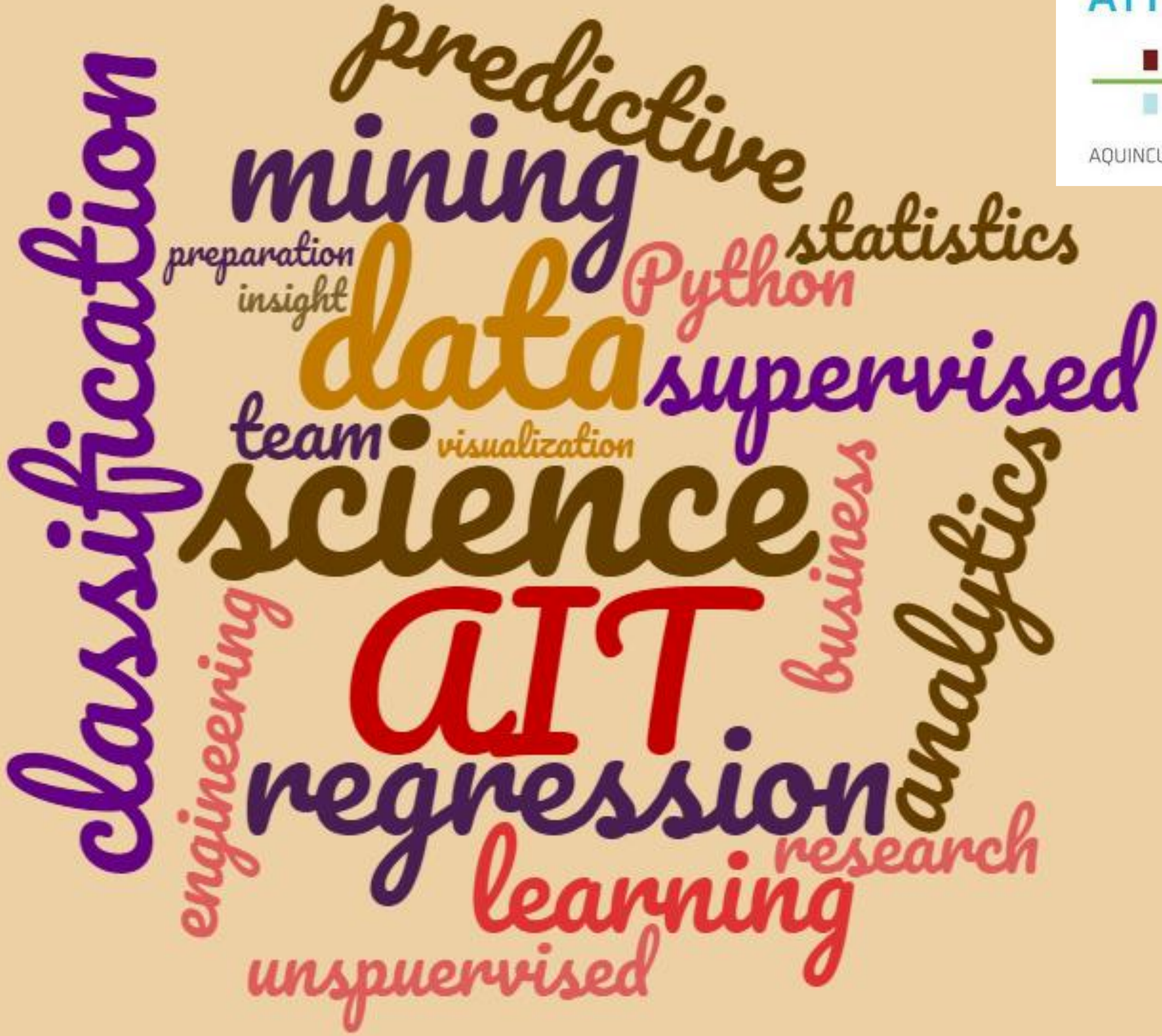
AIT-BUDAPEST

AQUINCUM INSTITUTE OF TECHNOLOGY

Data Science

April 24, 2020
Ensemble methods



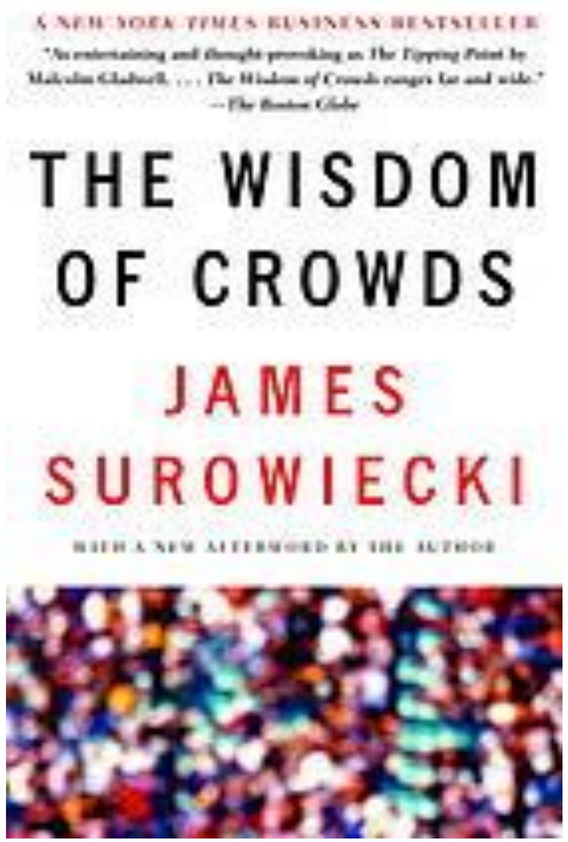
Roland Molontay

Galton and the weight of an ox

- The story of Sir Francis Galton (1906)
  - English statistician, polymath
  - He visited a country fair where 800 people participated in a contest to estimate the weight of an ox
  - To his surprise, the median guess, 1207 pounds was accurate within 1% of the true weight of 1198 pounds
    - None of the experts were as accurate as the crowd itself
  - Several similar experiments have been conducted with similar results

The wisdom of crowds

- Criteria required to form a wise crowd (collective intelligence)
  - Diversity of opinion
  - Independence
  - Decentralization, specialization
  - Appropriate aggregation



