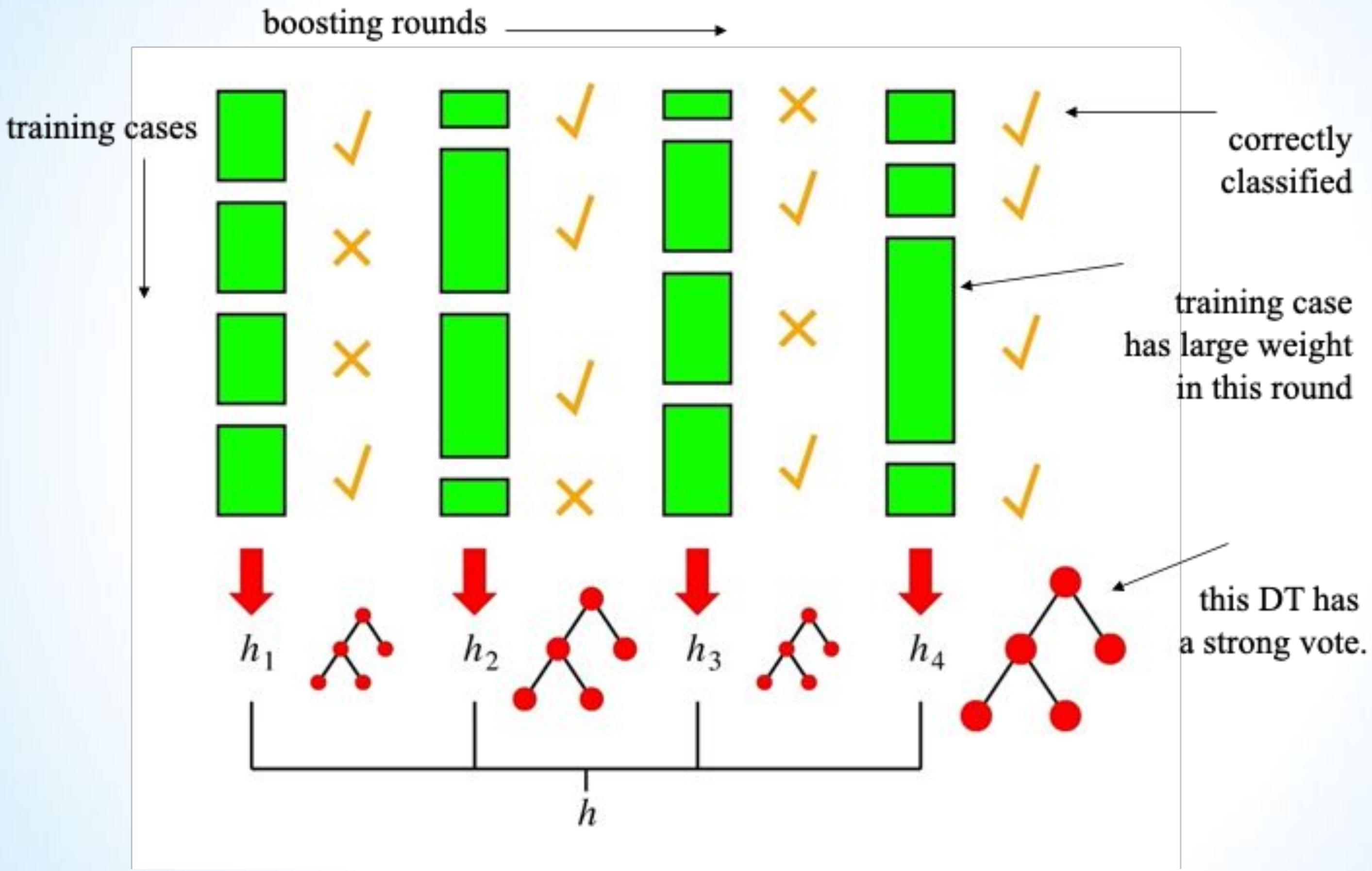
Round 3

h3e	ε
1	0.05
1	0.05
0	0.00
0	0.00
0	0.00
0	0.00
0	0.00
0	0.00
0	0.00
1	0.05
ε3	0.14
α3	0.92



