Classification
&
MCS

By:

Kaveh Kavousi

Institute of Biochemistry and Biophysics (IBB)

University of Tehran, Tehran, Iran.

Agenda

- Classification Problem
- Single Classifier
- Motivation for using MCS
- What is MCS?
- History of MCS
- Is MCS better than Single Classifier?
- Factors affecting MCS performance
- How to implement MCS
- Application of MCS
- Future Research Problem

Classification Problem

What is Classification?

- The goal of classification is to organize and categorize data into distinct classes
  - A model is first created based on the previous data (training samples)
  - This model is then used to classify new data (unseen samples)
- A sample is characterized by a set of features
- Classification is essentially finding the best boundary between classes

What is Classification?

- Applications:
  - Biological Data Analysis
  - Personal Identification
  - Medical Diagnosis
  - Text Categorization
  - Denial of Service Detection
  - Character recognition
  - Biometrics
  - Image classification
  - …

Classification Formulation

- Given
  - an input Feature space $\Re^n$
  - a set of classes $\omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$
- the ***Classification Problem*** is
  - to define a mapping f: $\Re^n \rightarrow \omega$ where each vector x in $\Re^n$ is assigned to one class
- This mapping function is called a Decision Function

Decision Function

- The basic problem in classification problem is to find c decision functions

  $$d_1(x), d_2(x), \dots, d_c(x)$$

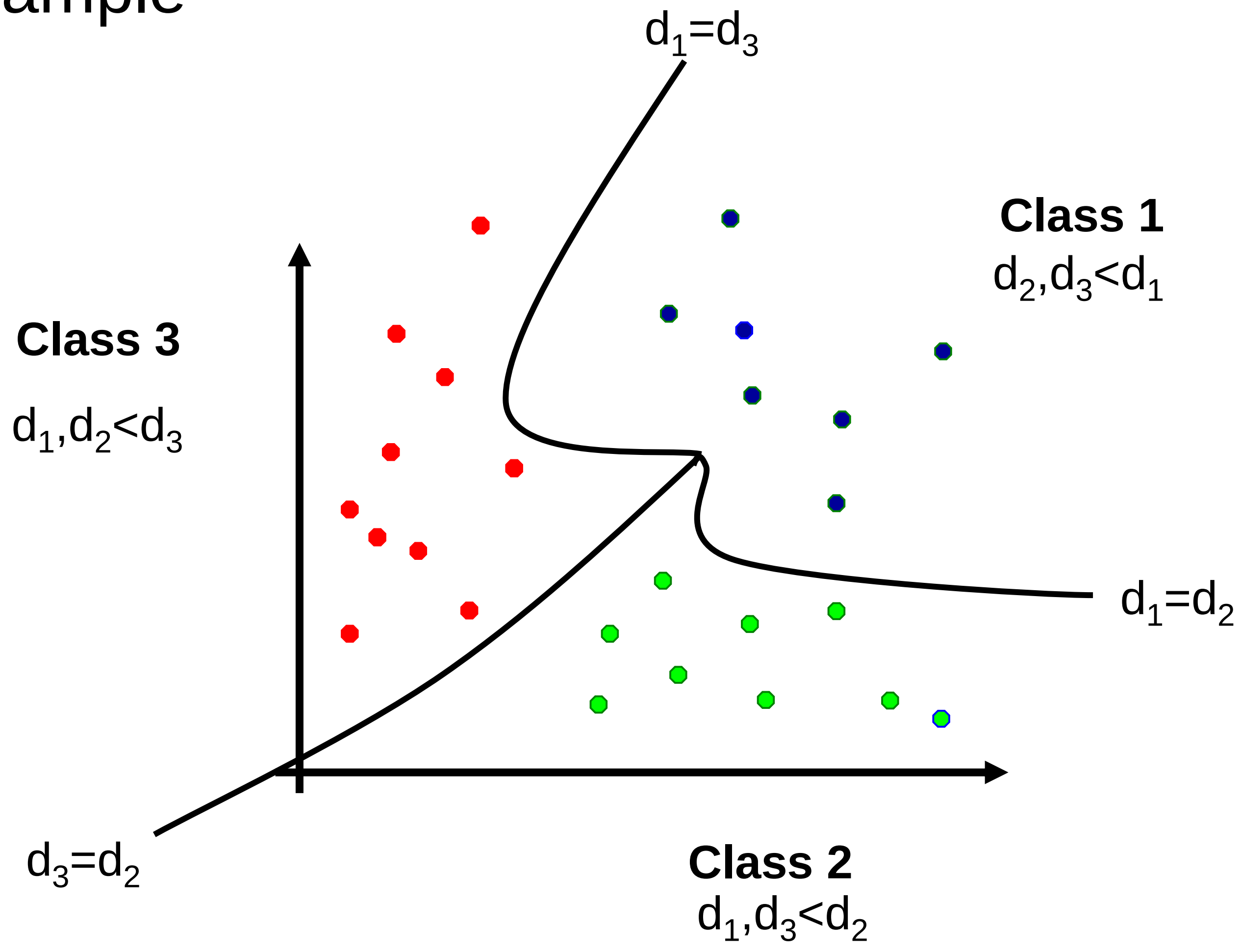
  with the property that, if a pattern x belongs to class i, then

  $$d_i(x) > d_j(x) \qquad i, j = 1,2,\dots c;\ j \neq i$$

  $d_i(x)$ is some similarity measure between x and class i, such as distance or probability concept

Decision Function

- Example



$d_1=d_3$

Class 1

$d_2,d_3<d_1$

Class 3

$d_1,d_2<d_3$

$d_1=d_2$

$d_3=d_2$

Class 2

$d_1,d_3<d_2$

Single Classifier

Single Classifier

- Most popular single classifiers:
  - Minimum Distance Classifier
  - Bayes Optimal Classifier (Min. Error , Min. Risk)
  - K-Nearest Neighbor
  - Decision Tree
  - Neural Network
  - Support Vector Machine

Minimum Distance Classifier

- Simplest approach to selection of decision boundaries
- Each class is represented by a prototype (such as mean) vector:

$$m_j = \frac{1}{N_j} \sum_{x \in \omega_j} x \qquad j = 1,2,\ldots,M$$

where N_j = the number of pattern vectors from ω_j

$M =$ Number of classes

Minimum Distance Classifier

- A new unlabelled sample is assigned to a class whose prototype is closest to the sample

$$d_j(x) = \left\| x - m_j \right\| \qquad j = 1,2,\ldots,M$$

Bayes Classifier

- Bayes rule

$$P(\omega_j \mid x) = \frac{P(x \mid \omega_j)P(\omega_j)}{P(x)}$$

- $P(x)$ is the same for each class, therefore

$$d_j(x) = P(x \mid \omega_j)P(\omega_j)$$

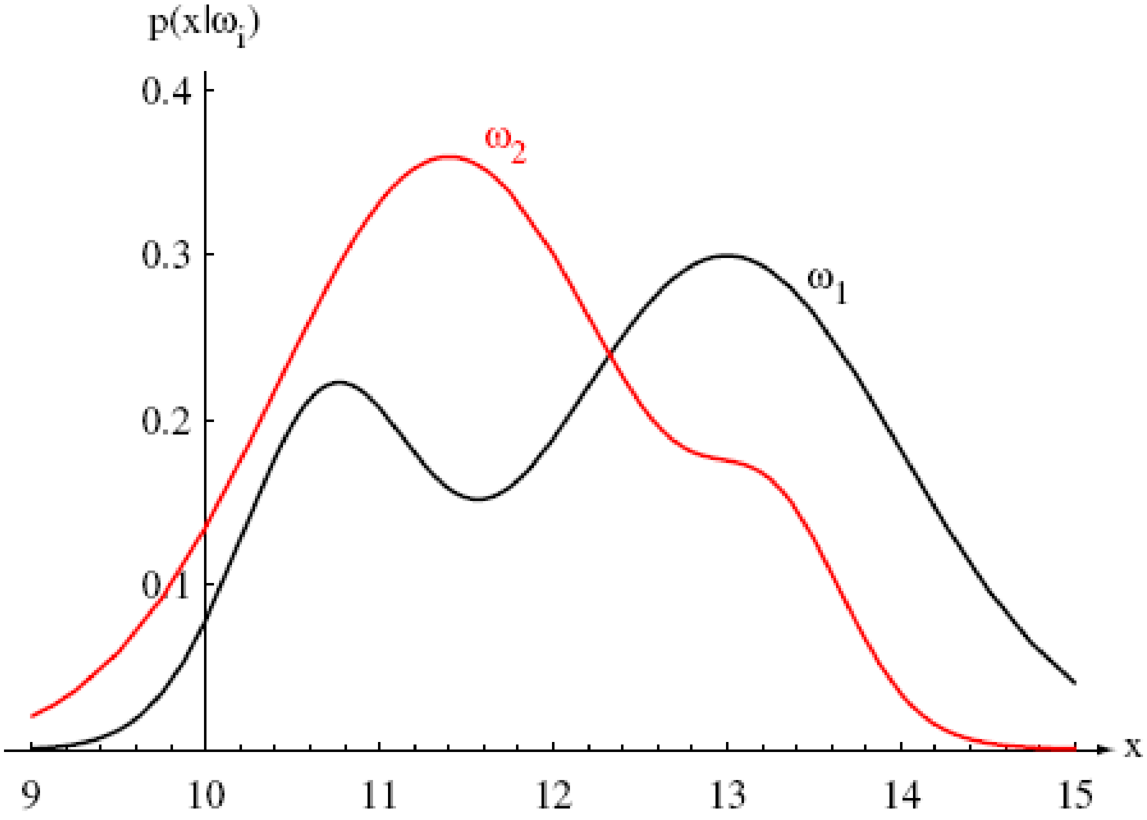
- Assign x to class j if
- $d_i(x) < d_j(x)$ for all $i \neq j$

Bayes Classifier

- The following information must be known:
  - The probability density functions of the patterns in each class $P(\omega_j)$ (a-priori knowledge)
  - The probability of occurrence of each class $P(x \mid \omega_j)$

- Training samples may be used to obtain estimations on these probability functions
- Samples assumed to follow a known distribution pattern

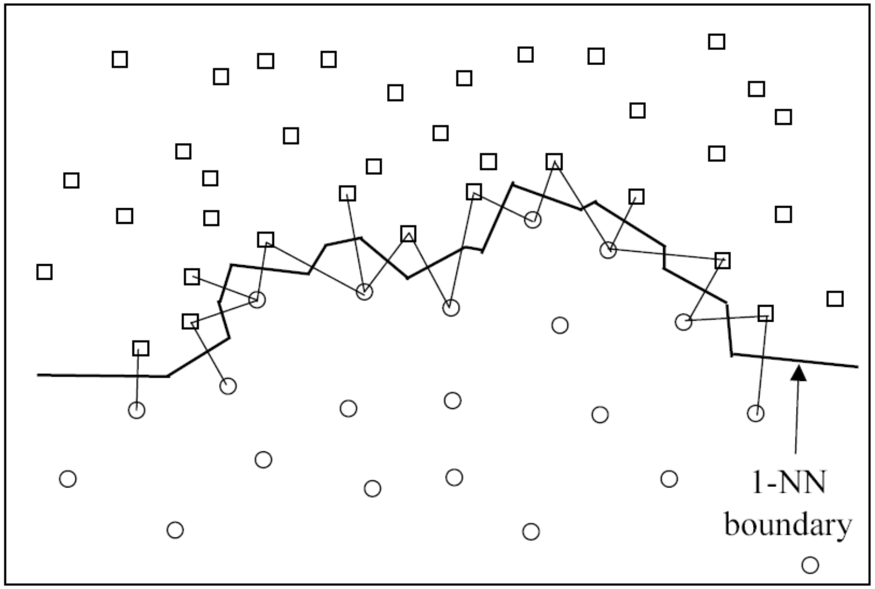
Bayes Classifier



Hypothetical class-conditional probability density functions show the probability density of measuring a particular feature value x given the pattern is in category ω_i. If x represents the length of a fish, the two curves might describe the difference in length of populations of two types of fish. Density functions are normalized, and thus the area under each curve is 1.0.

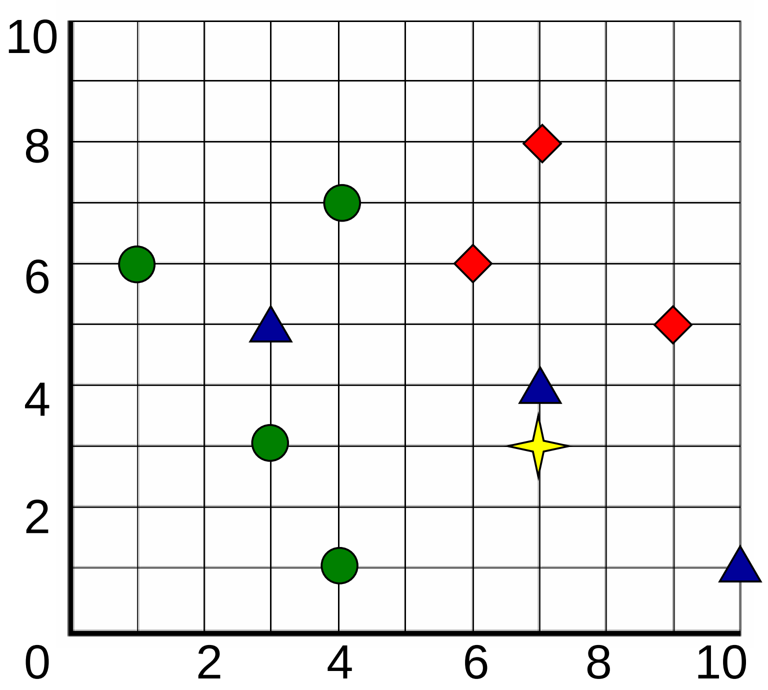
K-Nearest Neighbor

- 1-Nearest Neighbor Rule
  - The new sample is assigned to its nearest class
    - Samples close in feature space most likely belong to the same class



K-Nearest Neighbor

- K-Nearest Neighbor Rule (k-NNR)
  - Examine the labels of the k-nearest samples and classify by using a majority voting scheme

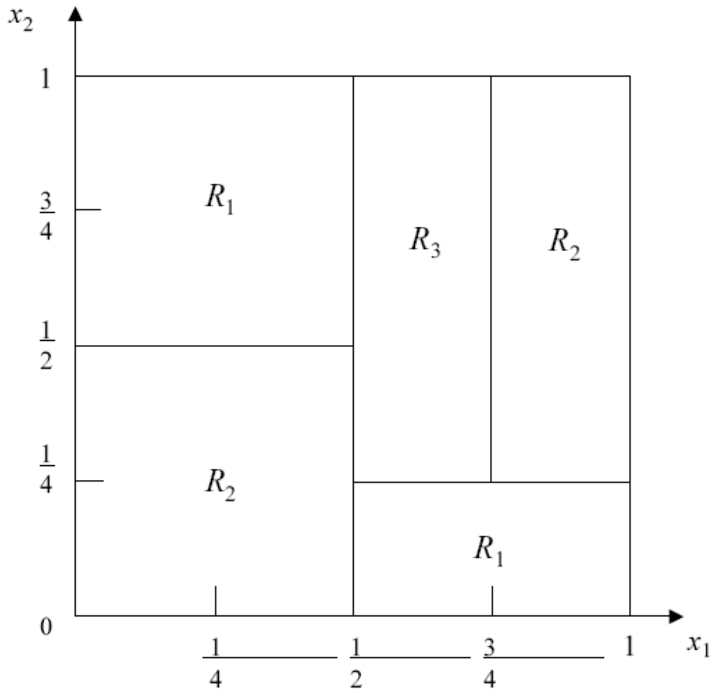


✦ (7, 3)

Order	Position	Distance	Class	
1	7,4	1	▲	1NN ▲
2	9,5	2.82	◆	
3	6,6	3.16	◆	3NN ◆
4	10,1	3.61	▲	
	4,1	3.61	●	5NN ◆▲
6	3,3	4	●	
7	3,5	4.47	▲	7NN ▲
8	7,8	5	◆	
9	4,7	5.66	●	9NN ◆▲●
10	1,6	6.71	●	