A NEW YORK TIMES BUSINESS BESTSELLER

"As entertaining and thought-provoking as The Tipping Point by Malcolm Gladwell. . . . The Wisdom of Crowds ranges far and wide." —The Boston Globe

THE WISDOM OF CROWDS

JAMES SUROWIECKI

WITH A NEW AFTERWORD BY THE AUTHOR

Ensemble methods in machine learning

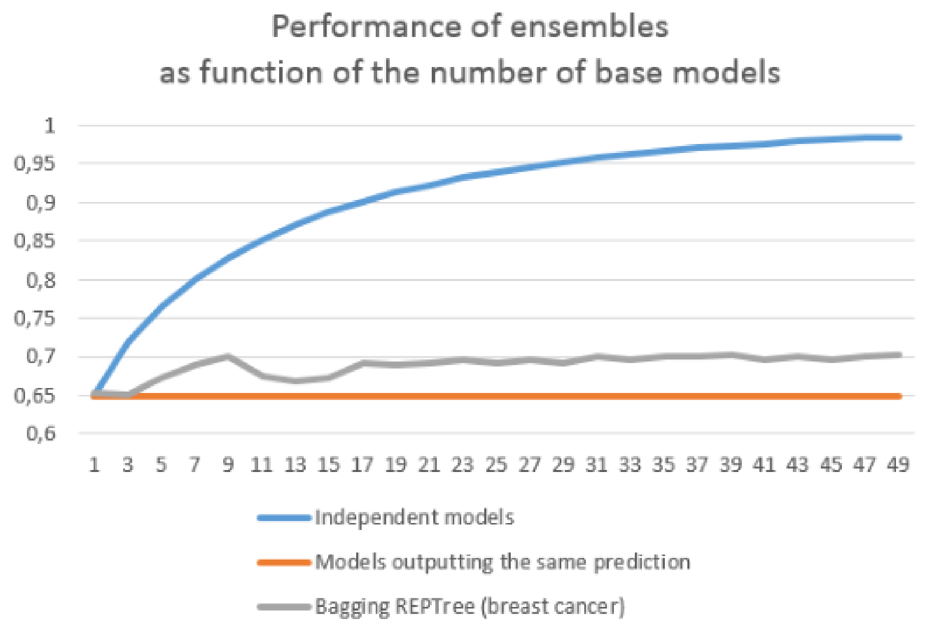
- Main idea: we combine multiple (simple) learning algorithms to have better performance
- Why is it good?
  - We have a binary classification task
  - We have 100 classification models
  - Assume that all of them misclassify a record with probability 0.4 (independently from each other)
  - The probability that more than half of the classifiers misclassify a single record is:

$$\sum_{k=50}^{100} \binom{100}{k} \cdot 0{,}4^k \cdot 0{,}6^{100-k} \approx 0{,}027.$$

    - Using majority voting we assign the right class with probability 0.973 (or 97.3%)

Performance of ensembles

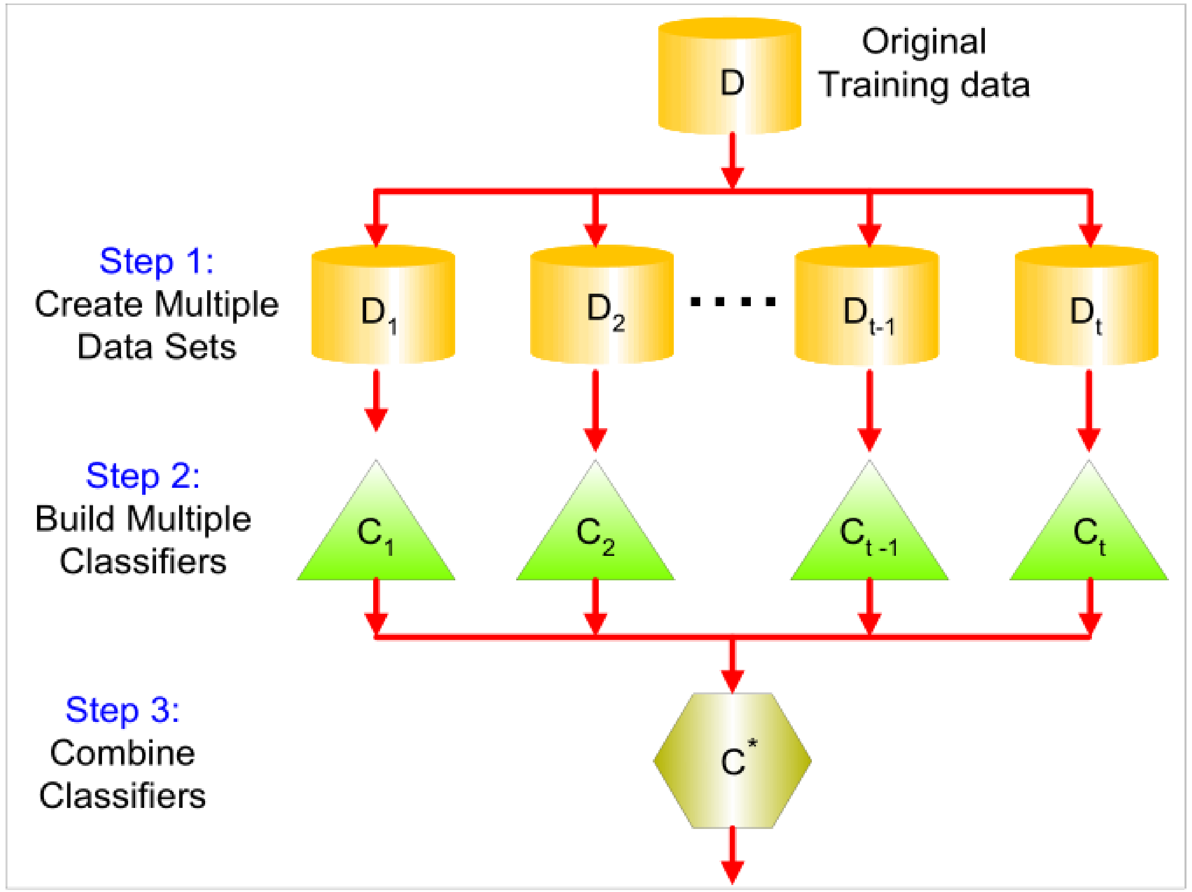
- In practice the increase in performance is not that sharp
  - In reality the models are not independent from each other
- But combining multiple classification algorithms usually results in better predictive performance than could be obtained from any of the composing learning algorithms alone
- It helps to reduce variance
- It helps to avoid overfitting