AdaBoost: Toy Example




boosting rounds
training cases
correctly
classified
training case
has large weight
in this round
this DT has
a strong vote.
h_1
h_2
h_3
h_4
h

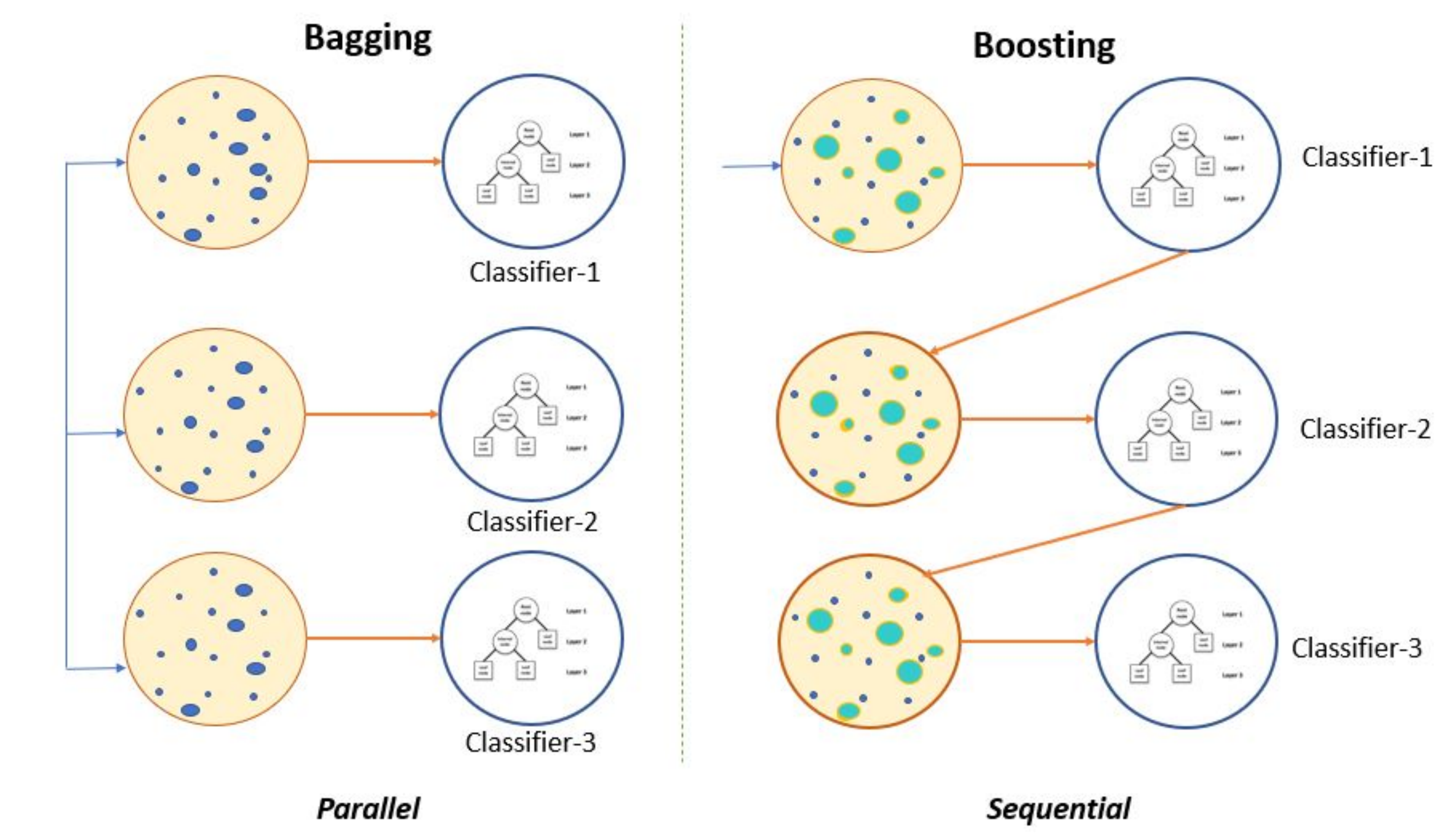
Intuition about the weights update

- Models with high error -> smaller weight in the decision function
- Models that worse than random -> just flip the predictions
- Models exactly like random -> no information -> eliminated
- Examples that Good models (high alpha) are mistaken -> get boosted

Its empirically shown that:

- **Bagging**: reduce variance.

