DATABASE SYSTEMS GROUP

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

LMU

Knowledge Discovery in Databases II

Winter Term 2015/2016

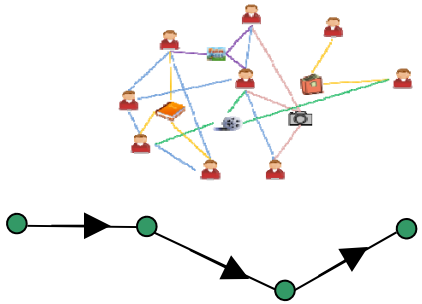
Lecture 10 & 11:
Variety: Multi-view data and Ensemble learning

Lectures : Dr Eirini Ntoutsi, PD Dr Matthias Schubert
Tutorials: PD Dr Matthias Schubert

http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_II_(KDD_II)

The (data) variety aspect

- So far, data is given by a set of feature vectors.
- In general though, data is structured and has links.
  - Graph data (social nets, authorship graphs, protein-interaction networks...)
  - Tree structures (XML documents, sensor networks,...)
  - Sequences and trajectories
  - Team compositions
  - Websites composed of webpages, composed of ...
  - Combinations of color, form, texture features for images



Core questions:

- Is the structure important for the description of the data?
- How can we apply data mining algorithms to structured data?
- Can we combine different views on the same data object to get better results?
- Does structured data yield additional data mining tasks?

Types of structured data objects

- Multi-view data:
  - Data is described by different feature sets or “views“.
  - E.g., an image can be described by its visual information and its textual tags
- Multi-instance data:
  - Each object is described as a set of objects from the same domain.
  - E.g., a team described by its players.
  - E.g., an image is described by a set of local feature descriptors
- Linked data or Graph data:
  - Objects may reference to other objects.
  - E.g., social graph data

Further cases not being discussed in the lecture:

- Sequential data: multi-instance data having a strict order
- Temporal data: Describes the same object over a time-period (time series, trajectories)
- Tree-structured data: acyclic graph data, describing hierarchies

Types of structured data objects

- Multi-view data:
  - Data is given by multiple object descriptions (records, objects in programming).
- Multi-instance data:
  - Each object is described as a set of objects from the same domain (arrays, lists, sets in programming).
- Linked data or Graph data:
  - Objects may reference to other objects (graph structured data, network data, …).

Further cases not being discussed in the lecture:

- Sequential data: multi-instance data having a strict order
- Temporal data: Describes the same object over a time-period (time series data, trajectories)
- Tree-structured data: acyclic graph data, describing hierarchies

Dealing with data variety

In general, there are three ways to handle data variety:

1. Transform data into a simpler format
   - Useful information about the structure is preserved in special features
   - Transformation from more complex to simpler descriptions (e.g. Multi-instance to vector, Graph to Multi-instance, ..)
2. Employ distance/ similarity measures for structured data
   - Having a distance/similarity function allows for the use of many data mining algorithms.
   - Comparing complex descriptions is usually more complex.
3. Employ specialized data mining algorithms
   - Especially in cases where the task is not standard (e.g., centrality in graphs)
   - Often solutions involve a workflow of several data mining tasks

Data variety topics covered in the course

- Ensemble learning and Multi-view object descriptions
- Multi-instance Data Mining
- Mining graph-structured objects
- Graph and link mining

Multi-view data mining and Ensembles

In general: Having more than one view on the same set of objects can be exploited to learn better results.

⇒ Ensemble theory: Learn better models by combining multiple base-learners.

⇒ Multi-View data mining: In most cases, specialized applications of Ensemble learning

⇒ Well established in application domains such as bioinformatics and multi-media retrieval.

Ensembles and Multi-View Data

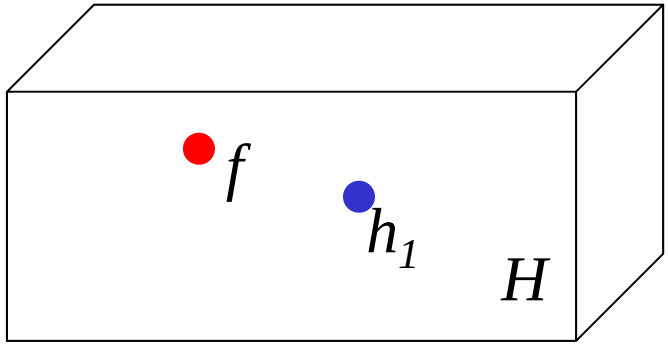
1. Introduction and Basic Principles of Ensemble Learning
2. Diversity in Ensemble Learning
3. Methods for Ensemble Construction
4. Mining Multi-View Data

Introduction to Ensemble Learning

- Input:
  - A dataset containing instances X from a data space D.
  - A set of classes C={c_i}
  - Each element x $\in$ X belong to class $c_i \in$ C.
- There is a function *f: D* $\rightarrow$ *C*, describing the connection between *x* and class c_i.
- The task of classification is to determine *f*.
- In general, learning algorithms compute an approximation of *f* which does not hold completely. They learn a hypothesis.
- A classifier is an *hypothesis* about the true function *f*

Introduction to Ensemble Learning

- The "true" function f is unknown.
- There is a labeled set of tuples $(x, c_i) \in f \subseteq D \times C$ (training set)
- A learning algorithm now determines the hypothesis h_i as classifier from the hypothesis space $H \subseteq D \times C$ which fits best to the training set.



- Caution: the "true" f does not need to be contained in H.

Introduction to Ensemble Learning

- A classifier (a learned hypothesis h) can be applied to all $x \in D$ to predict the $c_i = f(x)$

- The accuracy is the relative frequency of correct predictions.

$$Acc(h) = P(h(x)=f(x))$$

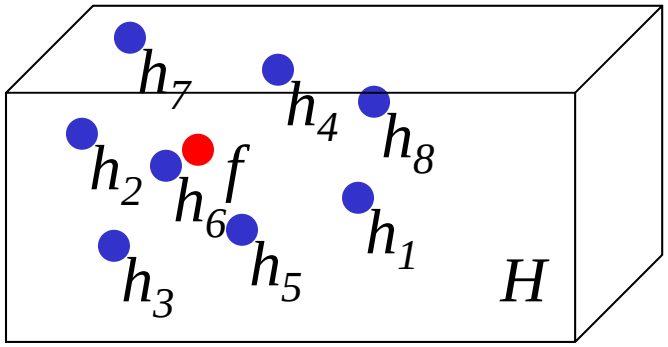
- Correspondingly, the classification error is the complement:

$$Err(h) = P(h(x)\neq f(x)) = 1 - Acc(h)$$

Introduction to Ensemble Learning

- Core idea of ensemble learning: Asking multiple “experts“ (classifiers) can avoid mistakes.

- From a mathematical point of view: building the average over multiple functions can smooth the decision surface.



Introduction to Ensemble Learning

- A simple decision rule for two classes $C=\{-1,1\}$:
  - Generate a set of hypotheses $\{h_1,\ldots,h_k\}$ and corresponding weights$\{w_1,\ldots,w_k\}$.
  - An ensemble-classifier $\hat{h}$ is defined by the following decision function:

$$\hat{h}(x) = \begin{cases} w_1 h_1(x) + \ldots + w_k h_k \geq 0 \rightarrow 1 \\ w_1 h_1(x) + \ldots + w_k h_k < 0 \rightarrow -1 \end{cases}$$

- often $w_1=\ldots=w_k=1$ (unweighted combination).
- Weights can by used to express the reliability of the classifiers
- More complex decision rules might be used, especially when having more than two classes

  $\rightarrow$ there is a large variety of ensemble-methods

Rationale for ensemble learning

- Let an ensemble of k=25 binary classifiers, each with an error rate e=0.35.
- Let majority voting defines the ensemble's decision
- What is the ensemble error?
  - If the base classifiers are identical, e=0.35
  - If they are independent, i.e., their errors are uncorrelated, the ensemble makes a wrong prediction if more than half of the base learners make a wrong prediction.
- The error rate of an ensemble learner is given by the frequency of the cases where at least half of the base classifiers are wrong:

$$Err(\hat{h}) = \sum_{i=\left\lceil \frac{k}{2} \right\rceil}^{k} \binom{k}{i} e^i (1-e)^{k-i}$$

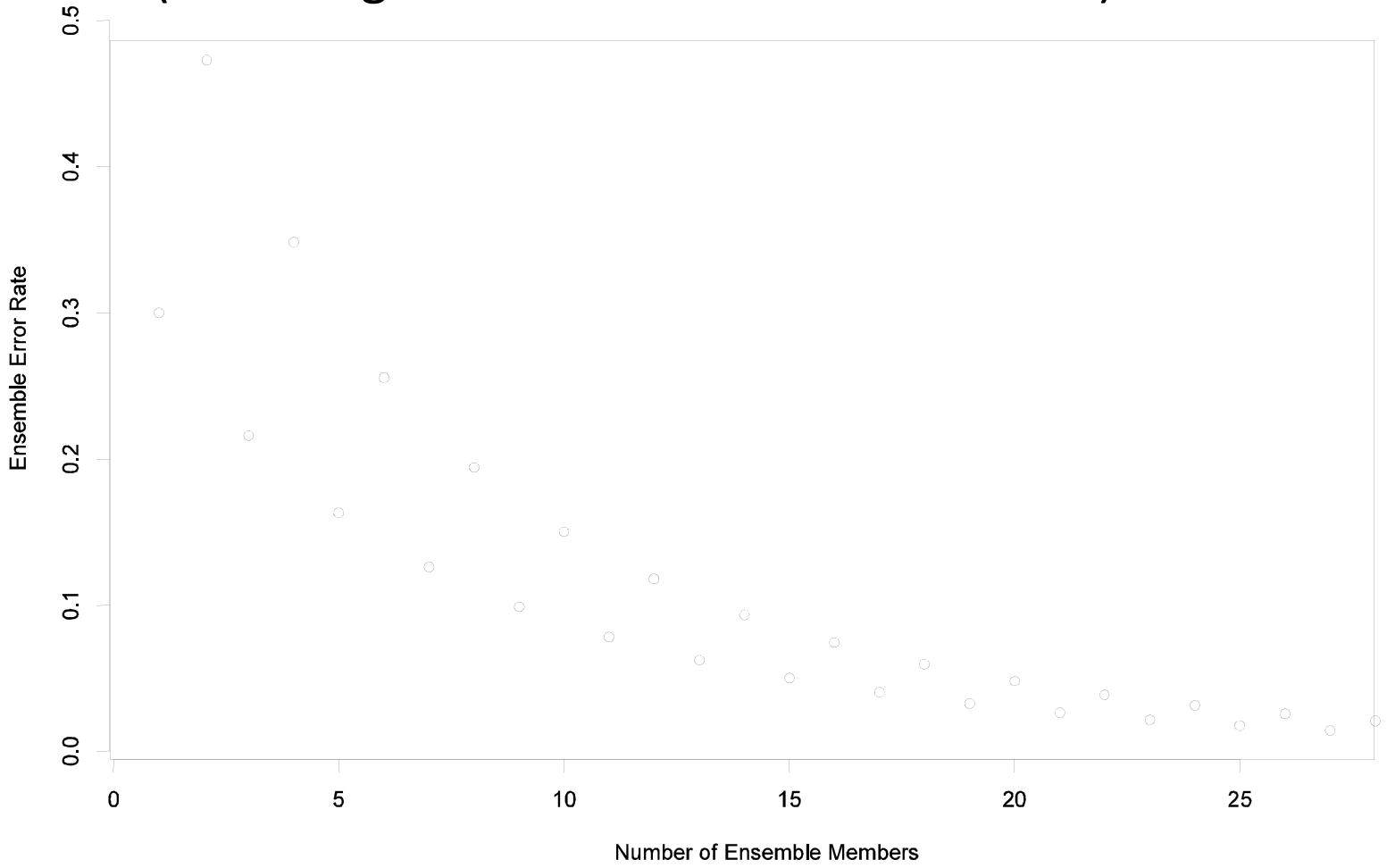
(Assumption here: $Err(h_1)=\ldots=Err(h_k)=e$)

  - it depends on the error-rate of its base-classifiers *e* and their amount *k*.

