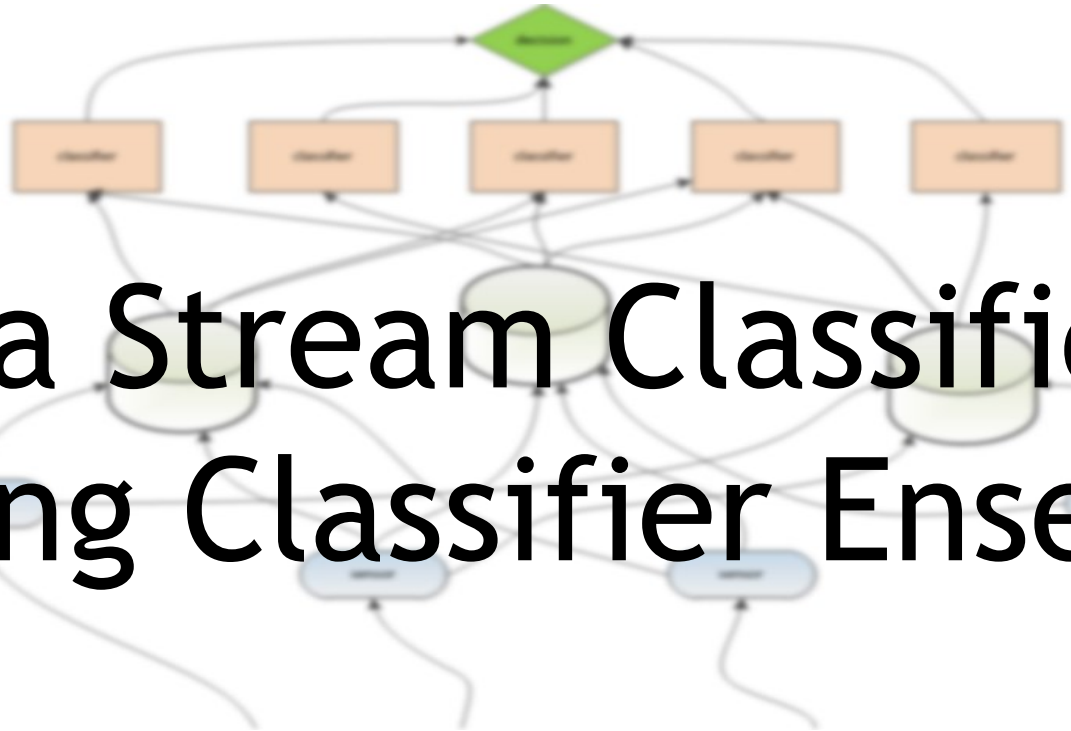


Data Stream Classification Using Classifier Ensemble

Michał Woźniak
Wroclaw University of Technology, Faculty of Electronics, Department of Systems and Computer Networks
E-mail: michal.wozniak@pwr.edu.pl Web: http://www.kssk.pwr.wroc.pl/wozniak/?lang=en

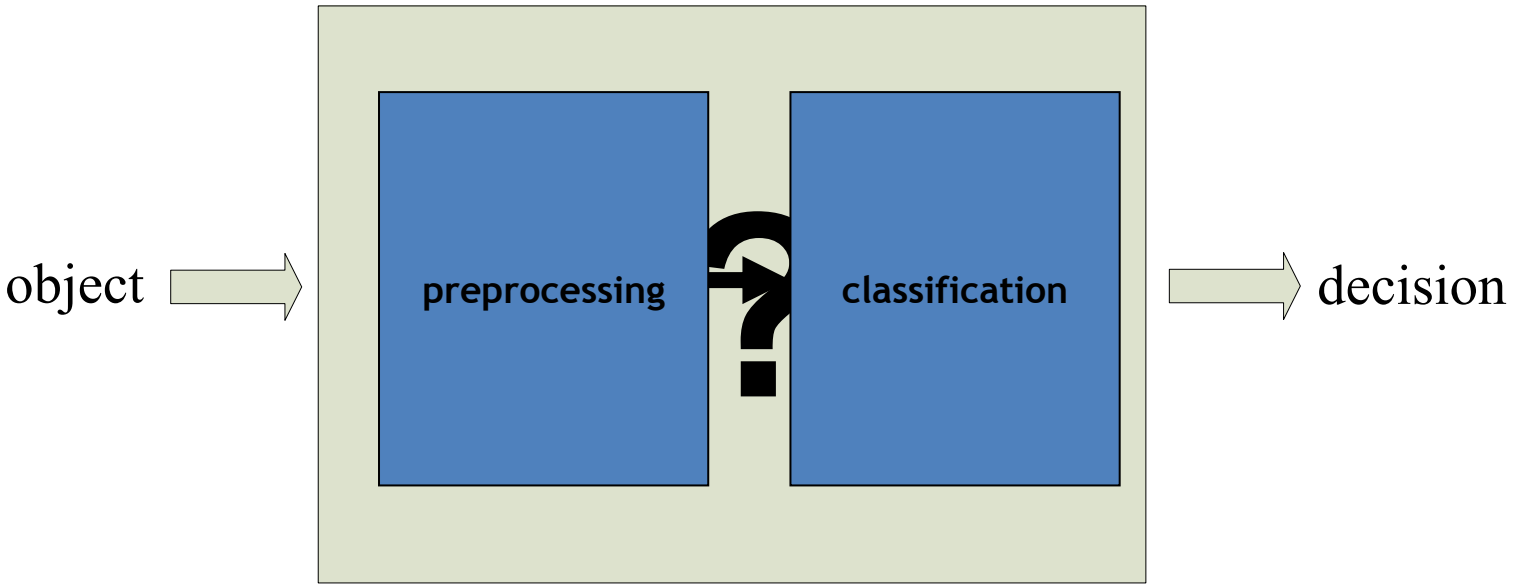
Wrocław University of Technology

engine

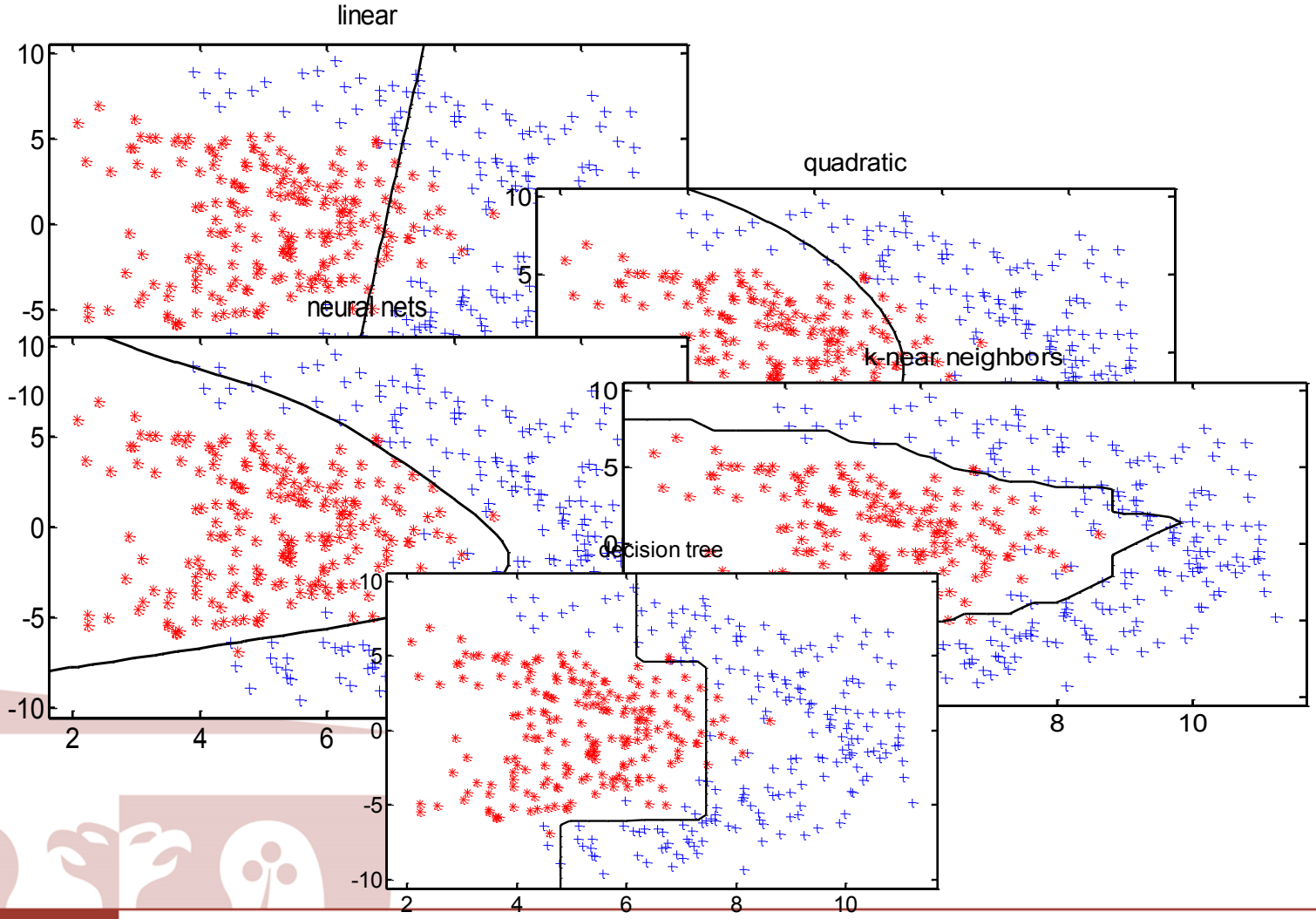
Outline

- Hybrid classifier
- Classifier ensemble
- Data stream classifiation

Pattern recognition task

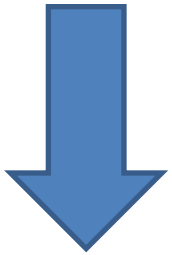


Pattern recognition task

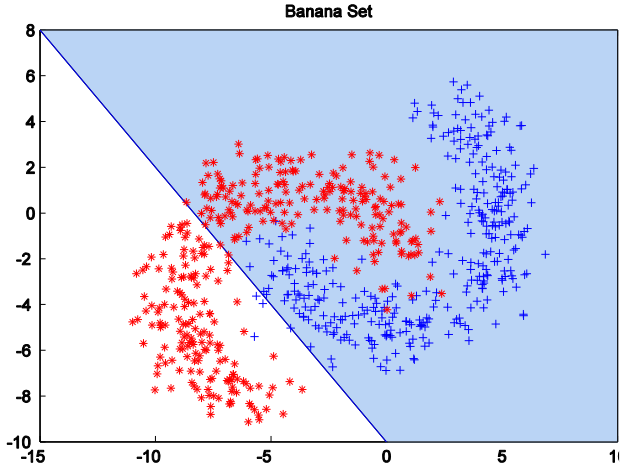


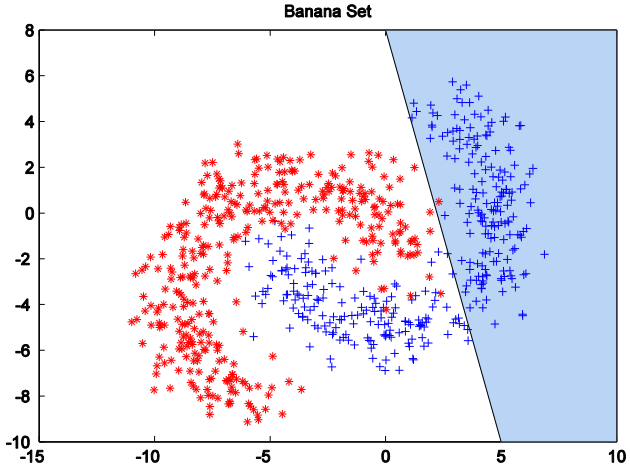
Motivation

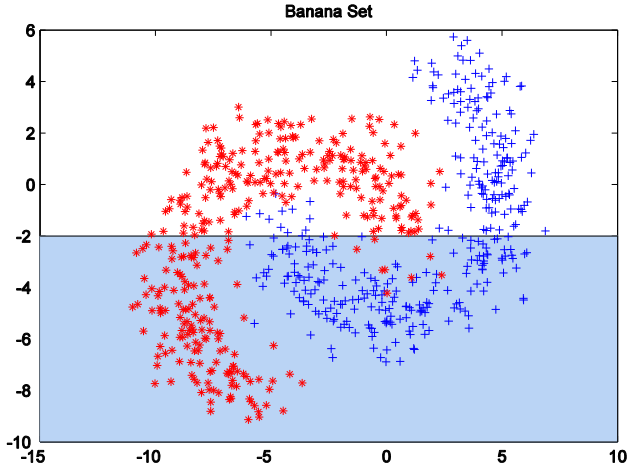
Based on Wolpert's theorem, there is not a single pattern recognition algorithm appropriate for all the tasks we deal with, as each classifier has its own domain of competence.

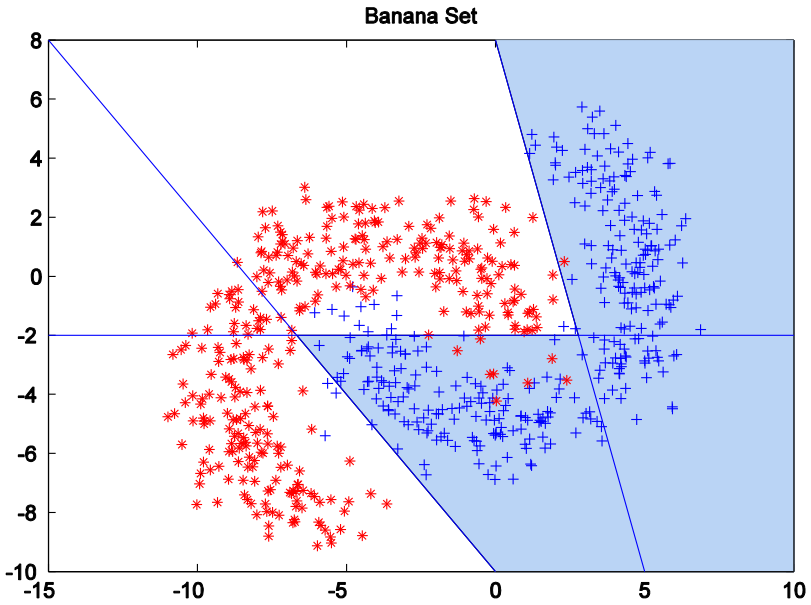


If it is not possible to generate a universal classifier, perhaps we could use the valuable components and ideas behind classifiers around us.