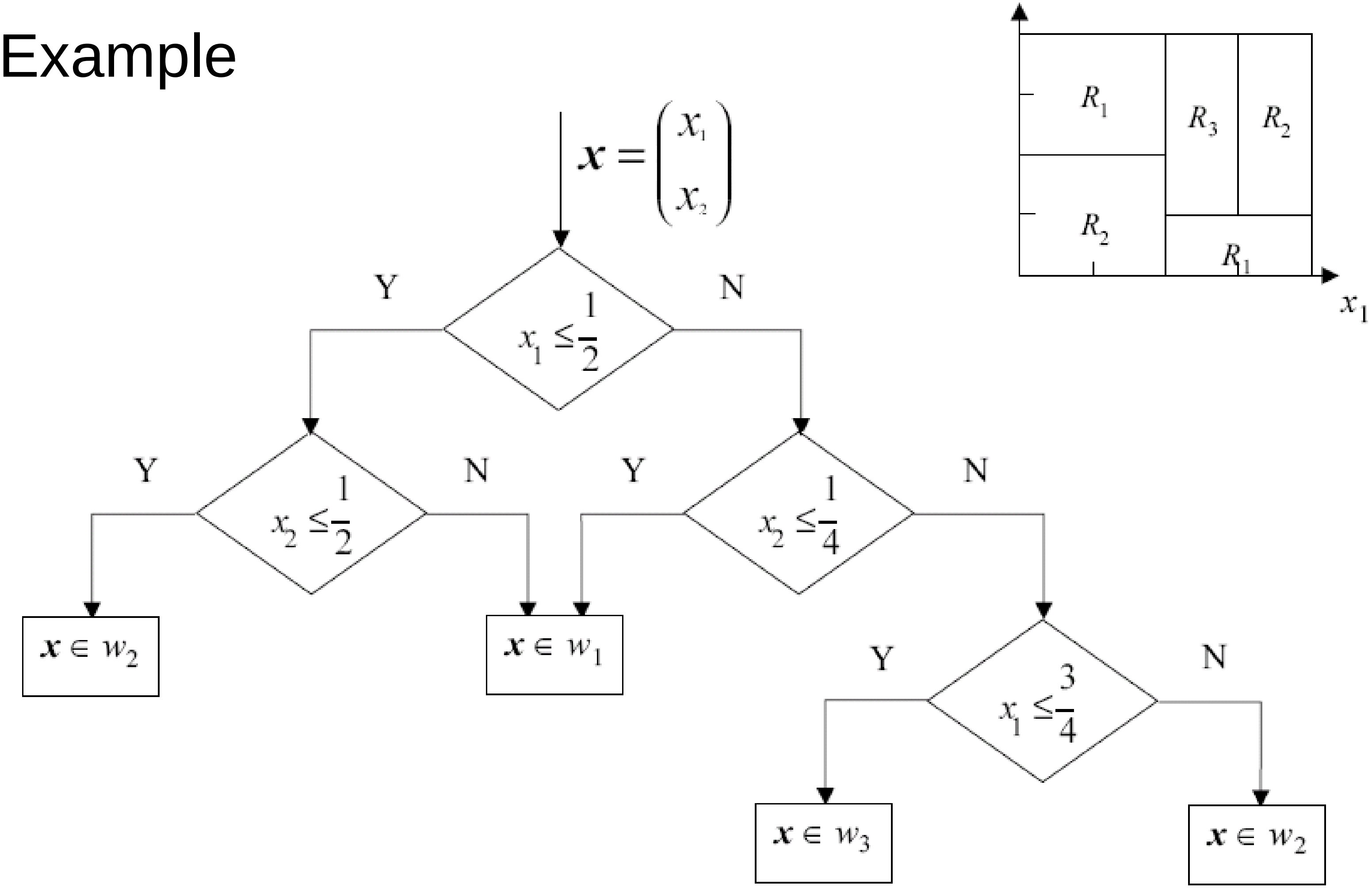
Decision Tree

- The decision boundaries are hyper-planes parallel to the feature-axis
- A sequential classification procedure may be developed by considering successive partitions of R



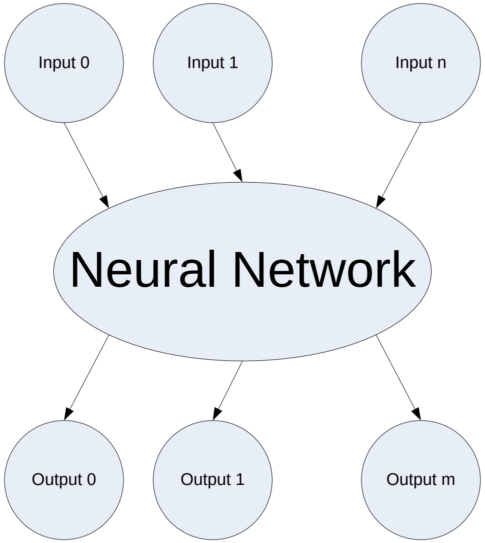
Decision Trees

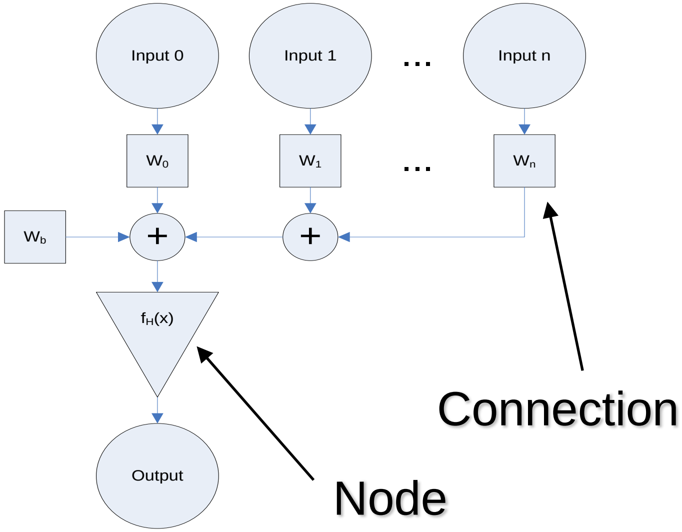
- Example



Neural Network

- A Neural Network generally maps a set of inputs to a set of outputs
- Number of inputs/outputs vary
- The network itself is composed of an arbitrary number of nodes with an arbitrary topology
- It is an universal approximator



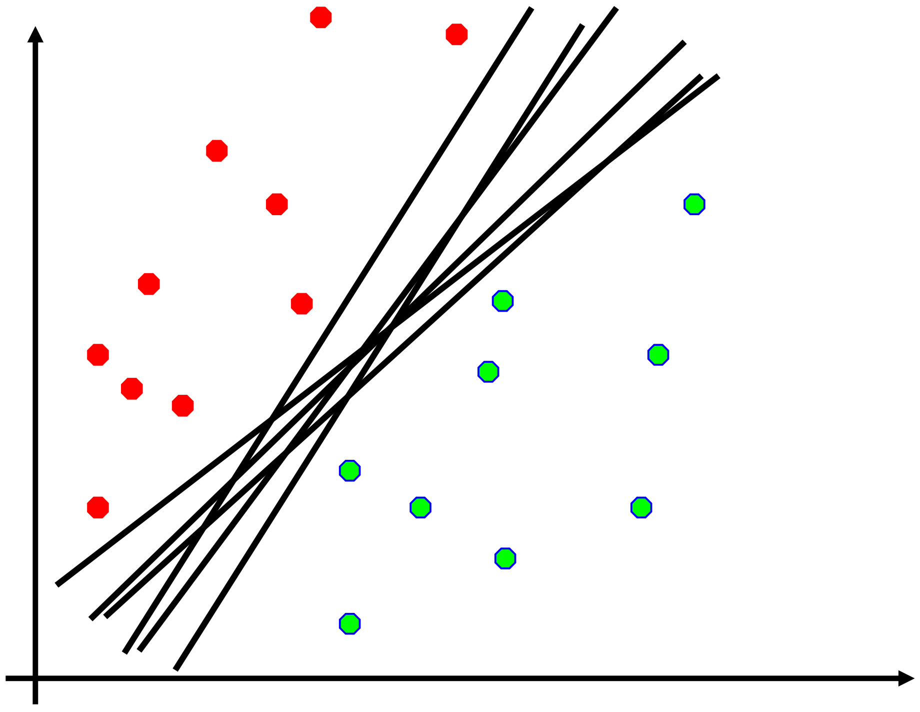


Neural Network

- A popular NN is the feed forward neural network
  - E.g.
    - Multi-layer Perceptron (MLP)
    - Radial-Based Function (RBF)
- Learning algorithm: back propagation
- Weights of nodes are adjusted based on how well the current weights match an objective

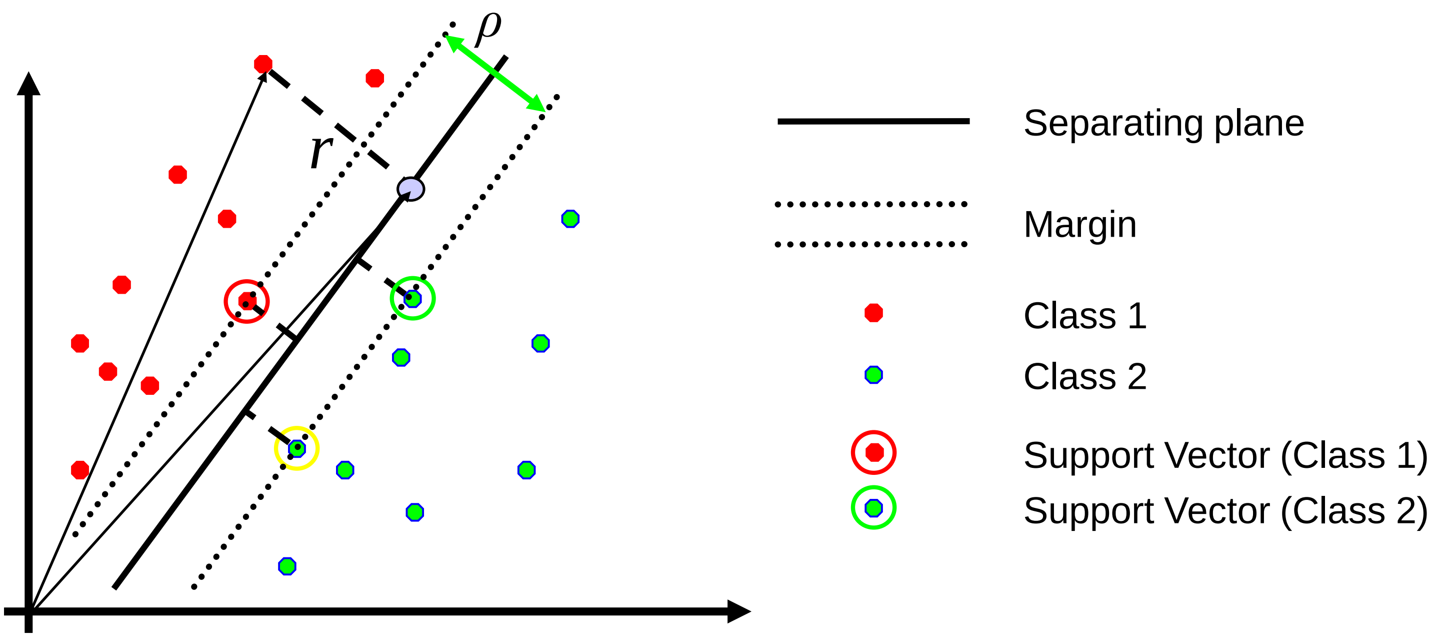
Support Vector Machine

- Basically a 2-class classifier developed by Vapnik and Chervonenkis (1992)
- Which line is optimal?



Support Vector Machine

- Distance from a pattern to the separating hyper-plane is *r* .
- Patterns closest to the hyper-plane are called ***support vectors***.
- ***Margin*** *ρ* of the separating hyper-plane is the width of separation between classes.



Support Vector Machine

- SVM finds the decision hyper-plane with the largest margin
- This hyper-plane has the lowest generalization upper bound

Motivation for MCS

Drawback of Single Classifier

- The “best” classifier not necessarily the ideal choice
  - When solving a classification problem, many individual classifiers with different parameters will be trained
  - The “best” classifier will be selected according to some criteria
    - e.g., training accuracy or complexity of the classifiers
  - Problems:
    - Which one is the best?
      - Maybe more than one classifiers meet the criteria (e.g. same training accuracy), especially in the following situations:
        - Without sufficient training data
        - The learning algorithm leads to different local optima easily
    - Potentially valuable information may be lost by discarding the results of less-successful classifiers
      - E.g., the discarded classifiers may correctly classify some samples

Drawback of Single Classifier

- Other drawbacks
  - The final decision must be wrong if the output of selected classifier is wrong
  - The trained classifier may not be complex enough to handle the problem

Motivations for MCS

- Combining a number of trained classifiers lead to a better performance than any single one
  - Errors can be complemented by other correct classifications
  - Different classifiers have different knowledge reagrding the problem
- To decompose a complex problem into sub-problems for which the solutions obtained are simpler to understand, implement, manage and update

T.G. Dietterich (2000). Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, Multiple Classifier Systems, volume 1857 of Lecture Notes in Computer Science (pp. 1-15), Cagliari, Italy

What is MCS

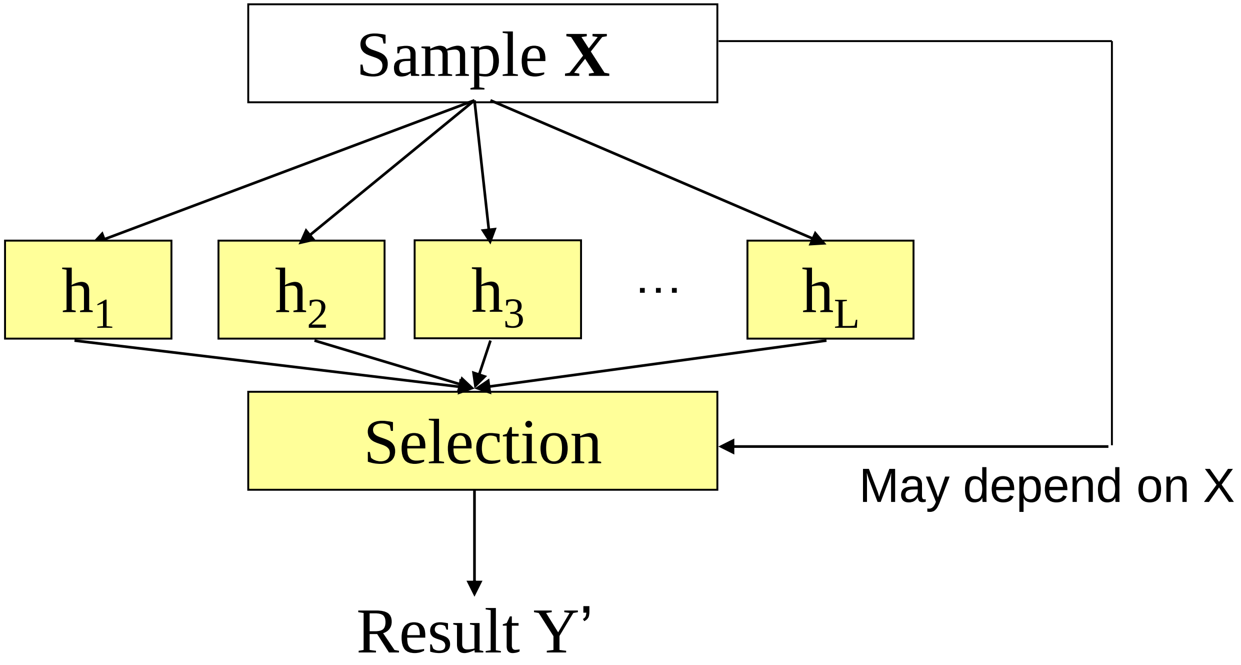
What is MCS?