- In our example: $Err(\hat{h}) = \sum_{i=13}^{25} \binom{25}{i} e^i (1-e)^{25-i} = 0.06$

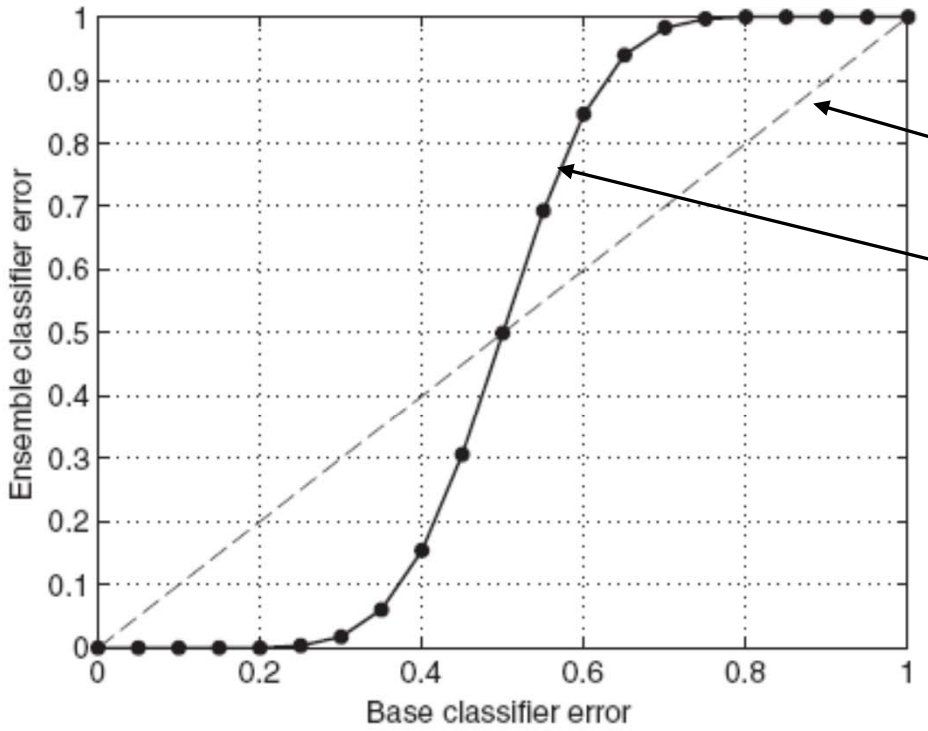
Introduction to Ensemble Learning

Dependency of the ensemble error rate on the number of base-learners (assuming a constant error rate of e= 0.3)



Introduction to Ensemble Learning

- Error rate of an ensemble of 25 binary classifiers, for varying error rates (*e*) of the base-classifiers :



Identical base learners

Independent base learners

!!! There is no improvement with identical base-learners

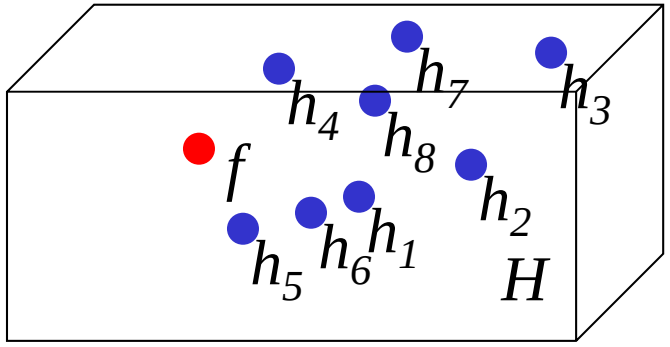
!!! If e>0.5, the ensemble performs worse than the base learner!

(from: Tan, Steinbach, Kumar: Introduction to Data Mining)

Introduction to Ensemble Learning

- Requirement for an improvement: The errors of each classifier are independent.

$$Err(\hat{h}) = \sum_{i=\left\lceil \frac{k}{2} \right\rceil}^{k} \binom{k}{i} e^i (1-e)^{k-i}$$



- if the base classifiers are too similar, they are making the same mistakes => no improvement

Introduction to Ensemble Learning

- Conclusion:

> Required conditions for an improved accuracy:
>
> 1. Base classifiers are sufficiently “accurate”.
> 2. Base classifiers are “diverse”.

- Accuracy: better than random predictions
- Diversity: no correlations or, at least no strong correlations between the predictions of the base-learners
- Is it possible to optimize both simultaneously?

Ensembles and Multi-View Data

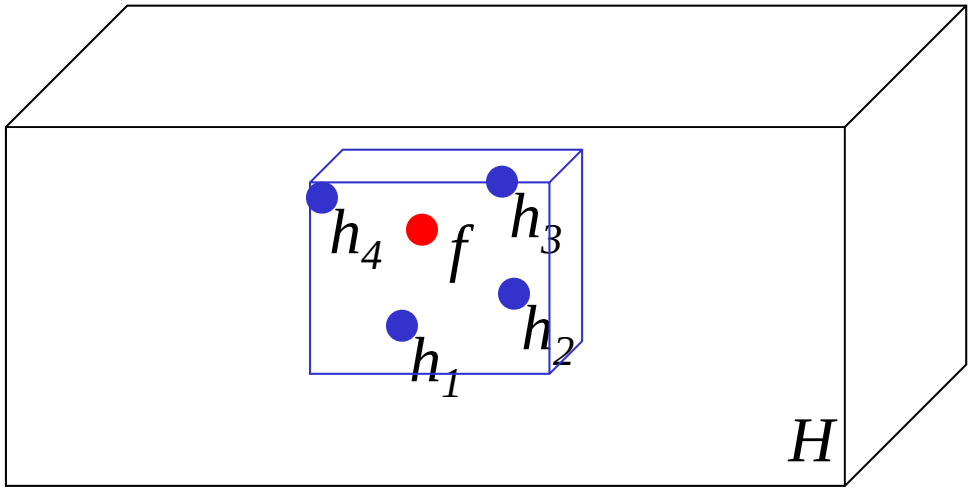
1. Introduction and Basic Principles of Ensemble Learning
2. Diversity in Ensemble Learning
3. Methods for Ensemble Construction
4. Mining Multi-View Data

Aspects of Diversity

- There three fundamental reasons that allow us to construct very good ensembles:
  - Statistical Variance
  - Computational Variance
  - Representation

Aspects of Diversity

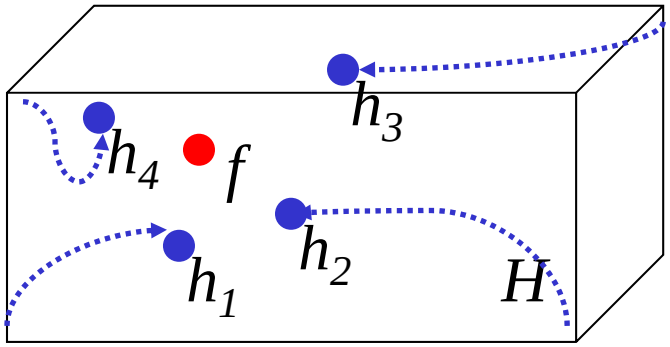
- Statistical Variance:
  - The number of potential hypotheses is to big to determine the best one based on a limited sample set.



  - Combining multiple hypotheses reduces the risk to make a large mistake

Aspects of Diversity

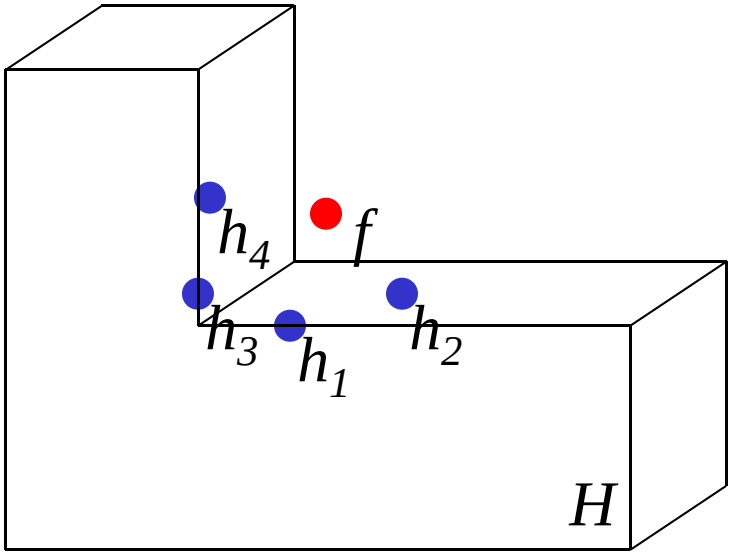
- Computation Variance:
  - Some learning algorithm cannot guarantee, to find the hypothesis fitting best to the training data due to the complexity of the learning algorithm
  - Even for cases where exist enough data (the statistical problem is absent)
  - For example, it is common to use heuristics computing local optima in case computation is too expensive.



  - Combination of multiple hypotheses (by running the local search by many different starting points) reduces the risk to take the wrong local optima of an error function.

Aspects of Diversity

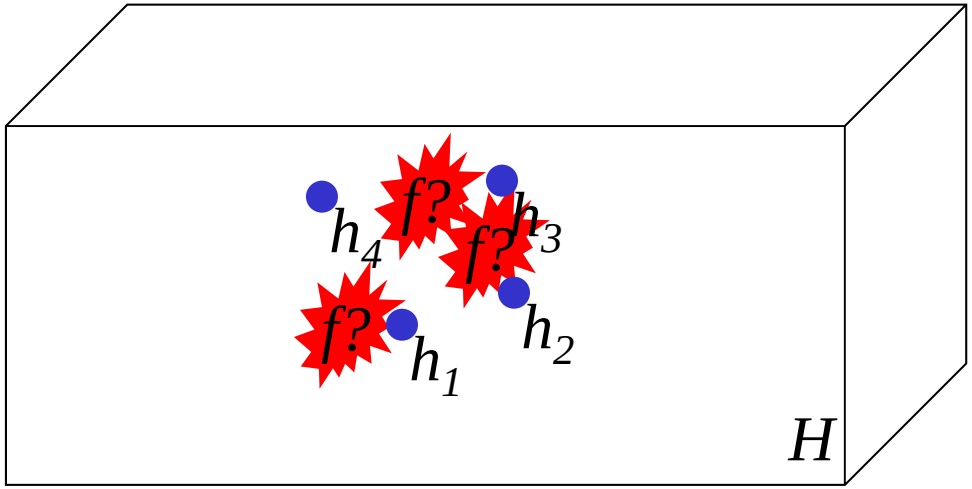
- Representation :
  - The space of representable hypotheses might not contain a good approximation of the “true” function f .



  - Combining multiple hypotheses can expand the space of representable hypotheses.

