Lecture 12:

Ensemble methods

Outline

- Motivation
- Idea of ensemble models
- Bucket of models (voting)
- Bagging
- Random forests
- Boosting
- Stacking

Motivation

Reminder:

- a classification task aims to estimate a relationship between the class attribute and the other attributes
- the construction of a classification model is based on:
  - a training dataset
  - some assumptions on the model (e.g. the decision boundary is linear or piecewise linear)
- Notations

y=f(x)=the true output (class) corresponding to a data instance x

D={(x1,y1),(x2,y2),…,(xL,yL)} = training dataset

g(x;D) = the estimated output produced by the model induced starting from the training dataset D

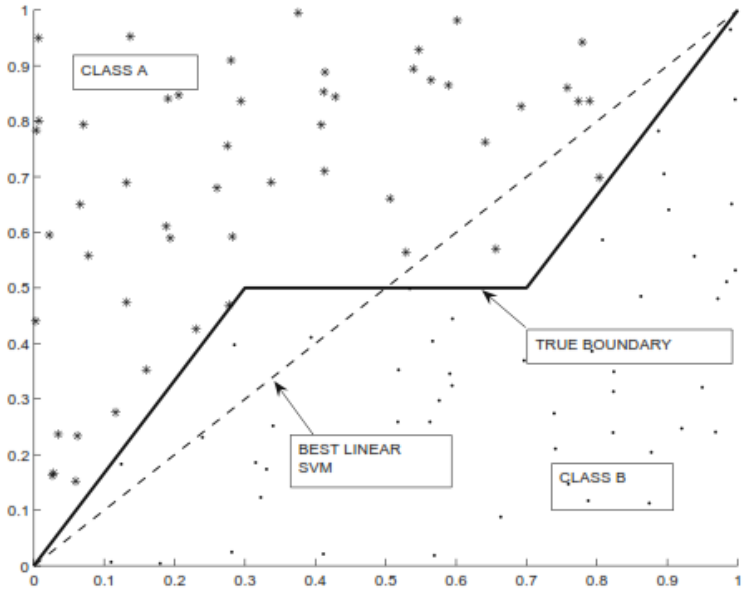
MSE = mean squared error

Model inference: estimate the model parameters such the MSE is minimized

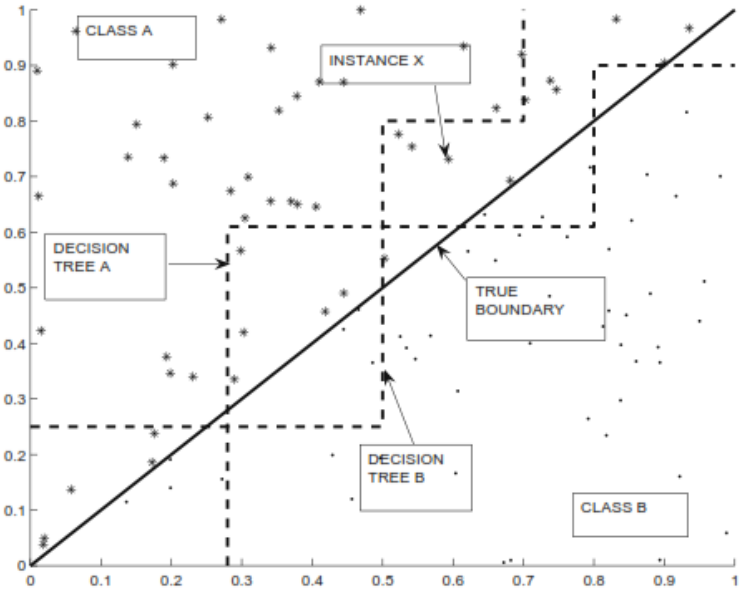
Motivation

Components of the error:

- Bias = caused by the limitations of the model (e.g. the model is characterized by linear decision boundary while the true boundary is not linear)
- Variance = caused by the limited amount of training data (e.g. there are differences in the performance of two classifiers based on the same model but trained using different training data)



(a) bias



(b) variance

C. Aggarwal, Data Mining. The textbook, 2015

Motivation

Components of the error:

- Bias = caused by the limitations of the model (e.g. the model is characterized by linear decision boundary while the true boundary is not linear)
- Variance = caused by the limited amount of training data (e.g. there are differences in the performance of two classifiers based on the same model but trained using different training data)

$$MSE = \frac{1}{L}\sum_{i=1}^{L}(y_i - g(x_i;D))^2 = \frac{1}{L}\sum_{i=1}^{L}(y_i^2 - 2y_i g(x_i;D) + g(x_i;D)^2)$$

$$E_D(MSE) = \frac{1}{L}\sum_{i=1}^{L}(y_i^2 - 2y_i E_D(g(x_i;D)) + E_D(g(x_i;D)^2) + (E_D(g(x_i;D)))^2 - (E_D(g(x_i;D)))^2)$$

$$= \frac{1}{L}\sum_{i=1}^{L}((y_i - E_D(g(x_i;D)))^2 + (E_D(g(x_i;D)^2) - (E_D(g(x_i;D)))^2)$$

Squared bias

(related to the model)

Variance (of answers produced by the models inferred from all possible training datasets) (related to the data)

Motivation

Components of the error:

- Bias = caused by the limitations of the model (e.g. the model is characterized by linear decision boundary while the true boundary is not linear)
- Variance = caused by the limited amount of training data (e.g. there are differences in the performance of two classifiers based on the same model but trained using different training data)

Remarks:

- A model with high bias will consistently make errors even if the training data set is changed
- A model with high variance will produce inconsistent results when trained with different data sets

How to reduce the error?

- By reducing the bias and/or by reducing the variance
- Is it possible to reduce both of them? How?

Motivation

Bias vs variance

- Simple models (e.g. linear models, simple rules, shallow decision trees, naïve Bayes)
  - High bias (because of oversimplification of the decision boundary)
  - Small variance (they are robust with respect to the changes in the datasets; simple models do not overfit)

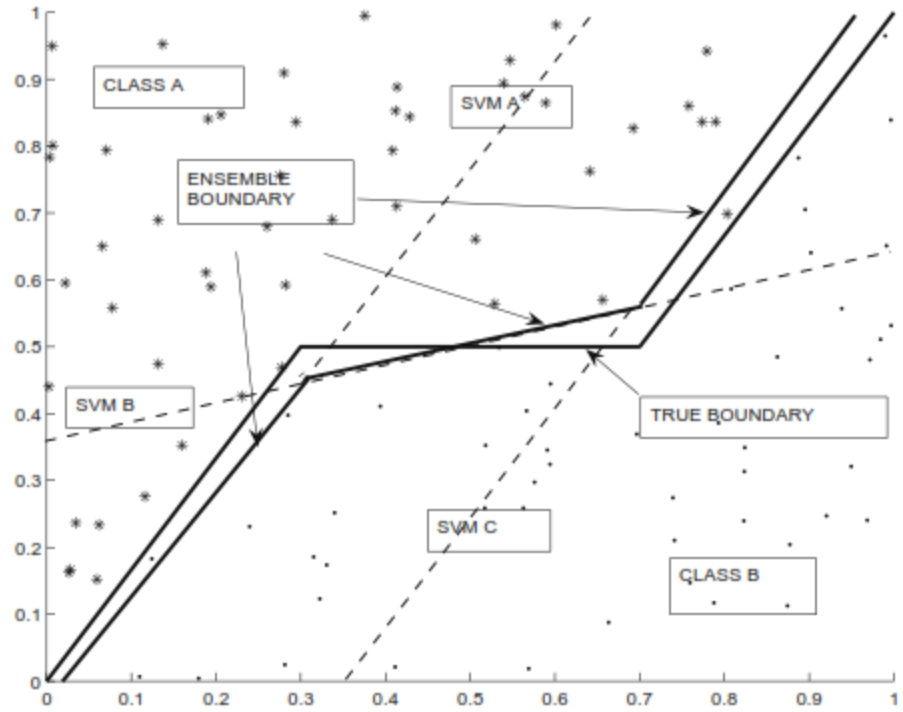
- Complex models (e.g. neural networks with many neurons/layers, deep decision trees)
  - Low bias (because they can model complex decision boundaries)
  - High variance (sensitive to data variation; prone to overfitting)

By using just one model it is necessary to identify a trade-off between bias and variance