- Multiple Classifier System (MCS) consists of
  - a set of individual classifiers
  - a fusion/selection method
    - to combine/select individual classifier outputs to give a final decision
- Types of MCS:
  - Classifier Selection
  - Classifier Fusion

Classifier Selection

- For each sample, identify a single classifier which is most likely to produce the correct classification label
- As a result, only the output of the selected classifier is taken as a final decision
- Usually, the input sample space is partitioned into smaller areas and each classifier learns the sample in each area
- It is similar to the “Divide and Conquer” approach

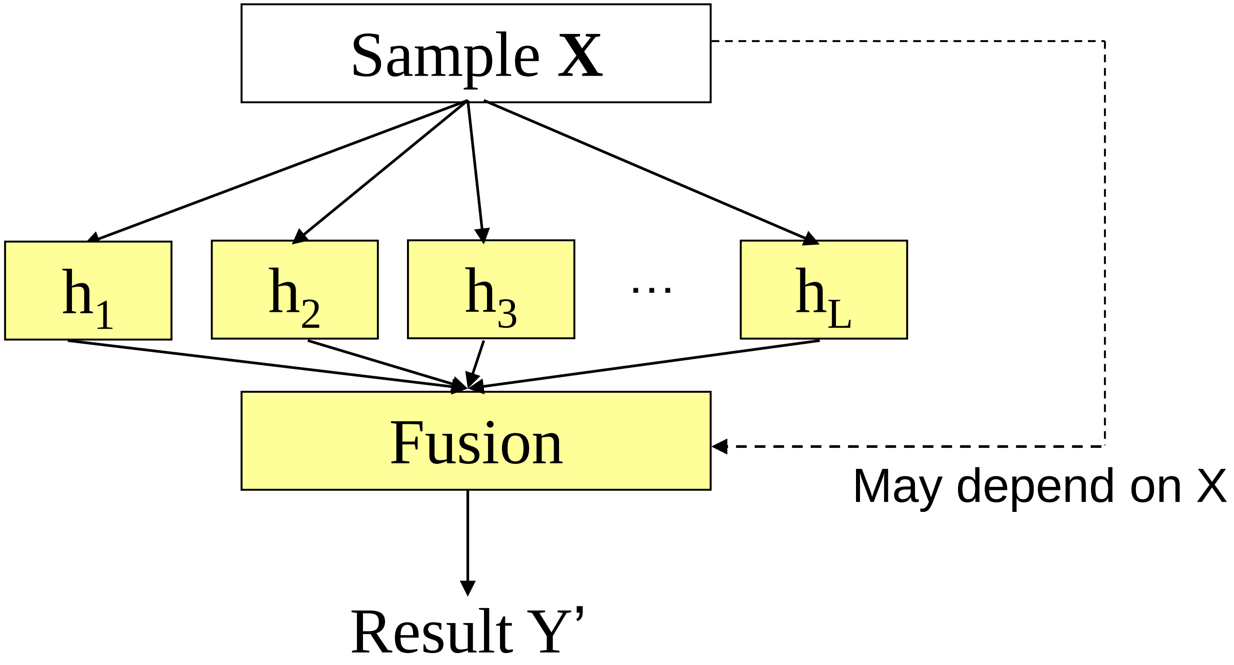
Classifier Selection



Classifier Fusion

- Mixing many different classifiers instead of extracting a single best classifier
- Each individual classifier tries to solve the same classification problem using different methods
  - E.g. Different training sets, different classifiers or different parameters
- The final output of a MCS is determined by fusing the outputs of the individual classifiers

Classifier Fusion



History of MCS

History of MCS

- Combining estimators to improve performance has quite a long history
- As we can find, the concept of combining classifiers was first proposed by Nilsson in 1965
- It was applied in many fields
  - Econometrics forecasting, (Clemen, 1989; Granger, 1989)
  - Evidence combination, (Barnett, 1981)
  - Software Engineering Diversity (Knight and Leveson, 1986)

- Extensive studies began in the 1990s.

N. J. Nilsson. Learning machines: Foundations of Trainable Pattern Classifying Systems. McGraw-- Hill, 1965

Clemen, R. Combining Forecasts: A review and annotated bibliography. International Journal of Forecasting, 5, pp559-583, 1989

Barnett, J.A. Computational methods for a mathematical theory of evidence, In Proc. Of IJCAI, pp868-875, 1981

Knight, J.C. & Leveson, N.G., An experimental evaluation of independence in multiversion programming. Trans on Software Eng., vol SE-12, no 1, 1986

Other Names of MCS

- 2003, Kuncheva has collected most of the aliases of MCS
  - Combination of multiple classifiers
  - Classifier fusion
  - Mixture of experts
  - Committees of neural networks
  - Consensus aggregation
  - Voting pool of classifiers
  - Dynamic classifier selection
  - Composite classifier systems
  - Classifier ensembles
  - Modular systems
  - Collective recognition
  - Stacked generalization
  - Dived-and-conquer classifiers
  - Pandemonium system of reflective agents
  - …

Kuncheva L.I. Combining classifiers: Soft computing solutions, in: S.K. Pal and A. Pal (Eds.) Pattern Recognition: From Classical to Modern Approaches, World Scientific Publishing Co., Singapore, 2001, 427-452

MCS vs Single Classifier

MCS must be better than Single?

- In all cases? NO!

- But in many cases, a MCS can have a better performance than a single classifier
  - Theoretical Discussion
  - Experimental Discussion

Theoretical Discussion

- Hansen and Salamon showed that a necessary and sufficient condition for MCS to be more accurate than any of its individual members
- Necessary and Sufficient conditions are:
  - Accurate
    - An error rate of better than random guessing
  - Diverse
    - Errors made by the classifiers are independent

Hansen LK, Salamon P (1990) Neural network ensembles. IEEE Trans Pattern Anal 12(10):993-1001

Theoretical Discussion

- Examples
  - All individual classifiers have an error rate of 0.4 (Lower than random) and their errors are independent
  - Majority Vote is used
  - The error rate of
    - MCS of 3 classifiers is 0.352
    - MCS of 5 classifiers is 0.317

    …
    - MCS of 21 classifiers is 0.026

$$P_{error} = \sum_{i=\lceil L/2 \rceil}^{L} \binom{L}{i} p^{i} (1-p)^{L-i}$$

L : the number of individual classifiers in MCS
P_{error} : probability of MCS error

L=3 , p=0.4 →

$$P_{error} = \sum_{i=\lceil 3/2 \rceil}^{3} \binom{3}{i} 0.4^{i} \, (1-0.4)^{3-i} = 0.3520$$

Experimental Discussion

- Many experimental researches show that MCS performs better than a single classifier
  - References:
    - H. Altincay and M. Demirekler. An information theoretic framework for weight estimation in the combination of probabilitistic classifiers for speaker identification. Speech Communication, 30:255-272, 2000
    - R. Battiti and A.M. Colla. Democracy in neural nets: voting schemes for classification. Nerual Networks, 7(4):691-707, 1994
    - G. Giacinto and F. Roli. An approach to the automatic design of multiple classifier systems, Pattern Recognition Letter, 22:25-33, 2001
    - Y.S Huang and C.Y. Suen. A method of combiing multiple experts for the recognition of unconstrained handwritten numerals. IEEE Transactions on Pattern Analysis and Machine Intelligence, 117:90-93, 1995
    - A.K. Jain, R.PW. Duin and J. Mao. Statistical pattern recognition: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1): 4-37, 2000
    - K. Woods, Jr. W.P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(4):405-410, 1997
    - …

Experimental Discussion

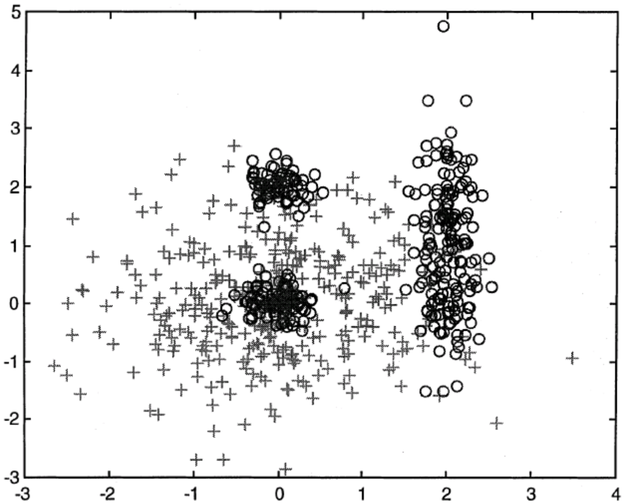
- Verikas, Lipnickas, etc, did detail experiments to show the performance of a single classifier and MCSs with different fusion methods

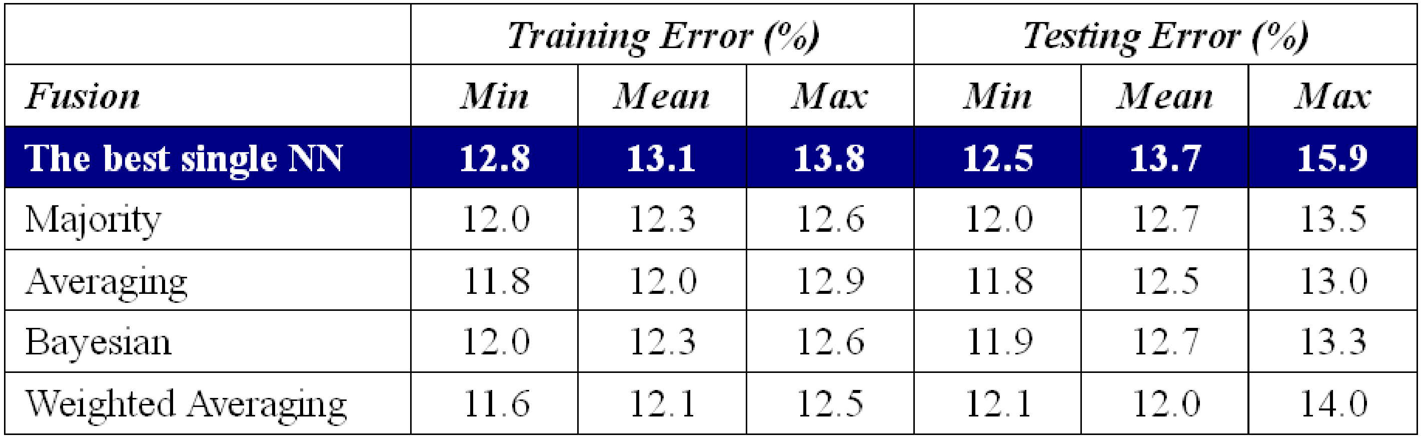
- Experiments
  - 50% training and 50% testing
  - Single Classifier
    - A single layer MLP with 10 sigmoid hidden neurons
  - MCS
    - A single layer MLP with 10 sigmoid hidden neurons
    - 5 individual classifiers
    - Training set is divided into 5 parts, each of them is used to train one individual classifier
  - The experiments are repeated eight times to take the minimum, mean and maximum

T. Windeatt and F. Roli (Eds.): MCS 2003, LNCS 2709, pp. 74–83, 2003.

Experimental Discussion

- Results of the ***Cloud*** dataset
  - 2 features
  - 2 classes
  - 150 training and 150 testing samples
  - Highly overlapping and nonlinear





	Training Error (%)			*Testing Error (%)*		
Fusion	*Min*	*Mean*	*Max*	*Min*	*Mean*	*Max*
The best single NN	**12.8**	**13.1**	**13.8**	**12.5**	**13.7**	**15.9**
Majority	12.0	12.3	12.6	12.0	12.7	13.5
Averaging	11.8	12.0	12.9	11.8	12.5	13.0
Bayesian	12.0	12.3	12.6	11.9	12.7	13.3
Weighted Averaging	11.6	12.1	12.5	12.1	12.0	14.0

Factors Affecting MCS Performance

Factors Affecting MCS Performance

- Three factors affecting the accuracy of MCS:
  - Accuracy of individual classifiers
    - How good are the individual classifiers?
  - Fusion Methods
    - How to combine classifiers?
  - Diversity among classifiers
    - What are the differences between decisions of various classifiers?

Accuracy of individual classifier

- The performance of an individual classifier is affected by
  - Training Dataset (sample and feature)
  - Learning Model (types of classifier)
  - Model's Parameters (e.g. the number of neurons in NN)
- If individual classifiers have low accuracies, a MCS not expected to perform well

Fusion method

- A method to combine individual classifier outputs to reach the final decision for MCS
- Since different fusion methods may have different final outputs for the same individual classifier outputs, MCS error is affected by its fusion method
- For Example

Sample x	Class1	Class2
Classifier 1	0.2	0.8
Classifier 2	0.7	0.3
Classifier 3	0.7	0.3

average	**Class 1: 0.53** Class 2: 0.47
max	Class 1: 0.7 **Class 2: 0.8**

Fusion method

- 3 types of fusion methods based on individual classifier outputs types:
  - Soft Output
  - Class Ranking
  - Class Label

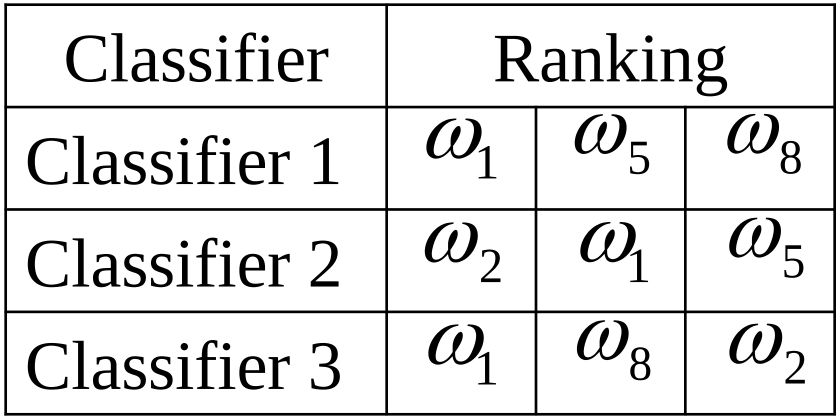
Soft Output

- For each sample, the classifier outputs a value representing the confidence of this sample belonging to each class
- The output of a classifier i is a c-dimensional vector $[d_{i,1}, d_{i,2}, \ldots, d_{i,c}]^T$, where c is number of classes
- It is called decision profile

	ω_1	ω_2
Classifier 1	0.5	0.5
Classifier 2	0.2	0.8
Classifier 3	0.6	0.4

Class Rankings

- The classifier outputs a number of class labels and rank them
- Not very popular



Classifier	Ranking		
Classifier 1	ω_1	ω_5	ω_8
Classifier 2	ω_2	ω_1	ω_5
Classifier 3	ω_1	ω_8	ω_2

Class Label

- A classifier outputs only one class label

Classifier	output
Classifier 1	ω_1
Classifier 2	ω_2
Classifier 3	ω_1

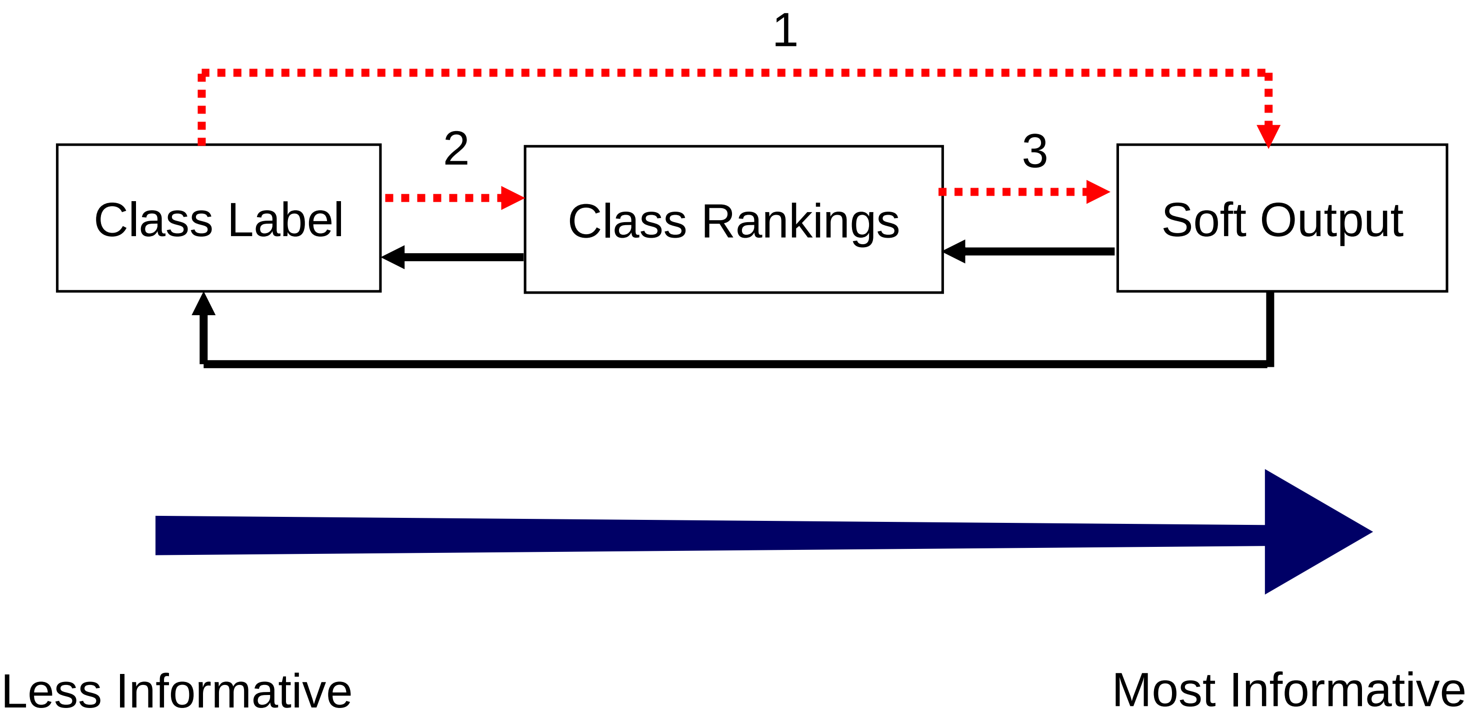
Transferability of Classifier Outputs

→ Always true

⇢ Assumption is needed

Examples of the assumptions:

1. the selected class is 1 and others are 0
2. the selected class is 1st and others are 2nd
3. the 1st class is 50%, 2nd class is 25% ….