Aspects of Diversity

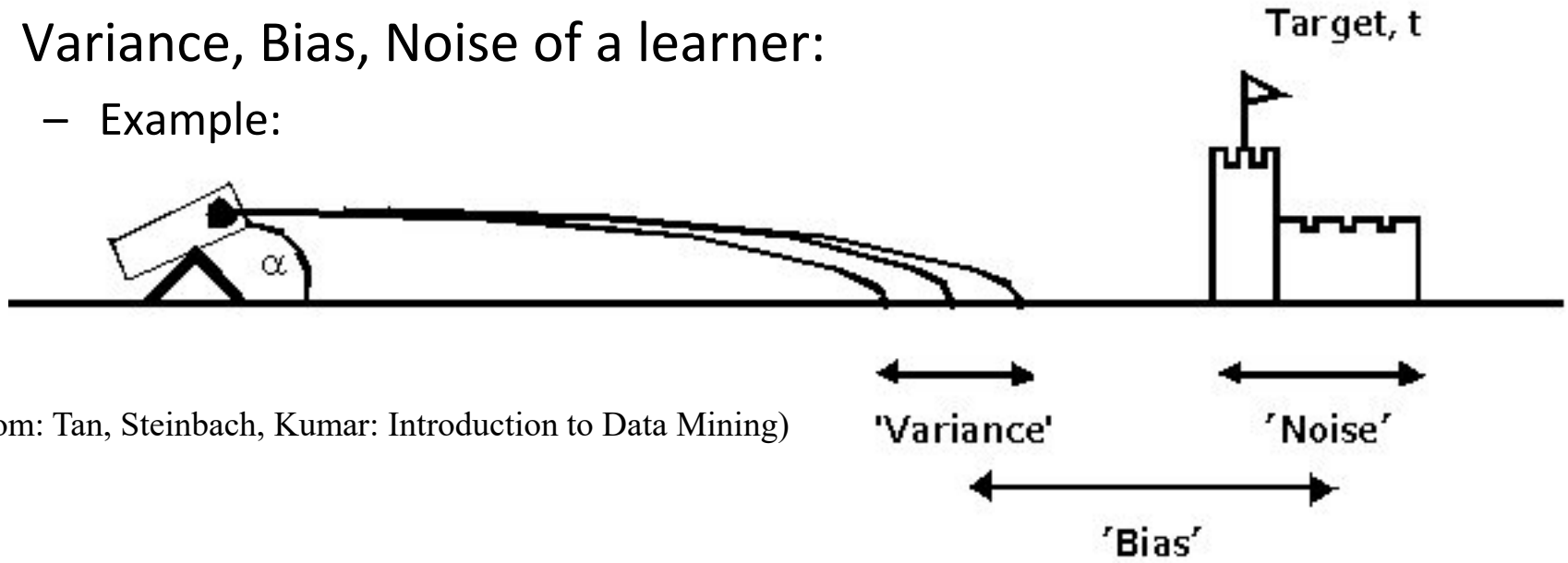
- Fuzzy target functions:
  - The training samples do not allow clear conclusions about the target function (e.g., training samples might be contradictive).



  - Combining multiple hypotheses reduces the risk to approximate a wrong hypothesis.

Aspects of Diversity

- Variance, Bias, Noise of a learner:
  - Example:



(from: Tan, Steinbach, Kumar: Introduction to Data Mining)

  - Variance, Bias and Noise represent different types of errors

$$err = Bias_{\alpha} + Variance_{f} + Noise_{t}$$

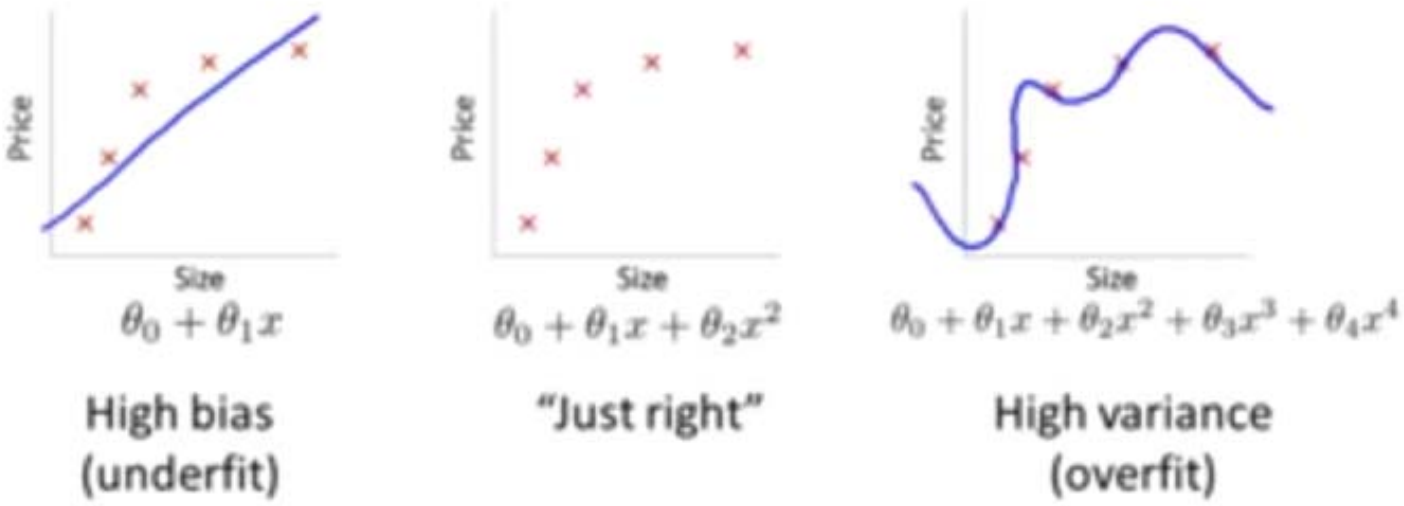
  - $Variance_f$: depends on the employed force *f*
  - $Noise_t$: the target is not stationary
  - $Bias_\alpha$: depends on the angle of the canon

Bias-Variance decomposition 1/2

- A classifier's error can be analyzed w.r.t. variance, bias, noise:
  - Error due to bias:
    - The amount by which the expected model prediction differs from the true value, over the training data.
    - Introduced at model selection: Represents the assumptions on the design choices of the classifier (e.g. linear separable, independent attributes,..).
    - High bias → underfit
    - "Bias-free learning is futile": A core part of learning is to abstract the observations within a model →learners are based on this model
  - Error due to variance:
    - It shows how sensitive the algorithm is to the training data
    - Variance is high if different training sets raise different classifiers.
    - High variance → overfit
  - Error due to noise:
    - The class for some of the training objects is not clearly defined or ambiguous.
    - E.g., instances with same values different class labels

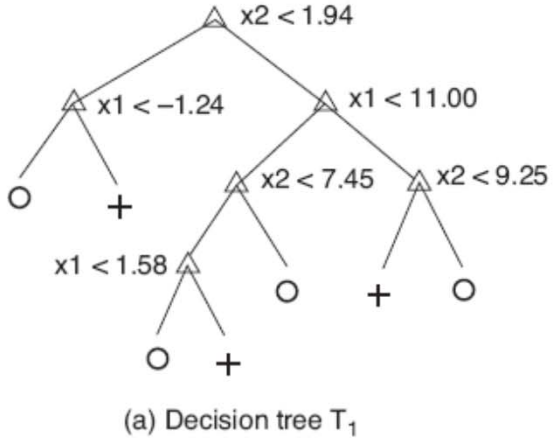
Bias-Variance decomposition 2/2

- Ideally, we want to choose a model that captures the regularities in the training data, but also generalized well to unseen data.
- Example from Andrew Ng, Machine Learning course, lecture 42
  - https://www.youtube.com/watch?v=4d3_VJqWaV0

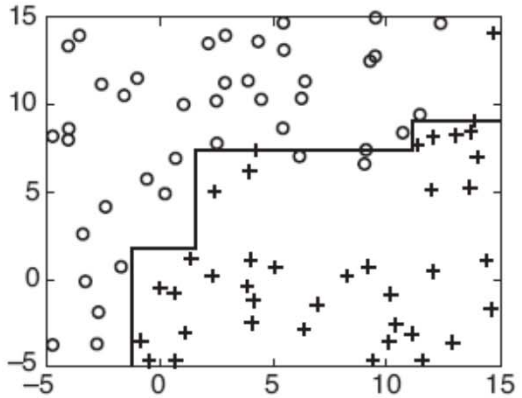


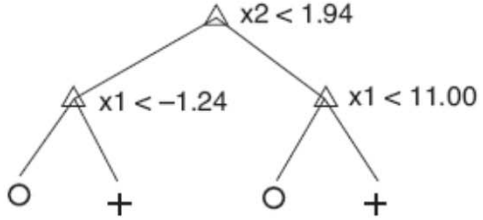
Example of bias 1/4

- Decision Trees of different complexities:



(a) Decision tree T_1





(b) Decision tree T_2

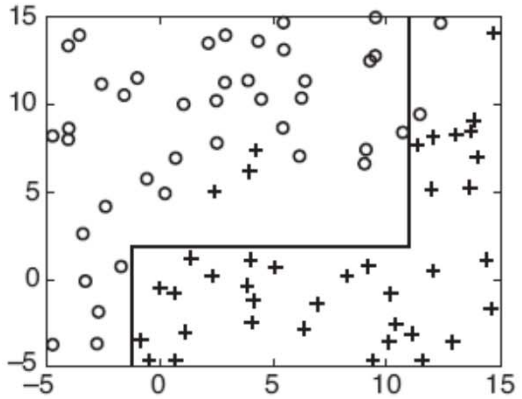


Figure 5.33. Two decision trees with different complexities induced from the same training data.

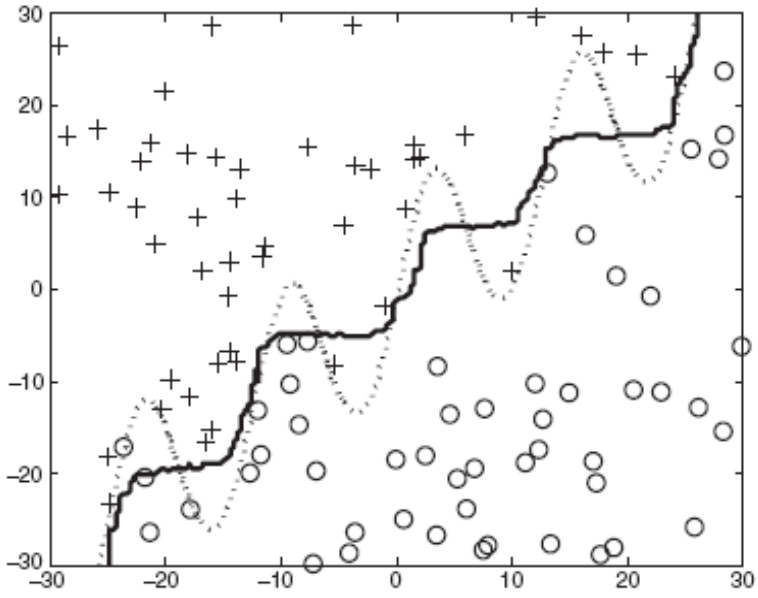
(from: Tan, Steinbach, Kumar: Introduction to Data Mining)

Example of bias 2/2

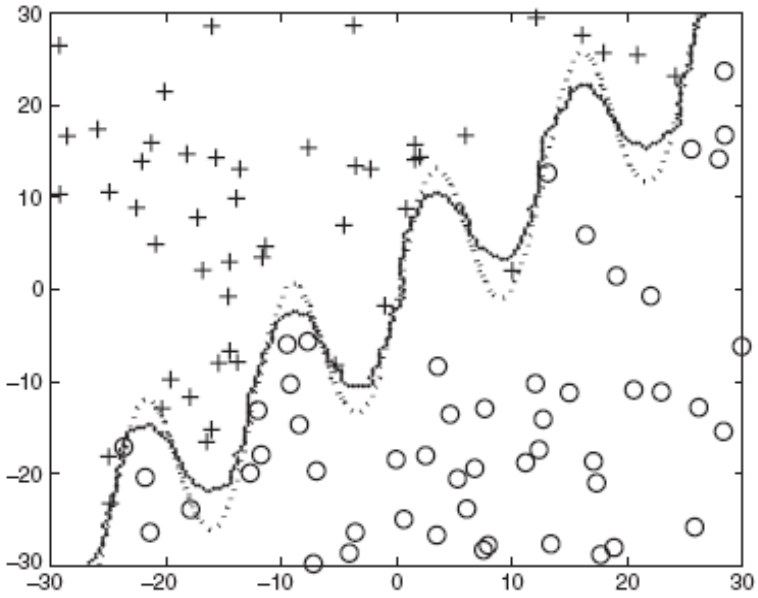
- Decision Trees as an example for bias:
  - T_1 and T_2 were trained on the same data
  - T_2 was generated from T_1 by pruning it to the maximal height of two
  - T_2 uses stricter assumptions about class separation (strong bias)

Example of bias 3/4

- The relative contribution of bias and variance to the complete error depends on the classifier type, e.g., DTs vs 1NN.