Banana Set


Banana Set


Banana Set


Banana Set

Hybrid classifier

Hybrid classifier means a classifier system which merges different components of individual classifiers to exploit their strengths and improve the performance of the classification system.

Hybrid classifier - levels

- Use different (distributed) data sources for training
- Apply different data types and knowledge representations to merge them into one unified representation
- Use trained models but take additional knowledge into consideration, e.g., additional constrains
- Use trained models to achieve the common decision based on combined classifier approaches

Classifier ensemble

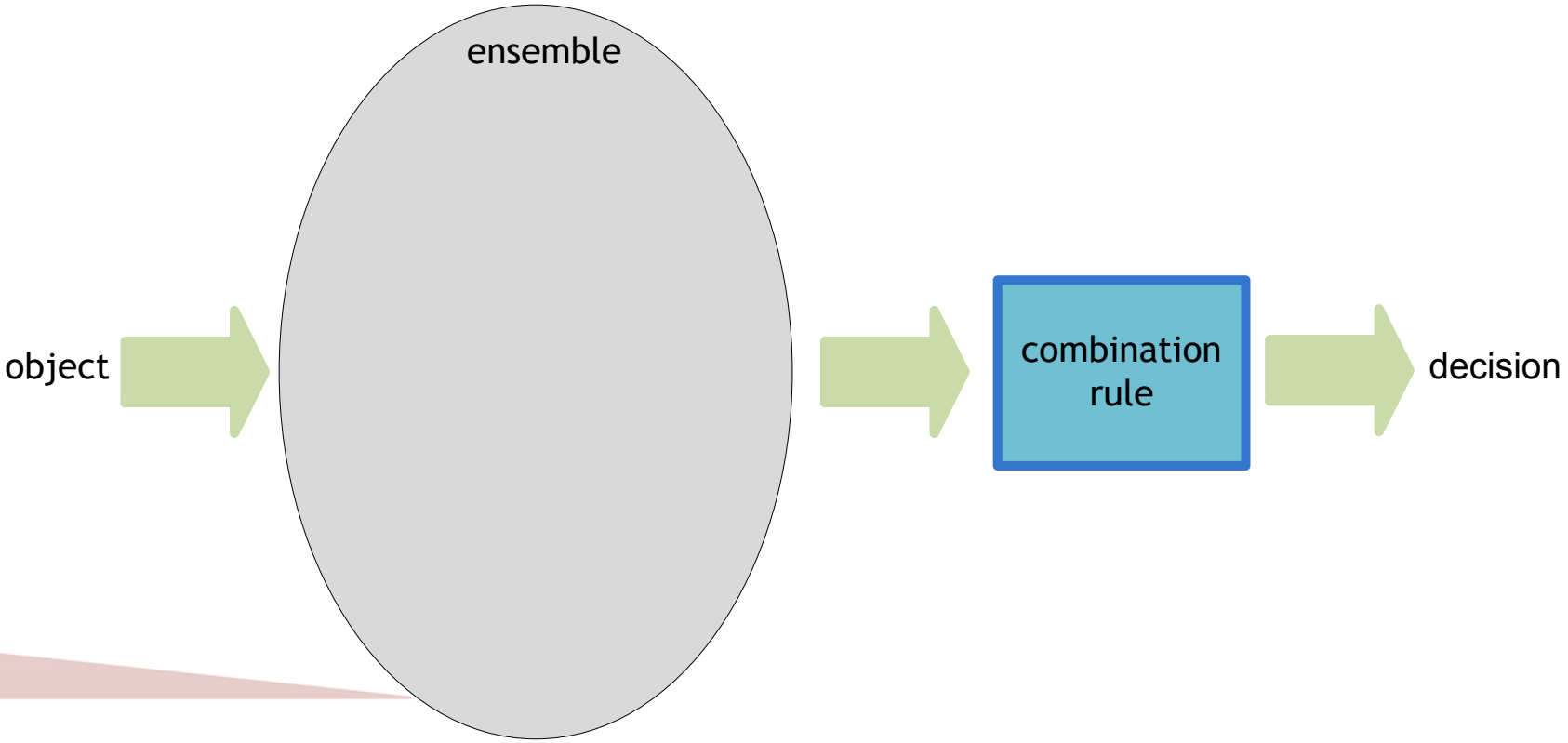
Problems related to classifier ensemble design:

- Forming valuable ensemble based on diversity measures and cost of classifier exploitation.
- Developing efficient combination rules.

Classifier ensemble - motivations

- Avoiding the selection of the worst classifier.
- Classifiers combination can improve the performance of the best individual ones and it can exploit unique classifier strengths.
- Computational complexity – finding an optimal classifier is NP-hard (many training algorithms suffer from the problem of local minima).
- This is natural way of classification in distributed environment, because of e.g., privacy reasons

Classifier ensemble - architecture

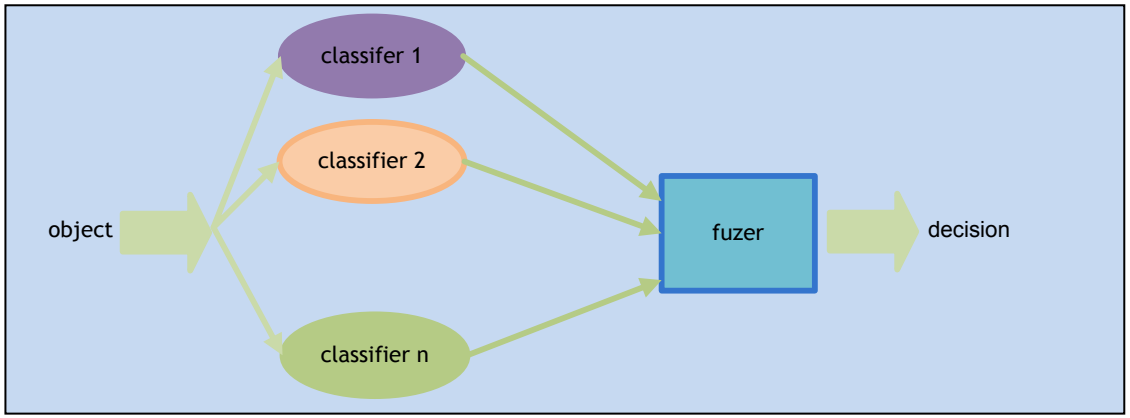


Combined classifier architecture

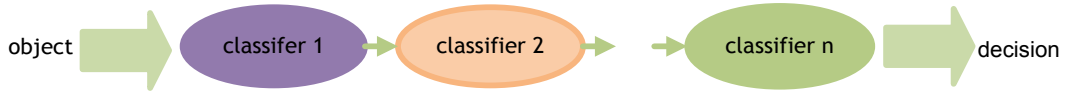
They are characterized by:

- Topology
- Ensemble line-up
- Combination rule

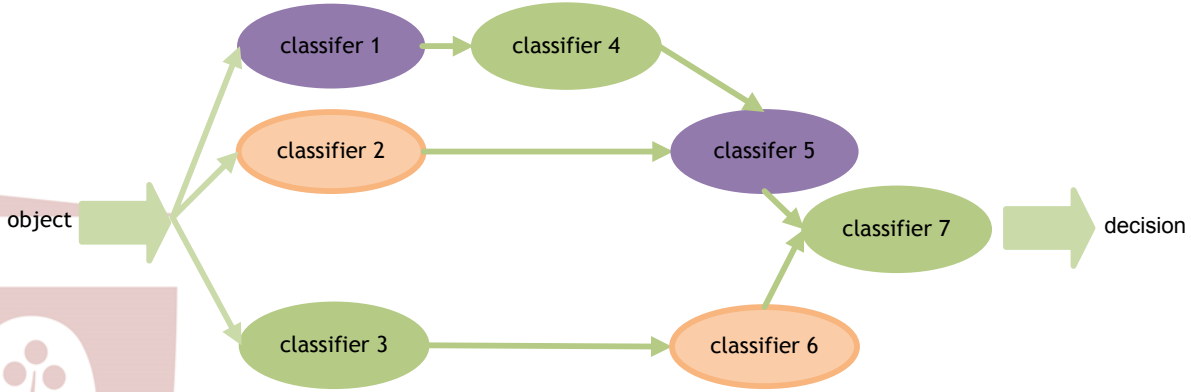
parallel



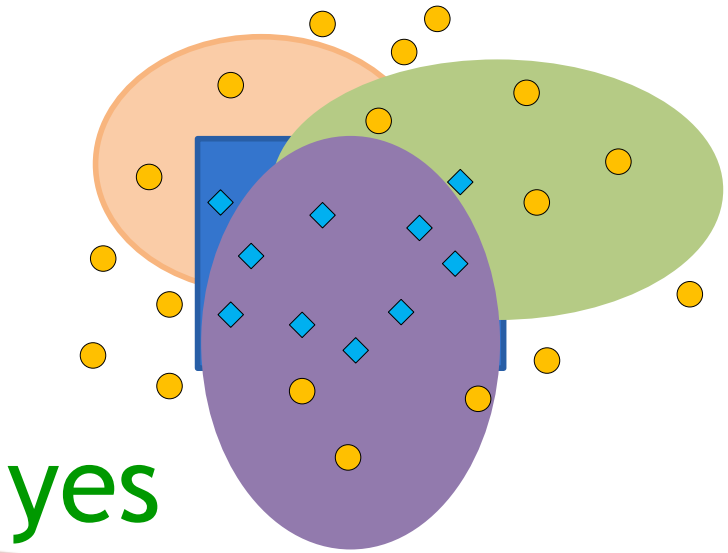
serial



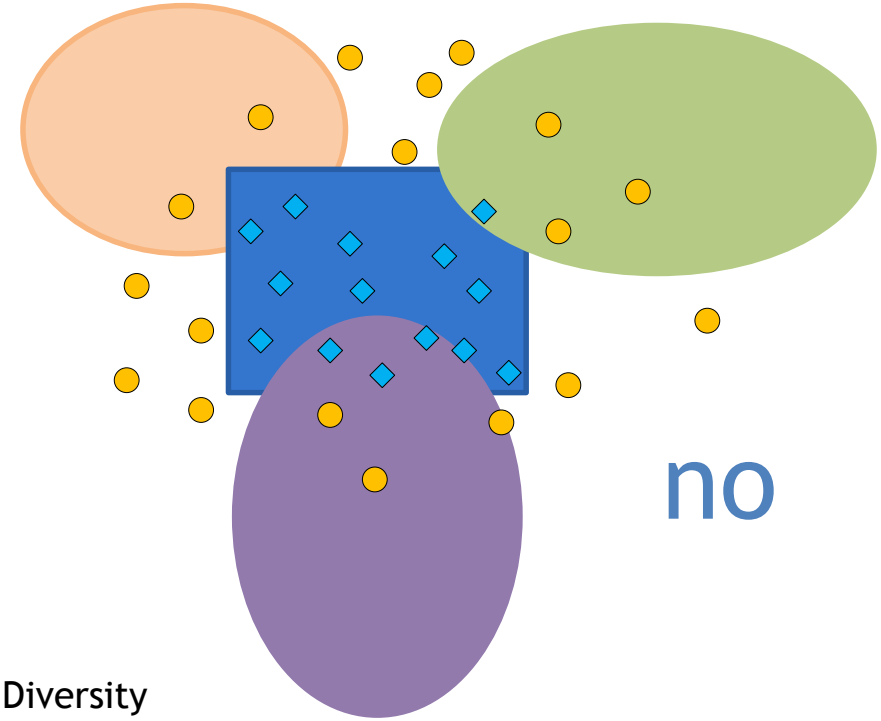
hybrid



- An ideal **ensemble** consists of classifiers with high accuracy and high diversity, i.e., mutually complementary



yes



no

See also: Brown G., Kuncheva L., "Good" and "Bad" Diversity in Majority Vote Ensembles, LNCS 5997, pp. 124–133, 2010

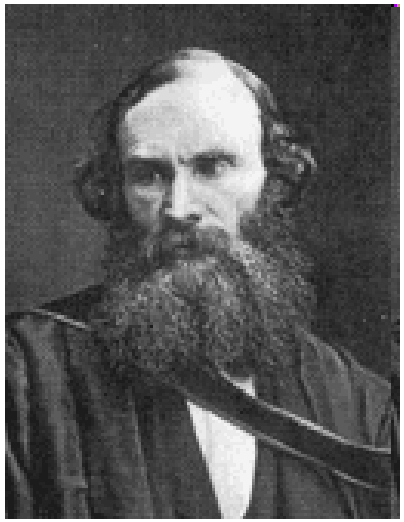
- How classifiers can be made more diverse:
  - Manipulate input
  - Manipulate model
  - Manipulate output

Diversity

It is hard to say what does mean.

"To measure is to know„

Lord Kelvin (1824-1904)



Diversity measure - examples

❷	Correlation coefficient	r_{AV}
❷	Product-moment correlation	r_{AV}
❷	Q statistics	Q_{AV}
❷	Disagreement measure	S_{AV}
❷	Double fault measure	F_{AV}
•	Entropy measure	q
•	'Difficulty' measure	H
•	Kohavi-Wolpert variance	k
•	Interrated agreement measure	KW
•	**'Fault majority' measure**	w

$$\rho_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{\sqrt{N^{1*}N^{0*}N^{*1}N^{*0}}}$$

$$r_{i,j} = \frac{N^{00}}{\sqrt{N^{*0}N^{0*}}}$$

$$Q_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$$

$$S_{i,j} = \frac{N^{01} + N^{10}}{N}$$

$$F_{i,j} = \frac{N^{00}}{N}$$

$$H = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M - \lceil M/2 \rceil}\min\{m(\mathbf{x}_i), M - m(\mathbf{x}_i)\}$$

$$\theta = \frac{1}{N}\sum_{i=0}^{M}(z_i - \bar{z})^2$$

$$KW = \frac{1}{NM^2}\sum_{i=1}^{N}[m(\mathbf{x}_i)(M - m(\mathbf{x}_i))]$$

$$\kappa = 1 - \frac{\sum_{i=1}^{N} m(\mathbf{x}_i)(M - m(\mathbf{x}_i))}{NM(M-1)\bar{e}(1-\bar{e})}$$

$$\omega = \sum_{j=\lceil L/2 \rceil}^{L}\sum_{i^*=1}^{\lceil L/2 \rceil} z_{i^*,j}$$

Ensemble pruning

- It is obvious that more does not mean better, especially in the case of combined classifiers.
- Zhou et al. {Zhou:2002} presented an appropriate analysis for regression problems, where they formulated condition once removing one model from ensemble has the positive impact for the ensemble performance.