Dymitr Ruta and Bogadn Gabrys, An Overview of Classifier Fusion Methods, Computing and Information System, 7 (2000) p.1-10

Fusion method for Soft Output

- Three most popular methods will be discussed:
  - Simple Summary Function (SSF)
  - Weighted Average (WA)
  - Decision Template (DT)

Simple Summary Function (SSF)

- Average (AVR), Maximum (MAX), Minimum (MIN), Product (PRO)
  - These fusion methods calculate the support ($\mu_j(x)$) for class j
  - No Training needed

AVR:

$$\mu_j(x) = \frac{1}{L}\sum_{i=1}^{L} d_{i,j}(x)$$

MIN:

$$\mu_j(x) = \min_{i} \{d_{i,j}(x)\}$$

MAX:

$$\mu_j(x) = \max_{i=1,..,L} \{d_{i,j}(x)\}$$

PRO:

$$\mu_j(x) = \prod_{i=1}^{L} d_{i,j}(x)$$

Simple Summary Function (SSF)

- An example:
  - Average
    - **Class 1: 0.5**
    - **Class 2: 0.5**
  - Minimum
    - Class 1: 0.2
    - **Class 2: 0.3**
  - Maximum
    - Class 1: 0.7
    - **Class 2: 0.8**
  - Product
    - Class 1: 0.084
    - **Class 2: 0.096**

Sample x	Class1	Class2
Classifier 1	0.2	0.8
Classifier 2	0.6	0.4
Classifier 3	0.7	0.3

Weighted Average (WA)

- The Weighted Average (WAVR)
  - The support for class j :

$$\mu_j(x) = \frac{1}{L}\sum_{i=1}^{L} w_i d_{i,j}(x)$$

$$\text{where} \quad \sum_{i=1}^{L} w_i = 1$$

  - Usually, the weight is calculated using the accuracy of the individual classifier

Weighted Average (WA)

- An example:
  - Case 1:
    - Assume that the weight of
      - Classifier 1: 0.7
      - Classifier 2: 0.2
      - Classifier 3: 0.1
    - Class 1: 0.33
    - **Class 2: 0.67**
  - Case 2:
    - Assume that the weight of
      - Classifier 1: 0.2
      - Classifier 2: 0.3
      - Classifier 3: 0.5
    - **Class 1: 0.57**
    - Class 2: 0.43

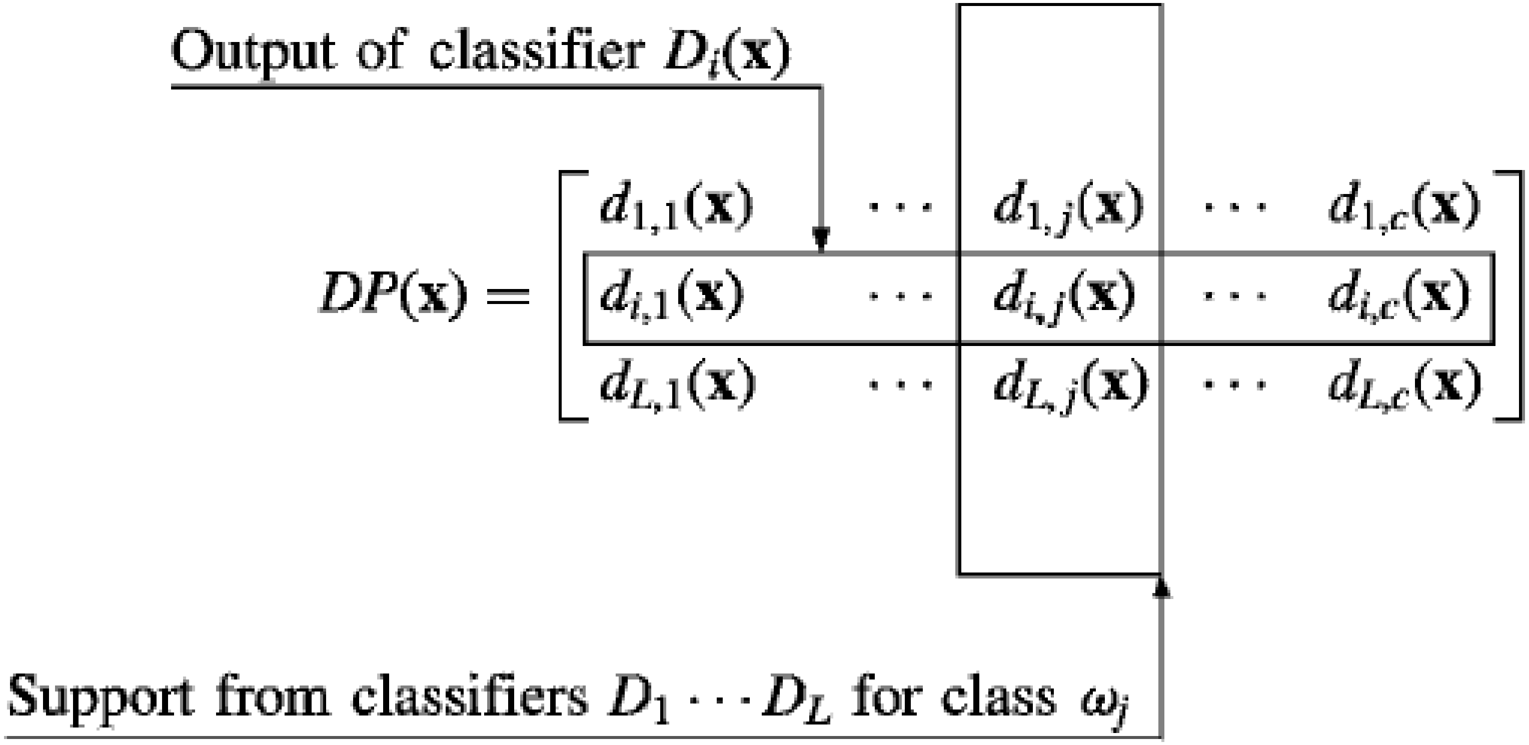
x	Class1	Class2
Classifier 1	0.2	0.8
Classifier 2	0.6	0.4
Classifier 3	0.7	0.3

Decision Template (DT)

The idea of the decision templates (DT) combiner is to remember the most typical decision profile for each class W_j, called the decision template, DT_j, and then compare it with the current decision profile DP(x) using some similarity measure *S. The* closest match will label x

Decision Template (DT)

The L classifier outputs for a particular input $\mathbf{x}$ can be organized in a *decision profile* ($DP(\mathbf{x})$) as the matrix



Output of classifier $D_i(\mathbf{x})$

$$DP(\mathbf{x}) = \begin{bmatrix} d_{1,1}(\mathbf{x}) & \cdots & d_{1,j}(\mathbf{x}) & \cdots & d_{1,c}(\mathbf{x}) \\ d_{i,1}(\mathbf{x}) & \cdots & d_{i,j}(\mathbf{x}) & \cdots & d_{i,c}(\mathbf{x}) \\ d_{L,1}(\mathbf{x}) & \cdots & d_{L,j}(\mathbf{x}) & \cdots & d_{L,c}(\mathbf{x}) \end{bmatrix}$$

Support from classifiers $D_1 \cdots D_L$ for class ω_j

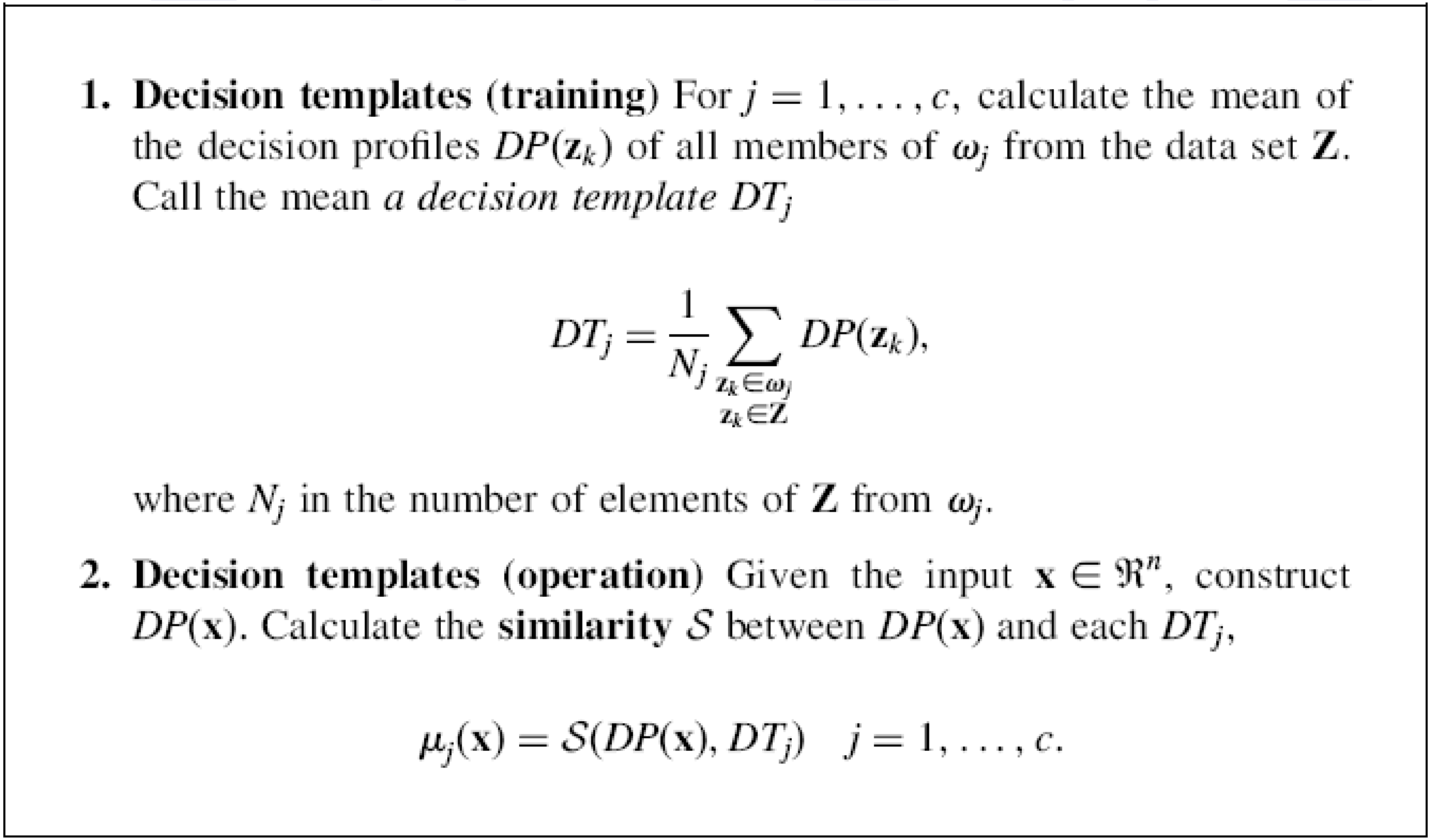
Decision Template (DT)

- The outputs of individual classifiers are organized as a decision profile
- A decision template is built for each class using the training sample
- The outputs of individual classifiers of a new sample (decision profile) will be compared with the decision template of each class by some measure, e.g. Euclidean distance
- The closest class will be selected

$$DT = \begin{bmatrix} d_{1,1}, d_{1,2}, \ldots, d_{1,c} \\ \ldots \\ d_{i,1}, d_{i,2}, \ldots, d_{i,c} \\ \ldots \\ d_{L,1}, d_{L,2}, \ldots, d_{L,c} \end{bmatrix}$$

Combining Pattern classifiers, Kuncheva , Page 170

Decision Template (DT)



1. **Decision templates (training)** For $j = 1, \dots, c$, calculate the mean of the decision profiles $DP(\mathbf{z}_k)$ of all members of ω_j from the data set $\mathbf{Z}$. Call the mean *a decision template* DT_j

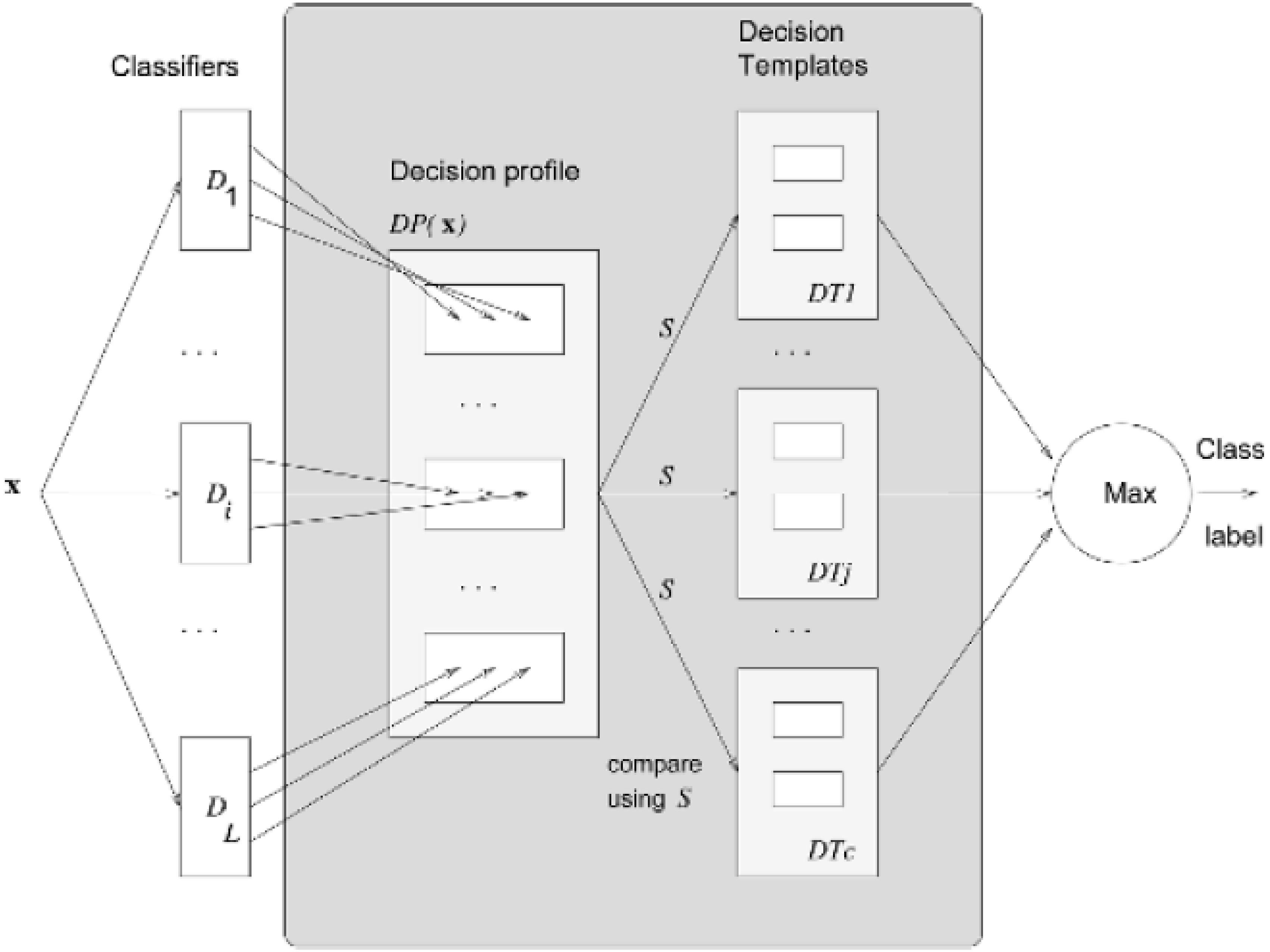
$$DT_j = \frac{1}{N_j} \sum_{\substack{\mathbf{z}_k \in \omega_j \\ \mathbf{z}_k \in \mathbf{Z}}} DP(\mathbf{z}_k),$$

   where N_j in the number of elements of $\mathbf{Z}$ from ω_j.