(a) Decision boundary for decision tree.



(b) Decision boundary for 1-nearest neighbor.

Figure 5.34. Bias of decision tree and 1-nearest neighbor classifiers.

(from: Tan, Steinbach, Kumar: Introduction to Data Mining)

Example of bias 4/4

- Example:
  - decision boundary for each classifier by averaging the models induced from 100 training sets, each containing 100 objects
  - dashed : true class border being used by the data generator
- The difference between the true border and the avg one reflects the bias of the classifier
  - 1-NN classifier has overall smaller distance to the class border

    ➔less bias
- But, the 1-NN classifier is more sensitive to the composition of the training set
  - the 100 1-NN classifiers display a larger variability in their decision boundaries

    ➔ larger variance

Ensembles and Multi-View Data

1. Introduction and Basic Principles of Ensemble Learning
2. Diversity in Ensemble Learning
3. Methods for Ensemble Construction
4. Mining Multi-View Data

Overview of the ensemble construction process

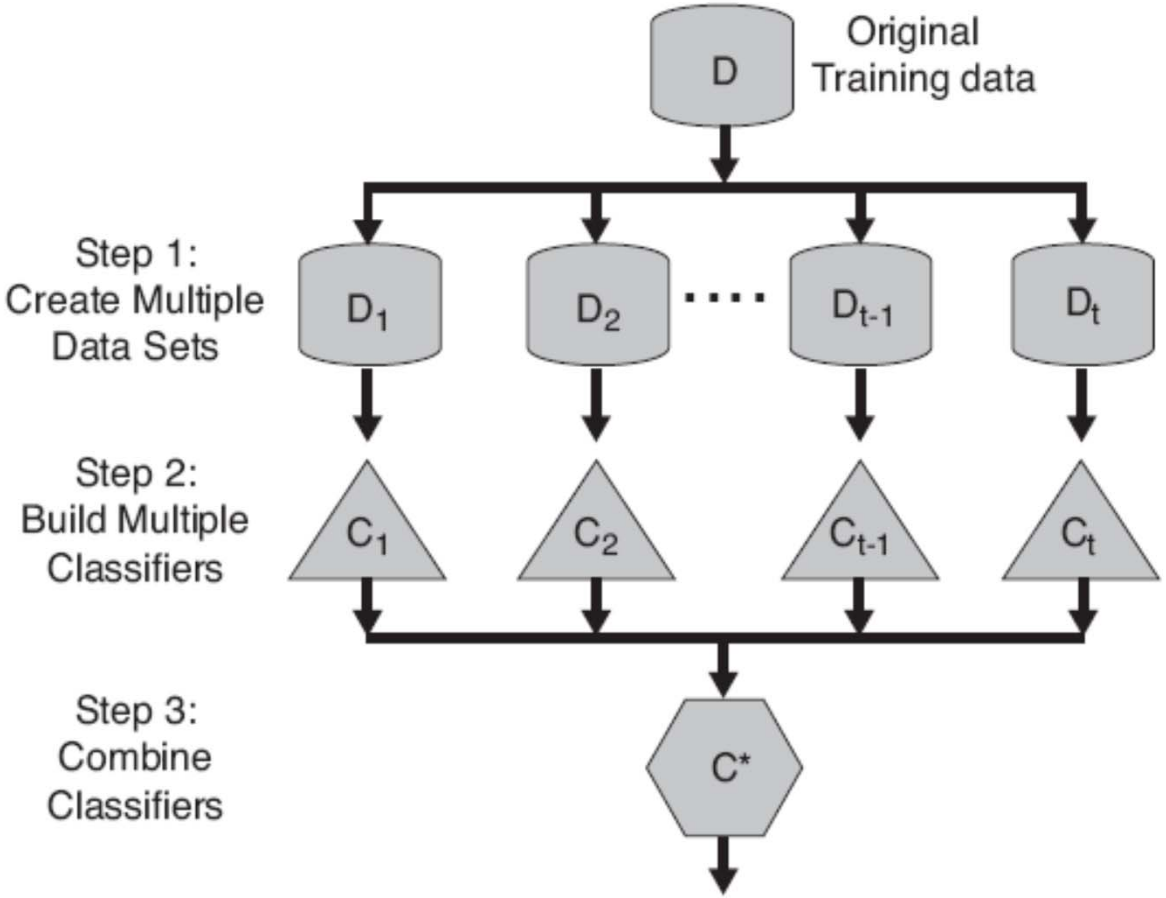


Figure 5.31. A logical view of the ensemble learning method.

(from: Tan, Steinbach, Kumar: Introduction to Data Mining)

Methods for Ensemble Construction

- How can we a achieve diversity of base learners?
  - Vary the training sets
    - Methods: Bagging and Boosting
  - Manipulate the input features
    - Learn on varying subspaces
    - Use multi-view data
  - Manipulate the class labels
    - various methods for mapping multi-class to 2-class problems
  - Manipulate the training algorithms
    - introduce randomness
    - employ varying initial models

Varying the training set

- Idea: create multiple training sets by resampling the original data
  - Approaches: bagging, boosting
- An important property of a learning algorithm is stability
  - The more stable a learner is, the less different are the classifiers on different training sets for the same classification task.
  - Unstable learners may strongly change even under small modifications of the training sample.

    ➔they are more suitable for ensemble learning
  - Examples of unstable learners:
    - Decision trees
    - Neural networks
    - Rule-based learners

Varying the training set: Bagging [Breiman96]

- Bootstrap: sampling with replacement
  - Each bootstrap sample D_i has the same size (n objects) as the original dataset
  - Some instances might appear >1 times, others might be omitted
  - On average, D_i contains 63% of the original training data
  - Each instance has a prob 1/n of being chosen and 1-1/n of not being chosen.
  - After making n draws each instance is not contained once with likelihood of

$$\left(1-\frac{1}{n}\right)^n$$

    - for large *n*'s $\left(1-\frac{1}{n}\right)^n \approx e^{-1} \approx 0.368$

  - therefore, the method is also called "0.632 bootstrap"

Bagging

- Bagging (Bootstrap Aggregating): build varying training sets by multiple bootstraps over the original dataset

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

- Bagging aggregates these bootstraps and trains a classifier on each of them
  - Using unstable methods results in multiple varying classifiers
- The final classifier is combined by a simple majority vote

Bagging example 1/2

- Training set

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Without bagging (singleton model): accuracy ~ 70%
- 10 bagging rounds and their models
- Decision stump models (1-level binary DTs)

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ==> y = 1
x > 0.3 ==> y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ==> y = -1
x > 0.05 ==> y = 1

Figure 5.35. Example of bagging.

(from: Tan, Steinbach, Kumar: Introduction to Data Mining)

Bagging example 2/2

- We classify the original data using the ensemble members
- The ensemble perfectly classify all the results

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

Figure 5.36. Example of combining classifiers constructed using the bagging approach.

(from: Tan, Steinbach, Kumar: Introduction to Data Mining)

More on bagging

- It enhances the representation of the target function
  - In our example: from decision stumps in the base learners to 2-depth DTS in the ensemble.
- Less susceptible to model overfitting when applied to noisy data
  - It doesn't focus on any particular instance in the training set
  - All instances have equal probabilities of being selected
- Bagging reduces the *variance* of the base learners by averaging
  - If the base learner is unstable, bagging helps to reduce the error associated with random fluctuations of the training data
- Bagging has little effect on *bias*
  - If the base learner is stable, the error in the ensemble is primary caused by bias in the base learners.

Varying the training set: boosting

- Boosting:
  - While bagging draws from a uniform distribution, boosting employs a weighted distribution.
  - Instances which are hard to classify are weighted higher in the next round
  - How weighting is used:
    - higher weights increase the likelihood of selection in the next bootstrap ➔ difficult examples appear more often in the next bootstrap
    - Can be used by the base learner to learn a model that is biased towards higher-weight instances

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

  - The ensemble is an aggregation of the base classifiers obtained from each boosting round.

Adaboost [FreSch95]

- Given a training set D of m instances $(X_1, y_1), \ldots, (X_m, y_m)$

- Initially, all instances have the same weight: $1/m$
- A weak learner is trained and its error is computed
- The weights are updated based on the weak learner's errors
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- The new weights are used in the next round

- The final decision is a linear combination of the weak learners decisions; the decision of each weak learner is weighted by its error.