Motivations for ensemble learning

- Statistical motivation
  - Using ensemble methods, the algorithm can average the different hypotheses and reduce the risk of choosing the wrong classifier
- Computational motivation:
  - Individual classifiers may be stuck in local optima, an ensemble algorithm may provide a better result
- Representational motivation:
  - Ensemble methods can expand the space of hypotheses searched by the individual learning algorithms

Main idea of ensemble learning

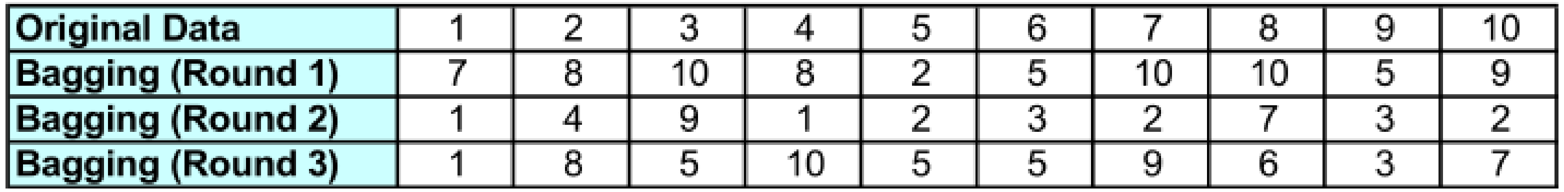


Multiple classifiers on a training set

- If the training set is very large: we partition the dataset into disjoint subsets, and we train an individual classifier on each subset
  - Usually the training set is not large enough to do that
- Bagging: random sampling with replacement
  - We create the samples independently from each other
  - The selection of all record is with equal probability
- Boosting: creating the sample in a sequential way, taking into account the previous classifier’s success
  - After each training step the selection probabilities (weights) are redistributed
  - Misclassified data increases its weights to emphasize the most difficult cases

Bagging

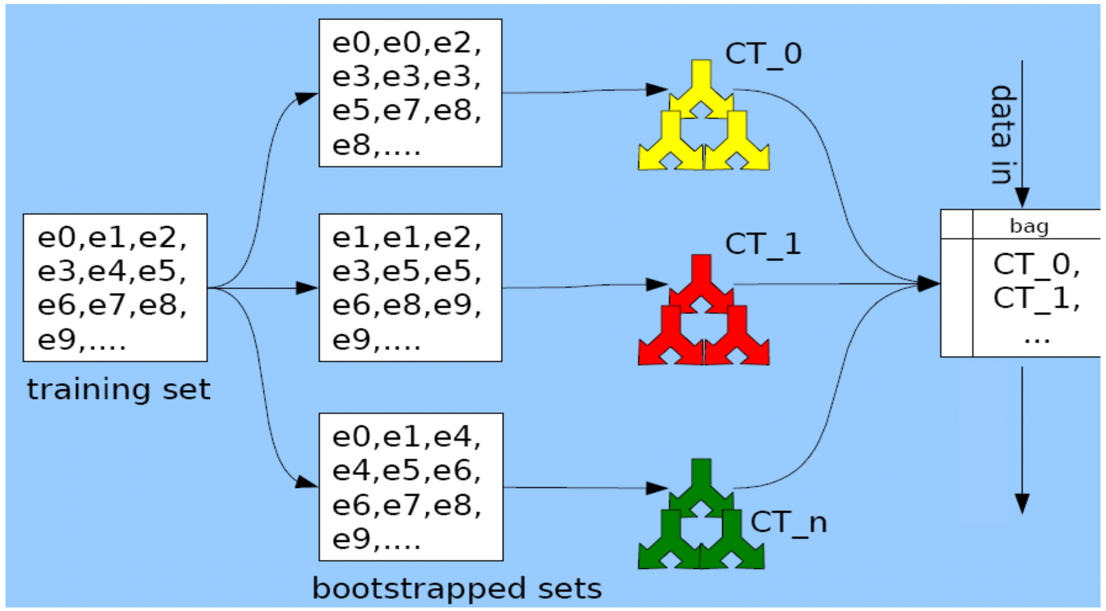
- Bagging (**B**ootstrap **Agg**regat**ing**)
- We train the individual models on random samples sampled with replacement

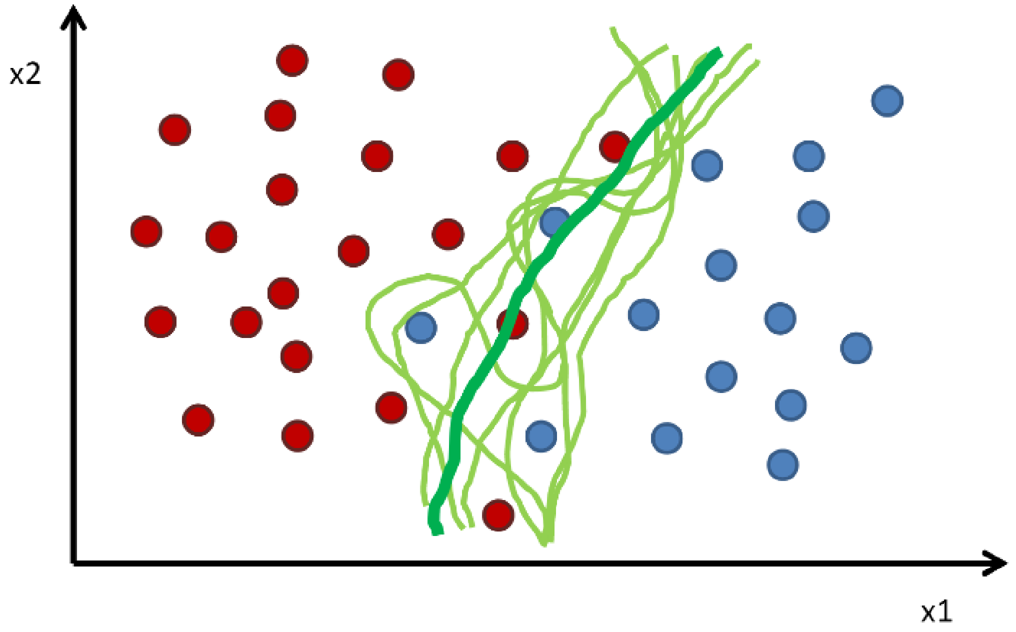


Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- The probability of selecting a record: $1-\left(1-\frac{1}{n}\right)^n \rightarrow 1-\frac{1}{e} \approx 0.632$
- The final result is aggregated using majority voting or averaging (for regression)