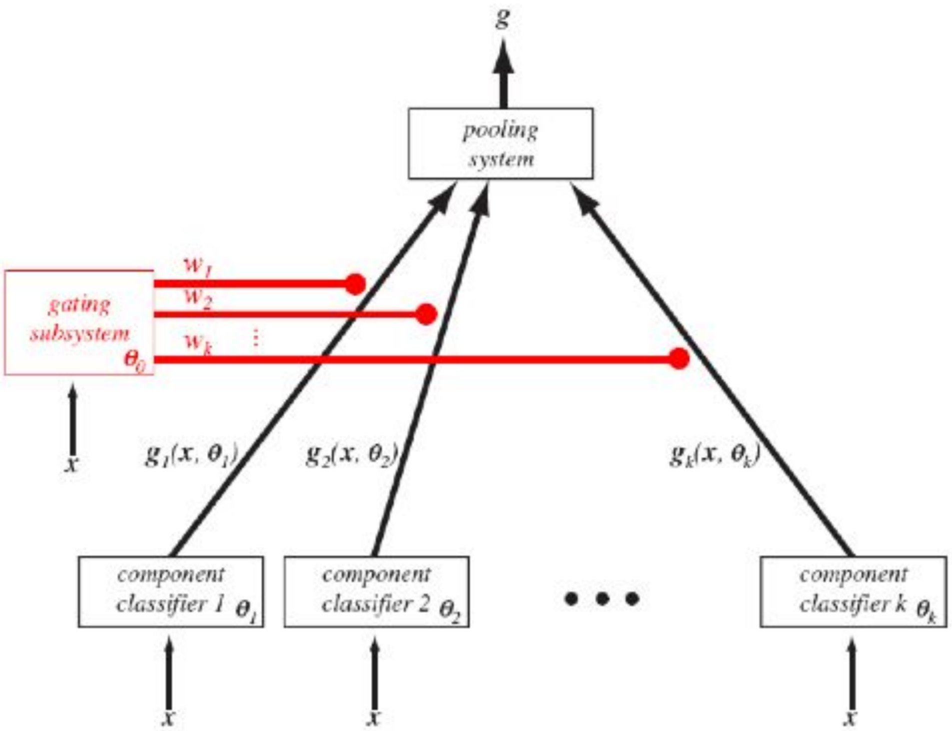
- **AdaBoost**: reduces both the bias and the variance.
  it seems that bias is mostly reduced in early iterations, while variance in later ones.


Bagging
Classifier-1
Classifier-2
Classifier-3
Parallel
Boosting
Classifier-1
Classifier-2
Classifier-3
Sequential