Adaboost pseudocode

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

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

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

Error of classifier M_t

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

the weight of classifier M_t

- Update:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

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

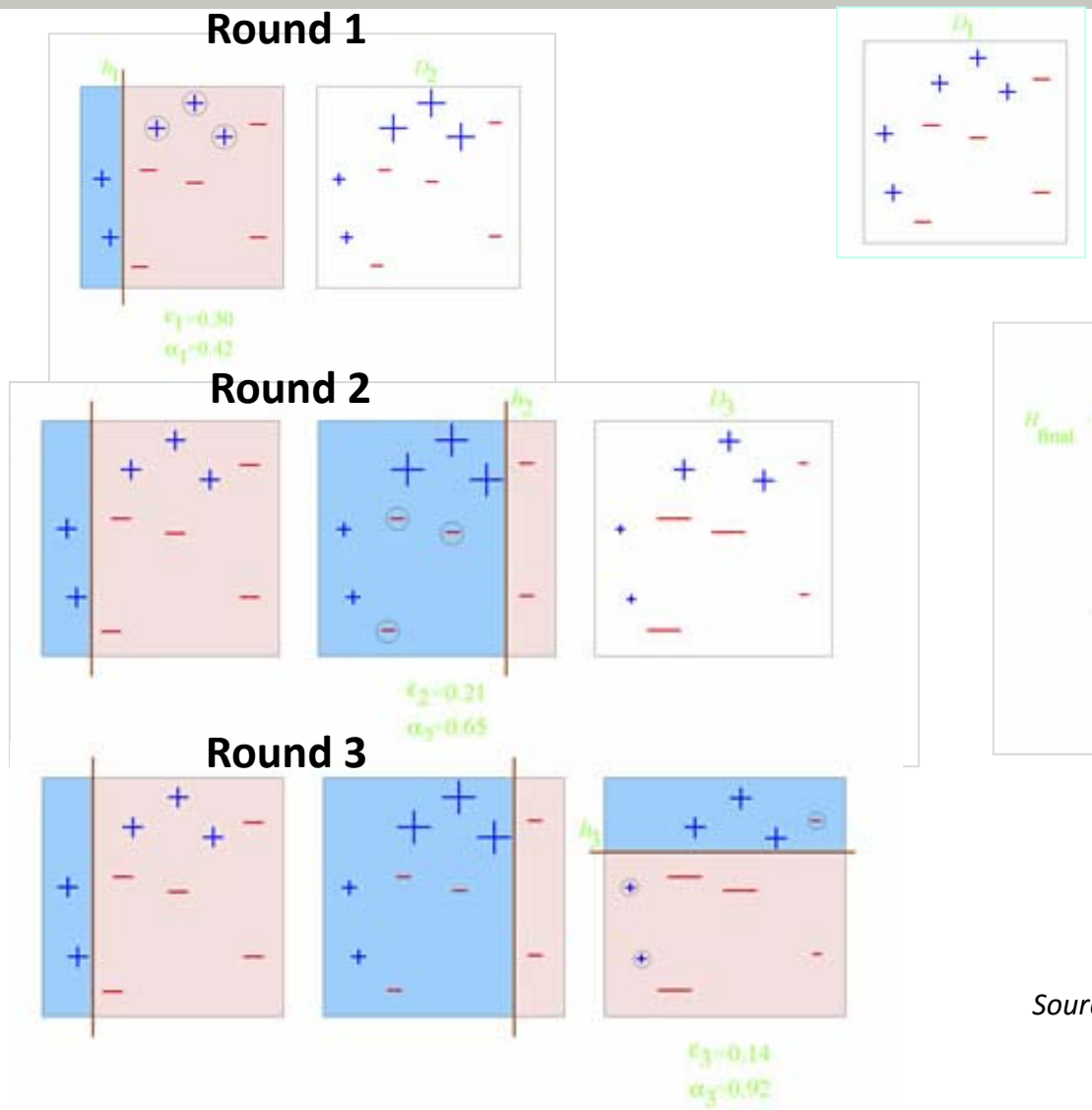
Weights update

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

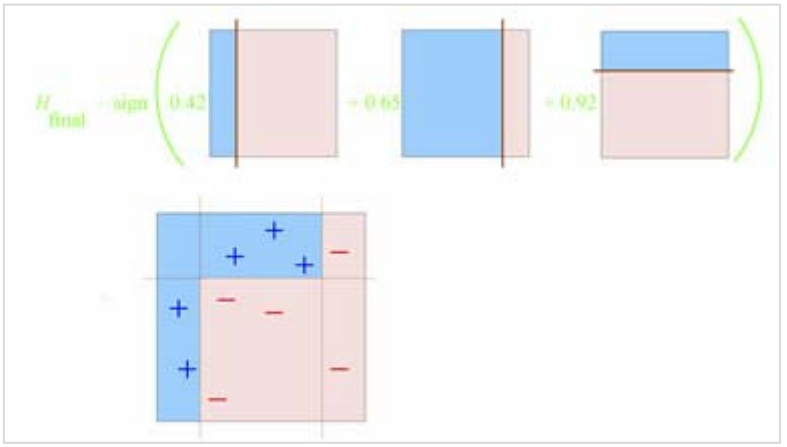
Output the final hypothesis:

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

Boosting example



Final classifier



Source: http://videolectures.net/mlss05us_schapire_b/

More on boosting

- Focus on more difficult examples
- Can be quite susceptible to overfitting
  - since it focuses on training examples that are wrongly classified

- Comparing to bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data.

Feature manipulation

- Manipulate the input features: Learn on varying subspaces or combine features
- Typical example: Random Forests, an ensemble of decision trees

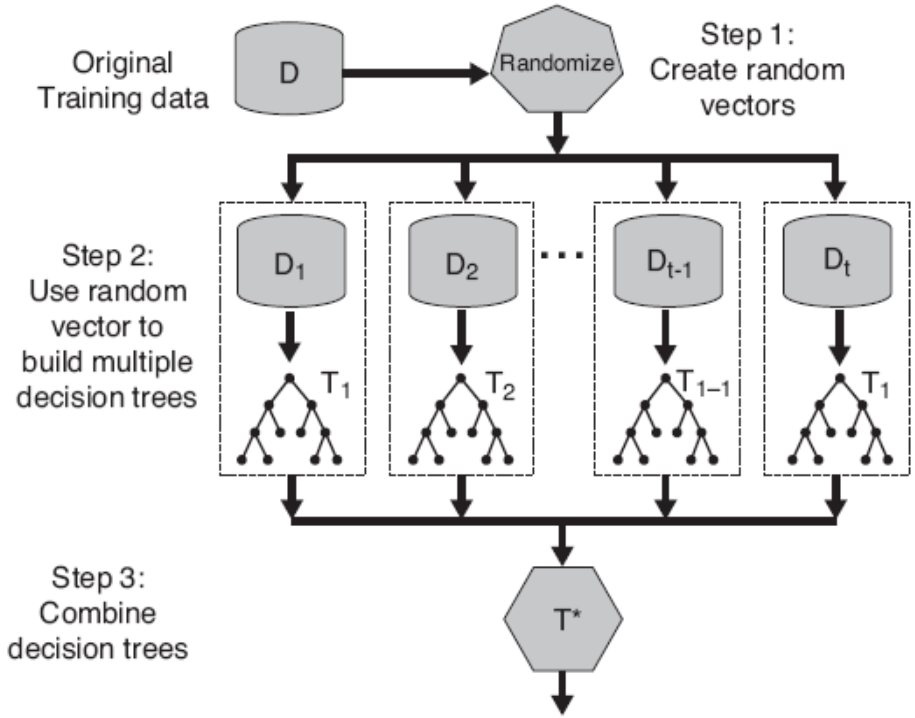


Figure 5.40. Random forests.

(from: Tan, Steinbach, Kumar: Introduction to Data Mining)

Random forests

- Intuition: Randomization helps to reduce the correlation among decision trees so that the generalization error of the ensemble can be improved.
- How to generate random trees?
  - random selection of features for each base-learner
  - (if the feature space is small), increase the feature space by constructing combinations of input features
  - for each node use one of the F best splits instead of the best split
- It combines classifiers being trained on different views/ feature sets (Multi-View Data Mining)

Manipulating the class labels

- Suitable when the number of classes is sufficiently large
- Idea: Map the multi-class problem to a set of two-class problems.
  - Partition the class labels into two sets, A_1+A_2.
    - instances belonging to A_1 → class 0
    - Instances belonging to A_2 → class 1
  - Use the re-labeled instances to train a base classifier.
  - By repeating the whole process, an ensemble is built

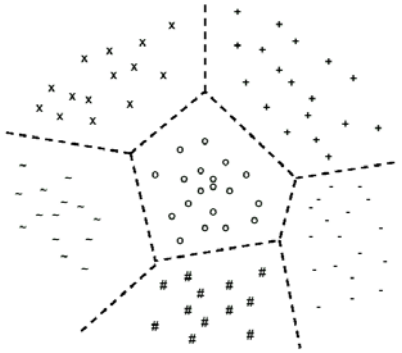


Image from: Fürnkranz 2002

  - For prediction, for each classifiers C_i in the ensemble
    - If C_i predicts 0, all class labels in A_1 receive a vote. If it predicts 1, then the vote goes to the class labels in A_2.
    - The class with the highest votes is the predicted class.
- General methods:
  - one-versus-rest, all-pairs, error correcting output codes

One-versus-rest

- *one-versus-rest* (also known as: *one-versus-all*, *one-per-class*):

  For *k* classes, *k* classifiers are trained. Each distinguishes one class from the combination.n of all other classes

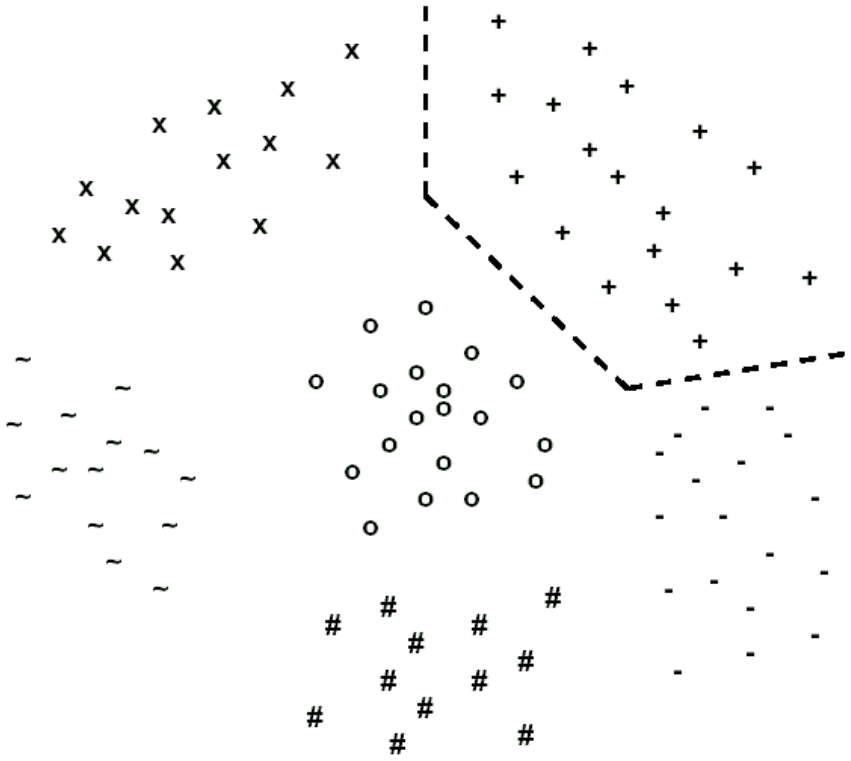


Image from: Fürnkranz 2002

All-versus-all

- *all-pairs* (also known as: *all-versus-all*, *one-versus-one*, *round robin*, *pairwise*):
  - For each pair of classes (y_i, y_j), a classifier is trained distinguishing between y_i, y_j
    - Only instances from y_i, y_j are employed, rest ignored
  - k(k-1)/2 binary classifiers

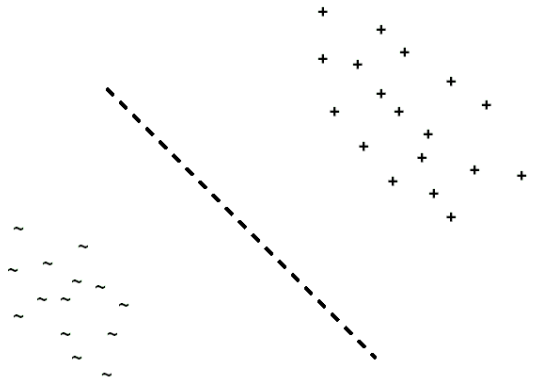


Bild aus: Fürnkranz 2002

Error correcting output coding (ECOC)

- Inspired by information theory for sending messages across noisy channels
- Idea: Encode each class using binary classifiers

- How the codes are created?
  - The class set *C* is randomly split into two disjunctive *A+B* subsets.
    - Dataset relabeling: instances belonging to class set *A* are labeled with *0*, all other objects belonging to the class set B are labelled with *1*.
    - A classifier h_l is built
  - Process is repeat L times → L classifiers
  - Each class c_i in C is encoded by an L-bit codeword
    - Bit l is 1 if and only if c_i belongs to class set B

- How the codes are used during prediction for a new instance x?
  - each h_l classifies x → codeword for x
  - the class with a codeword whose Hamming distance is closest to the codework of x, is the predicted one.
    - Hamming distance: # positions that the corresponding symbols are different

ECOC example

- Example: $C=\{c_1,c_2,c_3,c_4\}$

class	code word						
c_1	1	1	1	1	1	1	1
c_2	0	0	0	0	1	1	1
c_3	0	0	1	1	0	0	1
c_4	0	1	0	1	0	1	0

- 7-bit codewords => 7 binary classifiers are trained
- Each bit in the codeword is derived by the corresponding binary classifier
- If a test instance is encoded/classified as (0,1,1,1,1,1,1), for which class would the ensemble decide?