Ensemble pruning

- **Ranking-based methods** use an evaluation measure for classifier ranking and choose only the first best ones.
- Margineantu and Ditterich proposed to use the kappa-statistics to order each possible pairs of classifiers and choose a fixed number of best models {Margineantu:1997}.
- They proposed also to apply the Reduce-Error Pruning adding a fixed number of classifiers one by one to ensemble according to their accuracies, and then processes the next phase of the algorithm iteratively to check that if replacing a selected classifier by an unselected classifier could improve ensemble accuracy.

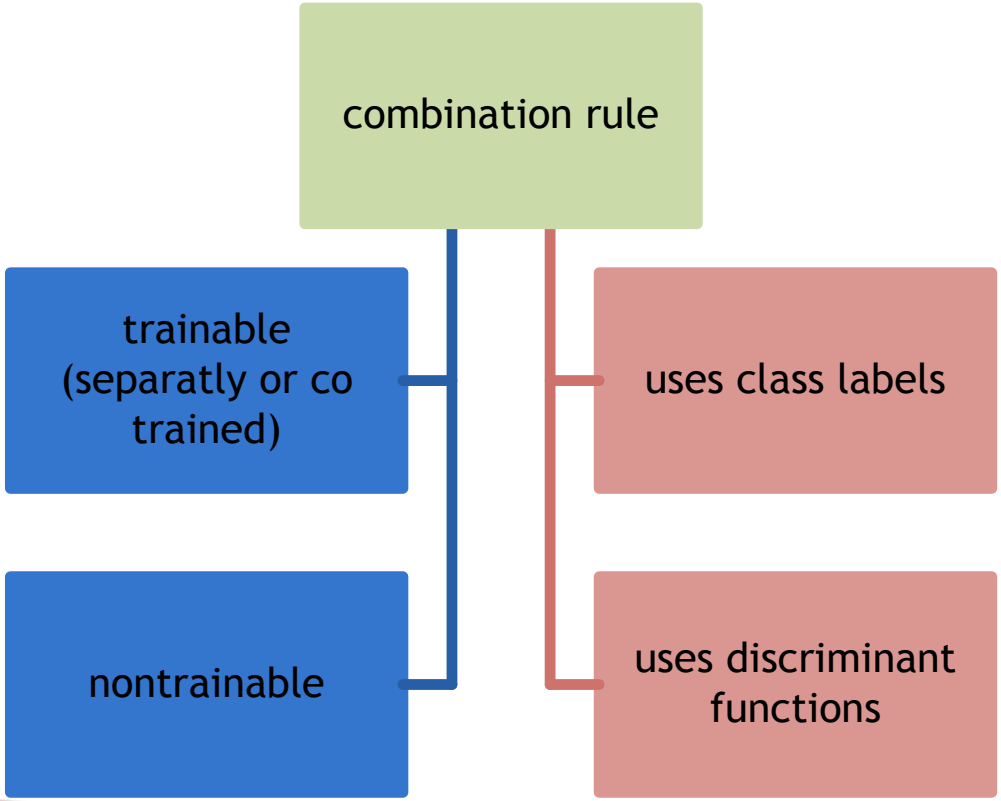
Ensemble pruning

- **Clustering-based methods** consist of two phases.
  - In the first phase, they cluster a pool of classifiers according to the criterion as Coincident measure {Giacinto:2001} or double-fault diversity {Kuncheva:2003}.
  - The next phase is responsible for pruning of each cluster.
  - Two main approaches can be found:
    - New classifier is trained for each cluster {Bakker:2003}
    - One classifier from each cluster is selected, e.g., which is most distant from remaining clusters {Giacinto:2001}, or is most accurate in a given cluster {Fu:2005}.
  - The important problem is how to fix the number of clusters which has an impact on ensemble performance {Lazarevic:2001}.
  - Further, there is a noteworthy work by Inoue and Narihisa {Inoue:2002} who applied SOM to the problem under consideration.

Ensemble pruning

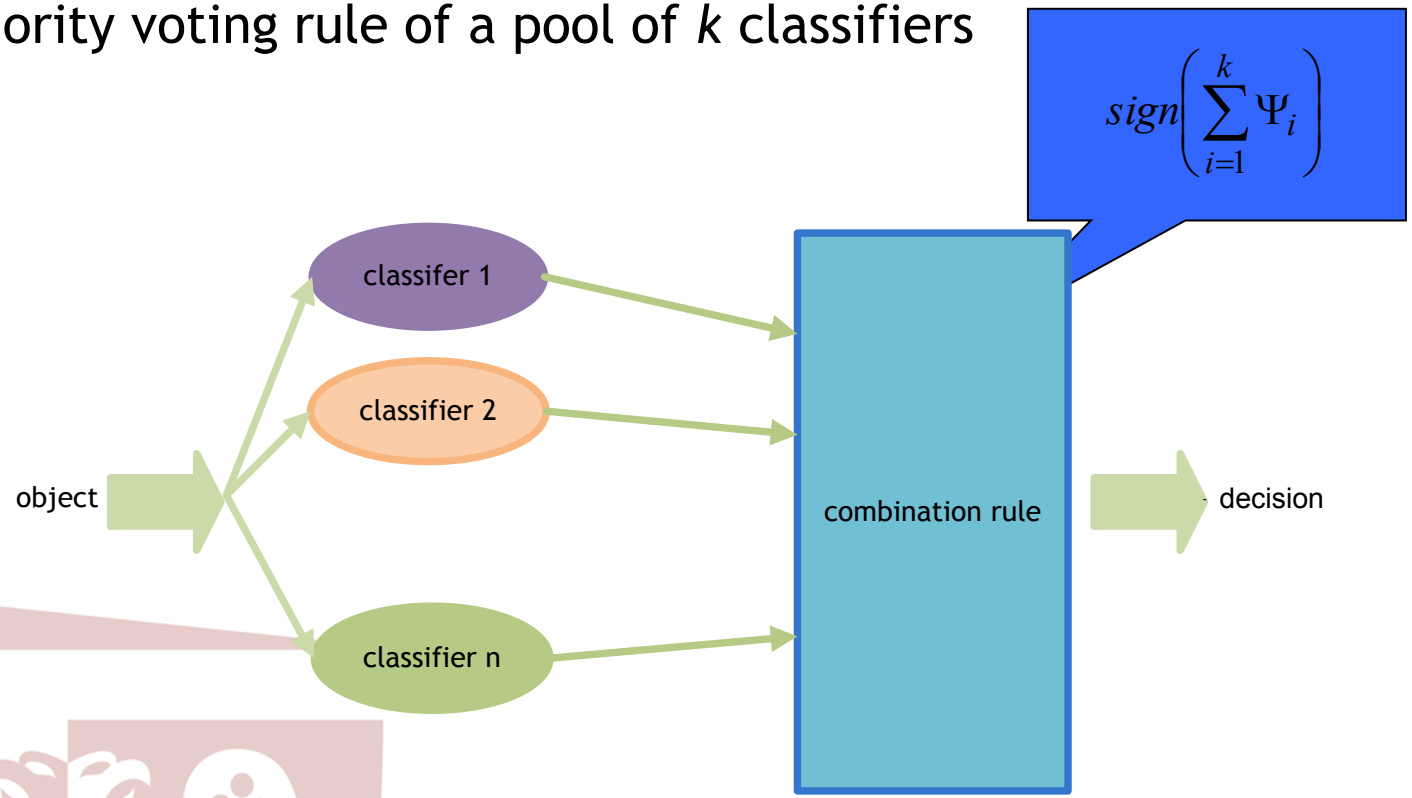
- **Optimization-based pruning** methods consider ensemble pruning as an optimization problem and most of them use heuristic techniques {Ruta:2005,Banfield:2005}, evolutionary algorithms {Zhou:2002,Gabrys:2006}, or competitive techniques based on cross-validation {Dai:2012} to enumerate only a few.
- Krawczyk and Wozniak used genetic approach to form an ensemble with minimal classification error within a fixed cost bounds {Krawczyk:2011}
- Jackowski et al. {Jackowski:2012} proposed a novel criterion based on EG2 {Nunez:1991} proposition. On the one hand it takes the ensemble accuracy into consideration, but on the other hand its cost is related to the sum of attribute acquisition costs used by the individual classifiers.

Combination rule design



Combination rule design

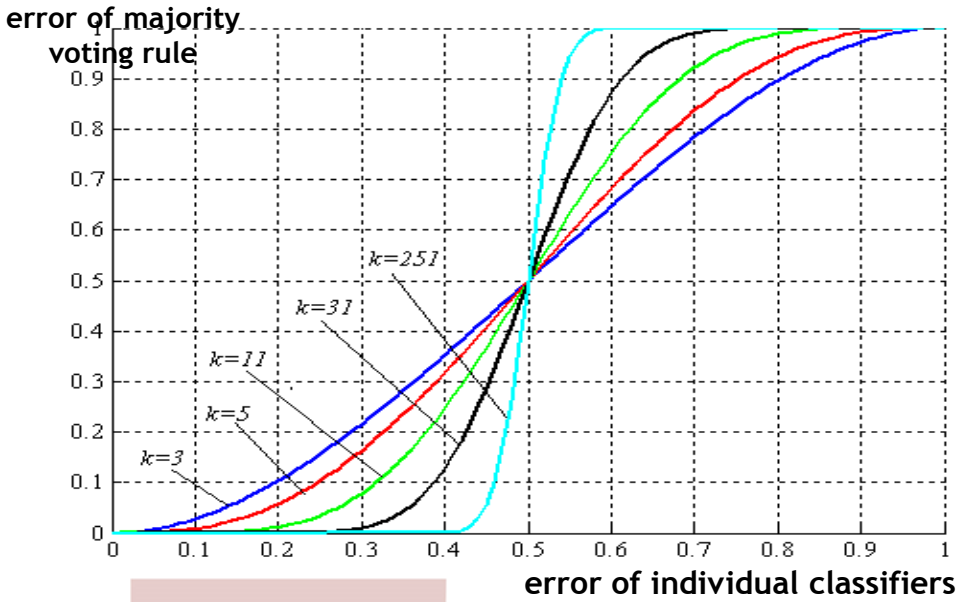
- *k* classifiers are given (1 means correct classification, -1 wrong one),
- Majority voting rule of a pool of *k* classifiers



Combination rule design

- Error of majority voting (for jointly independent errors and probability error *p* of each classifier) according to Bernoulli's equation

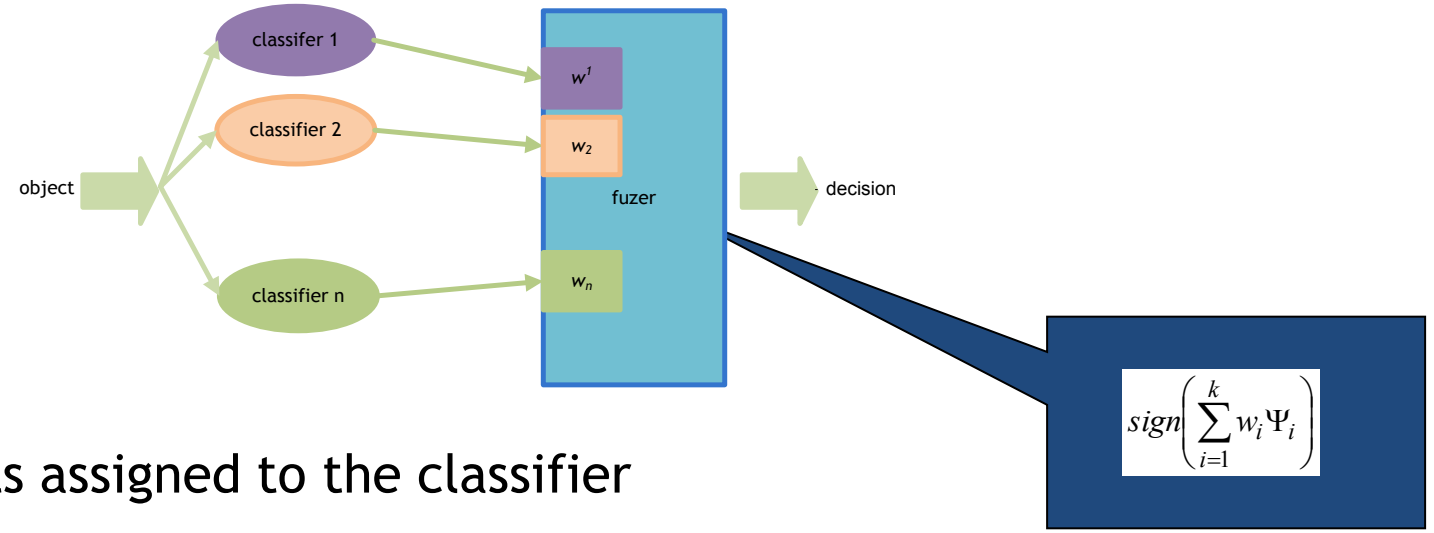
$$P_e^{(k)}(p) = \sum_{i=1}^{\frac{k+1}{2}} \binom{k}{\frac{k-1}{2}+i} p^{\left(\frac{k-1}{2}+i\right)} (1-p)^{\left(\frac{k+1}{2}-i\right)}.$$



- This result is known as the Condorcet Jury Theorem (1786)

Combination rule design

- Weighted voting rule of a pool of *k* classifiers



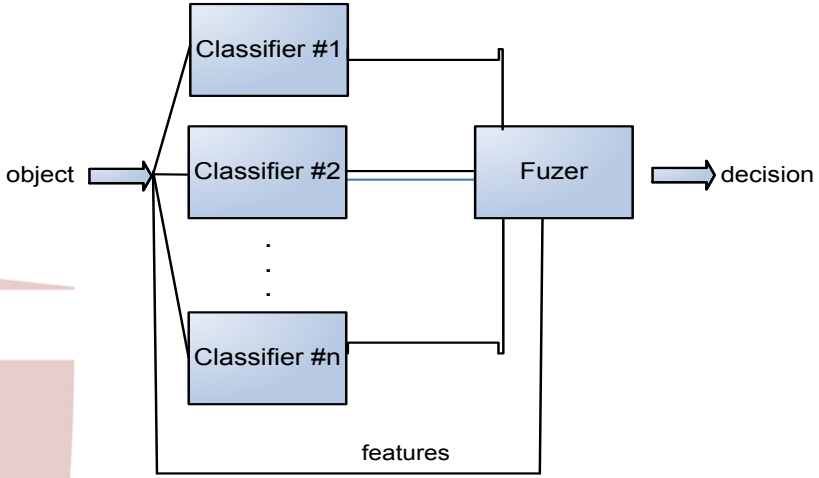
- Weights assigned to the classifier
- Weights assigned to the classifier and the class
- Weights dependent on features values and assigned to the classifier and the class

Combination rule design

- For the aforementioned voting models it is not possible to obtain a classifier which is better than the Oracle, because any decision rule might point to the correct class if at least one classifier produces the correct class label.