Stacking

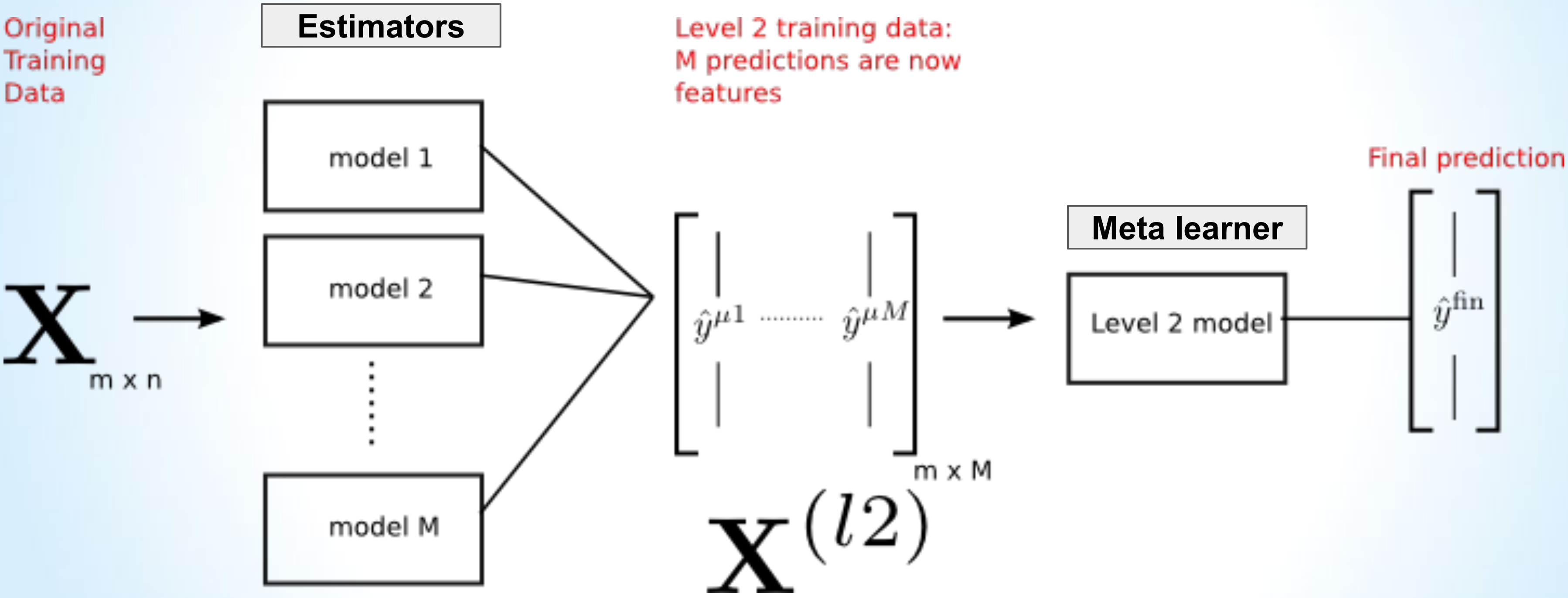
Ridge GLM
Maxout DNN
Random Forest
Lasso GLM
Rectifier DNN
GBM

Mixture of experts



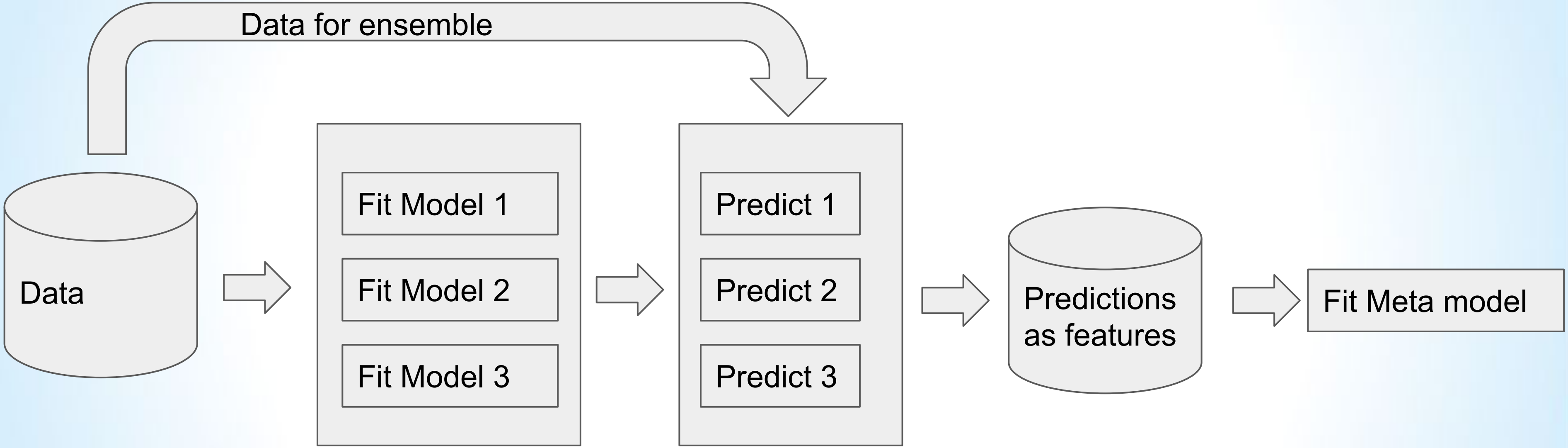
Nowlan, S. J. and Hinton, G. E. (1991) Evaluation of Adaptive Mixtures of Competing Experts *Advances in Neural Information Processing Systems 3.*