More on ECOC

- the name “Error Correcting Output Codes” reflects the idea that training multiple classifiers introduces redundant class borders
- the “code words” are binary codes which reflect the assignment of the classes for each classifier
- To build a high quality ensemble each of the class borders has to be sufficiently represented:
  - Row Separation: Each pair of code words should display a large Hamming distance (=large amount of mismatching bits).
  - Column Separation: the binary classifiers should be uncorrelated.

More on ECOC

class	code word						
c_1	1	1	1	1	1	1	1
c_2	0	0	0	0	1	1	1
c_3	0	0	1	1	0	0	1
c_4	0	1	0	1	0	1	0

- A large Hamming distance between the rows allows a unique class prediction of the ensemble.
- How large is the Hamming distance between the result (0,1,1,1,1,1,1) and the codes for the classes c_1, c_2, c_3 and c_4?

Manipulate the learning algorithm

- Manipulating the learning algorithm by randomization:
  - Start with varying initial models (e.g., different starting weights for back-propagation in neural networks)
  - Randomized splits in decision trees
  - It depends on the type of the classifier used

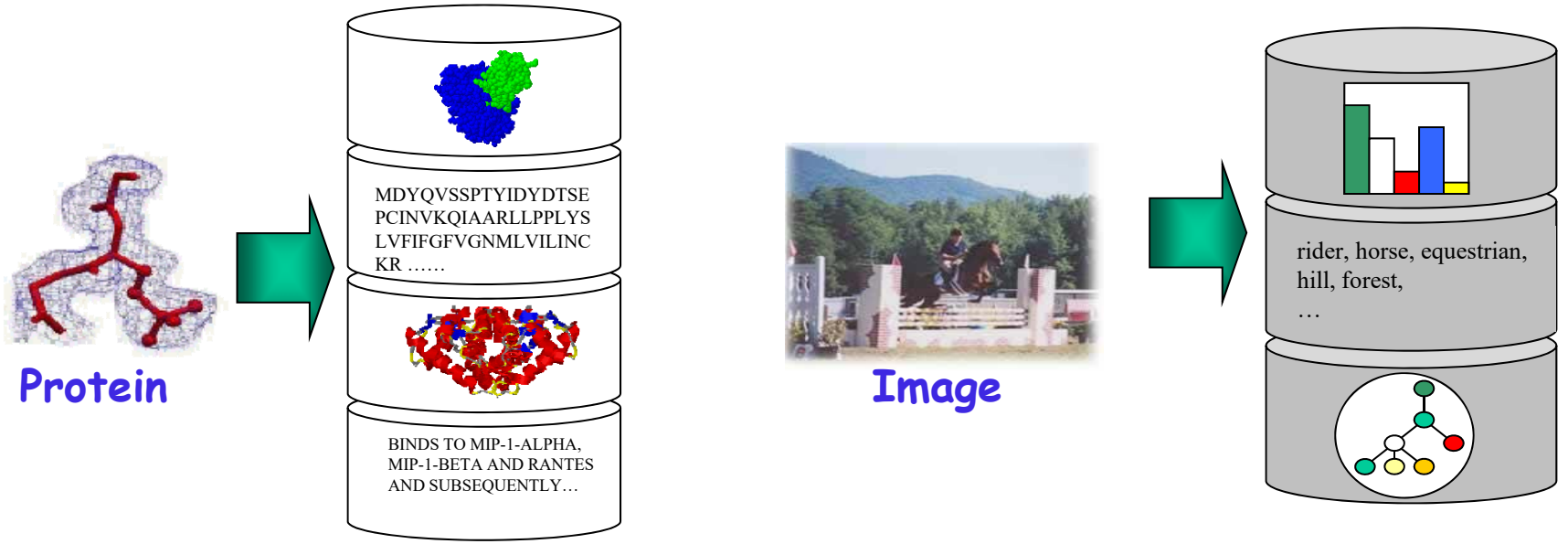
Ensemble learning overview

- A well-established method for obtaining highly accurate classifiers by combining less accurate base-learners.
- Ensemble = set of base learners + a voting strategy
- Important aspects for ensemble generation: accuracy and diversity of the base learners.
- Efficiency: easy to parallelize, each base learner is generated independently.
- Bagging and boosting approaches

- Ensemble learning for data streams: both base learners and their voting weights should be tuned over time.

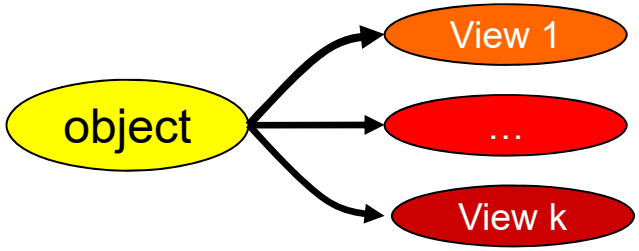
1. Introduction and Basic Principles of Ensemble Learning
2. Diversity in Ensemble Learning
3. Methods for Ensemble Construction
4. Mining Multi-View Data

Multi-View Data Mining



Reasons for the existence of multiple views:

- varying aspect of the same object
- varying measuring techniques
- varying feature transformations



$\Rightarrow$ Multi-View Data

$\Rightarrow$ $o \in R_1 \times .. \times R_n$, where R_i is the feature space of view i.

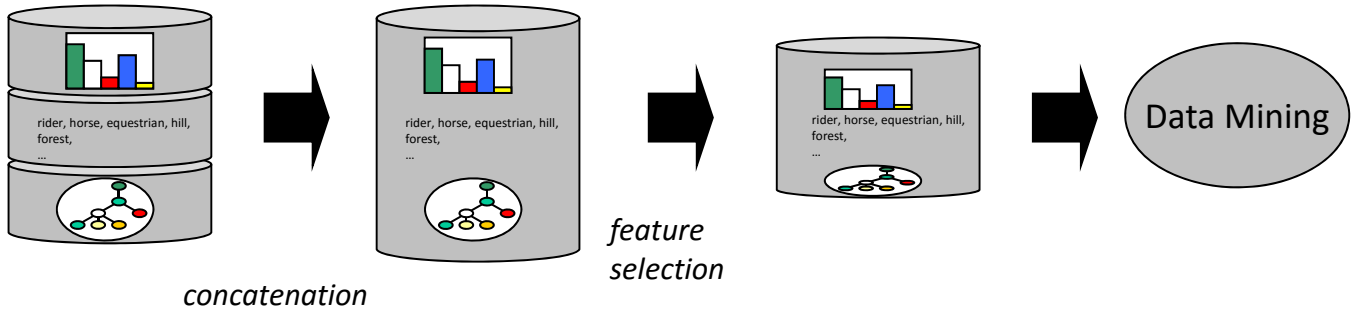
Problem setting 1/2

Challenges:

- all necessary information should be available to the data mining algorithm
  => employ all available information
- too many unnecessary features might have a negative influence on mining
  => use only the necessary features (compare to feature selection)

Naïve approach:

1. Build a joint feature space.
2. Use feature reduction or feature selection
3. Employ the data mining algorithm to lower dimensional representation.



Problem setting 2/2

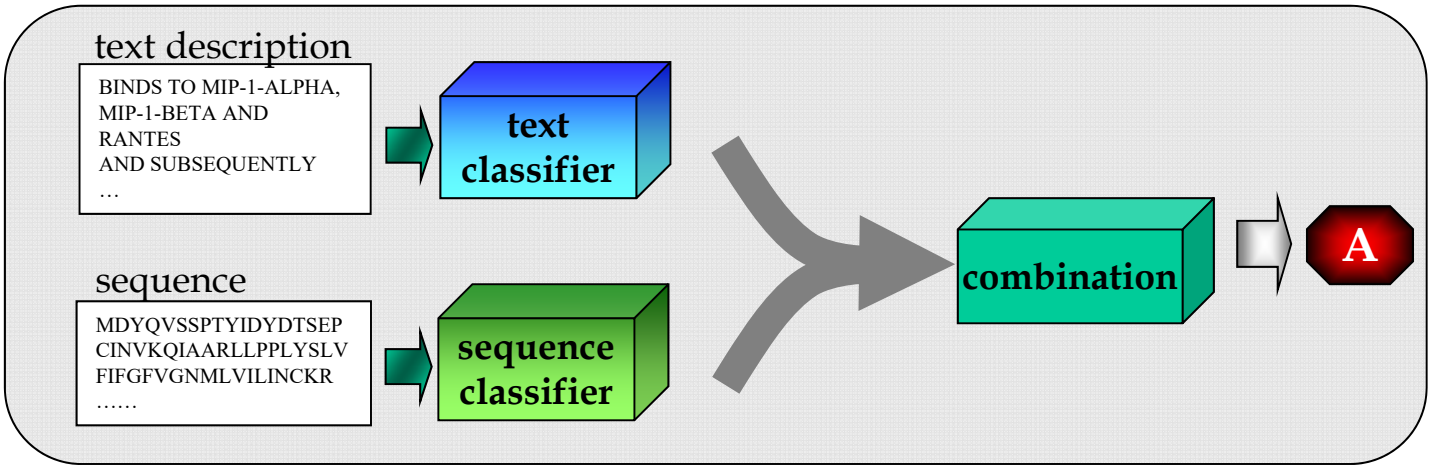
- Approaches (discussed in this lecture)
  - Multi-view classifiers
  - Co-training
  - Incorporate the multi-view aspect in the distance/similarity function
  - Multi-view clustering

Multi-view classifiers

Input: $o \in R_1 \times .. \times R_n$, where R_i is the feature space of view i.

Multi-view classifiers:

1. Train a classifier for each view
2. Classify a new object in each view
3. Combine the results to a global class prediction



Classifier combination

How to combine class predictions for an unknown instance *x* in the light of varying prediction qualities?