- The only model based (partially) on the class label which could achieve better results than the Oracle is a classifier which produces decision on the basis of class labels given by individual classifiers and feature vector values.

(Inoue H., Narihisa H. (2002), Optimizing a Multiple Classifier Systems, LNCS, Vol.2417, 285–294.
Raudys S.(2006), Trainable fusion rules. I. Large sample size case, Neural Networks, 19, 1506-1516.
Raudys S.(2006), Trainable fusion rules. II. Small sample-size effects , Neural Networks 19, 1517-1527)



Combination rule design

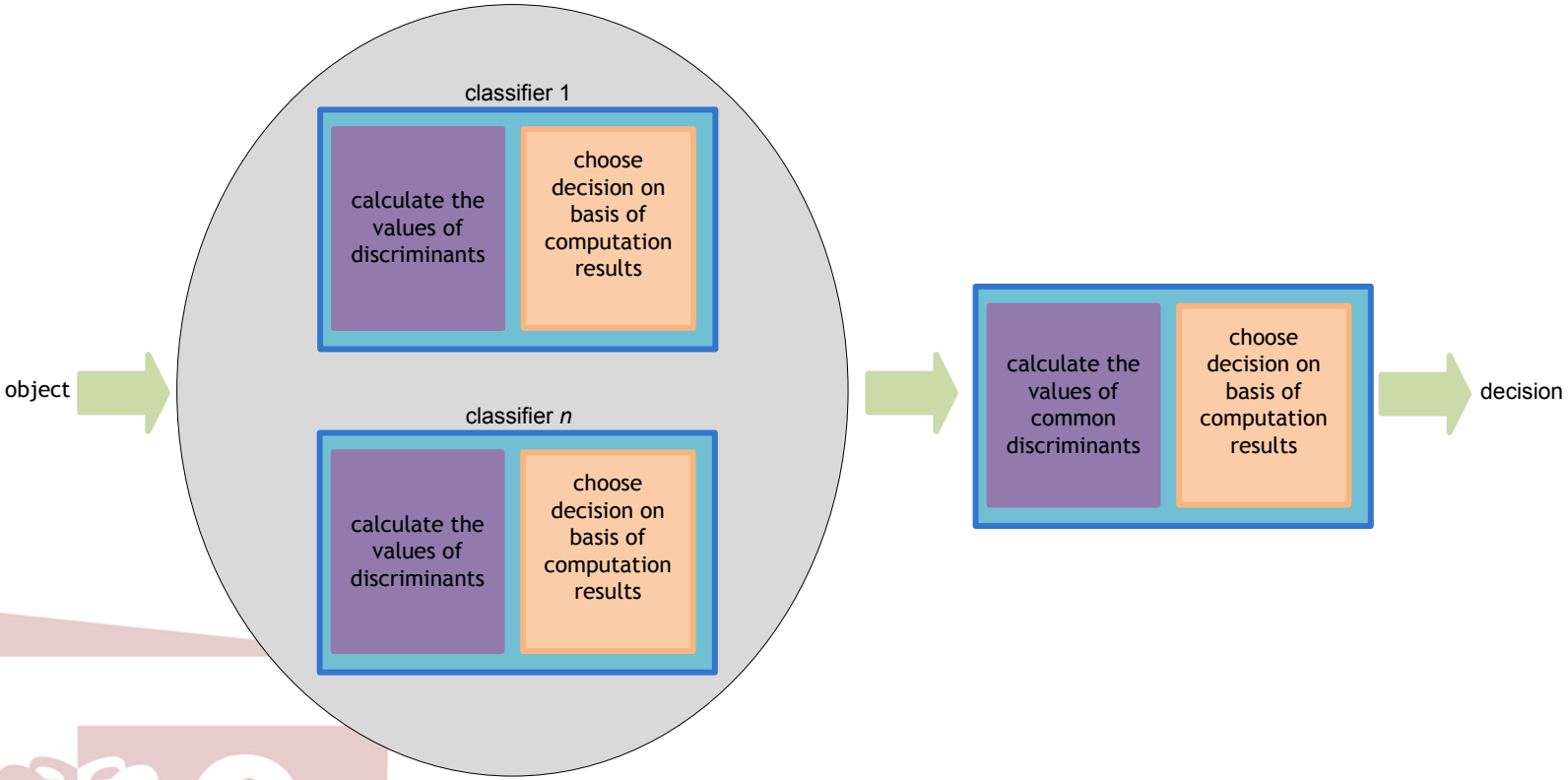
- The Behavior Knowledge Space (BKS) method was proposed by Huang and Suen {Huang:1995}
- The training phase of BKS aims to assign the most popular label from a learning set to each *n*-combination of the individual classifiers' responses.
- The crucial stage of this process is to assign the label to a given *n*-combination of the individual classifiers' response.
- In the case of ambiguous decision, if more than one class are classified by the set of individual classifiers as another one, we should establish decision randomly or choose the class with the highest support (e.g., most probably).
- It is better if we can get the support value, or we can use the most popular decision among individual classifiers.
- The number of *n*-combinations is pretty high M^n, so BKS requires a quite big learning set. The analytical estimation of dependency between BKS's error and learning set size was presented by Raudys {Raudys:2003}

Combination rule design

- **Stacking (Stacked generalization)** is the most general framework for classifier combination based on class labels.
- Many approaches presented before as weighted voting, could be recognized as stacking
- It is not limited by the oracle combination rule.
- Basically, the training is divided into two phases.
  - The first phase is related to training individual classifiers,
  - the second phase is responsible for training a combination rule (also called meta-classifier or meta-level classifier).

Combination rule design

Some classifiers provide class support, i.e., they make decisions on the basis of the values of discriminant functions.



Combination rule design

- **The Borda count** makes a decision by giving each class a certain number of points corresponding to the position in which it is ranked by each individual classifier.

Jean-Charles, chevalier de Borda (1733 – 1799) was a French mathematician, physicist, political scientist, and sailor. *(Wikipedia)*

Combination rule design – Borda count

rank			rank value
classifier 1	classifier 2	classifier 3	
B	C	B	5
A	A	A	4
E	D	D	3
D	E	E	2
C	B	C	1

rank for class A	4+4+4=12
rank for class B	5+1+5=11
rank for class C	1+5+1=7
rank for class D	2+3+3=8
rank for class E	3+2+2=7

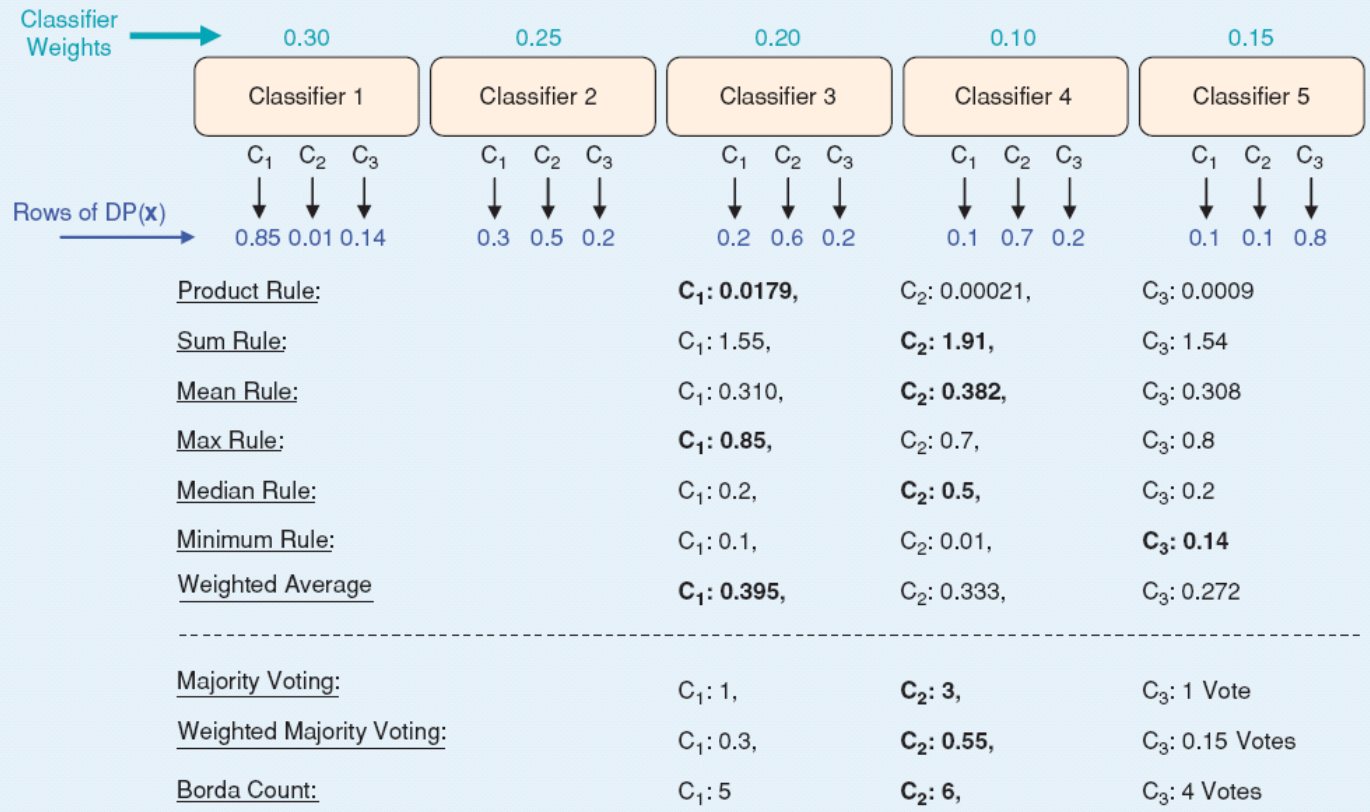
Let us note that Oracle and other voting methods use only the most highly ranked class and they decide that given object belongs to class B.

Combination rule design

- An alternative model for the construction of a combining classifier, one that performs classifier fusion on the basis of the discriminants of classifiers.
- The main form of discriminants are *posterior* probability estimators, but it could be given for e.g., by the output of neural networks or that of any other function whose values are used to establish the decision of the classifier.
- The aggregating methods (nontrainable) perform fusion with the help of simple operators, such as the maximum or average, but they are typically relevant only in specific, clearly defined conditions.

Duin R. P.W.(2002), The Combining Classifier: to Train or Not to Train?, *Proc. of the ICPR2002*, Quebec City.

Combination rule design



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

Figure 15. Example on various combination rules.

Figure reproduced by permission of R.Polikar

Polikar R., "Ensemble based systems in decision making," *IEEE Circuits and Systems Mag.*, vol. 6, no. 3, pp. 21-45 , 2006.

Combination rule design

- Weighting methods are an alternative and the selection of weights has a similar importance as in the case of weighted voting.
- The advantages of this approach include an effective counteraction against the occurrence of elementary classifier overtraining.

Mixture of experts

Mixture of experts proposed by Jacobs et al. {Jacobs:1991} is based on the *divide-and-conquer* principle.

- It proposes to select the most competent classifier (expert) for a given class and for a given observation supervised by so-called gating network.
- Therefore, it can be recognized as the dynamic selection of the classifiers. Also, the final decision of such a system is made on the basis of sampling a classifier pool according to their competence, decision of the most competent classifier, or weighted averaging, where weights depend on the competence of individual classifier for a given problem, i.e., they depends on attributes and class labels.
- The most important issue related to the discussed model is how to train it, and optimization methods usually used for neural networks learning are applied to deal with this problem as gradient descent {Fancourt:1998}.

Decision template

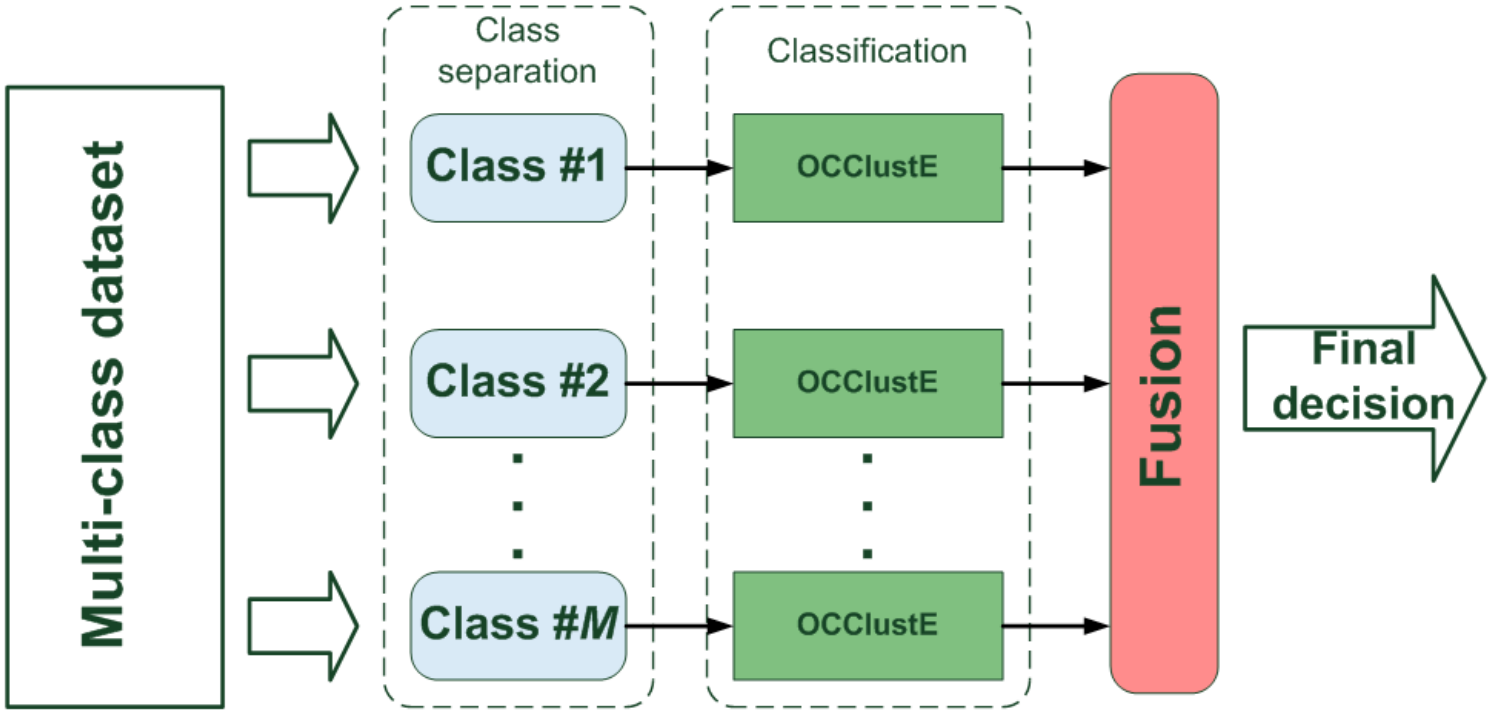
Kuncheva et al. proposed **Decision Template** framework for classifier fusion {Kuncheva:2001}

- It estimates the most typical profile of individual answers for each class.
- The decision about a given object *x* is made on the basis of similarity measure, which returns the most similar class *i* represented by decision template DT*i* to the object.
- The similarity measure plays a crucial role during decision making.
- Kuncheva et al. made discussions and comparisons on the basis of computer experiments, and several measures had been applied, such as negative squared Euclidian distance or fuzzy logic.
- Rogova {Rogova:1994} proposed Dempster-Shafer fuser, and which can be recognized as a kind of Decision Template method using similarity distance based on Dempster-Shafer theory of evidence.