2. **Decision templates (operation)** Given the input $\mathbf{x} \in \Re^n$, construct $DP(\mathbf{x})$. Calculate the **similarity** $\mathcal{S}$ between $DP(\mathbf{x})$ and each DT_j,

$$\mu_j(\mathbf{x}) = \mathcal{S}(DP(\mathbf{x}), DT_j) \quad j = 1, \dots, c.$$

Training and operation of the decision templates combiner.



Operation of the decision templates combiner.

Decision Template (DT)

- An example:
  - Given the decision templates of 2 classes:

$$DT_1 = \begin{bmatrix} 0.9 & 0.3 \\ 0.5 & 0.5 \\ 0.6 & 0.2 \end{bmatrix} \qquad DT_2 = \begin{bmatrix} 0.4 & 0.6 \\ 0.5 & 0.4 \\ 0.2 & 0.8 \end{bmatrix}$$

  - If the decision profile of a new sample x is

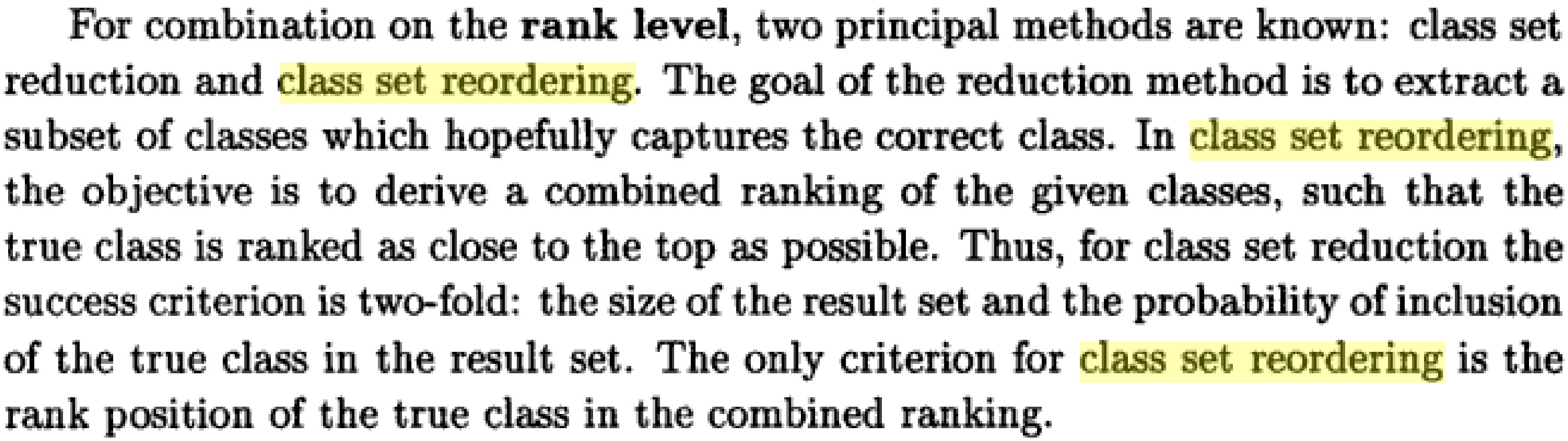
$$DP_x = \begin{bmatrix} 0.8 & 0.1 \\ 0.5 & 0.8 \\ 0.7 & 0.1 \end{bmatrix}$$

  - Euclidean Distance
    - **Class 1: 0.16**
    - Class 2: 1.31

Fusion method for Class Ranking

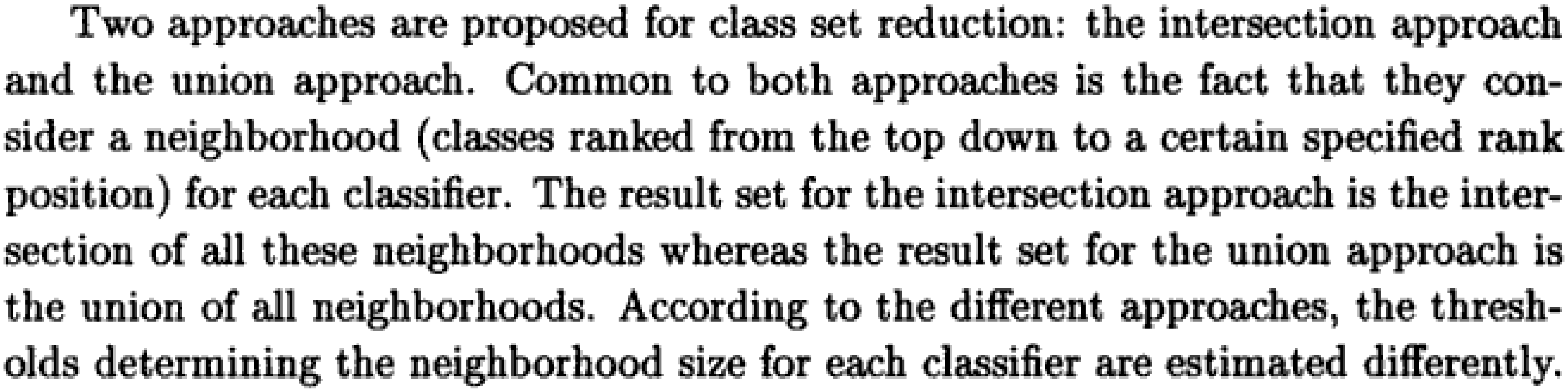
- Two major methods:
  - ***Class Set Reduction*** Method
    - Reduce the set of possible classes by certain criteria but ensure that correct class is still represented in the reduced set
  - ***Class Set Reordering*** Method
    - Obtain the true class ranked as close to the top as possible by certain criteria

Class Set Reduction & Class Set Reordering



For combination on the **rank level**, two principal methods are known: class set reduction and class set reordering. The goal of the reduction method is to extract a subset of classes which hopefully captures the correct class. In class set reordering, the objective is to derive a combined ranking of the given classes, such that the true class is ranked as close to the top as possible. Thus, for class set reduction the success criterion is two-fold: the size of the result set and the probability of inclusion of the true class in the result set. The only criterion for class set reordering is the rank position of the true class in the combined ranking.

Class Set Reduction Method



Two approaches are proposed for class set reduction: the intersection approach and the union approach. Common to both approaches is the fact that they consider a neighborhood (classes ranked from the top down to a certain specified rank position) for each classifier. The result set for the intersection approach is the intersection of all these neighborhoods whereas the result set for the union approach is the union of all neighborhoods. According to the different approaches, the thresholds determining the neighborhood size for each classifier are estimated differently.

Class Set Reduction Method

- ***Intersection of Neighborhoods (Pessimistic)***
  - The neighborhoods of all classifiers are determined by the ranks of true classes for the worst case in the training data set
  - The lowest rank by any of the classifier is taken as the threshold and only classes ranked above threshold are used for further processing

	True Class	Classifier 1	Classifier 2	Classifier 3
X1	ω_1	Rank 5	Rank 2	Rank 2
X2	ω_4	Rank 3	Rank 1	Rank 4
X3	ω_3	Rank 3	Rank 2	Rank 1
Threshold		**5**	**2**	**4**

The most bad decision of each classifier

T.H. Ho, J.J. Hull, S.N. Srihari, Decisision Combination in Multiple Classifier System, IEEE Trans. On Pattern Analysis and Machine Intelligence, vol. 16, pt. 1, pp, 66-75, 1994

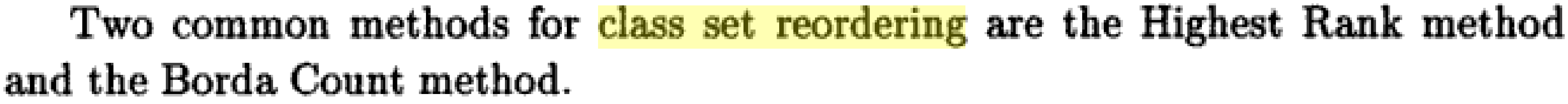
Class Set Reduction Method

- ***Union of Neighborhoods***
  - The threshold for each classifier is calculated as the maximum (worst) of the minimums (best) of ranks of true classes over the training data set
  - The redundant classifier can be easily determined, as its threshold equal to zero meaning that its output is always incorrect

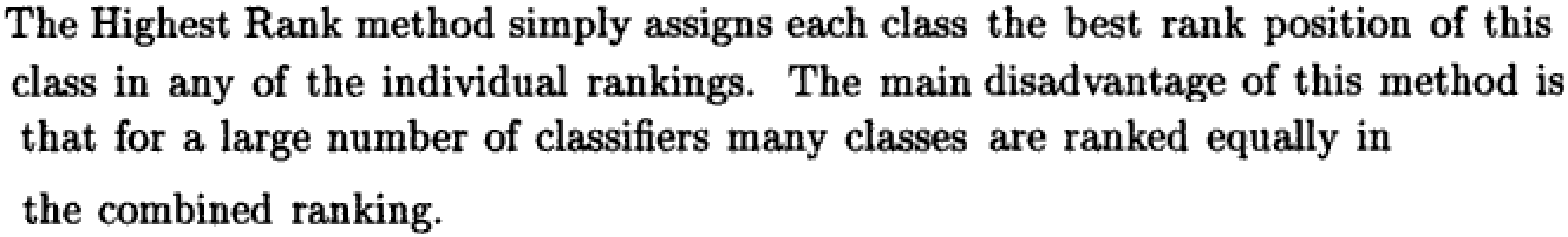
		Classifier Output			**Row Minimum**			
	True Class	Classifier 1	Classifier 2	Classifier 3	Classifier 1	Classifier 2	Classifier 3	
X1	ω_1	Rank 5	Rank 2	Rank 2	0	2	2	$\min(5,2,2)$
X2	ω_4	Rank 3	Rank 1	Rank 4	0	1	0	$\min(3,1,4)$
X3	ω_3	Rank 3	Rank 2	Rank 1	0	0	1	$\min(0,0,1)$
Threshold					**0** **(redundant)**	**2** $\max(2,1,0)$	**2** $\max(2,0,1)$	

T.H. Ho, J.J. Hull, S.N. Srihari, Decisision Combination in Multiple Classifier System, IEEE Trans. On Pattern Analysis and Machine Intelligence, vol. 16, pt. 1, pp, 66-75, 1994

Class Set Reordering Method



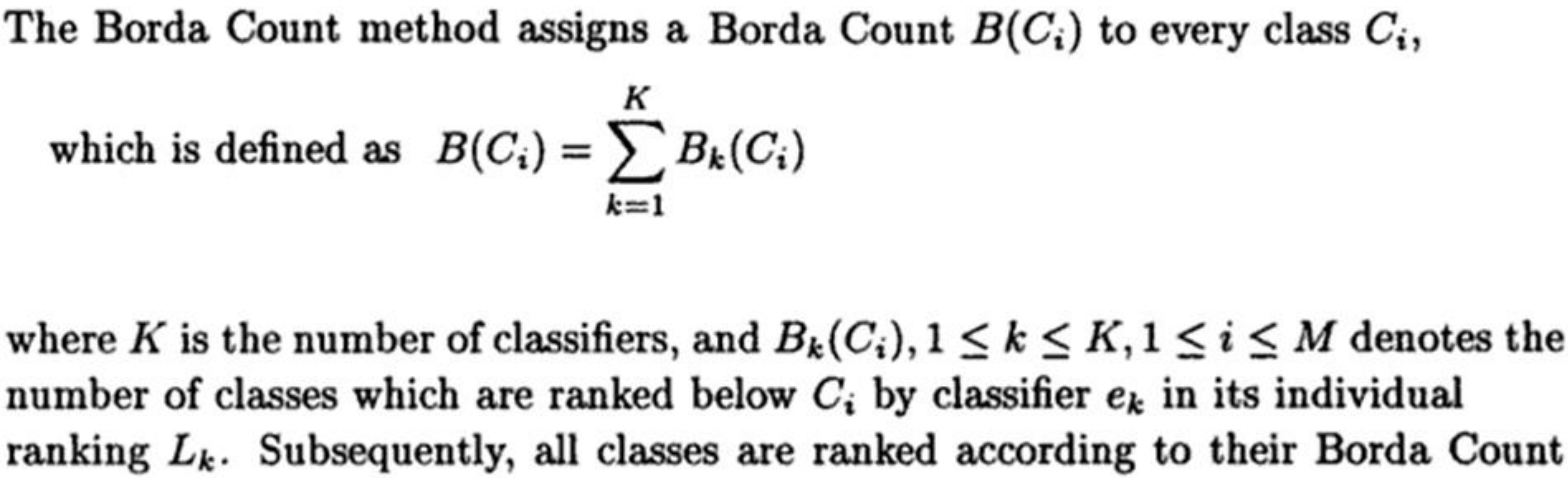
Two common methods for class set reordering are the Highest Rank method and the Borda Count method.



The Highest Rank method simply assigns each class the best rank position of this class in any of the individual rankings. The main disadvantage of this method is that for a large number of classifiers many classes are ranked equally in the combined ranking.