Stacking models

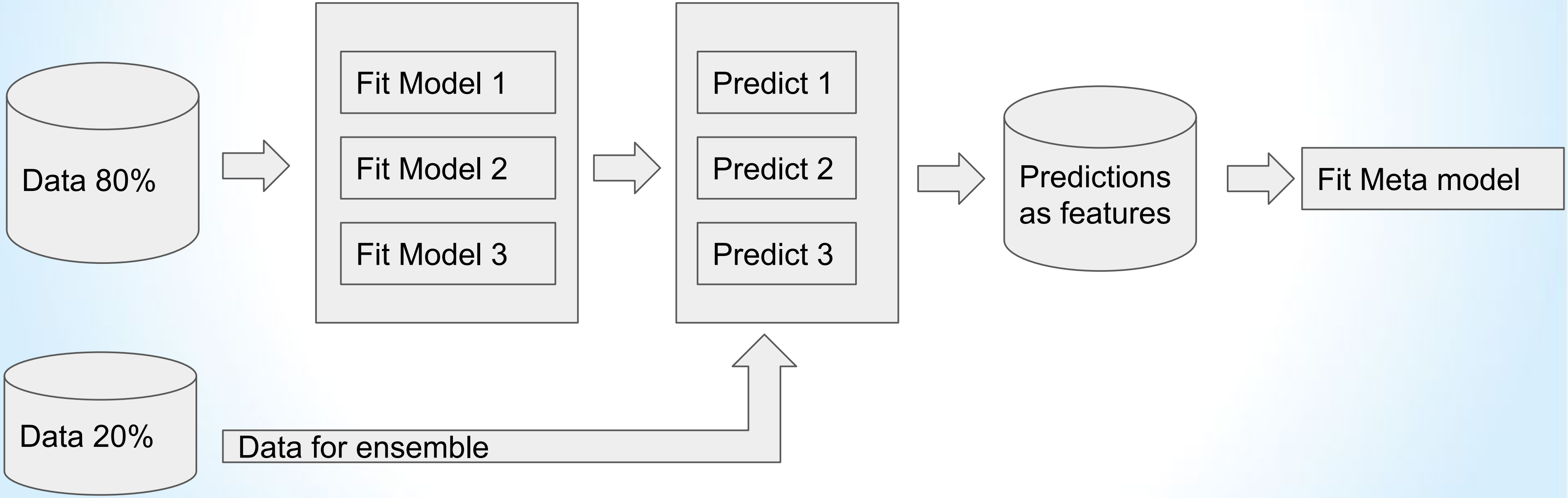


Wolpert, David H. "Stacked generalization." *Neural networks*5.2 (1992): 241-259.

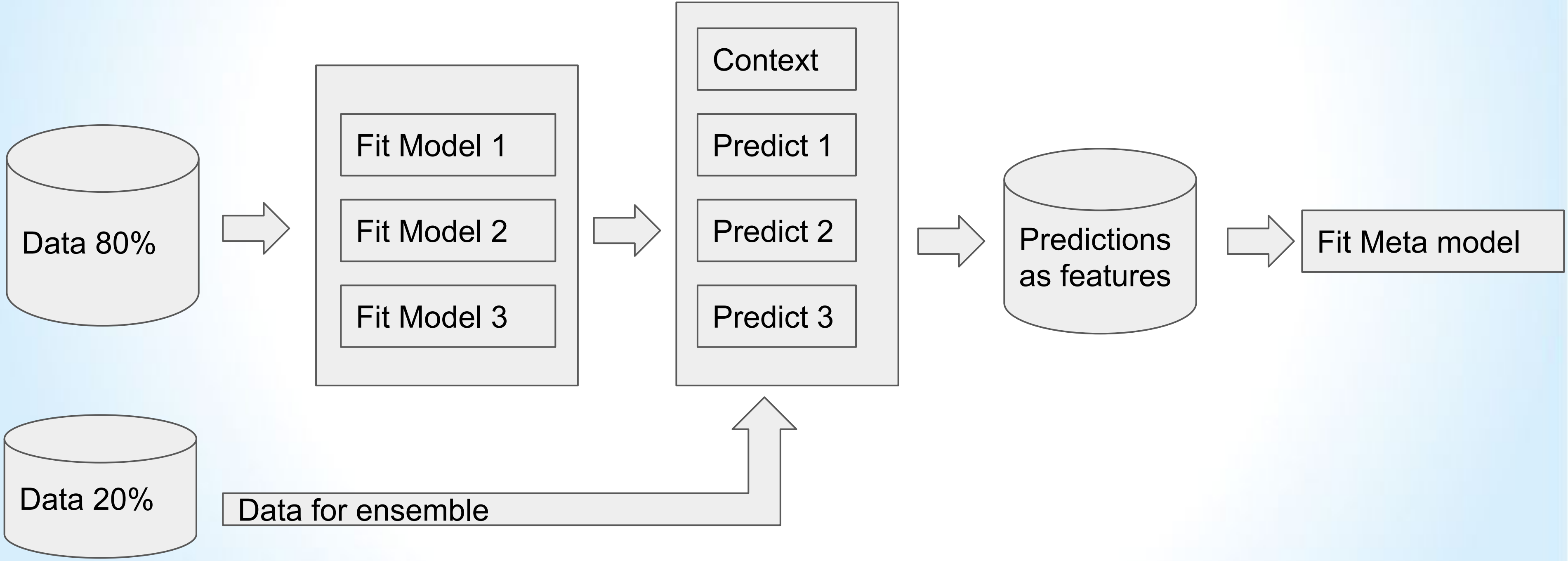
Is there a Leakage?