Classifier selection

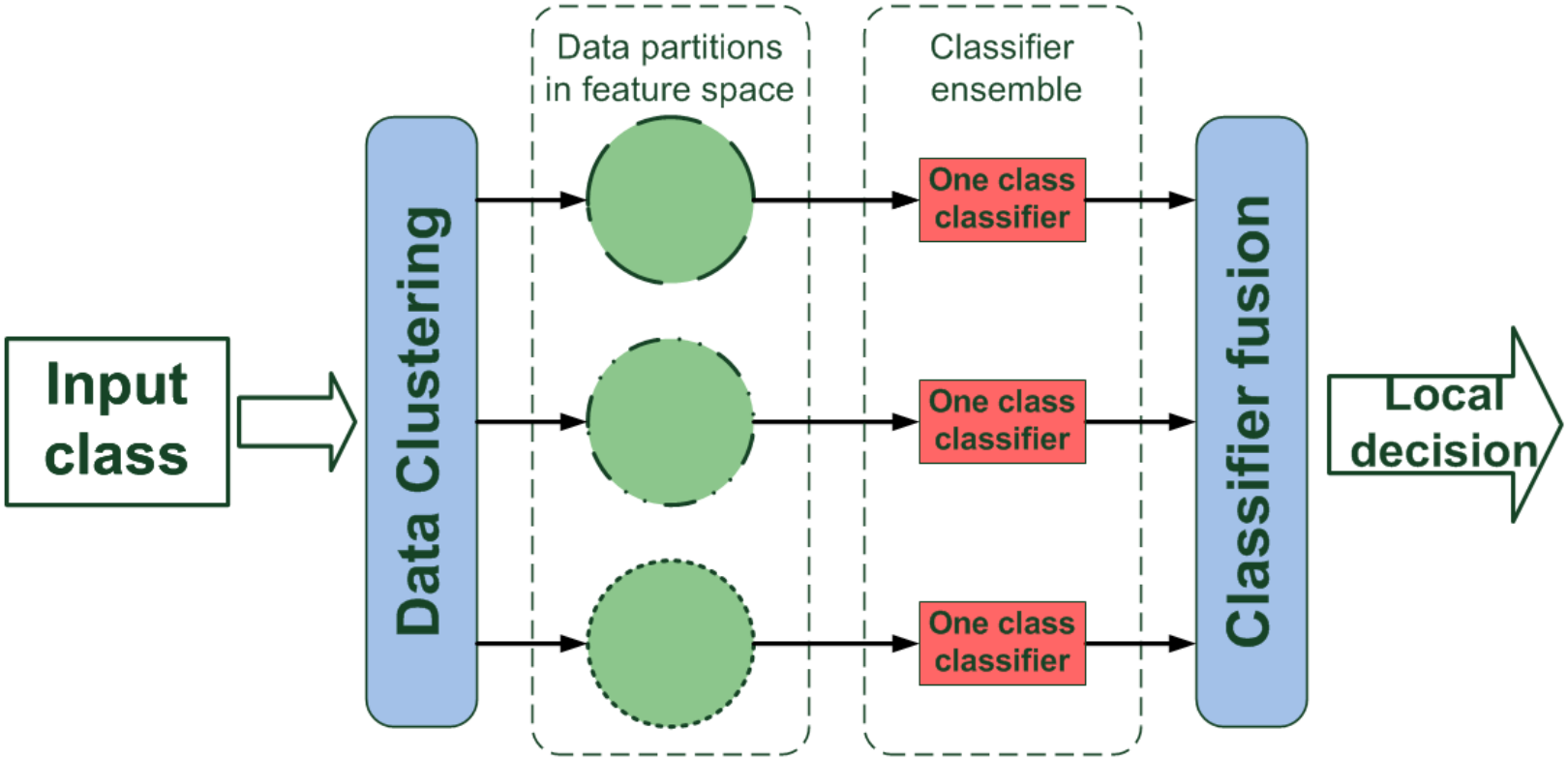
- The classifier selection assumes a local specialization of individual classifiers. According to this proposal, a single classifier that achieves the best results is chosen from a pool for each demarcated partition of the feature space. Its answer is treated as the system answer, for all objects included in the partition.
- This methodology was described by Rastrigin and Erenstein (Rastrigin and Erenstein, 1981). Certain proposals based on this idea assume a local specialization of particular classifiers and only search for locally optimal solutions (Baram, 1998; Cordella *et al.*, 2000; Giacinto *et al.*, 2000; Goebel and Yan, 2004; Ruta and Gabrys, 2005), while other methods propose dividing the feature space and selecting (or training) a classifier for each partition (Kuncheva, 2000; Baruque *et al.*, 2011).
- Jackowski and Wozniak propose AdaSS (Adptive Splitting and Selction), which uses evolutionary algorithm to split the feature space into competence areas and assining and a dedicated classifier esemble to each o them. The final decision is made on the basis on weighted aggregation.

OCClustE - One-class Clustering-based Ensemble

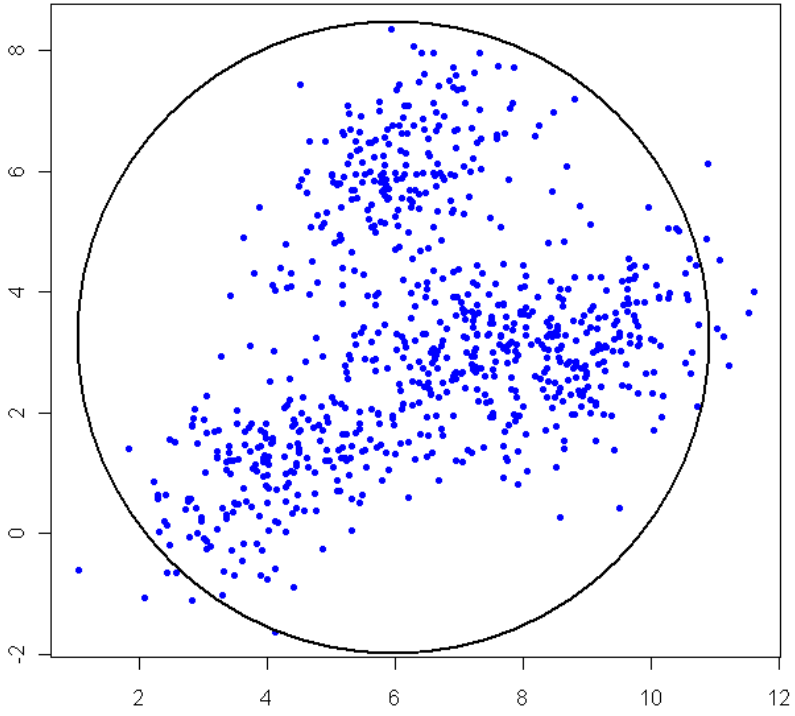


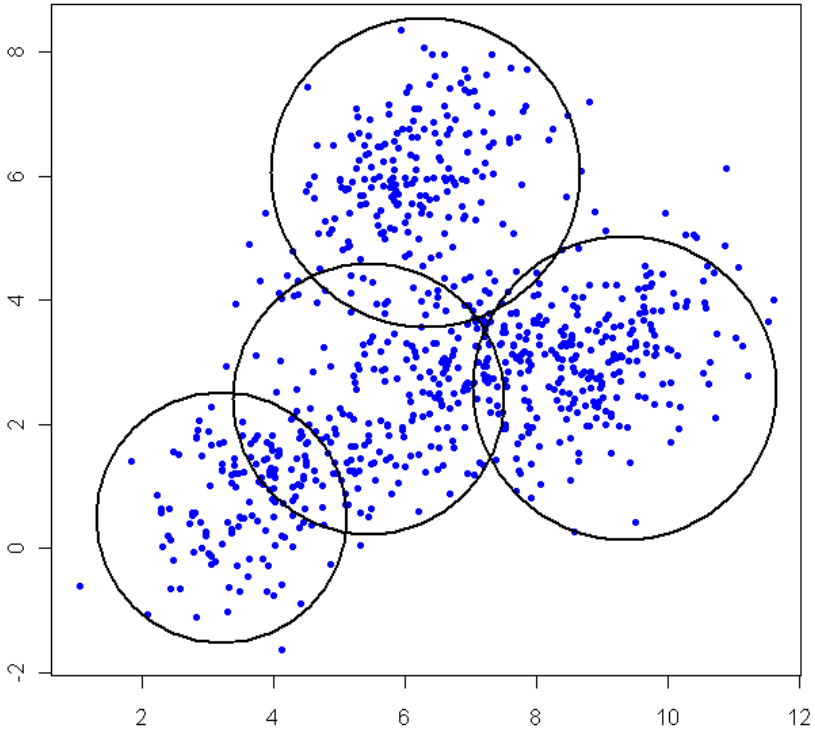
Krawczyk B., Woźniak M., Cyganek B., Clustering-based ensembles for one-class classification, Information Sciences, Volume 264, 20 April 2014, Pages 182–195

OCClustE - One-class Clustering-based Ensemble



OCClustE - One-class Clustering-based Ensemble





More on MCSs see

Wozniak M., Grana M., Corchado E., A survey of multiple classifier systems as hybrid systems, Information Fusion, Volume 16, March 2014, Pages 3–17

Data stream classification

- The market-leading companies realize that smart analytic tools which are capable of analyzing the collected and fast-growing data could lead to business success.
- In designing such solutions, we have to seriously consider that in the modern world most of the data arrive continuously, and it causes that the analytic tools should realize the relevant nature and be able to interpret so-called data streams.

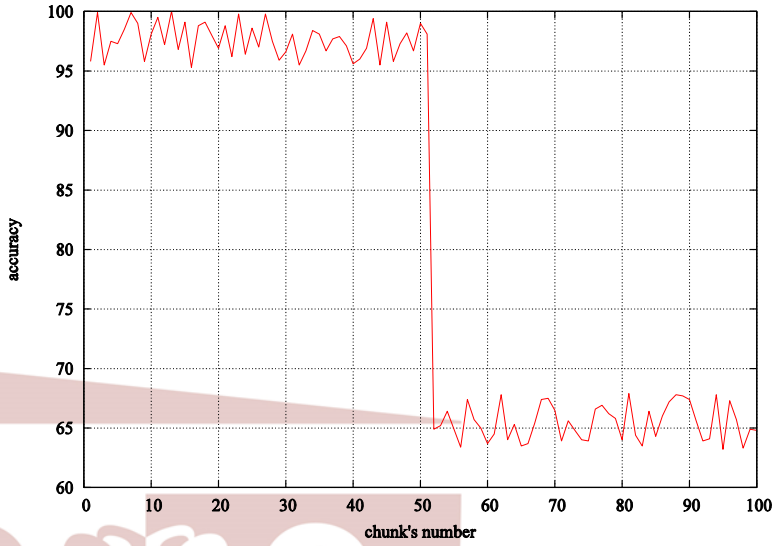
Data stream classification

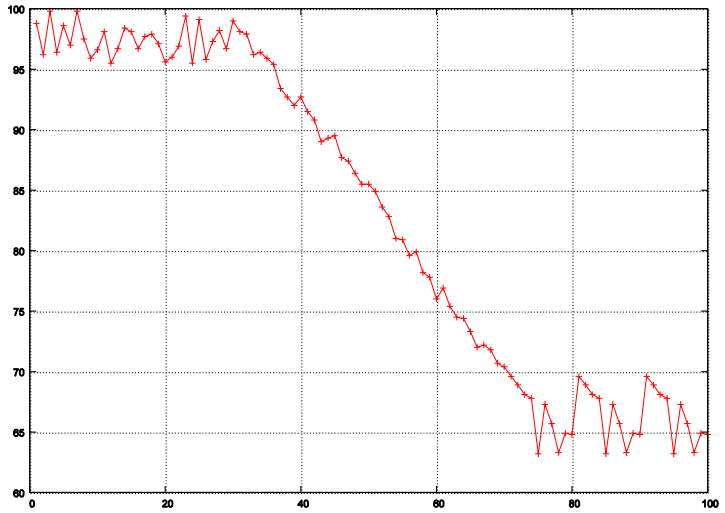
Most of the traditional classifier design methods do not take the following points into consideration:

- The statistical dependencies between the observations of the given objects and their classifications could change
- Data can come flooding in the analyzer, what causes that it is impossible to label all records

Concept drift

Appearance of concept drift can potentially cause a significant accuracy deterioration of an exploiting classifier