Bagging - examples





Bagging for attributes

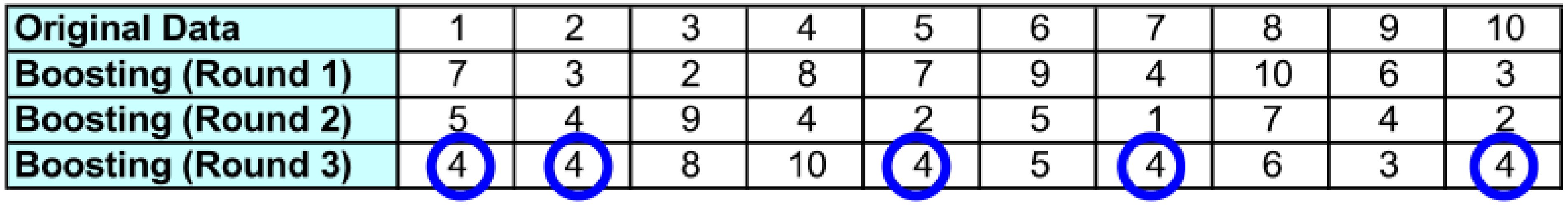
- Attribute bagging (random subspace method)
- Here the difference between the models is not due to sampling the training set but sampling the feature set
- We create (not necessarily disjoint) random samples of the features and train individual models using the sampled feature set

Using meta-models (stacking)

- The results of the individual (base-level) models are combined using a meta-classifier or meta-regressor
  - More sophisticated than just using majority voting or averaging
- The training set is partitioned into two disjoint sets: S_1 and S_2
  - On S_1 the base classifiers are trained
  - The records of set S_2 are classified using the base models, the meta-model is trained on the outputs of the base level model as features

Boosting

- Boosting: creating the sample in a sequential way, taking into account the previous classifier's success
  - After each training step the selection probabilities (weights) are redistributed
  - Misclassified data increases its weights to emphasize the most difficult cases
    - In the example record 4 seems to be a difficult one



Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

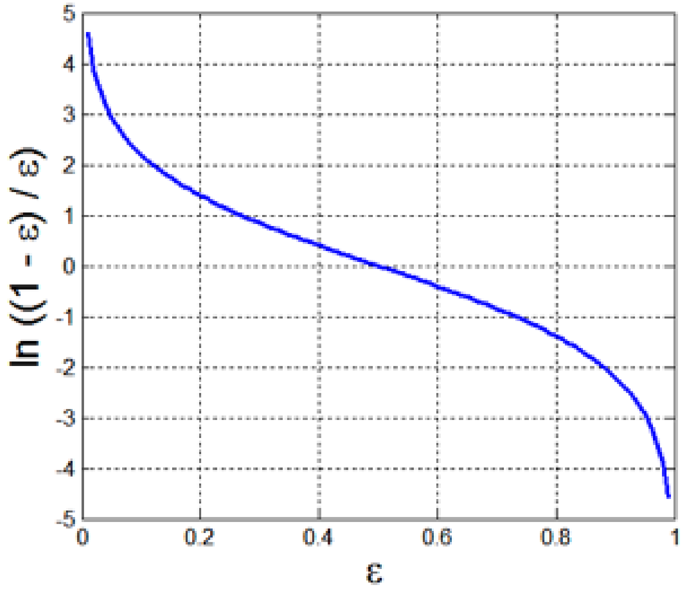
AdaBoost

- **Ada**ptive **boost**ing algorithms
- Base classifiers: $C_1, C_2, \dots, C_T$
- In step *j* classifier C_j is trained, and in step *j* the record *i* has weight $w_i^{(j)}$
- The error rate of classifier C_j:

$$\varepsilon_j = \frac{1}{N}\sum_{i=1}^{N} w_i \delta\left(C_j(x_i) \neq y_i\right)$$

- The importance of classifier C_j :

$$\alpha_j = \frac{1}{2}\ln\left(\frac{1-\varepsilon_j}{\varepsilon_j}\right)$$



AdaBoost II.