Class Set Reordering Method

- The Borda Count Method



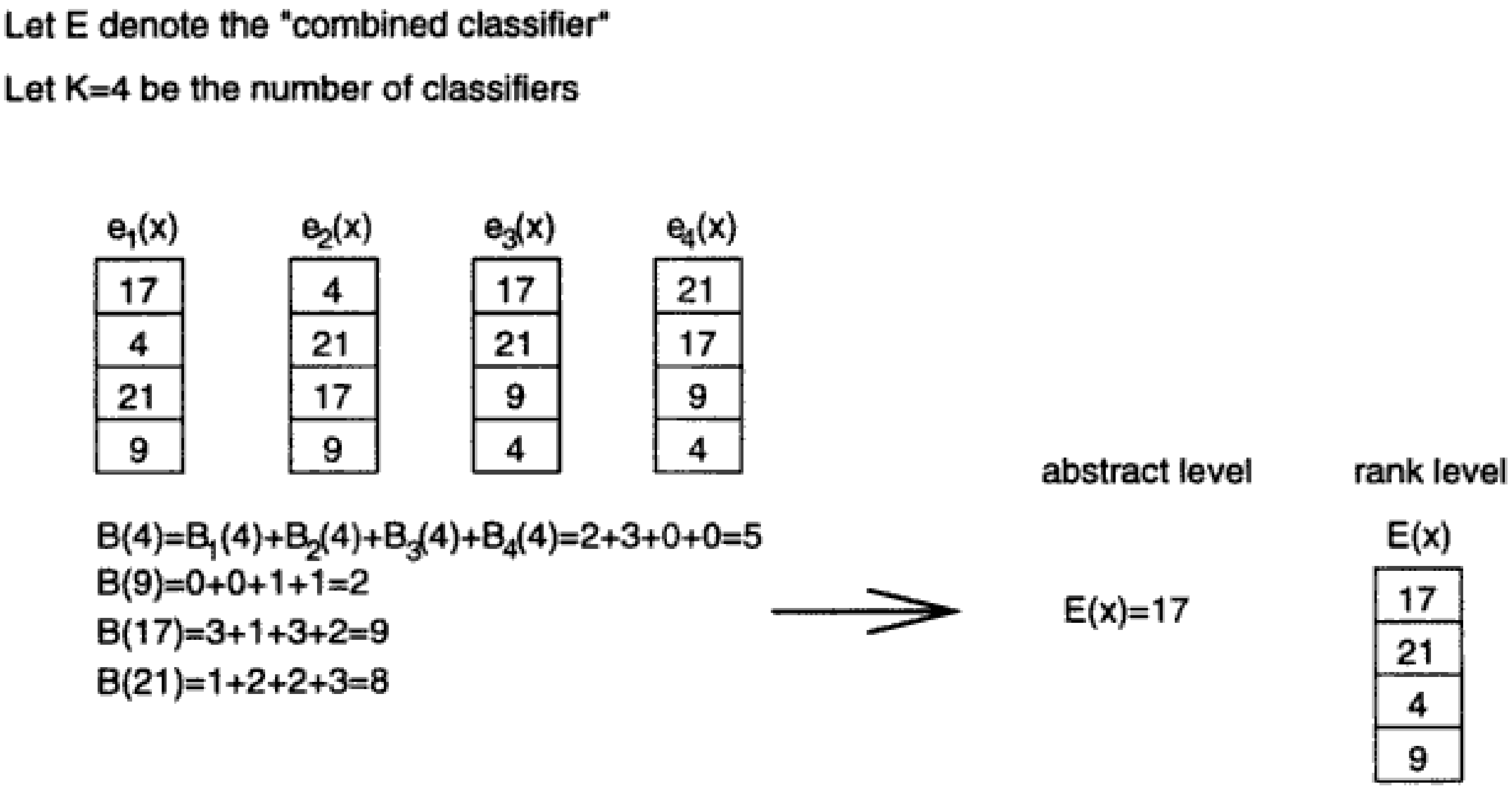
The Borda Count method assigns a Borda Count $B(C_i)$ to every class C_i,

which is defined as $$B(C_i) = \sum_{k=1}^{K} B_k(C_i)$$

where K is the number of classifiers, and $B_k(C_i), 1 \leq k \leq K, 1 \leq i \leq M$ denotes the number of classes which are ranked below C_i by classifier e_k in its individual ranking L_k. Subsequently, all classes are ranked according to their Borda Count

Class Set Reordering Method

- The Borda Count Method Example



Let E denote the "combined classifier"

Let K=4 be the number of classifiers

$e_1(x)$	$e_2(x)$	$e_3(x)$	$e_4(x)$
17	4	17	21
4	21	21	17
21	17	9	9
9	9	4	4

$B(4)=B_1(4)+B_2(4)+B_3(4)+B_4(4)=2+3+0+0=5$

$B(9)=0+0+1+1=2$

$B(17)=3+1+3+2=9$

$B(21)=1+2+2+3=8$

$\longrightarrow$

abstract level

$E(x)=17$

rank level

$E(x)$
17
21
4
9

Fusion method for Class Label

- Four most popular methods will be discussed:
  - Majority Vote (MV)
  - Weighted Majority Vote (WMV)
  - Naïve Bayes (NB)
  - Behavior Knowledge Space Method (BKS)

Majority Vote (MV)

- This method may be the oldest and the most well known strategy for decision making
- Assume that the label outputs of the classifiers are given as c-dimensional binary vectors ,

$$[d_{i,1}, d_{i,2}, \ldots, d_{i,c}]^T \in \{0,1\}^c$$

i = 1,…,L where $d_{i,j}$ =1 if label x in class j, and 0 otherwise

- The majority vote results in an ensemble decision for class k if

$$\arg Max_{j=1}^{c} (\sum_{i=1}^{L} d_{i,j}) == k$$

Majority Vote (MV)

- An example:
  - Class 1: 2 votes
  - **Class 2: 3 votes**

Sample x	Result Label
Classifier 1	1
Classifier 2	2
Classifier 3	1
Classifier 4	2
Classifier 5	2

Weighted Majority Vote (WMV)

- Similar to majority vote method but the influence of each vote to the final decision is not the same
- The weighted majority vote results in an ensemble decision for class k if

$$\arg\max_{j=1}^{c} \sum_{i=1}^{L} w_i d_{i,j} == k$$

Weighted Majority Vote (WMV)

- An example:
  - Case 1:
    - Assume that the weight of
      - Classifier 1: 0.1
      - Classifier 2: 0.2
      - Classifier 3: 0.2
      - Classifier 4: 0.3
      - Classifier 5: 0.2
    - Class 1: 0.3
    - Class 2: 0.7
  - Case 2:
    - Assume that the weight of
      - Classifier 1: 0.4
      - Classifier 2: 0.2
      - Classifier 3: 0.2
      - Classifier 4: 0.1
      - Classifier 5: 0.1
    - Class 1: 0.6
    - Class 2: 0.4

Sample x	Result
Classifier 1	1
Classifier 2	2
Classifier 3	1
Classifier 4	2
Classifier 5	2

Naïve Bayes (NB)

- The term “**naïve**” (strong independence assumption) is used since this method relies on the assumption that the classifiers are mutually independent.

$$\mu_j(x) \propto \prod_{i=1}^{L} \hat{P}\left(\omega_j \mid d_{i,j}(x)=1\right)$$

- $\hat{P}\left(\omega_j \mid d_{i,j}(x)=1\right)$ is learned from training samples

i: Classifier ; j:Class No.

Naïve Bayes (NB)

- An example: 100 Samples

$$\hat{P}(\omega_1 \mid d_{1,1}(x)=1)=\frac{\boxed{40}}{70} \qquad \hat{P}(\omega_2 \mid d_{1,2}(x)=1)=\frac{\boxed{20}}{30}$$

$$\hat{P}(\omega_1 \mid d_{2,1}(x)=1)=\frac{\boxed{20}}{40} \qquad \hat{P}(\omega_2 \mid d_{2,2}(x)=1)=\frac{\boxed{30}}{60}$$

$$\hat{P}(\omega_1 \mid d_{3,1}(x)=1)=\frac{\boxed{45}}{85} \qquad \hat{P}(\omega_2 \mid d_{3,2}(x)=1)=\frac{\boxed{40}}{90}$$

$d_{i,j}(x)$: Support degree of belonging x to class j by classifier i

- **Class 1: 0.1513**
- Class 2: 0.1111

Classifier Decision

		Class1	Class2
True	Class1	40	10
	Class2	30	20

Classifier 1

Classifier Decision

		Class1	Class2
True	Class1	20	30
	Class2	20	30

Classifier 2

Classifier Decision

		Class1	Class2
True	Class1	45	5
	Class2	40	10

Classifier 3

Behavior Knowledge Space Method (BKS)

- Every possible combination of class labels is regarded as an index to a cell in a look-up table (BKS table).
- The table is designed using a labeled training data set
- Each sample is placed in the cell indexed by the class labels of all individual classifiers

Behavior Knowledge Space Method (BKS)

- An example:
  - MCS contains 3 individual classifiers
  - In a binary classification problem, BKS table is trained by using the training samples
  - For example, the individual classifiers output **1 2 2** for a new sample. According to the table, the final decision of MCS is **class 1**

Classifiers Output **1 2 3**	**Class 1**	**Class 2**
1 1 1	90	5
1 1 2	60	1
1 2 1	100	0
1 2 2	60	40
2 1 1	50	20
2 1 2	41	62
2 2 1	20	10
2 2 2	15	20

Diversity among individual classifiers

- The difference between classifier outputs is called diversity
- Two classifiers are diverse if they produce different outputs
- Diversity is an important concept in MCS since, if all classifiers are the same, no MCS is needed
- When a classifier makes an error, we complement it with other classifiers which may make errors on different objects

Diversity among individual classifiers

- A very simple example:
  - x1 – x5 are samples
  - C1, C2, C3 are classifiers
  - Majority vote is used for MCS
  - 1 indicate correct classification; otherwise 0

Not Diverse

	Classifer1	Classifer2	Classifer3	MCS
x1	1	1	1	1
x2	0	0	0	0
x3	0	0	0	0
x4	1	1	1	1
x5	1	1	1	1
%	**0.6**	**0.6**	**0.6**	**0.6**

Diverse

	Classifer1	Classifer2	Classifer3	MCS
x1	1	1	0	1
x2	1	0	1	1
x3	0	1	1	1
x4	1	1	0	1
x5	0	0	1	0
%	**0.6**	**0.6**	**0.6**	**0.8**

Diversity among individual classifiers

- Diversity is an abstract concept
- Many different definitions for diversity
- Two major categories based on the type of individual classifiers' output
  - Diversity of Soft Output
  - Diversity of Label Output

Diversity of Soft Output

- the correlation coefficients between classifier outputs
- Correlation coefficient
  - ○ 1: not diverse (identical)
  - ○ 0: independent
  - ○ -1: the most diverse

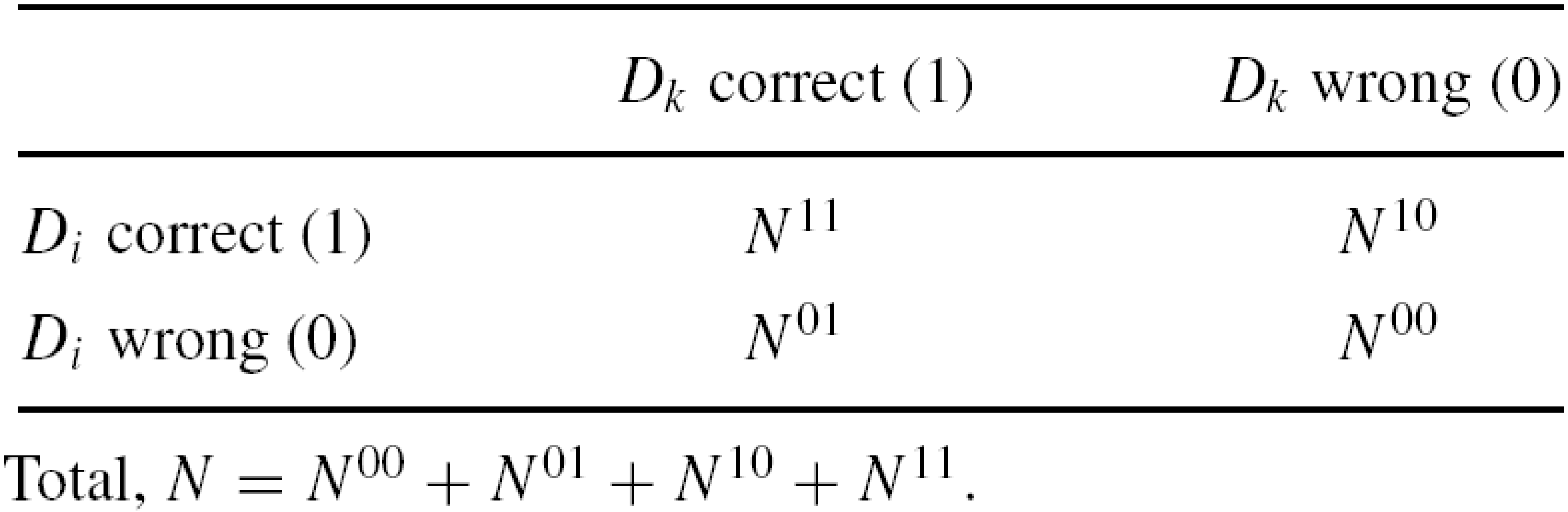
Tumer, K., & Ghosh, J. (1996a). Analysis of decision boundaries in linearly combined neural classifiers. Pattern Recognition, 29:2, 341–348.

Diversity of Label Output

- Two kinds of diversity measure:
  - Pairwise Measure
    - Consider a pair of classifiers at a time
    - An ensemble of L classifiers will produce L(L-1)/2 pairwise diversity values
    - To get a single value we average across all pairs
  - Nonpairwise Measure
    - Consider all the classifiers together and calculate directly one diversity value for the MCS

Pairwise Diversity Measure

- When only two classifiers are considered, four cases of outputs:



	D_k correct (1)	D_k wrong (0)
D_i correct (1)	N^{11}	N^{10}
D_i wrong (0)	N^{01}	N^{00}

Total, $N = N^{00} + N^{01} + N^{10} + N^{11}$.

$N^{01} = N^{10} = 0$ → No Diversity

Pairwise Diversity Measure

- Correlation

$$\rho = \frac{N^{11}N^{00} - N^{01}N^{10}}{\sqrt{(N^{11}+N^{10})(N^{00}+N^{10})(N^{11}+N^{01})(N^{00}+N^{01})}}$$

$\rho = 1$ → No Diversity

$\rho = -1$ → Complete Diversity

Sneath, P., & Sokal, R. (1973). Numerical Taxonomy. W.H. Freeman & Co.

Pairwise Diversity Measure

- Q statistic
  - It is a statistical method to assess the similarity of two classifier outputs

$$Q = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$$

  - Q and ρ have the same sign and |Q| >= | ρ |

$Q = 1$ → No Diversity

$Q = -1$ → Complete Diversity

Yule, G. (1900). On the association of attributes in statistics. Phil. Trans., A, 194, 257–319.

Pairwise Diversity Measure

- The Disagreement Measure
  - ○ Ho (1998) uses disagreement measure to build a decision forest
  - ○ Ho claimes that sufficient independence or dissimilarity between the trees can achieve better accuracy

$$Dis = \frac{N^{01} + N^{10}}{N^{11} + N^{10} + N^{01} + N^{00}}$$

$Dis = 0$ → No Diversity

$Dis = 1$ → Complete Diversity

Ho, T. (1998). The random space method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20:8, 832–844.

Skalak, D. (1996). The sources of increased accuracy for two proposed boosting algorithms. In Proc. American Association for Artificial Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop.

Pairwise Diversity Measure

- The Double-Fault Measure
  - Giacinto and Roli (2001) clusters a classifier pool based on their probabilities of double fault and then select least related classifiers

$$DF = \frac{N^{00}}{N^{11} + N^{10} + N^{01} + N^{00}}$$

Giacinto, G., & Roli, F. (2001). Design of effective neural network ensembles for image classification processes. Image Vision and Computing Journal, 19:9/10, 699–707.

Nonpairwise Diversity Measure

- For nonpairwise measures, there are 3 different types:
  - The number of correct classifiers
  - The proportion of correct classifiers
  - The probability of classifiers that are wrong

The number of correct Classifiers

- These methods consider two terms:

$l(x)$: number of correct classifier $L-l(x)$: number of incorrect classifier

  - ○ The Entropy Measure

- E=0 All classifiers are correct or incorrect
- E=1 50% of classifiers are correct

$$E=\frac{1}{N}\sum_{j=1}^{N}\frac{1}{(L-\lceil L/2\rceil}\min\{l(x_j),L-l(x_j)\}$$

  - ○ KW Variance

- KW=0 All classifiers are correct or incorrect

$$KW=\frac{1}{NL^2}\sum_{j=1}^{N}l(x_j)(L-l(x_j))$$

  - ○ Measurement of Inter-rater Agreement

$$\kappa=1-\frac{\frac{1}{L}\sum_{j=1}^{N}l(x_j)(L-l(x_j))}{N(L-1)\bar{p}(1-\bar{p})}$$

$\bar{p}$: the average individual classification accuracy

$\kappa\in\left(-\infty,0\right]$

Kohavi, R., & Wolpert, D. (1996). Bias plus variance decomposition for zero-one loss functions. In L. Saitta (Ed.), Machine Learning: Proc. 13th International Conference (pp. 275–283). Morgan Kaufmann.

Dietterich, T. (2000b). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. Machine Learning, 40:2, 139–157.