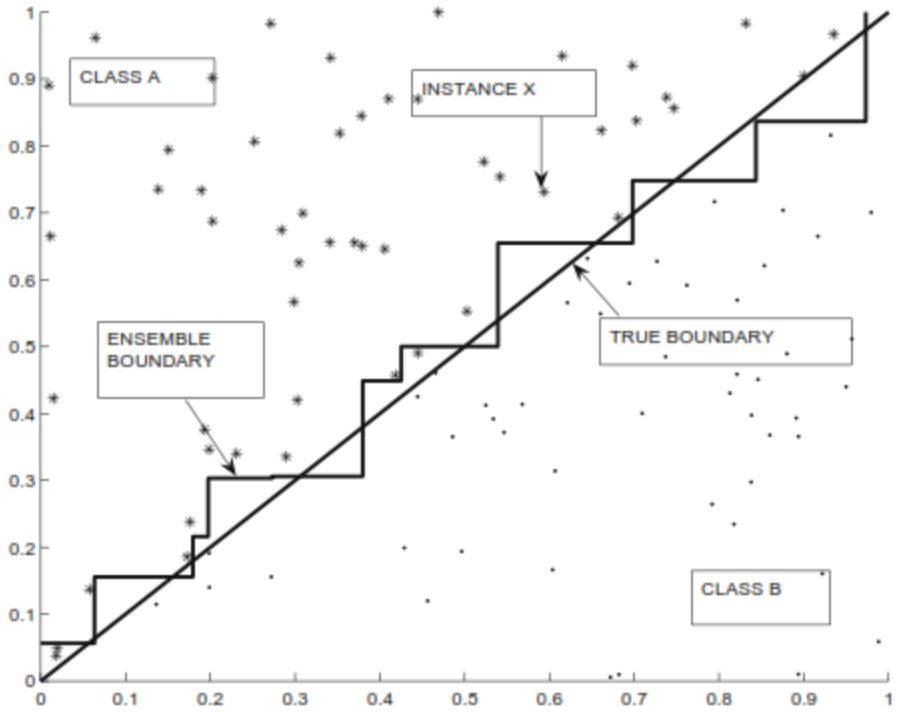
However, by combining several models one can reduce both the bias and variance

Motivation

Combining several models → ensemble model



(a) bias



(b) variance

By combining three linear SVMs one can generate nonlinear boundaries

By combining several decision trees trained on different datasets one can reduce the variance

C. Aggarwal, Data Mining. The textbook, 2015

Is it useful to combine models?

A simple probabilistic analysis

- Suppose there are 25 base classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$ (probability to make an error)
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction (in the case of an aggregation based on the majority rule):

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Ensemble models

How can be constructed the ensemble models?

- By generating different models (based on different structural assumptions) from the same training dataset - this corresponds to model-centered ensembles

- By training the same model based on different training datasets (usually extracted randomly from the full dataset) – this corresponds to data-centered ensembles

How are used the ensemble models?

- For a given input instance all models in the ensemble are applied and the final result is aggregated from the results provided by components through:
  - Voting (the most frequent answer) – in the case of classification problems
  - Averaging – in the case of regression problems

Ensemble models

A generic algorithm for ensemble models

- Input: a dataset D; set of methods/algorithms $\{A_1, A_2, \ldots A_r\}$
- Output: an ensemble model consisting of K individual models $\{M_1, M_2, \ldots, M_K\}$

REPEAT

- k=1
- select an algorithm A from the set $\{A_1, A_2, \ldots A_r\}$
- create a training dataset D_k (by sampling from D)
- construct the model M_k by using applying algorithm A to the dataset D_k
- k=k+1
- Evaluate the performance of the current ensemble $\{M_1, M_2, \ldots, M_k\}$ (for each model the data not included in the corresponding training set are used)

UNTIL desired performance

Ensemble models

A generic algorithm for ensemble models

- Input: a dataset D; set of methods/algorithms $\{A_1, A_2, \ldots A_r\}$
- Output: an ensemble model consisting of K individual models $\{M_1, M_2, \ldots, M_K\}$

Particular cases:

- Different algorithms, one training set (e.g. bucket of models)

- Same algorithm, different datasets
  - Bagging
  - Random forests
  - Boosting

Bucket of models

Main idea: several algorithms, one dataset → an aggregated meta-model

Variant 1:

- use several models trained for the same dataset
- aggregate the results of the component models by
    - Majority based voting
    - Averaging the results of individual models

Variant 2:

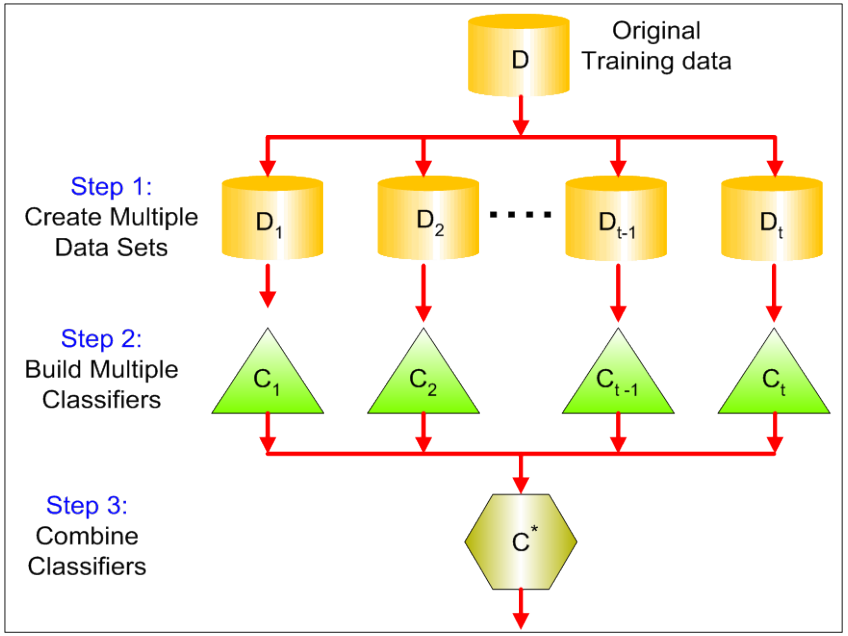
- Divide the dataset D in two subsets A and B
- Train all models using subset A
- Select the model with best behaviour on subset B
- Retrain the selected model on the entire dataset D

Remark:

- It might reduce the bias as for different parts of the dataset one can have different algorithms most appropriate

Bagging

- Main idea: one algorithm, several training datasets → several classifiers



Training datasets:

- Sampling with replacement from the full dataset D
- If the full data set has L elements then the probability that a given instance is selected is $1-(1 - 1/L)^L$;

- Example:

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

[Slides by Kumar/ Introduction to Data Mining, 2004]

Bagging

Impact of bagging:

- It reduces the variance
- It does not reduce the bias (as the same model is used for all training datasets thus its inherent limitations are not removed)

Remark:

- The reduction in the variance is ensured if the models composing the ensemble are independent
- An idea to limit the correlation between models is to introduce randomness → random forests

Random forests

Random forest = collection of random trees constructed based on a bagging approach (by using training datasets based on random sampling with replacement)

Construction of a random forest:

- construct a random tree for each training dataset

Usage of a random forest:

- use each tree in the forest for the data instance to be classified
- select the dominant resulting class (by a simple voting scheme)

Random forests

Random tree = decision trees constructed by using random-split

Main steps:

- If the number of cases in the training set is L, sample L cases at random - but *with replacement*, from the original data. This sample will be the training set for growing the tree.
- If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
- Each tree is grown to the largest extent possible (in order to ensure low bias). There is no pruning.

[Leo Breiman
https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm]

Random forests

Remark: the forest error rate depends on two things:

- The correlation between any two trees in the forest. Increasing the correlation increases the forest error rate.

- The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

[Leo Breiman https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm]

Random forests

Remark: Influence of m (the number of attributes selected during the splitting process)

- Reducing m reduces both the correlation and the strength.
- Increasing m increases both the correlation and the strength
- Somewhere in between is an "optimal" range of m

Remark: instead of cross-validation, in the case of random forests one can use the estimation of error based on the “out-of-bag” data (the data from the training set which have not been selected by sampling with replacement)

- Using the “out-of-bag” error rate, an adequate value of m can be identified

[Leo Breiman https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm]

Boosting

Main idea:

- Each instance in the training set has a weight which can be used
  - Either directly in the model (if it allows working with weighted instances)
  - Or in defining selection probabilities
- The weights are adaptive (they are increased for instances which are wrongly classified)

Approach:

- Initially, all instances in the dataset have equal weights
- During the iterative training process:
  - The instances that are wrongly classified will have their weights increased
  - The instances that are classified correctly will have their weights decreased

Remark: It is supposed that the main component of the error is the bias and such an approach tries to reduce the bias on the instances which are wrongly classified

AdaBoost

Training algorithm:

- Input: Base classification algorithm: A; training dataset: D
- Output:
  - set of classification models (M_1,…, M_T) constructed during the adaptive steps
  - set of weights corresponding to the models

- Main idea

- at each step t of the algorithm, one obtains a component of the ensemble (M_t) – see next slide
- the base classifier is usually a weak classifier

AdaBoost

AdaBoost (A,D)

t=1; initialize the weights of the training instances: w(t,i)=1/L for all i=1..L

REPEAT

t=t+1; construct M_t using the current values of the instances' weights

compute the weighted error rate of model M_t on D ($\varepsilon(t)$)

compute the model weight $\alpha(t)=\ln((1-\varepsilon(t))/\varepsilon(t))/2$

FOR i=1,L **DO**

IF x_i is wrongly classified **THEN** $w(t+1,i)=w(t,i)*\exp(\alpha(t))$

ELSE $w(t+1,i)=w(t,i)*\exp(-\alpha(t))$

FOR i=1,L **DO** w(t+1,i)=w(t+1,i)/sum(w(t+1,j),j=1..L)

UNTIL (t>=T) or ($\varepsilon(t)$=0) or ($\varepsilon(t)$>=0.5)

Remark: If any intermediate rounds produce error rate higher than 50%, instead of stopping the algorithm one can revert the weights back to 1/L and repeat the resampling procedure

AdaBoost

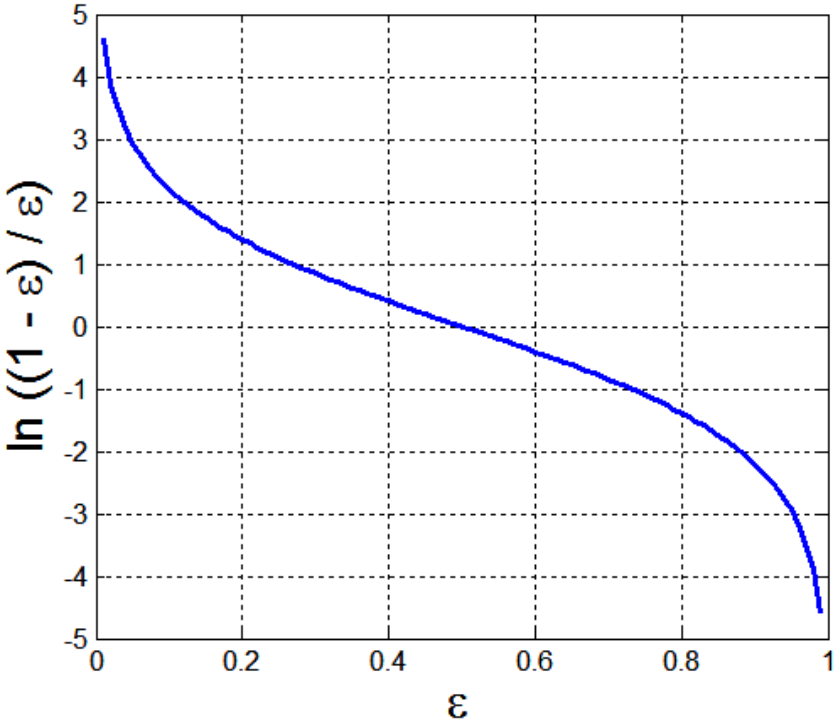
Some details

- Weighted error rate:

$$\varepsilon_t = \frac{1}{L}\sum_{i=1}^{L} w_i \delta\left(M_t(x_i) \neq y_i\right)$$

- Importance (weight) of a model/ classifier:

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



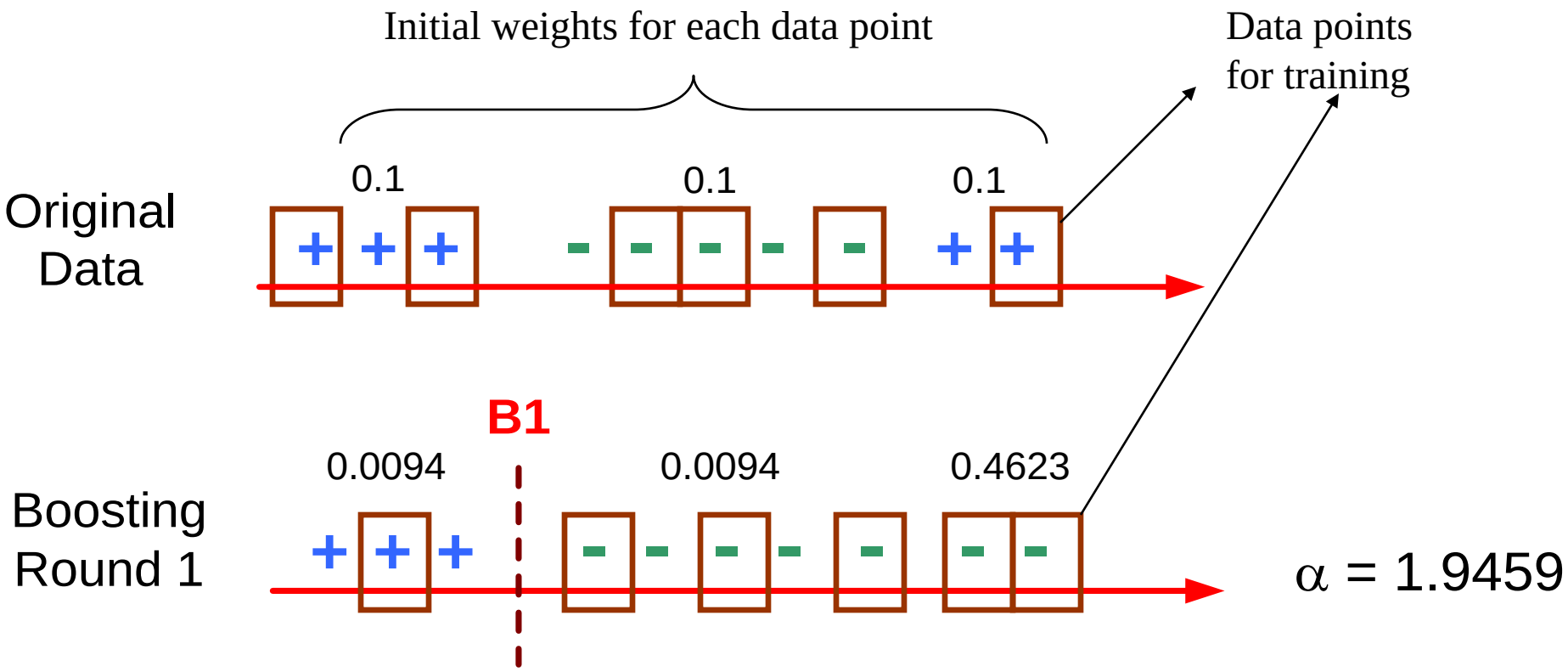
AdaBoost

Classification step

(in the case of a binary classifier producing outputs in {-1,1})