- Updating the weights

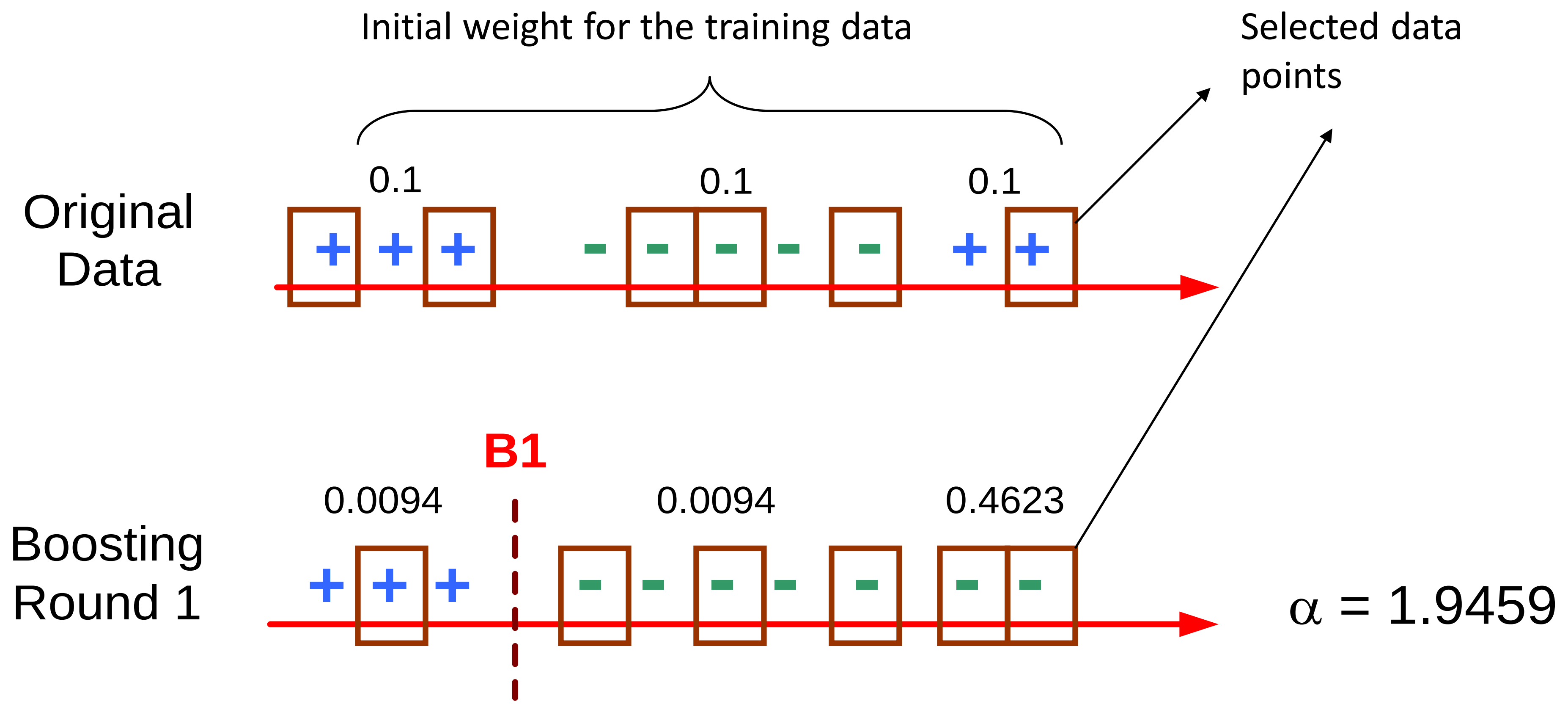
$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} e^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ e^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalizing factor to ensure that the sum of the weights is equal to 1