1. Each classifier Cl_r returns a confidence value c_A for each class A ➔ the confidence vector $c^r(x)$: $\sum_{A \in C} c_A^r(x) = 1$

2. Classify *x* by combining confidence vectors $c^r(x)$ of the different views

$$pred(X) = \mathbf{argmax}_{A \in C} \left(\Theta_{r \in R} \left(c_A^r \right) \right)$$

where $\Theta \in \{\min, \max, \sum, \prod\}$

Example

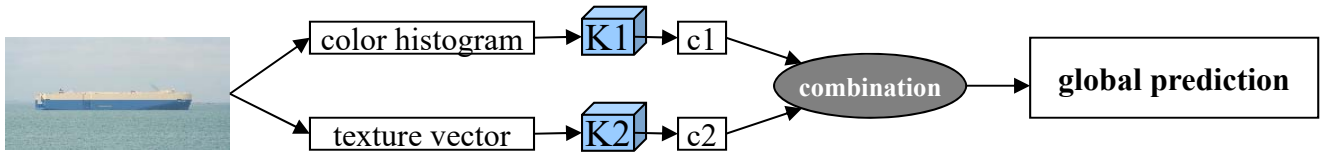
Input: 2 views for bitmap images: color histograms(R_1) and texture vectors (R_2).

classes = {"contains water"=A, "no water"=B}

Bayes classifiers K_1 (for R_1) and K_2 (for R_2)

Classification of a new image *b*:

$K_1(b)$=c1 =(0.45, 0.55); $K_2(b)$ = c2 = (0.6, 0.4)



- combination by maximum: c_{global} = (0.6, 0.55) and argmax(c_{global}) = A
- combination by mininum: c_{global} = (0.45, 0.4) and argmax(c_{global}) = A
- combination by average (sum):

  c_{global} = (0.45+0.6,0.55+0.4) * ½ = (0.525 , 0.475) and argmax(c_{global}) = A
- combination by product:

  c_{global} = (0.45*0.6,0.55*0.4)=(0.27, 0.22) and argmax(c_{global}) = A

Co-Training [BluMit98]

- Semi-supervised learning paradigm that exploits two mutually independent and sufficient views.

- Input: 2 views and only a limited amount of labeled instances.
  - The instance space: $X = X_1 \times X_2$
  - Each example: $x = (x_1, x_2)$

- How to deal with lack of labels?
- Idea: Use unlabeled data for training → co-training

- Why does this approach require multi-view data to succeed? Why a single view is not adequate?

Generating samples using a single view

Naïve approach:

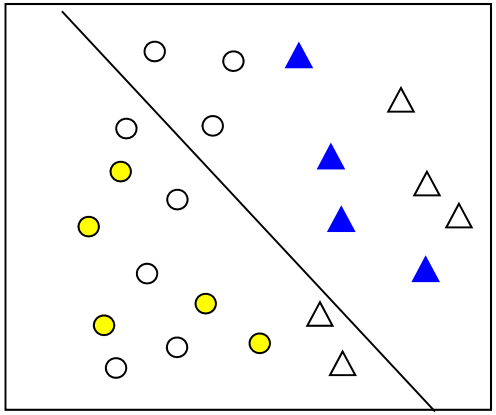
- Train a classifier *CL* on all labeled objects
- Classify *k* unlabeled objects and add them to the training set
- Train a new classifier on the extended set

Problem:

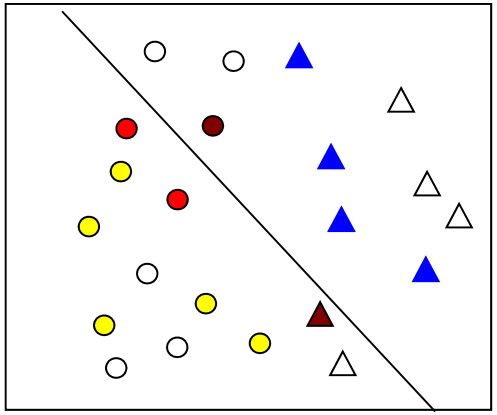
- new data is labeled by the classifier CL
- CL is trained on the original samples
- CL is based on the same distribution
- the new labels do not add any diversity

Generating samples using a single view: an example

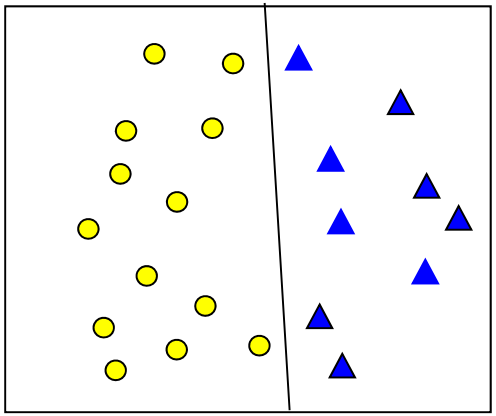
- blue = labelled objects (class triangle)
- yellow = labelled objects (class circle)
- red = labelled objects by CL_1



Training on original data



Training on newly labeled data



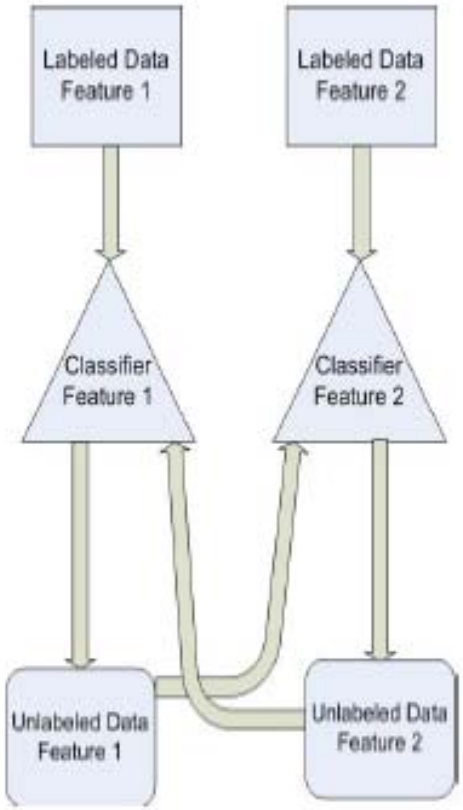
optimal solution

Conclusion:

- the red objects only confirm the given assumptions of the classifier
- to add new diversity an additional source of information is required

Co-Training overview

- Idea: Employ at least two classifiers for labeling objects being used to train classifiers for other views.



Source:
http://homes.cs.washington.edu/~santosh/presentations/coTraining_miscRead_web.pdf

The Co-Training Algorithm

Input: 2 sets of multi-view data

TR = labeled training set

U = set of unlabeled samples

Co-Training Algorithm

For *k* times do

For each R_i do

train CL_i for view *i.*

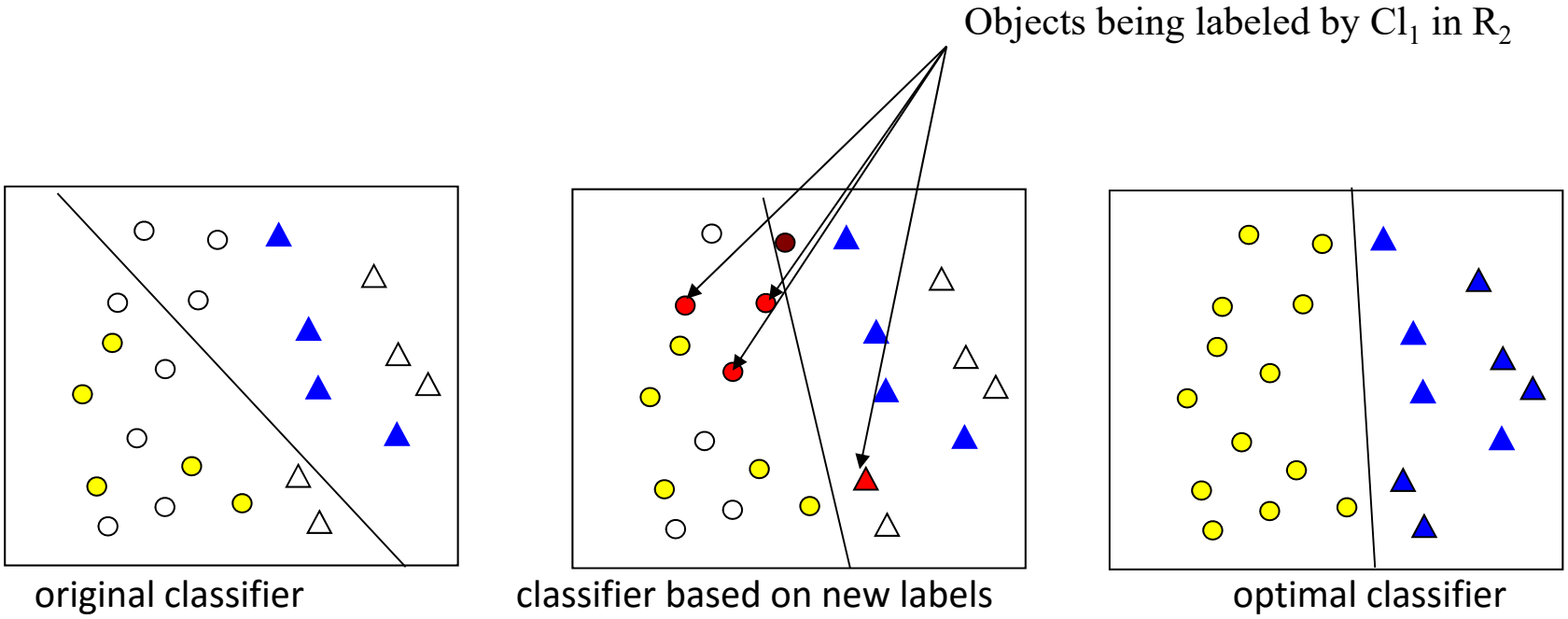
draw a sample from *U*.

generate new labels using CL_i.

add newly labeled objects to *TR*

Co-Training: an example

- blue = labelled objects (class triangle)
- yellow = labelled objects (class circle)
- red = labelled objects by CL_1



=> classification can be improved by using classifiers being trained on other views

Multiview similarity/distance measures

- Integrate multiple views on the object comparison level.
- Idea: Keep the separated feature spaces and combine the derived similarity values over all views.
- Example: weighted linear combination
  - $o \in R_1 \times .. \times R_n$, where R_i is the feature space of view i.
  - $d_i(o_1,o_2)$: local metric or kernel in R_i.

$$D_{combi}(o_1, o_2) = \sum_{R_i \in R} w_i \cdot d_i(o_1, o_2)$$

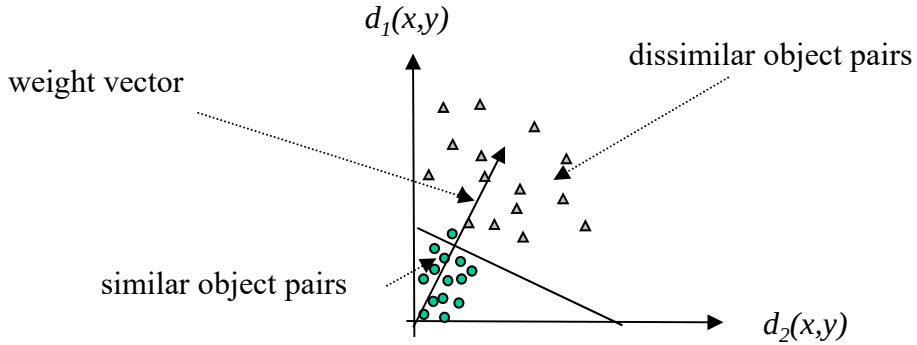
Learning weights for each view

Formulate the weighting problem as a linear classification task:

- use classes {“similar“,“dissimilar“} for pairs of objects *(x,y)*
  (if x.c =y.c then (x,y).c = similar else dissimilar)
- The normal vector of the separating hyperplane represents the weights
- Feature space: distance vectors $v_i = d_i(x,y)$ for all views R_i $1 \leq i \leq n$
- Training set: set of all pairs of vectors in the training set ($O(n^2)$

Method:

- Determine the distance vectors for all object pairs
- Train a linear classifier
- Determine the normal vector of the separating hyperplane as weights (MMH).



Combining similarity and distance functions

Remarks:

- Caution: linear classifiers must result in positive normal vectors!