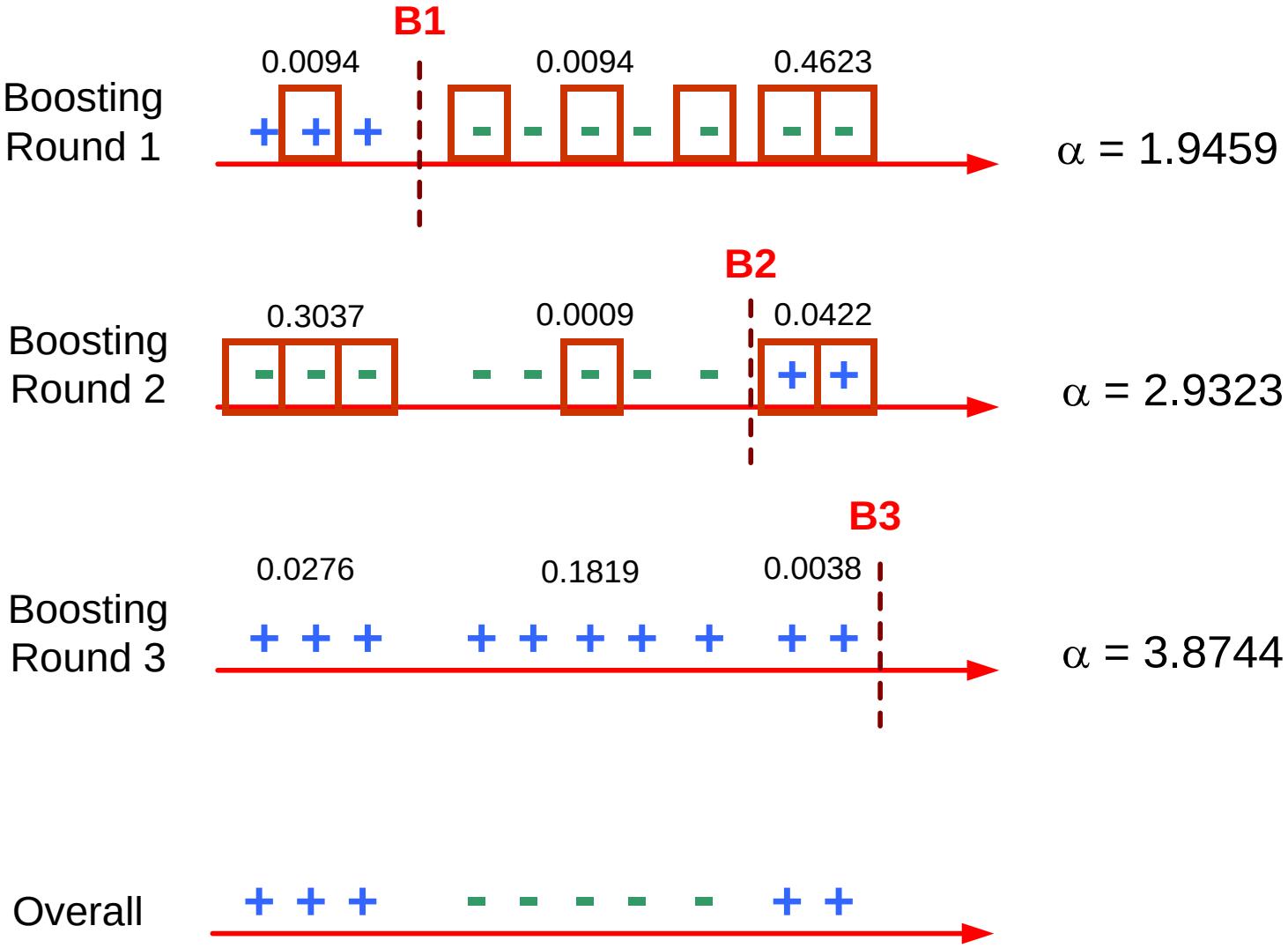
- Apply each of the ensemble components (M_1, M_2,…, M_T) and collect the results (r_1,r_2,…r_T) (in {-1,1})
- Aggregate the results:
  - compute $r=\alpha_1 r_1 + \alpha_2 r_2 +\ldots+ \alpha_T r_T$
  - IF r<0 THEN return -1 ELSE return +1

AdaBoost



[Slides by Kumar/ Introduction to Data Mining, 2004]

AdaBoost


B1
0.0094
0.0094
0.4623
Boosting Round 1
α = 1.9459
B2
0.3037
0.0009
0.0422
Boosting Round 2
α = 2.9323
B3
0.0276
0.1819
0.0038
Boosting Round 3
α = 3.8744
Overall

Stacking

Main idea: two levels of classification

Main steps:

- Divide the training dataset D in to subsets A and B
- First level: train an ensemble of k classifiers based on A (it could be a bucket of heterogeneous classifiers, it could be based on bagging or on k rounds of boosting)
- Second level:
  - Determine the k outputs (as class labels) of the classifiers trained at the first level for each of the data instances from the subset B
  - Construct a new dataset having as input attributes these k outputs and as class attribute the true label corresponding to the instance from subset B.
  - Train a classifier based on this new dataset

Stacking

Remarks:

- The result of stacking is a set of k first level classifiers and a combiner classifier
- For a test instance, the first level classifiers are used to create a new k-dimensional instance while the second classifier provides the output result based on transformed instance
- The original attributes can be combined with the new k attributes when the second level classifier is constructed; also it is possible that the new k attributes generated at the first level are probabilities, not class labels
- The stacking approach is able to reduce both bias and variance, since the second level combiner learns from the errors of different ensemble components.

Summary

Impact of ensemble methods on the components of the error

- Bagging and random forests are designed to reduce the variance
- Boosting and stacking are designed to reduce both the variance and the bias

Extension of the ensemble methods idea to clustering

- Same idea as for classification:
  - Apply different clustering methods (or the same method but with different values of the parameters)
  - Aggregate the results by using clustering algorithms for hypergraph (as each data represents a node and each clustering represents a hyperedge)