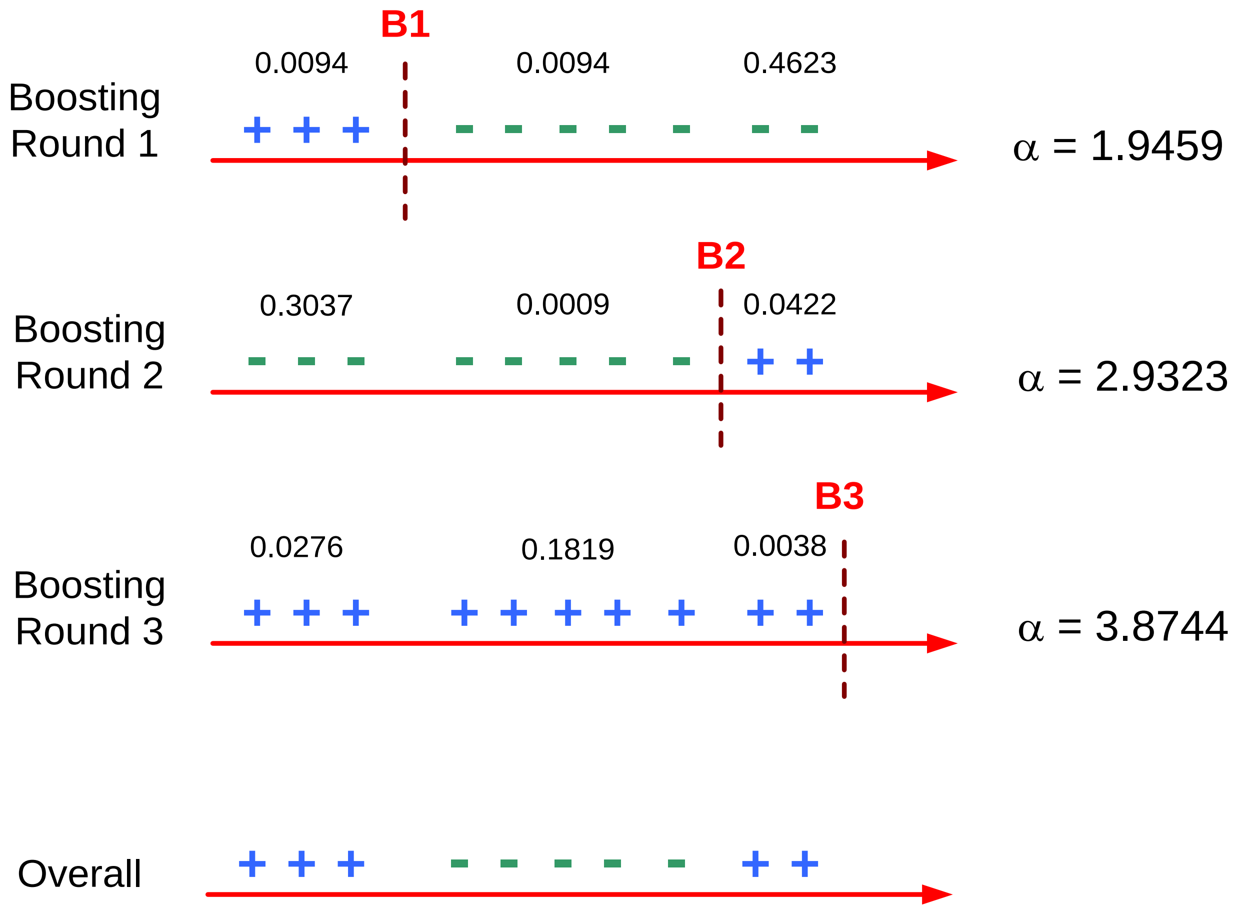
- Classification

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

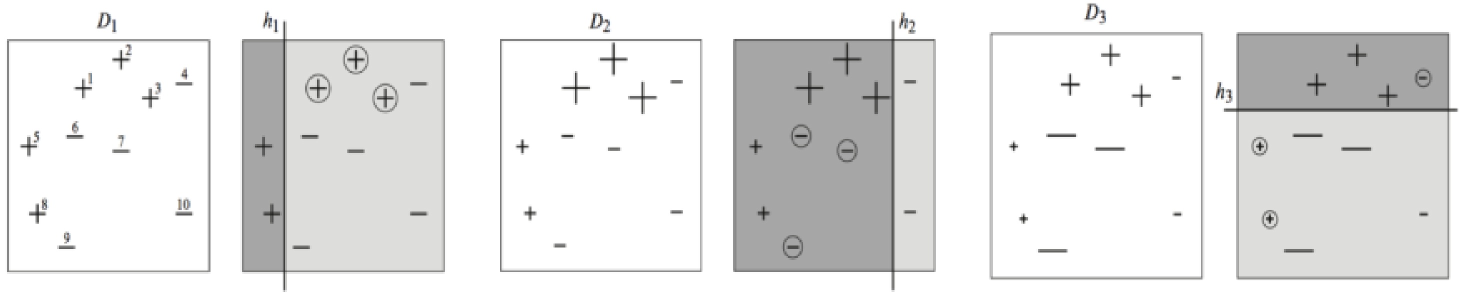
AdaBoost - example

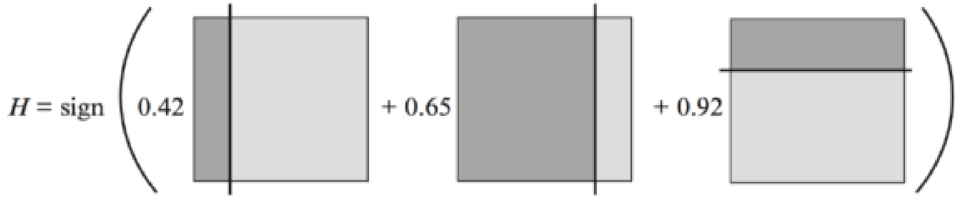


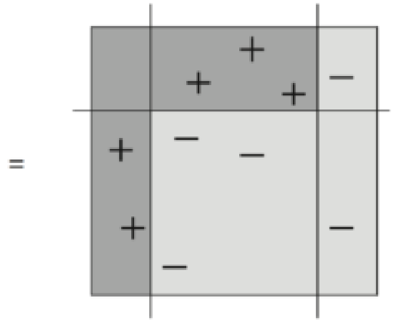
AdaBoost – example II.