- It is possible to use the learned classifier directly as a combination function. In this case, the class probability for the class dissimilar is used as distance measures

- Combining distance values using more complex combination functions requires that the mathematical properties which are required for the data mining algorithm still to hold.

  (symmetry, triangular inequality, positive definiteness)

Clustering multi-view data

Requirements for clustering multi-view data :

- employ all views.
- use specific similarity/distance measures
- employ index and data structures for the employed data types
- the effort should only increase linearly with the number of views

Union method

Idea: An object is in a dense region if there are *k* neighbors sufficiently similar over all views. (similarity can be reliably observed from each view)

used for : sparse data

Union Core-Object:

Given: $\varepsilon_1, \varepsilon_2, .., \varepsilon_m \in IR^+$, $MinPts \in N$. $o \in O$ is an union core object if

$$\left| \bigcup_{R_i(o) \in o} N_{\varepsilon_i}^{R_i}(o) \right| \geq MinPts$$ where $N_{\varepsilon_i}^{R_i}(o)$ denotes the local ε-*neighborhood in view i* .

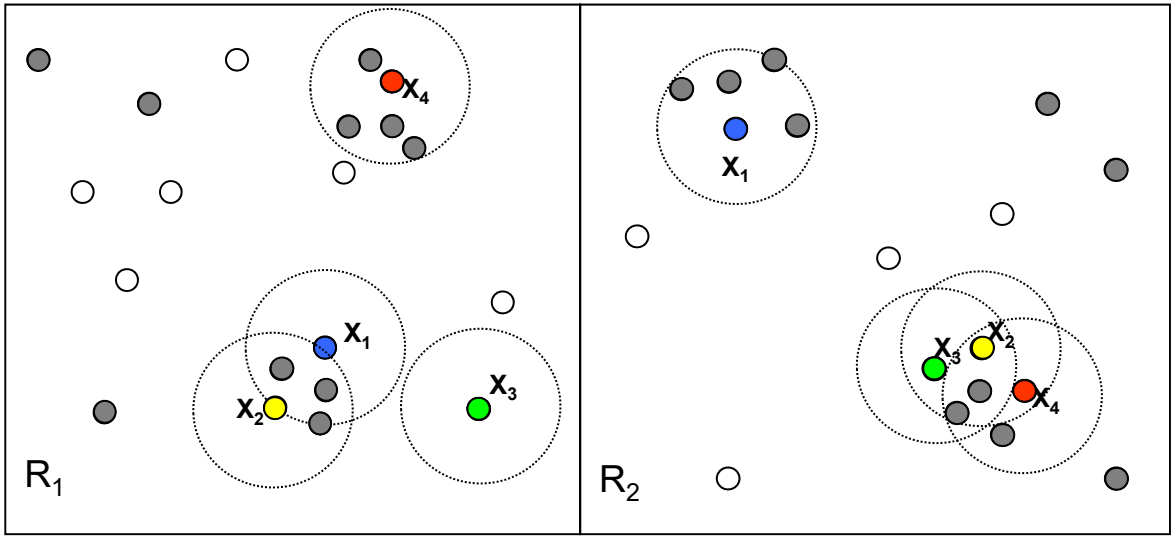
Direct union reachabibility:

Object $p \in O$ is direct union reachable by $q \in O$ w.r.t. $\varepsilon_1, \varepsilon_2, .., \varepsilon_m$ and *MinPts* if *q* is a union core object and p is a member of at least one local neighborhood, i.e.:

$$\exists \; i \in \{1, \ldots, m\} : R_i(p) \in N_{\varepsilon_i}^{R_i}(q)$$

Union Method

Cluster expansion using the union method



MinPts = 3

Intersection Method

Idea: An object is in a dense region if the ε-neighborhoods in all views are dense.

Used for: dense views and unreliable similarity functions.

Intersection core object:

Given: $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m \in IR^+$, $MinPts \in N$. o $\in$ O is an intersection core object if

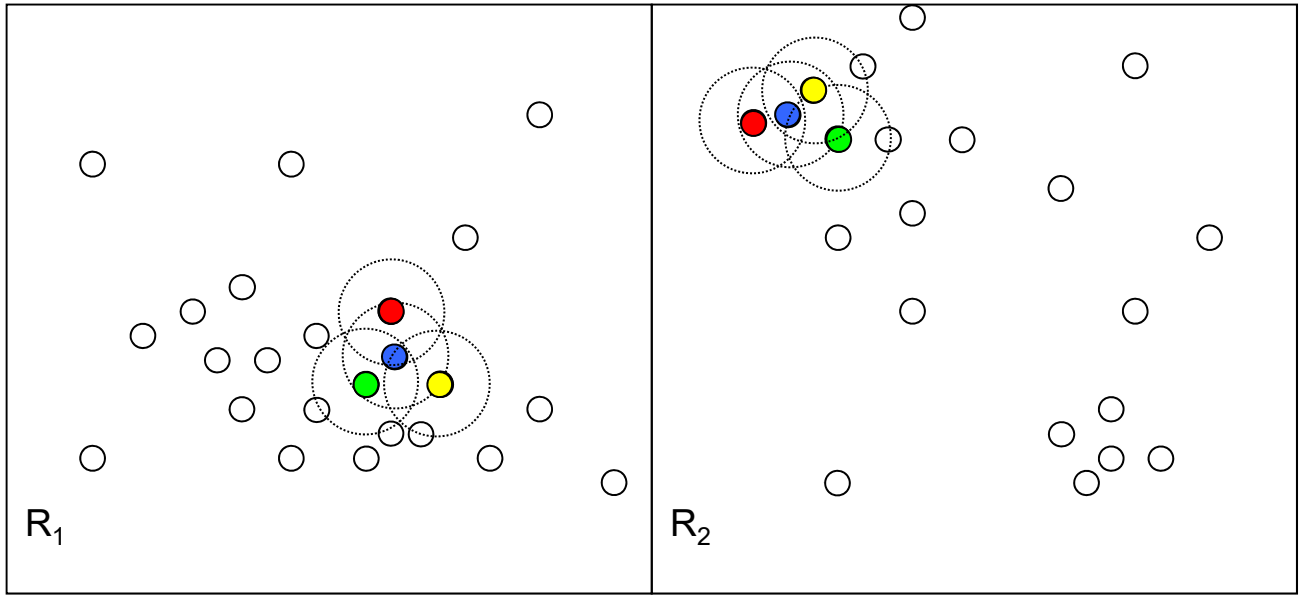
$$\left| \bigcap_{R_i(o) \in o} N_{\varepsilon_i}^{R_i}(o) \right| \geq MinPts$$ where $N_{\varepsilon_i}^{R_i}(o)$ denotes the local ε-*neighborhood in view i* .

Direct intersection reachable:

Object $p \in O$ is direct intersection reachable by $q \in O$ w.r.t. $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m$ *and MinPts* *if q* is an intersection core object and $\forall \; i \in \{1, \ldots, m\} : R_i(p) \in N_{\varepsilon_i}^{R_i}(q)$

Intersection Method

Cluster expansion using the intersection method



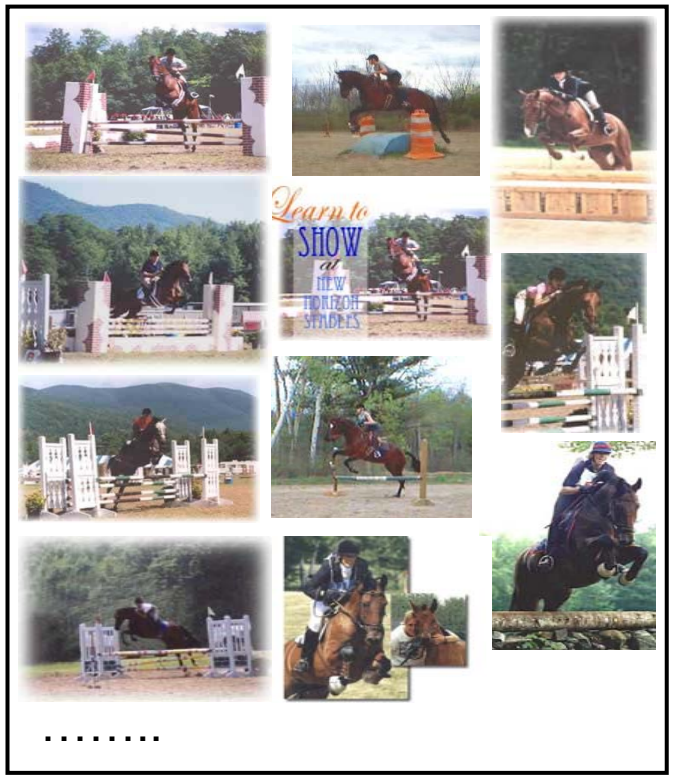
MinPts = 3

Example results

Cluster in single views.

Example images for cluster IC 5
(color histograms)

Example images for cluster IC 5
(segment trees)

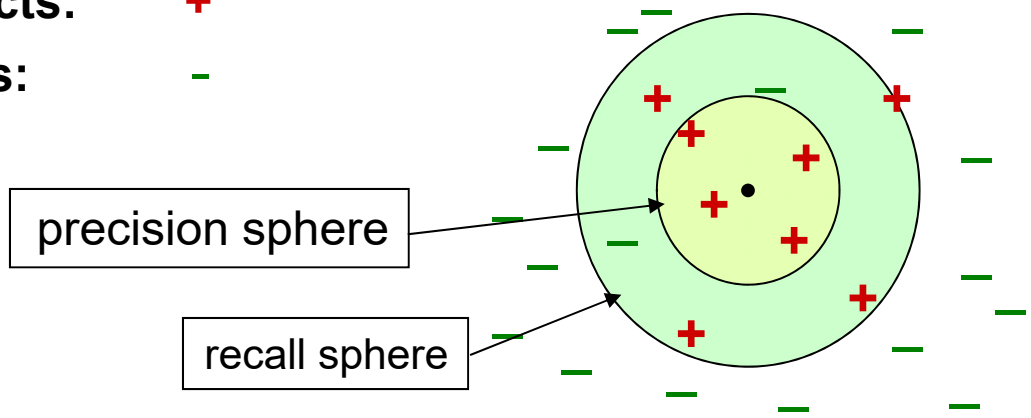


Cluster IC5 based on the intersection method.

Similarity in different views

truly similar objects: +

dissimilar objects: -



- optimally precision sphere = recall sphere
  (one view is enough)
- the intersection methods tries to remove false positive from the recall sphere.
- the union method tries to add false negatives to the precision sphere

References

- T. G. Dietterich, Ensemble Methods in Machine Learning, Multiple Classifier Systems , 2000.
- T. G. Dietterich: Ensemble learning. In: M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks. MIT Press 2003.
- J. Fürnkranz: Round robin classification. In: Journal of Machine Learning Research, 2:721-747, 2002.
- P.-N.Tan, M. Steinbach, and V. Kumar: Introduction to Data Mining, Addison-Wesle, 2006, Kapitel 5.6+5.8.
- Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman and Hall/CRC, 2012.
- G. Valentini and F. Masulli: Ensembles of learning machines. In: Neural Nets WIRN Vietri 2002.
- K. Kailing, H.-P. Kriegel, A. Pryakhin, M. Schubert. Clustering Multi-Represented Objects with Noise, PAKDD, 2004.
- Blum. A, Mitchell T.: *Combining Labeled and Unlabeled Data with Co-Training*, Workshop on Computational Learning Theory (COLT 98),1998.