Is there a Leakage? Yes



Improve with context features



Stacking best practices

Dependency - ?

Aggregate - ?

Diversify - ?

Stacking best practices

Dependency - train different models, separate dataset per level
Aggregate - learn it - the stacked model
Diversify - train on different features or/and labels

Important Advantages:

- Easy integration between model versions
- Try advance modeling without risking current model
- Data scientists can work in parallel on same code base

Methods Comparison

	Bagging Random Forest	**Boosing** AdaBoost	**Stacking**
Diversity			
Dependency			
Aggregate			
Estimator			

Methods Comparison

	Bagging Random Forest	**Boosing** AdaBoost	**Stacking**
Diversity	Sample space	Sample Weights	
Dependency	Independent	Dependant	
Aggregate	Equal	Weighted	
Estimator	Complex	Simple	

Y-DATA SCHOOL OF DATA SCIENCE

Yandex

Methods Comparison

	Bagging Random Forest	**Boosing** AdaBoost	**Stacking**
Diversity	Sample space	Sample Weights	Both
Dependency	Independent	Dependant	Both
Aggregate	Equal	Weighted	Both
Estimator	Complex	Simple	Both

Summary

“Two heads are better than none. One hundred heads are so much better than one”

Dearg Doom, The Tain, Horslips, 1973

“Great minds think alike, clever minds think together”

L. Zoref, 2011.

- But they must be different and specialised.
- And it might be an idea to select only the best of them for the problem at hand