Concept drift

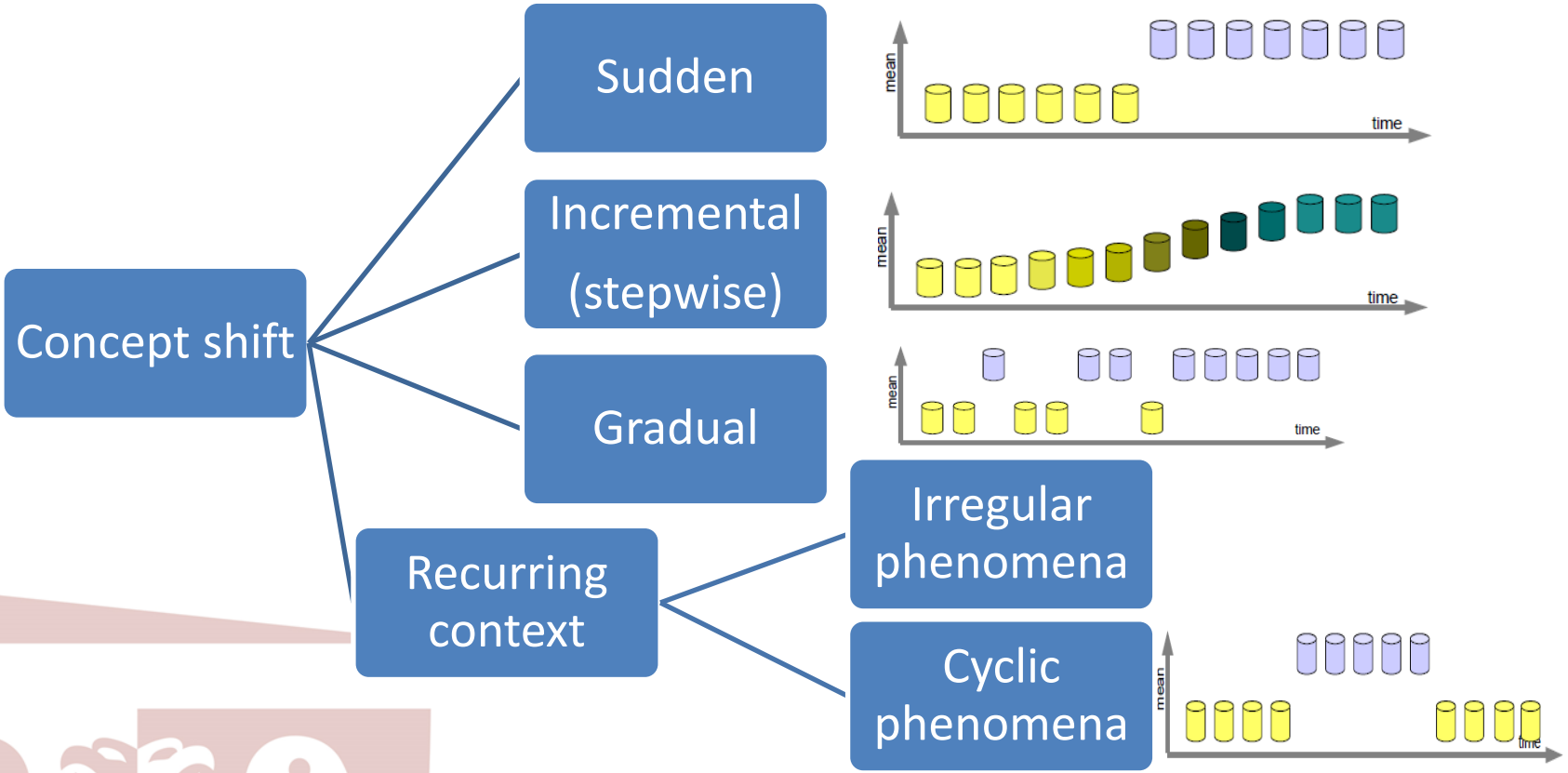
- SPAM detection
- Customer's behavoiur may change over time
- Fraud detection
- Energy consuming
- Sensors etc.

Real and virtual concept drift

We can classify the drift on the basis of its rapidity (abrupt or smooth), or according to its influence on the probabilistic characteristics of the classification task:

- **virtual concept drift** means that changes do not impact the decision boundaries (*posterior* probabilities), but affect the conditional probability density functions.
- **real concept drift** means that changes affect the decision boundaries (*posterior* probabilities) and may impact unconditional probability density function.

Concept drift



Concept drift

- The data stream can be noisy and includes outliers, but they are not considered as the concept drift, because outliers and noise have the random nature and should be ignored.
- We require such a data stream classifier which is robust to noise and sensitive to concept drift.

Concept drift

The following approaches can be considered to deal with the above problem.

- Rebuilding a classification model if new data becomes available. It is very expensive and impossible from a practical point of view, and especially for which the concept drift occurs rapidly.
- Detecting concept changes in new data, and if these changes are sufficiently significant then rebuilding the classifier.
- Adopting an incremental learning algorithm for the classification model.

Concept drift

We can divide these algorithms into four main groups:

1. Online learners
2. Instance based solutions (also called sliding window based solutions)
3. Ensemble approaches
4. Drift detection algorithms

Online learners

This group relates to the family of algorithms that continuously update the classifier parameters while processing the incoming data. Not all types of classifiers can act as online learners; they have to meet some basic requirements (Domingos:2003):

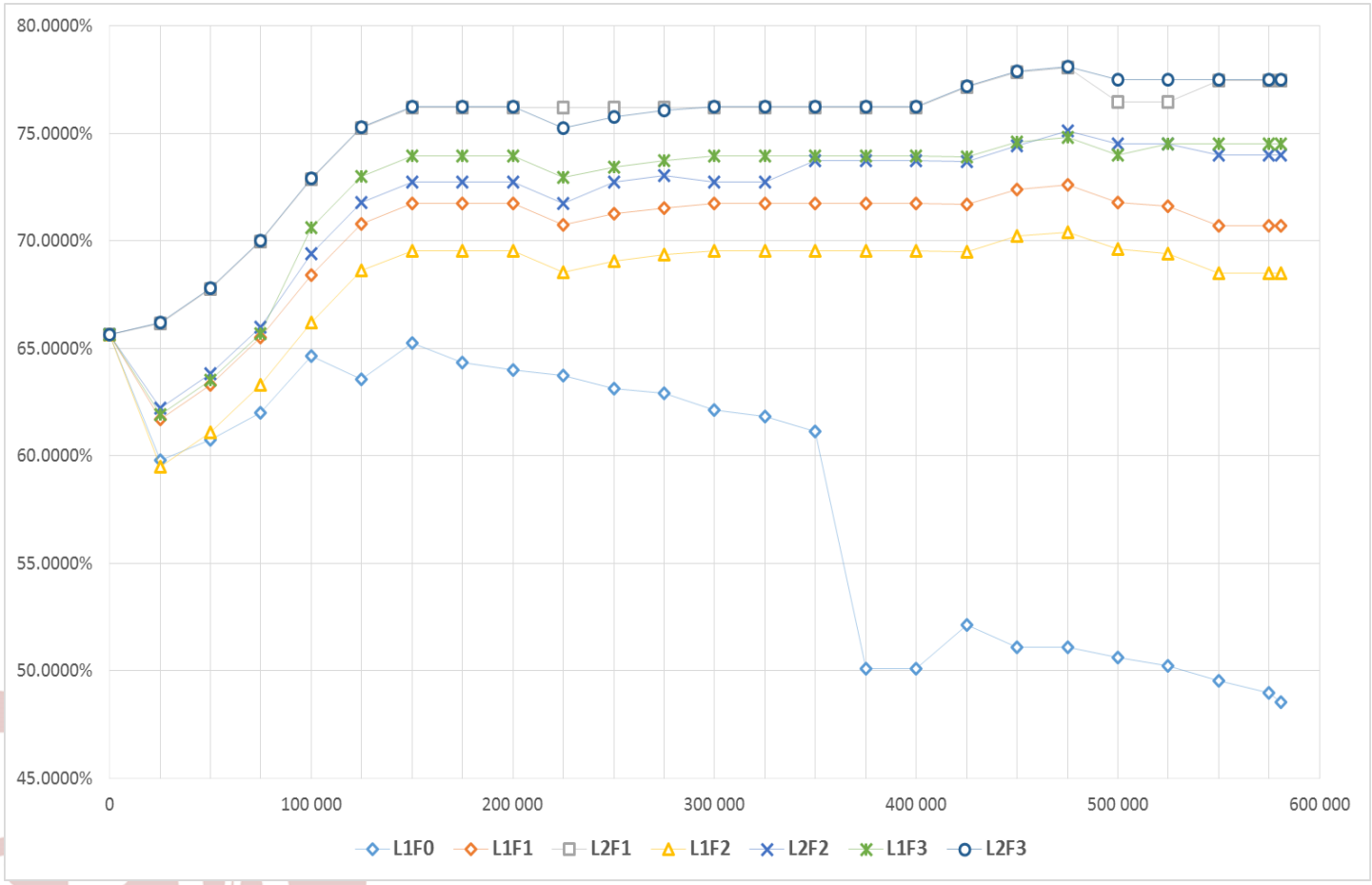
- Each object must be processed only once in the course of training.
- The system should consume only limited memory and processing time, irrespective of the execution time and amount of data processed.
- The training process can be paused at any time, and its accuracy should not be lower than that of a classifier trained on batch data collected up to the given time.

Classifiers that fulfill these requirements work very fast and can adapt their model in a very flexible manner.

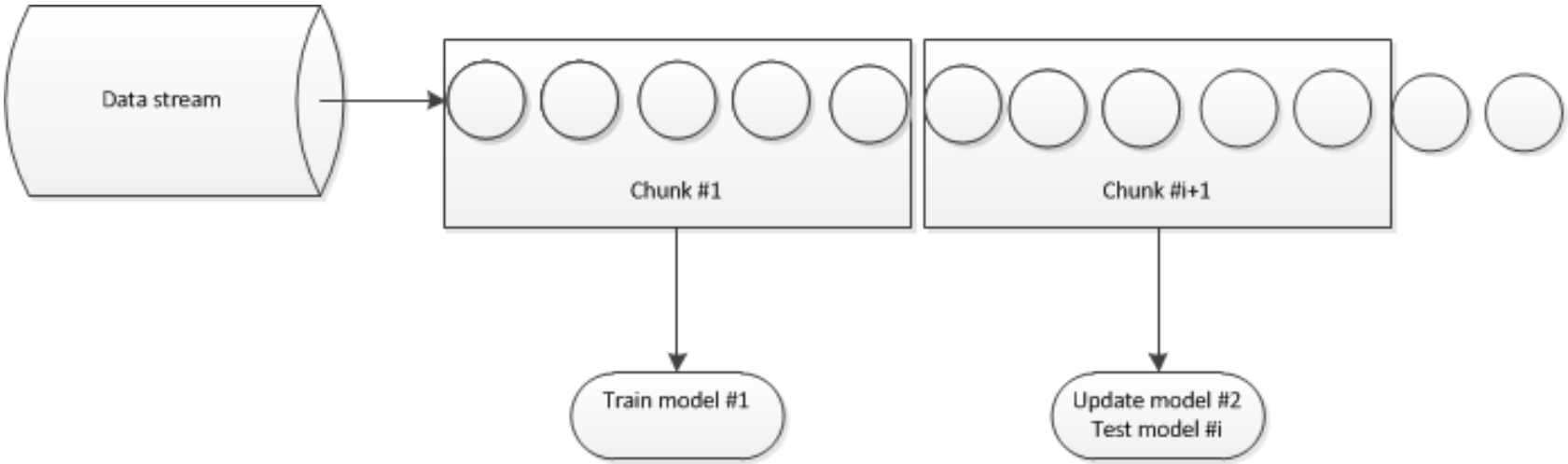
Online learners

- Among the others, the following are the most popular online learners: Naïve Bayes, Neural Networks, and Nearest Neighbour.
- A more sophisticated solution CVFDT (Concept-adapting Very Fast Decision Tree) was proposed by Hulten.It is an extended version of the ultra fast decision tree, which ensures consistency with incoming data by maintaining alternative subtrees. CVFDT replaces the outdated tree when its respective alternative is more accurate.
- Krawczyk and Wozniak propose the WOCSVM (Weighted One Class SVM) with forgetting (see Krawczyk B., Wozniak M., One-class classifiers with incremental learning and forgetting for data streams with concept drift, Soft Comput, DOI 10.1007/s00500-014-1492-5, 2014)

COV data set (accuracy)



Test/Train phases



Sliding windows

- This group consists of algorithms that incorporate the forgetting mechanism.
- This approach is based on the assumption that the recently arrived data are the most relevant, because they contain characteristics of the current context.
- However, their relevance diminishes with the passage of time.

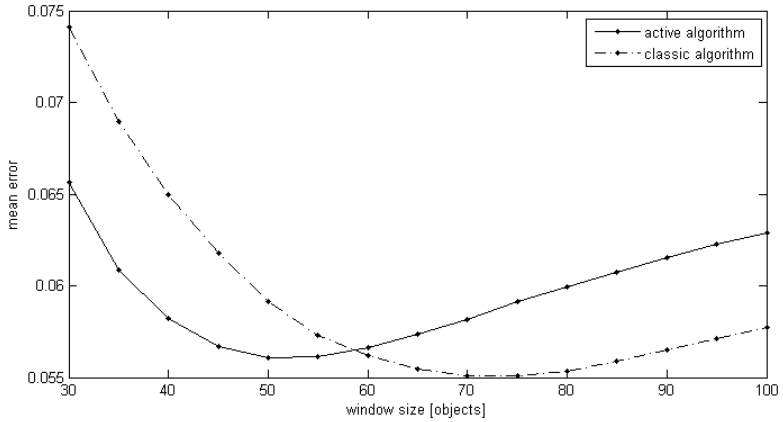
Sliding windows

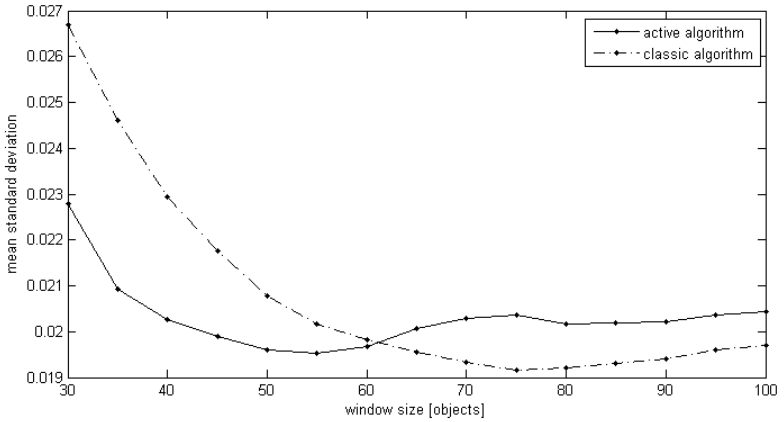
- Therefore, narrowing the range of data to those that were most recently read may help form a dataset that embodies the actual context.
- There are three possible strategies here:
  - selecting the instances by means of a sliding window that cuts off older instances (Widmer:1996);
  - weighting the data according to their relevance;
  - applying bagging and boosting algorithms that focus on misclassified instances (Bifet:2009, Chu:2004}.

Sliding windows

When dealing with the sliding window the main question is how to adjust the window size.

- A shorter window allows focusing on the emerging context, though data may not be representative for a longer lasting context.
- A wider window may result in mixing the instances representing different contexts.





Sliding windows

- Therefore, certain advanced algorithms adjust the window size dynamically depending on the detected state (e.g., FLORA2 {Widmer:1996} and ADWIN2 {Bifet:2007}).
- In more sophisticated algorithms, multiple windows may even be used {Lazarescu:2004}.
- In object weighting algorithms the relevance of the instance is used to calculate its weight, which is usually inversely proportional to the time that has passed since the instance was read {Klinkenberg:1998,Koychev:2000}. Cohen and Stauss propose to use 3 types of decay functions (expotential, polinominal and chordal one).