AdaBoost – another example

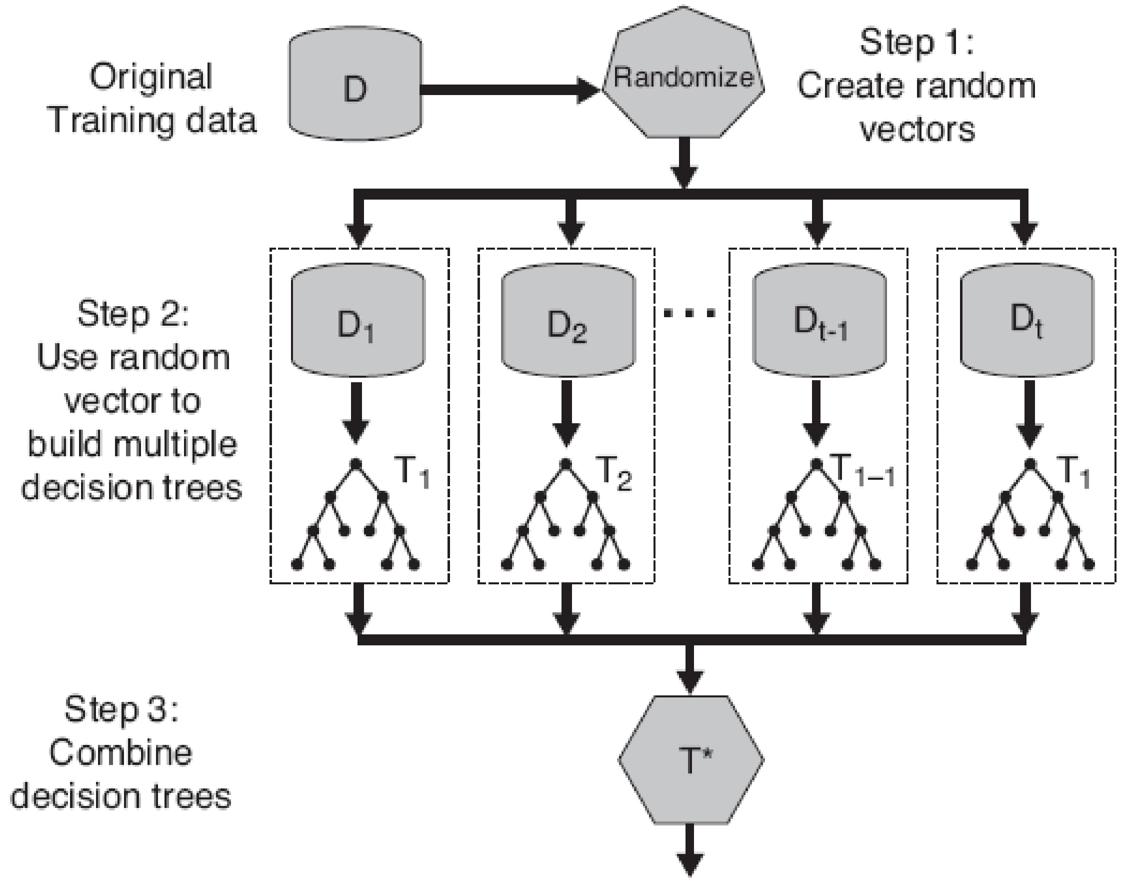






Random forest

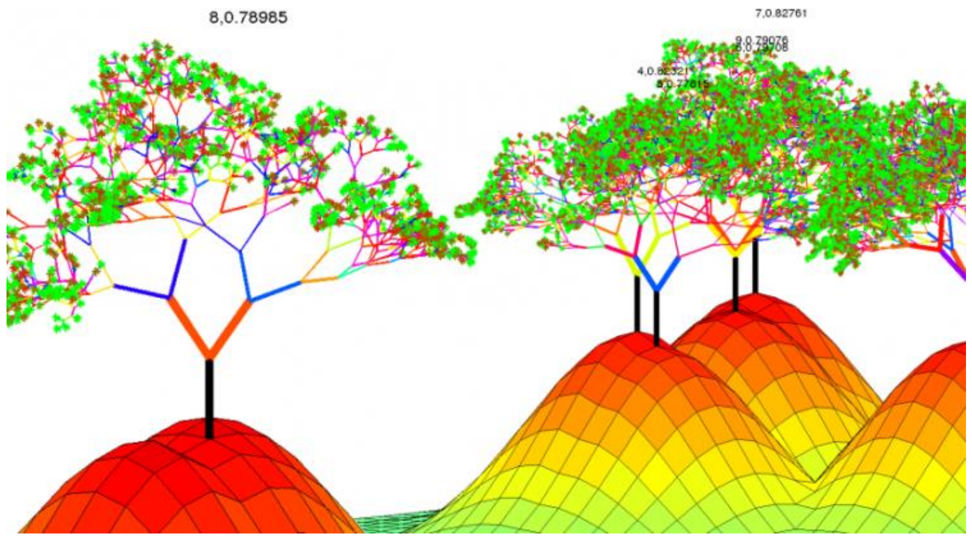
- We build multiple decision trees in such a way that
  - We create random samples from the training set using bagging method
  - We use attribute bagging for the features for training the individual trees
    - the feature set for an individual tree is much smaller than the original feature set



Original Training data
D
Randomize
Step 1: Create random vectors
Step 2: Use random vector to build multiple decision trees
D_1
D_2
D_{t-1}
D_t
T_1
T_2
T_{1-1}
T_1
Step 3: Combine decision trees
T^*

Random forest II.

- The base classifiers are decision trees
  - Trained on random samples (bagging)
  - For an individual tree we restrict the feature space for a much smaller number of features
  - The individual trees grow according to a decision tree algorithm
- The random forest uses majority voting on the results of the individual trees

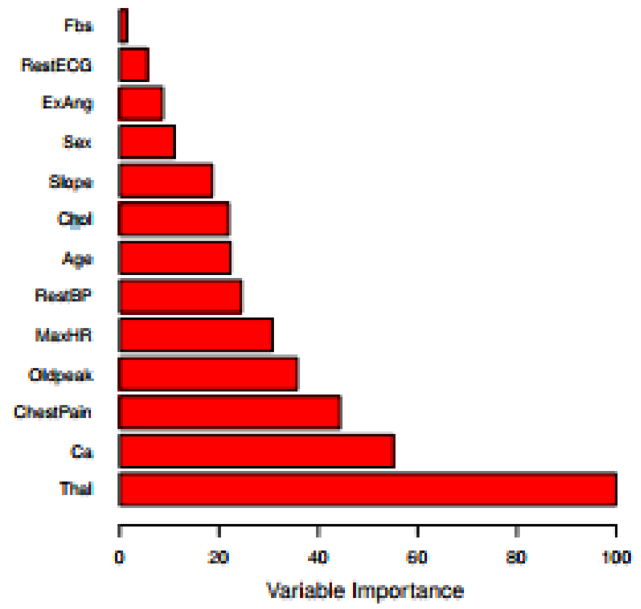


Efficiency of random forest

- Usually we generate a high number of decision trees
- The efficiency of RF model depends on the followings:
  - Number of generated trees (the more trees, the better result)
  - The correlation between the trees (the higher correlation, the worse result)
  - BUT: The more tree we have, the higher the correlation is
  - The more attributes we let the decision trees to split on, the higher the correlation between the trees is
  - We can optimize for these parameters:
    - number of trees
    - used number of features for each tree

Evaluation of random forest

- The interpretability of the model is worse than for a simple decision tree
- Fast, easy to parallelize
- It can be used to measure the feature importance
  - How many times the decision trees use a certain attribute to split on
- Reduce overfitting
- Can handle high dimension
- Small number of parameters (number of trees, cardinality of the reduced feature space)

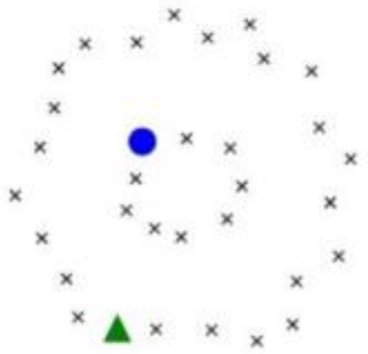


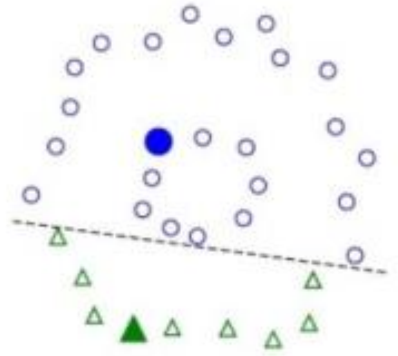
Semi-supervised learning

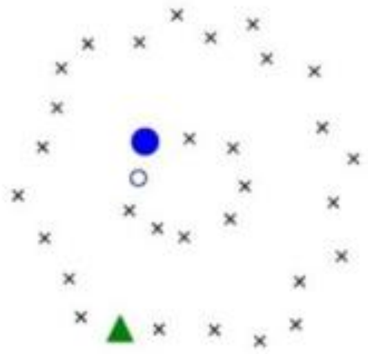
- We use this method if the training set is small or non-representative
- Procedure
  - We classify the object whose label we are the most confident in
  - We add the previously classified object (with the predicted label) to the training set
  - We retrain the model, and repeat the procedure on the unlabeled the data points
- Computationally expensive
- The algorithm is able to take into consideration the structure of the unlabeled data points