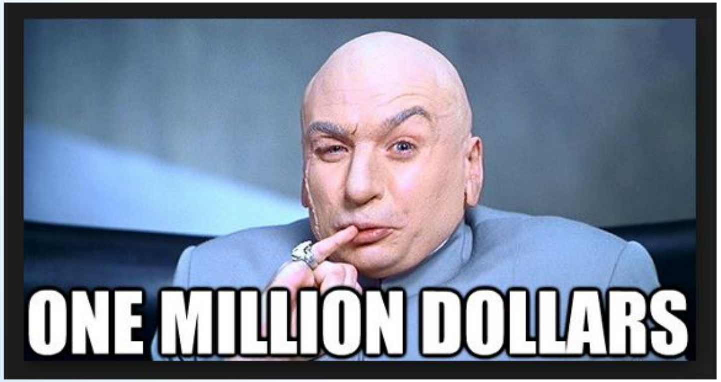
Y-DATA
SCHOOL OF DATA SCIENCE

Yandex

End

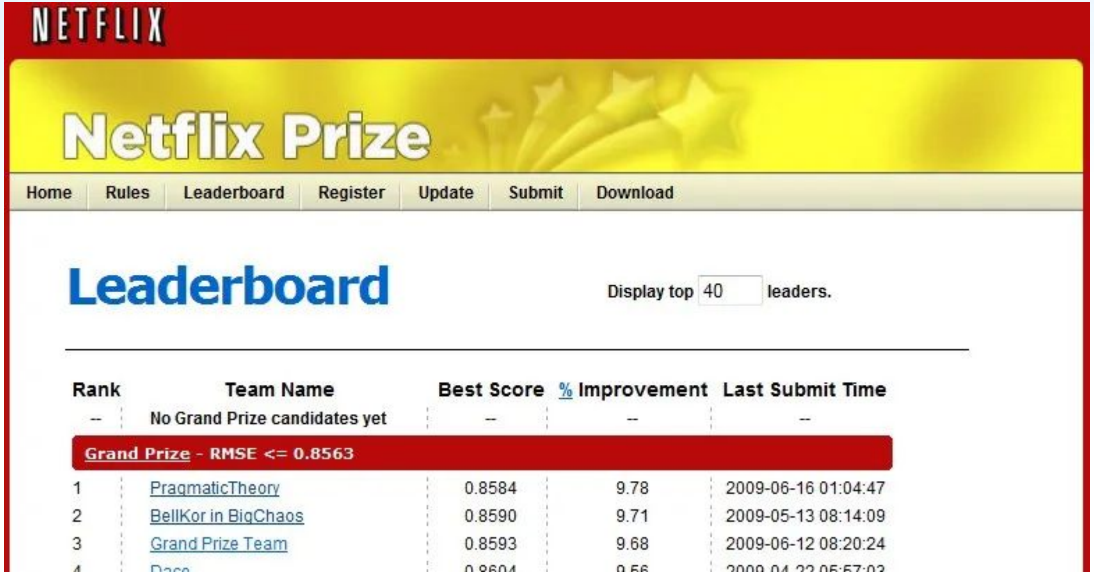
Netflix Prize

Task: Predict number of stars given to a movie by a user
Goal: Improve current model by 10%
Data: 100M+ Ratings, 480K+ users, 17K+ movies
Competitors: 20K+ teams, 150+ countries.
Prize: 1,000,000$



Y-DATA
SCHOOL OF DATA SCIENCE

Yandex


NETFLIX

Netflix Prize

Home | Rules | Leaderboard | Register | Update | Submit | Download

Leaderboard

Display top 40 leaders.

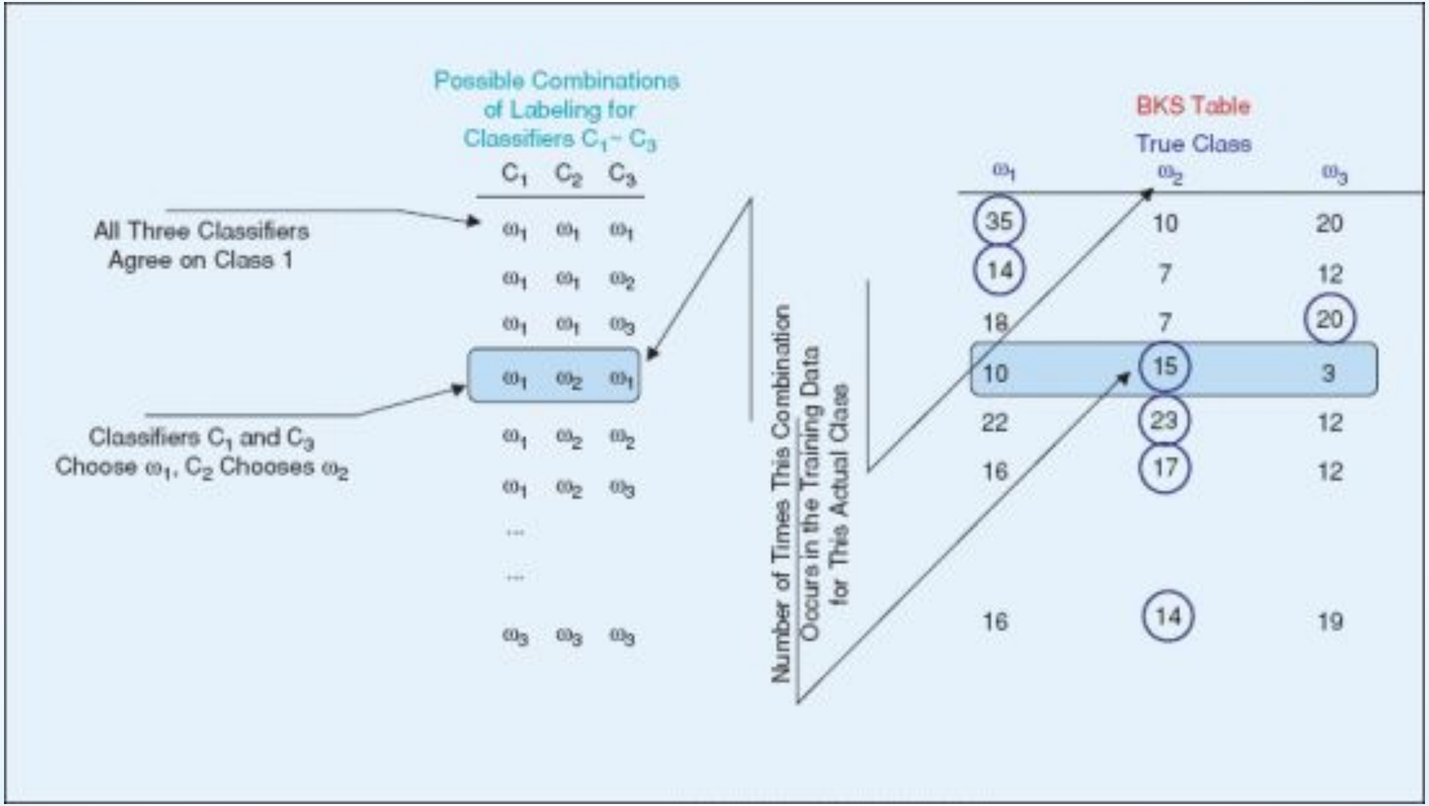
Rank	Team Name	Best Score	% Improvement	Last Submit Time
--	No Grand Prize candidates yet	--	--	--
Grand Prize - RMSE <= 0.8563				
1	PragmaticTheory	0.8584	9.78	2009-06-16 01:04:47
2	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
3	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24