Sliding windows

- Ziolbate proposed FISH algorithms (FISH, FISH2, FISH3), which combine distance in the attribute space and the distance in time with the k-NN to select object to be labelled.
- Kurlej and Wozniak proposed an active learning approach {Kurlej:2012} to select valuable examples for the k-NN classifier.
- The computer experiments confirmed that active learning approach behaves well, especially for the small windows size.

Drift detection

- Algorithms that address the question of when drift occurs.
- Not all classification algorithms dealing with concept drift, require drift detection. Some evolving systems continuously adjust the model to incoming data {Zliobaite:2010}.
- This technique is called implicit drift detection {Kuncheva:2008} as opposed to explicit drift detection methods that raise a signal to indicate change.

Drift detection

- The detector can be based on changes
  - in the probability distribution of the instances {Gaber:2006,Markou:2003,Salganicoff:1993}
  - or classification accuracy {Klinkenberg:2000,Baena-Garcia:2006}.
- Many detection algorithms base on a knowledge of object labels after the classification in order to detect concept drift, however as pointed out in {Zliobaite:2010}, such approach does not fit in the real scenarios.
- In general, concept drift detection algorithms can be divided into three types, depending on the assumption about the amount of costly knowledge regarding the true class labels available for the algorithm.

Drift detection - supervised algorithms

- assuming access to classification performance measures or true class labels,
- detecting concept drift on the basis of classifier's accuracy or analysis of class distributions,
- although an access to this knowledge is often very expensive and in many practical cases it is impossible to label data e.g., because objects are coming very fast {Kifer:2004}.

Drift detection - supervised algorithms

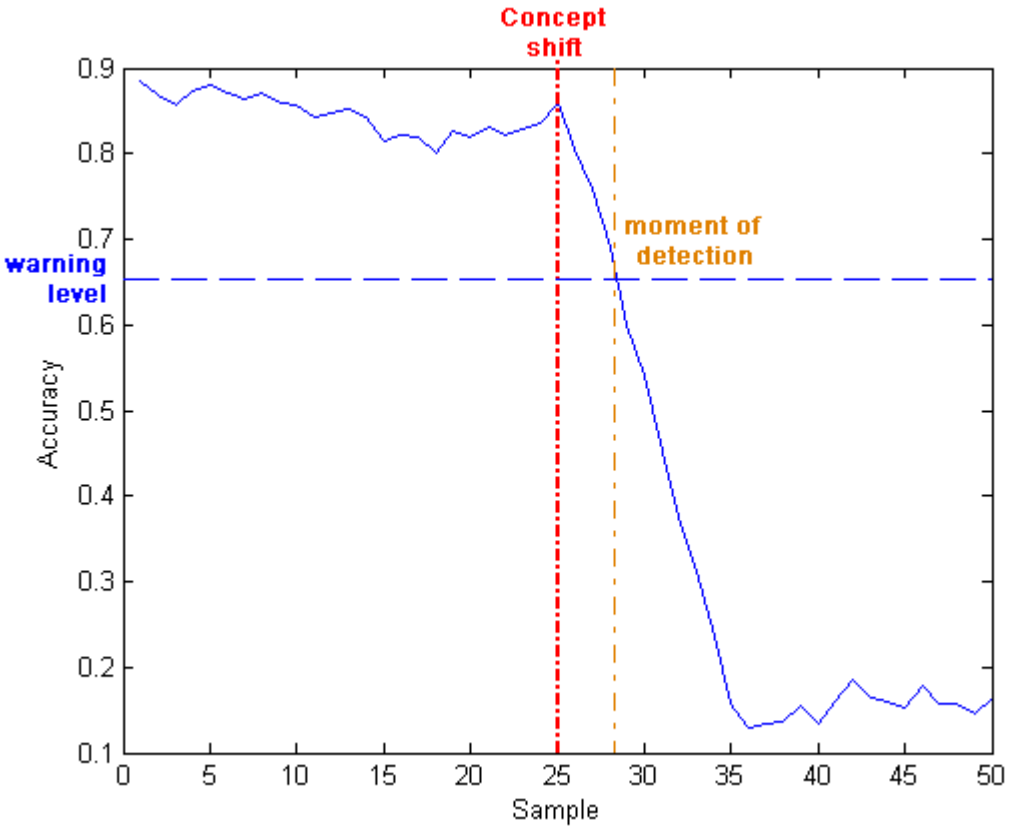
Data model

Classification system

Detector

Expert

Drift detection - supervised algorithms



Drift detection - supervised algorithm

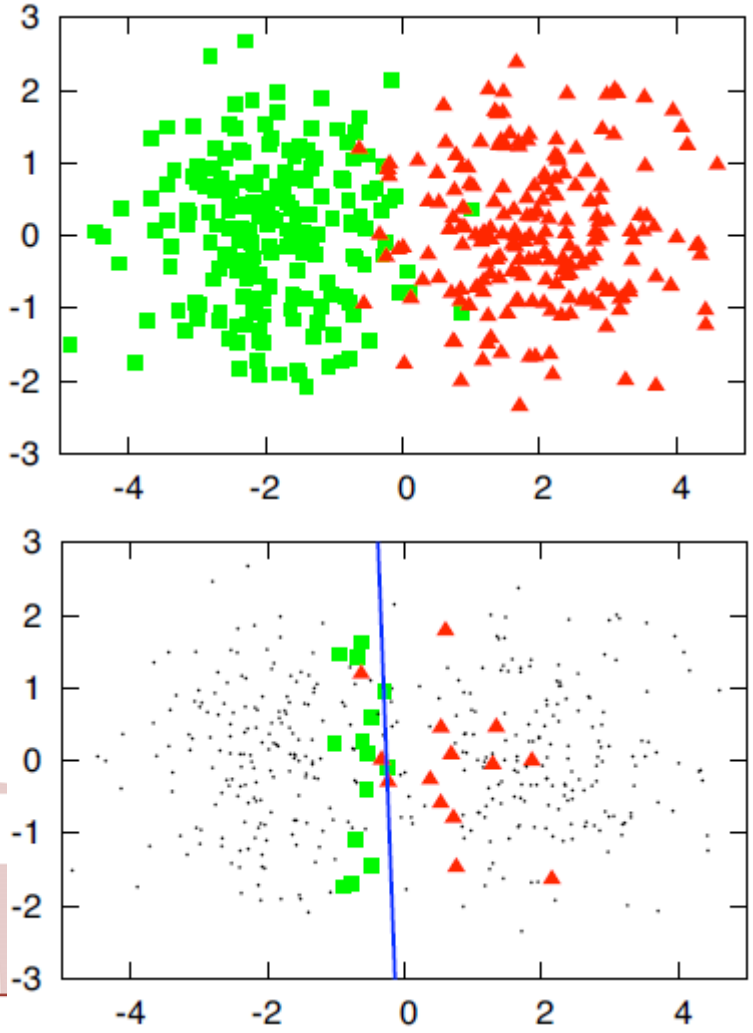
Weakness

- Access to classification performance measures or true class labels – usually it is hard to be granted, e.g.,
  - Medical diagnosis – human expert should verifies the diagnosis;
  - Credit application (the true label is available ca. 2 years after the decision);
  - Spam filtering – user should confirm the decision

Drift detection - semisupervised algorithms

- Assuming limited access to classification performance measures or true class labels,
- also detecting concept drift on the basis of the properties of data when such knowledge is not available – a more “rigorous” approach, taking into account a cost of labeling,
- a flag example in this category is active learning {Kurlej:2011a,Greiner:2002}, which selects the samples for labeling

Drift detection - semisupervised algorithms



Drift detection – non-supervised algorithms

- assuming no access to classification performance measures or true class labels,
- basing only on the properties of data,
- detecting concept drift on basis of attribute value distribution, cluster memberships or classifier's support levels – after detecting concept drift,
- usually the labels or knowledge about classification error is still necessary to train a new classification model.

Drift detection – non-supervised algorithms

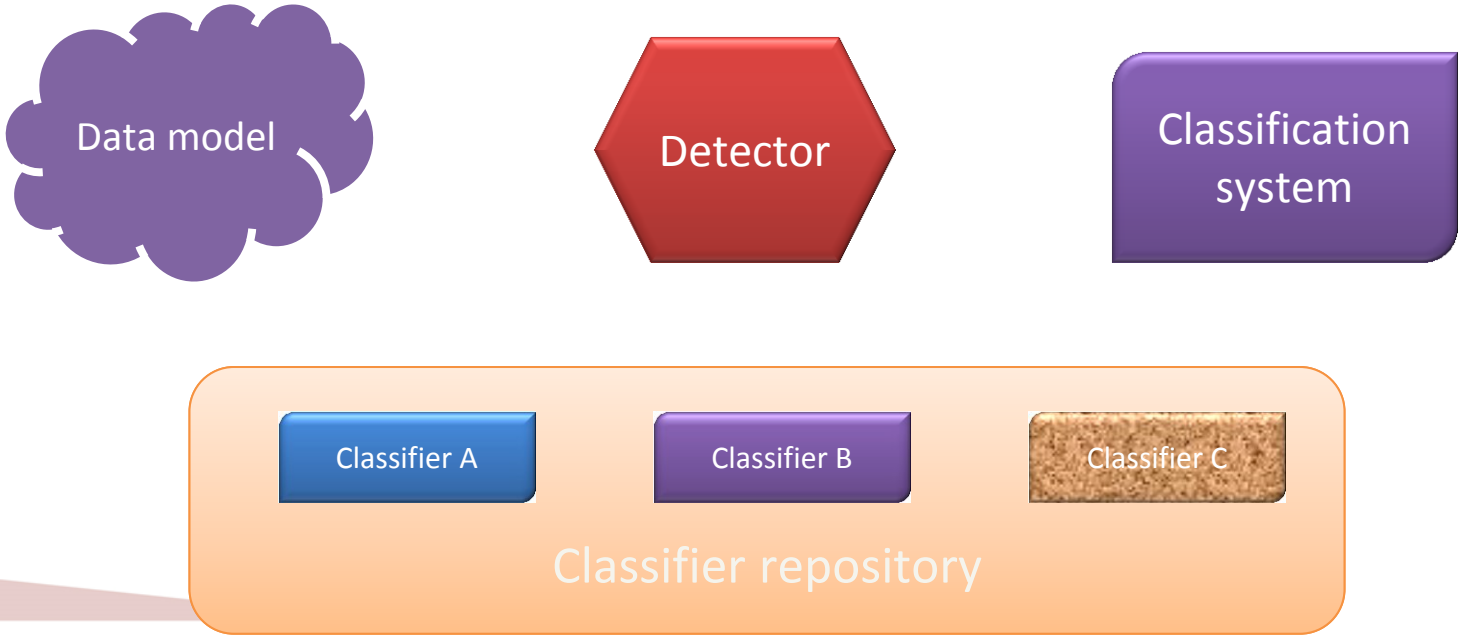
Drift detection – non-supervised algorithms

- Sobolewski and Wozniak explored the possibilities of detecting concept drift in data streams without any supervision {Sobolewski:2013},
- such approach has some limitations:
  - it is suited only for virtual concept drift, as the real concept drift is undetectable by analysing solely the properties of data,
  - there are certain situations when also virtual concept drift is impossible to detect, e.g. when classes swap places.

Drift detection – recurring concept

- Among the machine learning methods dealing with concept drift, a new class has recently emerged {Ramamurthy:2007}, comprising algorithms that process data streams featuring a recurring context.
- Two additional requirements are imposed on algorithms in this class:
  - the system should maintain knowledge of previously emerged contexts,
  - it should be effective in recognizing contexts and switching to a valid one.
- Both issues can be effectively solved by an ensemble system.

Drift detection – non-supervised algorithms for recourring concept drift



Ensemble approach

-
- As it was mentioned a collective decision can increase classification accuracy because the knowledge that is distributed among the classifiers may be more comprehensive. This premise is true if the set consists of diverse members {Shipp:2002}.
- In static environments, diversity may refer to the classifier model, the feature set, or the instances used in training

Ensemble approach

- In a changing environment diversity can also refer to the context.
- Several strategies are possible for a changing environment:
  - Dynamic combiners, where individual classifiers are trained in advance and their relevance to the current context is evaluated dynamically while processing subsequent data. The level of contribution to the final decision is directly proportional to the relevance {Littlestone:1994,Jacobs:1991}. The drawback of this approach is that all contexts must be available in advance; emergence of new unknown contexts may result in a lack of experts.
  - Updating the ensemble members, where each ensemble consists of a set of online classifiers that are updated incrementally based on the incoming data {Fern:2003,Oza:2000,Kolter:2007,Bifet:2011,Bifet:2009,Rodriguez:2008}.
  - Dynamic changing line-up of ensemble e.g., individual classifiers are evaluated dynamically and the worst one is replaced by a new one trained on the most recent data {Jackowski:2013a,Kolter:2003}.

Dynamic combiners

- **Horce racing approach**

Let's $b \in [0,1]$

1: train all classifiers in ensemble

2: establish the same weight for each individuals.

3: weight for new example

4: use weighted voting to classify the example

5: update the weights of the classifiers that err (weight<-b*weight)

6: go to 3

Dynamic combiners

- **Winnow algorithm**

Let's a>1

1: train all classifiers in ensemble

2: establish the same weight for each individuals.

3: weight for new example

4: use weighted voting to classify the example

5: update the weights of the classifiers that err (weight<-weight/a)

6. update the weights of the classifiers that do not err (weight<-a*weight)

7: go to 3

Updating the ensemble members

Online bagging {Oza:2001}

1: Initialize set of *L* individuals

2: For each new example randomly choose how many times the example would have appeared in the data set used by each individuals (usually according to Poisson distribution).

3: Add examples to datasets and retraing individuals

4. Go to 2

Online boosting {Oza:2001}

2:We should retrain the whole ensemble, i.e., if the example is hard for the first individual then it is more representative in the treining set for the second individual etc.

Changing line-up

- Among the most popular ensemble approaches, the following are worth noting:
  - the Streaming Ensemble Algorithm (SEA) {Street:2001}
  - the Accuracy Weighted Ensemble (AWE) {Wang:2003}.
- Both algorithms keep a fixed-size set of classifiers. Incoming data are collected in data chunks, which are used to train new classifiers.
- All the classifiers are evaluated on the basis of their accuracy and diversity (SEA) and the worst one in the committee is replaced by a new one if the latter has higher accuracy.
- The SEA uses a majority voting, whereas the AWE uses the more advanced weighted voting.