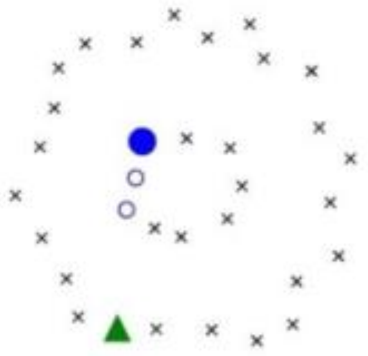
Example – semi supervised learning

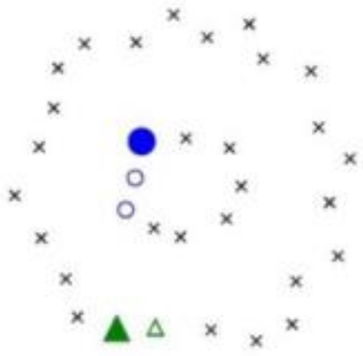
- kNN classifier
    - Two classes
    - The filled objects mark the records with known labels

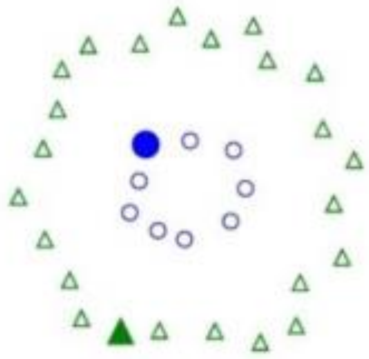

(a) The training set.


(b) Decision boundary with supervised training.


(c) 1st iteration of self-training.

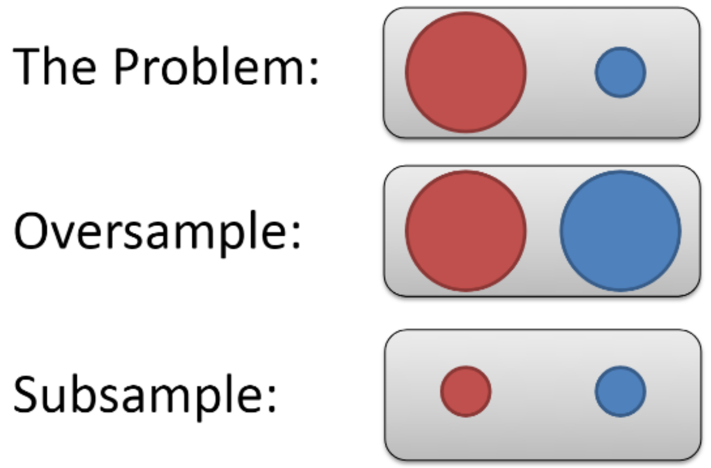

(d) 2nd iteration of self-training.


(e) 3rd iteration of self-training.


(f) Classification with self-training.

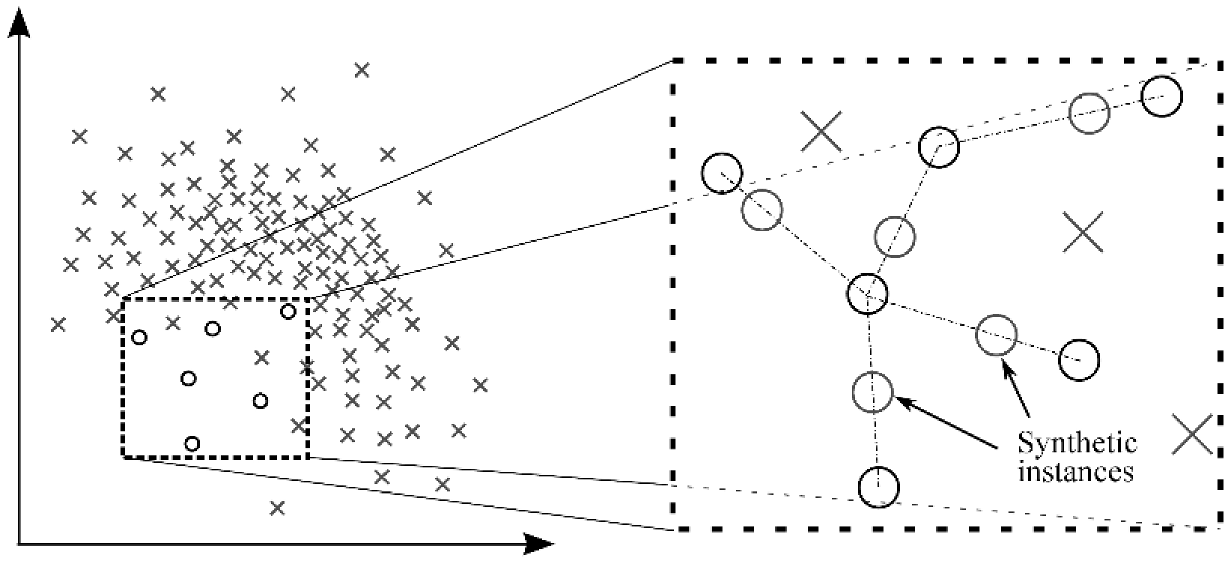
Classifying imbalanced data

- Goal: balancing the class distribution
- Methods:
  - Under-sampling, sub-sampling:
    we reduce the size of the frequent class by eliminating the objects that are easy to classify (that are far away from the decision boundary)
    - Disadvantage: we lose data
  - Over-sampling: synthetically supplementing the training data of the minority class
    - We won’t have more information on the minority class, still it can help to improve the performance of classifiers



SMOTE

- SMOTE (**S**ynthetic **M**inority **O**ver-sampling **Te**chnique)
- The most common oversampling procedure
- For all instances from the minority class we consider their *k* nearest neighbors from the minority class.
- To create a synthetic data point by taking a random point on the segment between the neighbor and current data point



Acknowledgement

- András Benczúr, Róbert Pálovics, SZTAKI-AIT, DM1-2
- Krisztián Buza, MTA-BME, VISZJV68
- Bálint Daróczy, SZTAKI-BME, VISZAMA01
- Judit Csima, BME, VISZM185
- Gábor Horváth, Péter Antal, BME, VIMMD294, VIMIA313
- Lukács András, ELTE, MM1C1AB6E
- Tim Kraska, Brown University, CS195
- Dan Potter, Carsten Binnig, Eli Upfal, Brown University, CS1951A
- Erik Sudderth, Brown University, CS142
- Joe Blitzstein, Hanspeter Pfister, Verena Kaynig-Fittkau, Harvard University, CS109
- Rajan Patel, Stanford University, STAT202
- Andrew Ng, John Duchi, Stanford University, CS229