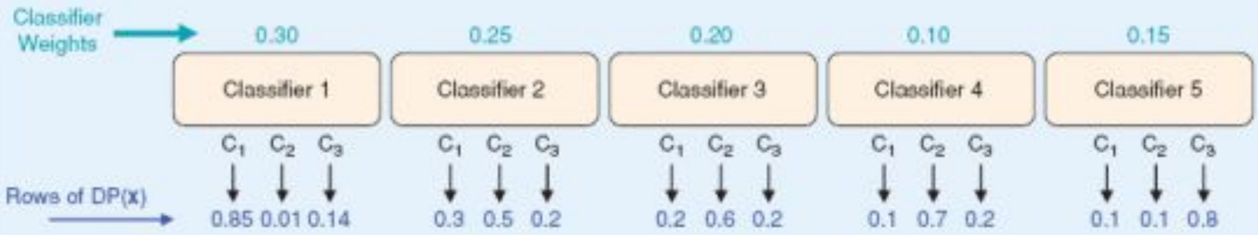
Yehuda Koren

Behavior Knowledge space

1. In training:
   a. Fit each classifier separately
   b. For each classification combination count how many times each classifier was correct and save it in a lookup table
2. In prediction:
   a. Perict each classifier
   b. Use lookup table for prediction combination
   c. Predict with the best model

Behavior Knowledge space





Product Rule:	**C_1: 0.0179,**	C_2: 0.00021,	C_3: 0.0009
Sum Rule:	C_1: 1.55,	**C_2: 1.91,**	C_3: 1.54
Mean Rule:	C_1: 0.310,	**C_2: 0.382,**	C_3: 0.308
Max Rule:	**C_1: 0.85,**	C_2: 0.7,	C_3: 0.8
Median Rule:	C_1: 0.2,	**C_2: 0.5,**	C_3: 0.2
Minimum Rule:	C_1: 0.1,	C_2: 0.01,	**C_3: 0.14**
Weighted Average	**C_1: 0.395,**	C_2: 0.333,	C_3: 0.272
Majority Voting:	C_1: 1,	**C_2: 3,**	C_3: 1 Vote
Weighted Majority Voting:	C_1: 0.3,	**C_2: 0.55,**	C_3: 0.15 Votes
Borda Count:	C_1: 5	**C_2: 6,**	C_3: 4 Votes