How to implement MCS

How to implement MCS

- More diverse MCS does not mean better
- We want to build a MCS with accurate individual classifiers and these classifiers are some what diverse (not very diverse)
- Steps for building MCS (can be in different order)
  - Decide the number of individual classifiers
  - Training the individual classifiers differently
    - Using different Training Sets
    - Using different Learning Models
    - Using different parameters
  - Select a fusion method to combine the classifiers

Ensemble Classifiers

- IDEA:
  - do not learn a *single* classifier but learn a *set of classifiers*
  - *combine the predictions* of multiple classifiers

- MOTIVATION:
  - reduce variance: results are less dependent on peculiarities of a single training set
  - reduce bias: a combination of multiple classifiers may learn a more expressive concept class than a single classifier

- KEY STEP:
  - formation of an ensemble of *diverse* classifiers from a single training set

Forming an Ensemble

- Modifying the data
  - Subsampling
    - bagging
    - boosting
  - randomly sampled feature subsets
- Modifying the learning task
  - pairwise classification / round robin learning
  - error-correcting output codes

- Exploiting the algorithm characterisitics
  - algorithms with random components
    - neural networks
  - randomizing algorithms
    - randomized decision trees
  - use multiple algorithms with different characteristics

How to implement MCS

- Implementation methods can be separated into two categories:
  - Implicit Diversity Training Methods
    - No diversity measurement is used directly
  - Explicit Diversity Training Methods
    - Diversity measurement is used in training

Implicit Methods

- Three most popular methods:
  - Bagging
  - Boosting (AdaBoost)
  - Random Feature Space Approach

Bagging (Bootstrap Aggregating)

- Breiman, 1996
- Derived from bootstrap (Efron, 1993)
- Several classifiers are trained independently by using different bootstrap sample sets (drawn with replacement) from the original dataset
- Then they are aggregated using Majority Voting (MV) or Averaging methods for each case

Bagging

- Given a standard training set *D* of size *n*, bagging generates *m* new training sets , each of size *n′* < *n*, from *D* uniformly by sampling examples and with replacement.
- By sampling with replacement, it is likely that some examples will be repeated in each . If *n′*=*n*, then for large *n* the set is expected to have the fraction (1 - 1/*e*) (≈63.2%) of the unique examples of *D*, the rest being duplicates.
- This kind of sample is known as a bootstrap sample. The *m* models are fitted using the above *m* bootstrap samples and combined by averaging the output (for regression) or voting (for classification).

Bagging Example

Original	1	2	3	4	5	6	7	8
Training set 1	2	7	8	3	7	6	3	1
Training set 2	7	8	5	6	4	2	7	1
Training set 3	3	6	2	7	5	6	2	2
Training set 4	4	5	1	4	6	4	3	8

Bagging

```
1. for $m = 1$ to $M$                    // $M$ ... number of iterations
   a) draw (with replacement) a bootstrap sample $S_m$ of the data
   b) learn a classifier $C_m$ from $S_m$
2. for each test example
   a) try all classifiers $C_m$
   b) predict the class  that receives the highest number of votes
```

- variations are possible
  - e.g., size of subset, sampling w/o replacement, etc.
- many related variants
  - sampling of features, not instances
  - learn a set of classifiers with different algorithms

Boosting (AdaBoost)

- AdaBoost is adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers.
- The classifiers it uses can be weak (i.e., display a substantial error rate), but as long as their performance is not random (resulting in an error rate of 0.5 for binary classification), they will improve the final model.
- Even classifiers with an error rate higher than would be expected from a random classifier will be useful, since they will have negative coefficients in the final linear combination of classifiers and hence behave like their inverses.

L. Breiman, "Arcing classifiers," The Annals of Statistics, 26 (3), pp. 801-849, 1998

Boosting (AdaBoost)

- AdaBoost generates and calls a new weak classifier in each of a series of rounds . For each call, a distribution of weights is updated that indicates the importance of examples in the data set for the classification.

- On each round, the weights of each incorrectly classified example are increased, and the weights of each correctly classified example are decreased, so the new classifier focuses on the examples which have so far eluded correct classification.

L. Breiman, "Arcing classifiers," The Annals of Statistics, 26 (3), pp. 801-849, 1998

Boosting (AdaBoost)

- Similar to bagging method, each individual classifier is also trained using a different training set
- But the training set is selected based on the error of the trained MCS
- The probability of a sample will be selected is higher if the MCS classifies it wrongly
- Finally, Weight Majority Voting (WMV) will be used as a fusion method
- When individual classifiers are not of identical accuracy, it is reasonable to attempt to give the more competent classifiers more power in making the final decision

L. Breiman, "Arcing classifiers," The Annals of Statistics, 26 (3), pp. 801-849, 1998

Boosting

- Basic Idea:
  - later classifiers focus on examples that were misclassified by earlier classifiers
  - weight the predictions of the classifiers with their error
- Realization
  - perform multiple iterations
    - each time using different example weights
  - weight update between iterations
    - increase the weight of incorrectly classified examples
    - this ensures that they will become more important in the next iterations
(misclassification errors for these examples count more heavily)
  - combine results of all iterations

Dealing with Weighted Examples

- directly
    - example e_i has weight w_i
    - number of examples n $\Rightarrow$ total example weight $\sum_{i=1}^{n} w_i$
- via sampling
    - interpret the weights as probabilities
    - examples with larger weights are more likely to be sampled
    - assumptions
        - sampling with replacement
        - weights are well distributed in [0,1]

Boosting – Algorithm AdaBoost

```
1. initialize example weights $w_i = 1/N$   $(i = 1..N)$
2. for $m = 1$ to $M$                    // $M$ ... number of iterations
   a) learn a classifier $C_m$ using the current example weights
   b) compute a weighted error estimate   $err_m = \frac{\sum w_i \textit{ of all incorrectly classified } e_i}{\sum_{i=1}^{N} w_i}$
   c) compute a classifier weight   $\alpha_m = \frac{1}{2} \log((1 - err_m)/err_m)$
   d) for all correctly classified examples $e_i$ :        $w_i \leftarrow w_i e^{-\alpha_m}$
   e) for all incorrectly classified examples $e_i$ :      $w_i \leftarrow w_i e^{\alpha_m}$
   f) normalize the weights $w_i$ so that they sum to 1
3. for each test example
   a) try all classifiers $C_m$
   b) predict the class  that receives the highest sum of weights $\alpha_m$
```

Random Feature Space Approach

- To choose a group of features for each classifier in an MCS
- The feature group size is chosen by an ad-hoc method
- Each individual classifier is then trained on the randomly selected group of features
- Their results are combined by a fusion rule.

T. K. Ho, "The random subspace method for constructing decision forests", IEEE Trans on pattern analysis and machine intelligence, 1998.

Explicit Method

- Explicit method can be separated into two types:
    - Overproduce and Select Method
        - The Most Diverse Ensemble
        - Kappa-error Plot Method

Overproduce and Select Method

- Different classifiers are trained to form a classifier pool
- A MCS is combined by selection from the pool based on some criteria, such as diverse measure or accuracy.

K. Tumer, & J. Ghosh . Analysis of decision boundaries in linearly combined neural classifiers. Pattern Recognition, 29:2, 341–348, 1996.

The Most Diverse Ensemble

- A pairwise diversity matrix is built based on double fault measure or Q statistics.
- The least related classifiers will be selected to form an MCS.
- The selection is carried out using a search method through the set of pairs of classifiers until the desired number of ensemble members is reached.

G. Giacinto and F. Roli, "Design of effective neural network ensembles for image classification processes", Image Vision and Computing Journal, 19 (9-10), 699-707, 2001

F. Roli, G. Giacinto, and G. Vernazza, "Methods for designing multiple classifier system," Proc. 2nd International Workshop on Multiple Classifier Systems, vol 2096 of Lecture Notes in Computer Science, Cambridge, UK, 2001, Springer – Verlag, pp. 79-87

Kappa-error Plot Method

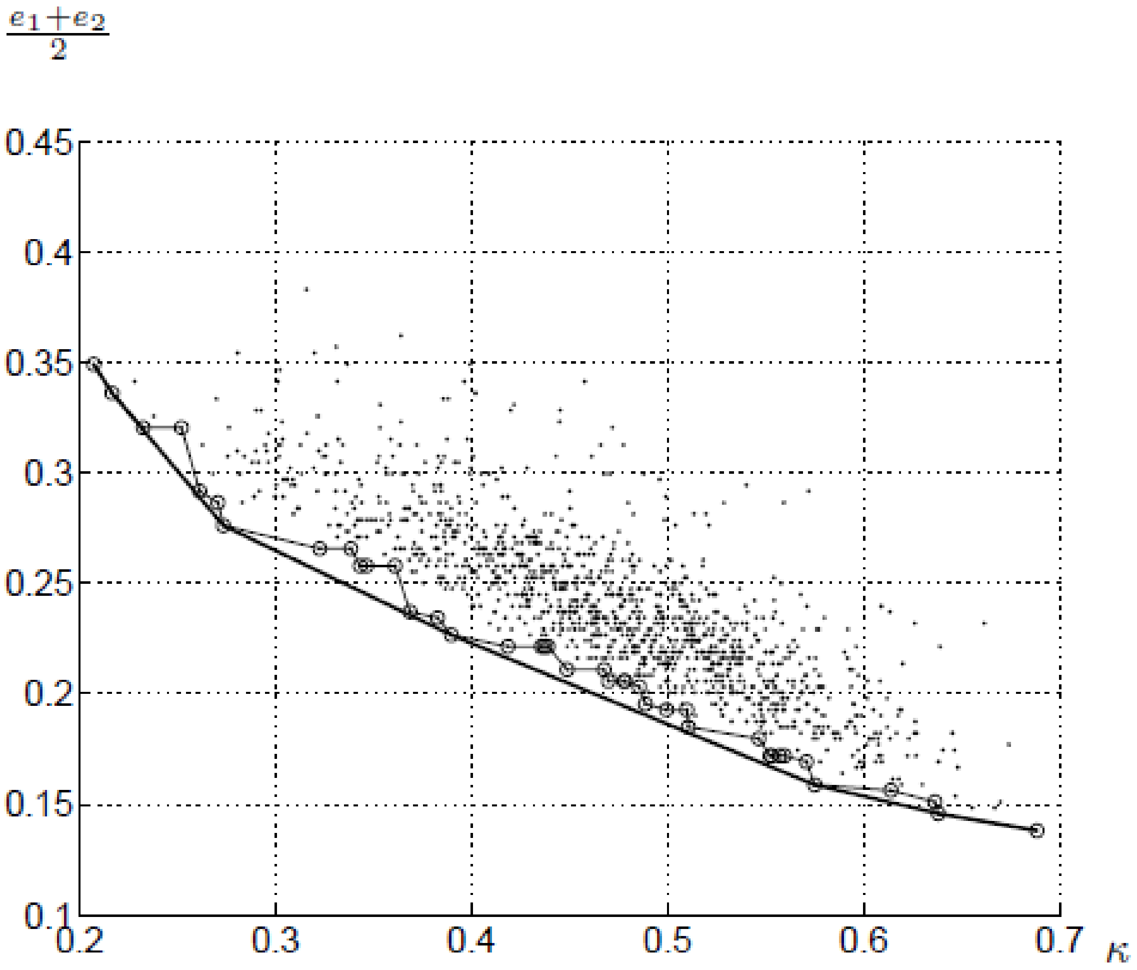
- Every pair of classifiers is plotted as a dot in a two-dimensional space
- X-axis: the pairwise diversity measure kappa

(Low values of kappa signify higher disagreement and hence higher diversity)

- Y-axis: the average of the individual training errors of the two classifiers
- There are L(L-1)/2 points in the scatterplot.
- The best pairs, which have low error and low kappa, are situated in the left bottom part of the plot.

D. D. Margineantu and T. G. Dietterich. "Pruning adaptive boosting", In Proc. 14th International Conference on Machine Learning, San Francisco, 1997, Morgan Kaufmann, pp. 378-387

Kappa-error Plot Method



D. D. Margineantu and T. G. Dietterich. "Pruning adaptive boosting", In Proc. 14th International Conference on Machine Learning, San Francisco, 1997, Morgan Kaufmann, pp. 378-387

Application of MCS

Application of MCS

- Bankruptcy Prediction & Credit Rating
- Network Intrusion Detection
- Handwriting Recognition
- Face Recognition

Bankruptcy Prediction & Credit Rating

- A set of financial ratios as input for MCS
- A decision for a given sample is bankrupt or non-bankrupt is an output

D. West, S. Dellana and J. Qian, "Neural network ensemble strategies for financial decision applications", Computers & operations research, Vol 32, pp. 2543-2559, 2005.

Bankruptcy Prediction & Credit Rating

- MCS with 100 MLP individual classifiers are used
- Each MLP has a single hidden layer, the number of hidden neurons is determined by random sampling with replacement
- 70% samples for training set, 15% for validation set and 15% for testing set

D. West, S. Dellana and J. Qian, ''Neural network ensemble strategies for financial decision applications'', Computers & operations research, Vol 32, pp. 2543-2559, 2005.

Bankruptcy Prediction & Credit Rating

- Three real world datasets are used
  - Dr. Hans Hofmann of the University of Hamburg contributed the German credit scoring data.
    - 700 examples of creditworthy applicants
    - 300 examples where credit should not be extended
    - 24 variables
      - 3 continuous and 21 categorical of which 9 are binary
      - E.g. credit history, account balances, loan purpose, loan amount, employment status, personal information, age, housing, and job.

D. West, S. Dellana and J. Qian, "Neural network ensemble strategies for financial decision applications", Computers & operations research, Vol 32, pp. 2543-2559, 2005.

Bankruptcy Prediction & Credit Rating

- German credit data
  - constructed by the authors from Standard and Poor's Compustat financial files
  - consists of five key financial ratios from Altman's research
    - These ratios constructed from financial statement information two years prior to bankruptcy include:
      - working capital/total assets
      - retained earnings/total assets
      - earnings before interest andtaxes/total assets
      - market value of equity/book value total liability
      - sales/total assets
  - There are a total of 329 observations in the data set with 93 bankrupt companies and 236 healthy companies

D. West, S. Dellana and J. Qian, "Neural network ensemble strategies for financial decision applications", Computers & operations research, Vol 32, pp. 2543-2559, 2005.

Bankruptcy Prediction & Credit Rating

- Australian credit scoring data
  - 307 and 383 examples of bad and good credit.
  - The data set contains a mixture of six continuous and eight categorical variables.

- We use labels of -1 (bad credit/bankrupt) and +1 (good credit/healthy) for all three data sets.

D. West, S. Dellana and J. Qian, "Neural network ensemble strategies for financial decision applications", Computers & operations research, Vol 32, pp. 2543-2559, 2005.

Bankruptcy Prediction & Credit Rating

Classifier type	Error % for Australian credit dataset	Error % for German credit dataset	Error % for bankruptcy dataset
MCS with CV	0.131	0.242	0.130
MCS with Bagging	0.128	0.251	0.126
MCS with Boosting	0.148	0.255	0.128
Single model	0.132	0.253	0.131

- The MCS with CV is the most accurate for the German credit data while the one with bagging is the most accurate for the Australian credit and bankruptcy data
- The MCS with boosting was competitive with bagging for the bankruptcy data

D. West, S. Dellana and J. Qian, "Neural network ensemble strategies for financial decision applications", Computers & operations research, Vol 32, pp. 2543-2559, 2005.

Bankruptcy Prediction & Credit Rating

- Each of the three ensemble strategies achieved a statistically significant reduction in error in at least one application
- The CV ensemble was most accurate for the German credit data
  - an application characterized by high noise levels, a relatively large training set size, and a large number of feature variables
- The bagging strategy was most effective for the Australian credit and the bankruptcy data set
  - both characterized by smaller training samples, fewer feature variables, and less noise
- The boosting strategy was effective for the bankruptcy data
  - the smallest data set with the fewest number of feature variables and the least amount of noise.

D. West, S. Dellana and J. Qian, "Neural network ensemble strategies for financial decision applications", Computers & operations research, Vol 32, pp. 2543-2559, 2005.