Changing line-up

- Adaptive Classifier Ensemble (ACE), where ACE's ensemble consists of one on-line learner and several batch classifiers and a drift detector.
- The drift detector checks follows the errors of each bach classifiers and if the best one falls outside user-defined confidence interval the signals the drift.
- If the drift is detected or number of the buffered examples exeded the chunk's size, then the new batch classifier is trained on the buffered examples and on-line classifier is reset.
- The final decision is made on the basis of the weighted voting (weights depend on individual's accuracy on the recent chunk).

Changing line-up

- Dynamic Weighted Majority (DWM) algorithm {Kolter:2003} forms the ensemble of online classifiers.
- DWM modifies the weights and updates the ensemble in a more flexible manner - the weight of the classifier is reduced when the classifier makes an incorrect decision (after the each example).
- Frequently, after fixed number of predictions, a new classifier is added to the ensemble when the committee makes a wrong decision.
- It could cause that number of individuals will increase endlesly, then ensemble pruning is required.

Changing line-up

- Jackowski proposes an classifier ensemble training methods dedicated so-called recurring context (i.e., when classification model can change, but the old models could reappear) {Jackowski:2013}.
- The proposed combined classifier collects information on emerging contexts in a pool of elementary classifiers trained on subsequent data chunks, and the pool is updated only when concept drift is detected.
- Classifiers are not removed from the pool, and therefore, knowledge of past contexts is preserved for future use.
- To select an ensemble for the current model the ensemble pruning method based on evolutionary programming is employed.

Changing line-up

- Wozniak et al. propose the dynamic ensemble model called *Weighted Aging Ensemble* (WAE) {Wozniak:2013} which can modify line-up of the classifier committee on the basis of diversity measure.
- Additionally the decision about object's label is made according to weighted voting, where weight of a given classifier depends on its accuracy and time spending in an ensemble.

WAE Weighted Aging Classifier Ensemble

- We assume that the classified data stream is given in a form of data chunks denotes as LS_k, where k is the chunk index.
- The concept drift could appear in the incoming data chunks.
- Instead of drift detection WAE tries to construct self-adapting classifier ensemble. Therefore on the basis of the each chunk one individual is trained and we check if it could form valuable ensemble with the previously trained models.
- The size of the ensemble if fixed an ensemble pruning procedure is used (based on linear combination of accuracy and diversity – generalized diversity).
- We propose to antiaging procedure of an individual if it has the higger accuracy than the average accuracy of iclassifier in the pool.. This could be useful especially in the case of recurring concept drift.

1. **The same weights for each classifier** in the pool, i.e., majority vote is use as the combination rule

$$w(\Psi_i) = \frac{1}{|\Pi|} \tag{14}$$

2. **Weights proportional to classifier accuracy**

$$w(\Psi_i) = P_a(\Psi_i) \tag{15}$$

3. **Aged weights proportional to classifier accuracy** used by the original WAE algorithm [50]

$$w(\Psi_i) = \frac{P_a(\Psi_i)}{\sqrt{itter(\Psi_i)}} \tag{16}$$

This weight calculation includes aging as well (i.e., forgetting).

4. **Kuncheva's weights** - suggested by Kuncheva in her book [13]

$$w(\Psi_i) = \frac{P_a(\Psi_i)}{1 - P_a(\Psi_i)} \tag{17}$$

5. **Weights proportional to accuracy related to the whole ensemble accuracy (denotes as $P_a^{ensemble}(\Pi)$)**

$$w(\Psi_i) = \begin{cases} \frac{P_a(\Psi_i)}{P_a^{ensemble}(\Pi)} \text{ if } \frac{P_a(\Psi_i)}{P_a^{ensemble}(\Pi)} > \theta \\ 0 \text{ otherwise} \end{cases} \tag{18}$$

where *theta* $\in [0, 1]$ is the parameter which responsible for the removing less important classifiers.

6. **Weights proportional to accuracy related to the whole ensemble accuracy (denotes as $P_a^{ensemble}(\Pi)$) using bell curve**

$$w(\Psi_i) = \begin{cases} \frac{1}{2\pi} exp \frac{P_a(P_a^{ensemble}(\Pi) - \Psi_i)}{2} \text{ if } \frac{1}{2\pi} exp \frac{P_a(P_a^{ensemble}(\Pi) - \Psi_i)}{2} > \theta \\ 0 \text{ otherwise} \end{cases} \tag{19}$$

where *theta* $\in [0, 1]$ as previously stands for the parameter responsible for the removing less important classifiers.

1. **Aged weights proportional to classifier accuracy** used by the original WAE algorithm [50] and presented in the previous section.

$$w(\Psi_i) = \frac{P_a(\Psi_i)}{\sqrt{itter(\Psi_i)}} \tag{20}$$

It was described in the previous section.

2. **Constant aging**

$$w(\Psi_i) = \begin{cases} w(\Psi_i) = w(\Psi_i) - \delta \text{ if } w(\Psi_i) = w(\Psi_i) - \delta > \theta \\ 0 \text{ otherwise} \end{cases} \tag{21}$$

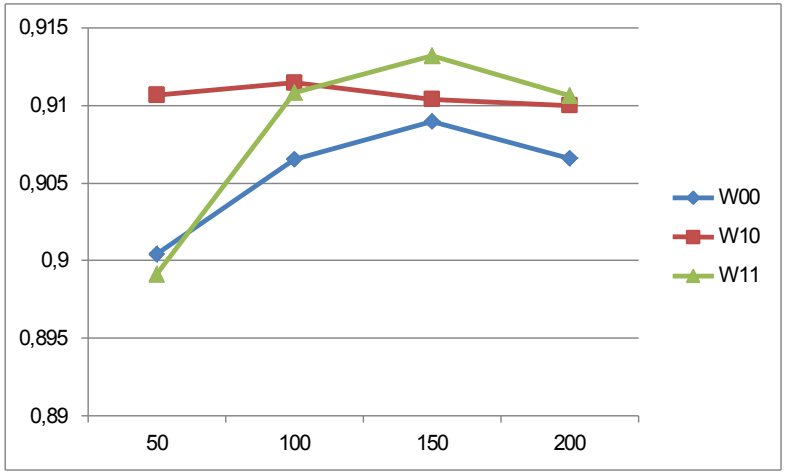
where $theta \in [0, 1]$ as previously stands for the parameter responsible for the removing less important (old enough) classifiers.

3. **Gaussian aging**

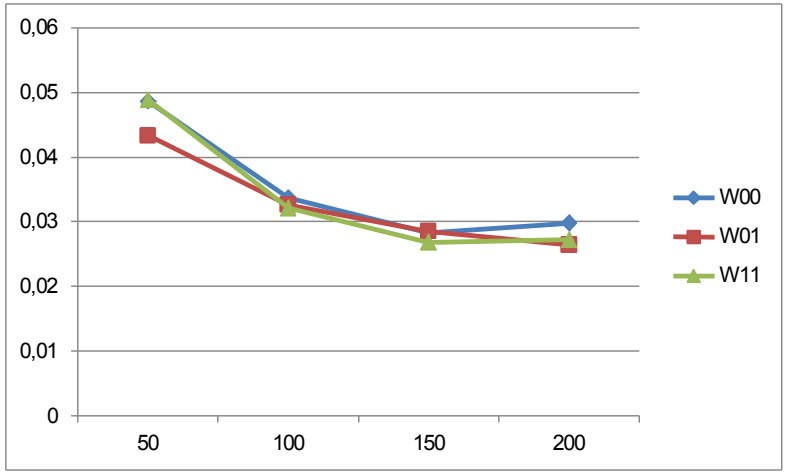
$$w(\Psi_i) = \begin{cases} \frac{1}{2\pi} exp \frac{itter(\Psi_i)\xi}{2} \text{ if } \frac{1}{2\pi} exp \frac{itter(\Psi_i)\xi}{2} > \theta \\ 0 \text{ otherwise} \end{cases} \tag{22}$$

where $theta \in [0, 1]$ as previously stands for the parameter responsible for the removing less important (old enough) classifiers and xi is used defined parameter.

Experiments –Naive Bayes

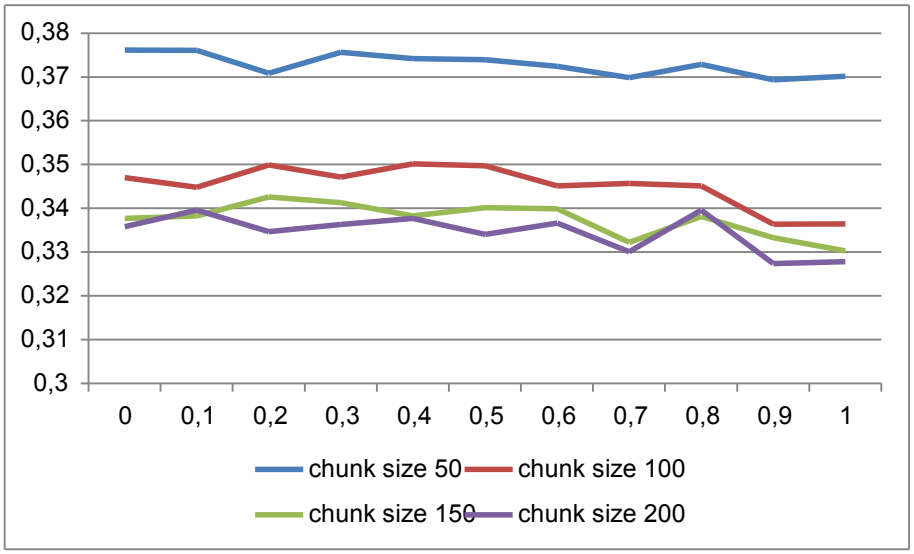


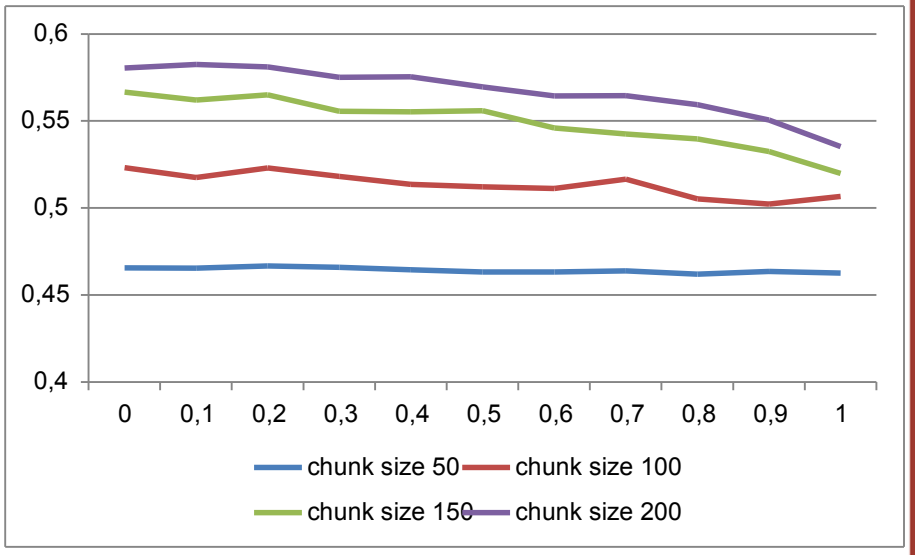
accuracy



standard deviation

Experiments - Hyper Plane Stream

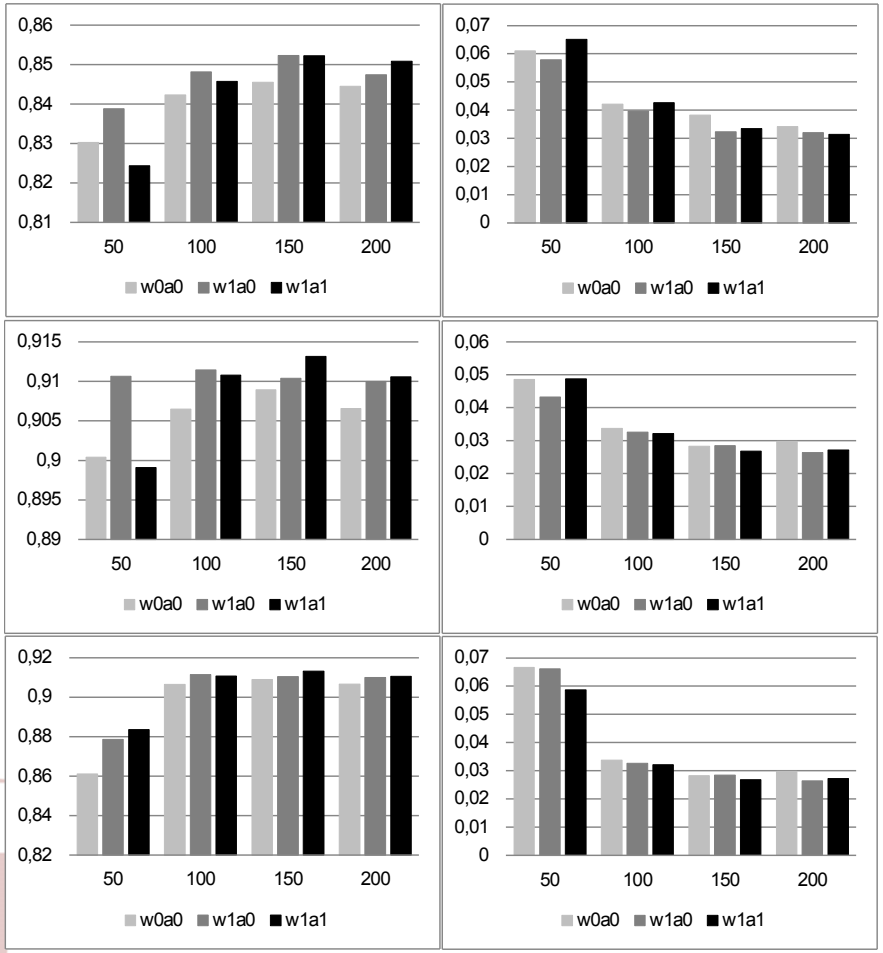




The computer experimental results for the Hyper Plane Stream dataset. Dependencies between *a* factor used in the pruning criterion and ensembles' accuracies (left) diversities (right)

Pruning criterion = a Diversity+(1-a) Accuracy

Experiments on SEA dataset



- Classifier ensemble is a promising research direction for data stream classification, but some problems still await for the proper solution, e.g.:
  - New combination ensemble approach with a drift detection algorithm, what could have a higher impact to the classification performance.
  - Proposing appropriate diversity measures which are able to take into consideration the nature of the task.
  - Proposing distributed ensemble algorithms.
  - How to evaluare classifiers for drifted data streams?