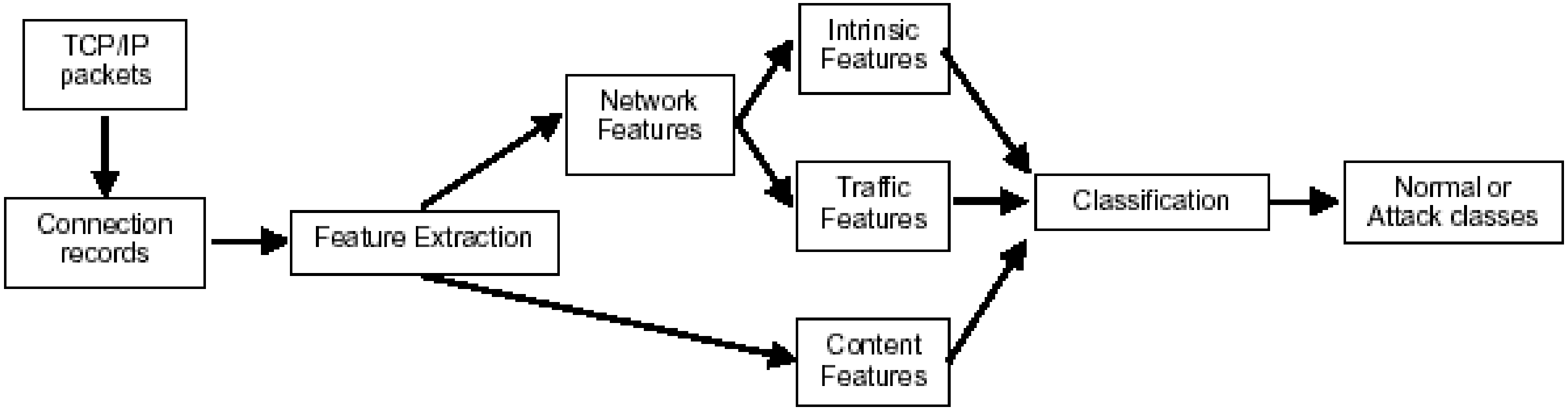
Network Intrusion Detection

- Each member of the classifier ensemble is trained on a distinct feature representation of patterns and combined using the fusion rules
- This approach is motivated by the observation that human experts try to design signatures that combine different attack characteristics in order to attain low false alarm rates and high attack detection rates
- But the manual development of such types of signatures is very difficult and tedious
- The fusion of multiple classifiers, using learning-by-examples, can automate such an approach, thus providing IDSs with the ability to detect new attacks without producing high false alarm rates.

G. Giorgio, R. Fabio & D. Luca, "Fusion of multiple classifiers for intrusion detection in computer networks", Pattern Recognition Letters 24, pp. 1795-1803, 2003.

Network Intrusion Detection

- The network intrusion detection problem can be formulated as given the information about network connections between pairs of hosts, assign each connection to one out of N data classes representing normal trace or different categories of intrusions e.g., Denial of Service, access to root privileges



G. Giorgio, R. Fabio & D. Luca, "Fusion of multiple classifiers for intrusion detection in computer networks", Pattern Recognition Letters 24, pp. 1795-1803, 2003.

Network Intrusion Detection

- The term “connection” refers to a sequence of data packets related to a particular service, e.g., the transfer of a web page via the http protocol.
- As the aim of a network intrusion detector is to detect connections related to malicious activities, each network connection can be defined as a pattern to be classified.

G. Giorgio, R. Fabio & D. Luca, “Fusion of multiple classifiers for intrusion detection in computer networks”, Pattern Recognition Letters 24, pp. 1795-1803, 2003.

Network Intrusion Detection

- The features are divided into three groups based on expert knowledge
  - content features,

    i.e., features containing information about the data content of packets that could be relevant to discover an intrusion, e.g., errors reported by the operating system, root access attempts, etc.
  - network related features

    intrinsic features,

    i.e., general information related to the connection. They include the duration, type and protocol

    trace features,

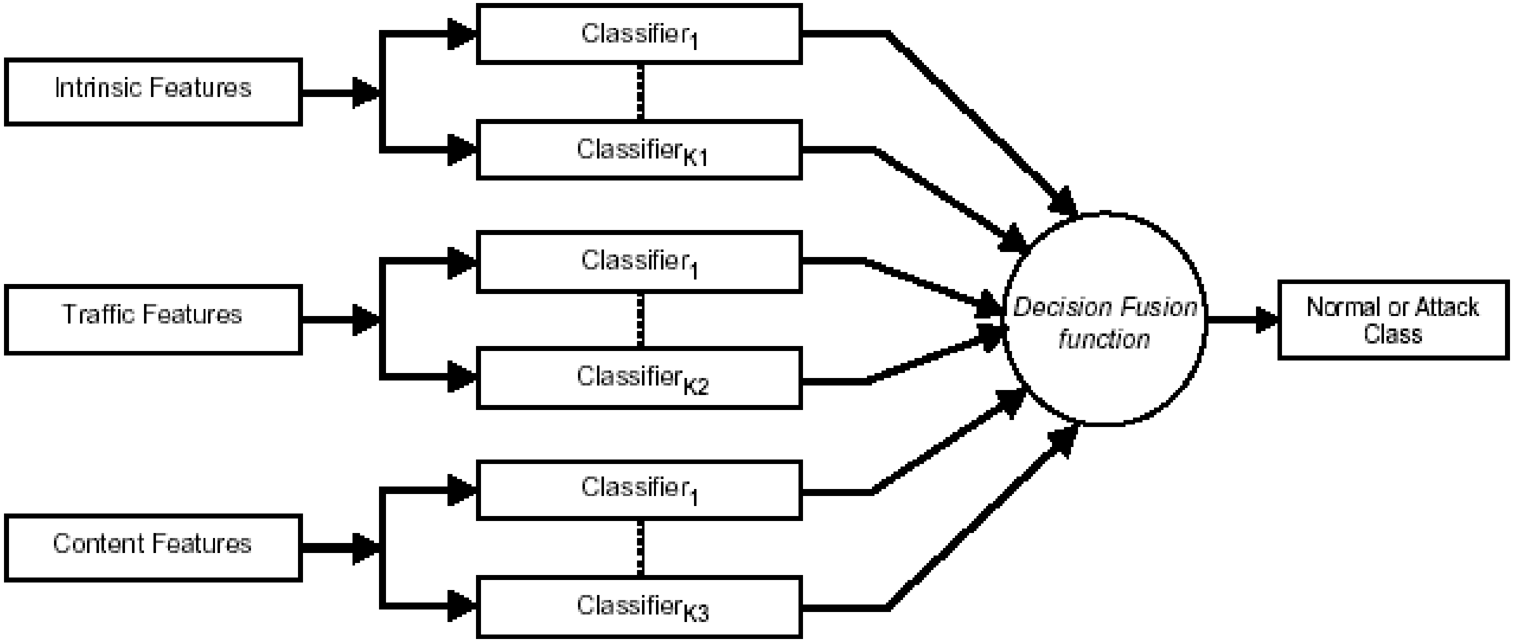
    i.e., statistics related to past connections similar to the current one e.g. number of connections with the same destination host or connections

    related to the same service in a given time window or within a predefined number of past connections

G. Giorgio, R. Fabio & D. Luca, "Fusion of multiple classifiers for intrusion detection in computer networks", Pattern Recognition Letters 24, pp. 1795-1803, 2003.

Network Intrusion Detection

- The architecture of MCS for network intrusion detection is shown as follows.



G. Giorgio, R. Fabio & D. Luca, "Fusion of multiple classifiers for intrusion detection in computer networks", Pattern Recognition Letters 24, pp. 1795-1803, 2003.

Network Intrusion Detection

- Experiments were carried out on a subset of the database created by DARPA in the framework of the 1998 Intrusion Detection Evaluation Program (http://www.ll.mit.edu/IST/ideval). The subset is preprocessed by the Columbia University and distributed as part of the UCI KDD archive.
- The available database is made up of a large number of network connections related to normal and malicious trace.
- Each connection is represented with a 41- dimensional feature vector
- Connections are also labeled as belonging to one out of five classes, i.e., normal trace, Denial of Service (DoS) attacks, Remote to Local (R2L) attacks, User to Root (U2R) attacks, and Probing attacks.

G. Giorgio, R. Fabio & D. Luca, "Fusion of multiple classifiers for intrusion detection in computer networks", Pattern Recognition Letters 24, pp. 1795-1803, 2003.

Network Intrusion Detection

- Only ftp connections are considered
- Discard 11 features out of 41, because they exhibited a constant value over all ftp connections
- A training set (725 samples)
    - 122 normal samples, 6 U2R attacks, 539 R2L attacks, 1 probing, and 57 DoS attacks
- A testing set (7436 samples)
    - 5128 normal samples and 2308 connections related to attacks
    - 125 attack connections were related to attack types not included in the training set
        - the classification results related to these patterns allowed to test the ability of the pattern recognition approach to detect novel attack types.

G. Giorgio, R. Fabio & D. Luca, "Fusion of multiple classifiers for intrusion detection in computer networks", Pattern Recognition Letters 24, pp. 1795-1803, 2003.

Network Intrusion Detection

MCS	Error %	False alarms %
Majority	0.89	1.29
Average	0.87	1.33
Naïve-Bayes	0.82	0.87
Decision templates	0.81	1.41
MLP-full feature set	1.55	3.57

- The detection performance of MCS with different fusion rules are comparative except the one with a posteriori DCS.

G. Giorgio, R. Fabio & D. Luca, "Fusion of multiple classifiers for intrusion detection in computer networks", Pattern Recognition Letters 24, pp. 1795-1803, 2003.

Handwriting Recognition

- each handwritten word input to the classifier has been normalized with respect to slant, skew, baseline location and height
- A sliding window of one pixel width is moved from left to right over the word and nine geometric features are extracted at each position of the window
- an input word is converted into a sequence of feature vectors in a 9-dimensional feature space
  - the fraction of black pixels in the window, the center of gravity, the second order moment, the position of the upper- and lowermost pixel, the contour direction at the position of the upper- and lowermost pixel, the number of black-to-white transitions in the window, and the fraction of black pixels between the upper- and lowermost black pixel

Simon Günter, Horst Bunke: Evaluation of Classical and Novel Ensemble Methods for Handwritten Word Recognition. SSPR/SPR 2004: 583-591

Handwriting Recognition

- Comparing different ensemble method
  - AdaBoost, Boositng, random subspace method, feature search and partition-based ensemble method
- a data set of 10,927 words with a vocabulary of size 2,296
  - That is, a classification problem with 2,296 different classes was considered.
  - The total number of writers who contributed to this set is 81.
  - A training set containing 9,861 words and a test set containing 1,066 words

Simon Günter, Horst Bunke: Evaluation of Classical and Novel Ensemble Methods for Handwritten Word Recognition. SSPR/SPR 2004: 583-591

Handwriting Recognition

Ensemble method	Recognition rate
AdaBoost	69.11%
Random subspace method	69.35%
Partition-based	70.83%
Boosting	69.82%
Feature search	72.04%
Single classifier	66.23%

- All ensemble methods are able to improve the performance of the single classifier
- The partition-based, boosting and feature search ensemble methods in handwriting recognition obtained better results than AdaBoost and random subspace method.

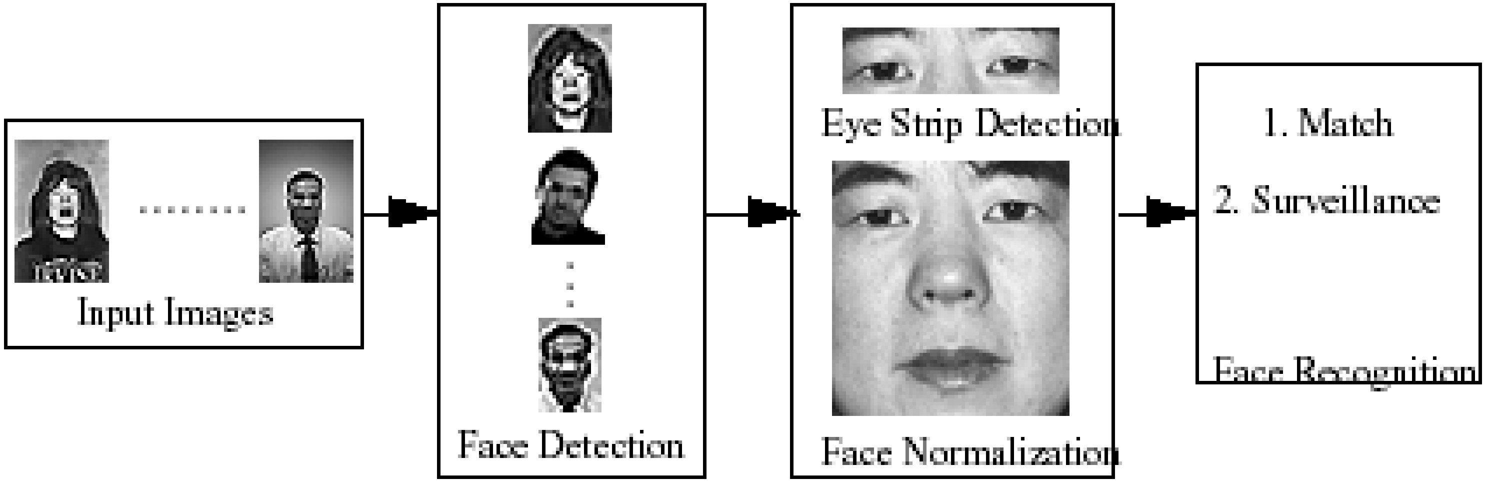
Simon Günter, Horst Bunke: Evaluation of Classical and Novel Ensemble Methods for Handwritten Word Recognition. SSPR/SPR 2004: 583-591

Face Recognition

- Face recognition starts through the detection of a pattern as a face and boxing it, proceeds then by normalizing the face image to account for geometry and illumination changes using information about the box surrounding the face and/or the eye location, and finally it identifies the face using appropriate classification algorithms.

Gutta, S.; Huang, J.; Takacs, B.; Wechsler, H., Face recognition using ensembles of networks, Pattern Recognition, 1996., Proceedings of the 13th International Conference on Volume 4, pp. 50 – 54, 1996.

Face Recognition



- The architecture of face recognition system is shown above

Gutta, S.; Huang, J.; Takacs, B.; Wechsler, H., Face recognition using ensembles of networks, Pattern Recognition, 1996., Proceedings of the 13th International Conference on Volume 4, pp. 50 – 54, 1996.

Face Recognition

- Use ensemble of Radial Basis Function and Decision trees
- The FERET facial database is used
  - It consists of 1109 sets comprising 8, 525 images
  - The lighting conditions and the size of the facial images can vary

- Example of facial images

Gutta, S.; Huang, J.; Takacs, B.; Wechsler, H., Face recognition using ensembles of networks, Pattern Recognition, 1996., Proceedings of the 13th International Conference on Volume 4, pp. 50 – 54, 1996.

Face Recognition

- 350 images for training and 372 images for testing
- Experiment on the dataset yields on the average 87 % correct match, and (ROC curves where) 99 % correct verification is achieved for a 2 % reject rate.

Gutta, S.; Huang, J.; Takacs, B.; Wechsler, H., Face recognition using ensembles of networks, Pattern Recognition, 1996., Proceedings of the 13th International Conference on Volume 4, pp. 50 – 54, 1996.

Research Problem

- Relationship between Diversity and MCS Accuracy
  - Current diversity measures cannot be shown to be directly related to MCS accuracy
  - How to find a diversity measure which can guarantee that?

Research Problem

- Fusion method
  - Current researches focus on:
    - Improving existing fusion method (e.g. how to handle tie cases in BKS? Dynamic weighted average vs weighted average)
    - Application specific fusion method
  - New fusion method which can guarantee MCS accuracy level

Research Problem

- MCS training method
  - Current
    - Static training sample selection for individual classifier
    - Feature selection for individual classifier
    - Parameter and classifier type selection
  - Dynamic training sample selection for individual classifier

Research Problem

- How to determine the number of classifiers in MCS
  - The number of classifiers is chosen by users in most of MCS implementation methods

Research Problem

- Generalization Error Model for MCS
  - Currently, the performance of MCS is estimated by using the testing accuracy which is dataset dependent
  - How to evaluate the generalization error of MCS?
  - Our research:
    - Generalization Error Model using sensitivity measure for single classifier has been developed
    - How to extend this model to MCS?