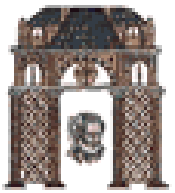
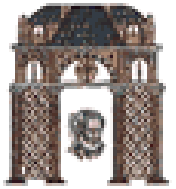


Αναγνώριση Προτύπων

Ομαδοποίηση Ταξινομητών – Data Fusion
(Fusion of Multiple Pattern Classifiers)

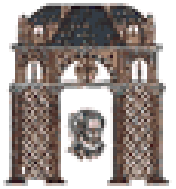
Χριστόδουλος Χαμζάς

Τα περιεχόμενο αυτής της παρουσίασης βασίζεται σε παρουσίαση του μαθήματος Visual Information Systems, Computer Science Department, Surrey University, και στο βιβλίο *L.Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, John Wiley & Sons, 2004 , καθώς και στην δημοσίευση Automatic Ranking of Retrieval Systems using Data Fusion (Nuray,R & Can,F, IPM 2006)*



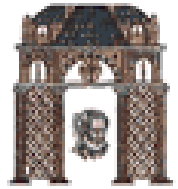
Το πρόβλημα

- Οι πιο πολλοί ταξινομητές που εξετάσαμε δεν δίνουν ακριβώς τα ίδια αποτελέσματα. (διαφορετικός αλγόριθμος, διαφορετικοί παράμετροι, διαφορετικό σύνολο εκμάθησης, κ,λ.π.)
- Οι ταξινομητές μπορούν να δώσουν και μία σειρά ταξινόμησης σε ποια κατηγορία ανήκει ένα πρότυπο, διότι μας δίνουν και ένα μέτρο confidence του αποτελέσματος (Bayesian, LMS, Neural Nets e.t.c.)
- Είναι επιθυμητό να μπορέσουμε να συνδυάσουμε τα αποτελέσματα, περισσοτέρων του ενός ταξινομητή.



Definitions

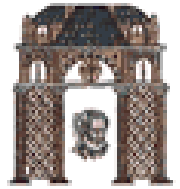
- A "classifier" is any mapping from the space of features(measurements) to a space of class labels (names, tags, distances, probabilities)
- A classifier is a hypothesis about the real relation between features and class labels
- A "learning algorithm" is a method to construct hypotheses
- A learning algorithm applied to a set of samples (training set) outputs a classifier



Combining Classifiers

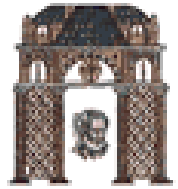
The basic philosophy behind the combination of different classifiers lies in the fact that even the "best" classifier fails in some patterns that other classifiers may classify correctly. Combining classifiers aims at exploiting this complementary information residing in the various classifiers.

Thus, one designs different optimal classifiers and then combines the results with a specific rule.



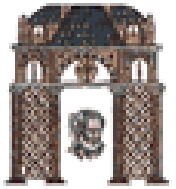
Data Fusion: Learn to Rank

- Merging the retrieval results of multiple systems.
- A data fusion algorithm accepts two or more ranked lists and merges these lists into a single ranked list with the aim of providing better effectiveness than all systems used for data fusion.



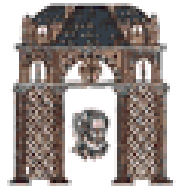
Fusion of Classifiers. Why?

- A natural move when trying to solve numerous complicated patterns
- Efficiency
  - Dimension;
  - Complicated architecture such as neural network;
  - Speed;
- Accuracy



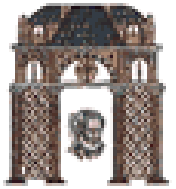
Traditional approach to pattern classification

- Unfortunately, no dominant classifier exists for all the data distributions, and the data distribution of the task at hand is usually unknown
- Not one classifier can discriminative well enough if the number of classes are huge
- For applications where the objects/classes of content are numerous, unlimited, unpredictable, one specific classifier/detector cannot solve the problem.



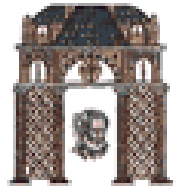
Combine individual classifiers

- Beside avoiding the selection of the worse classifier, under particular hypothesis, fusion of multiple classifiers can improve the performance of the best individual classifiers and, in some special cases, provide the optimal Bayes classifier
- This is possible if individual classifiers make "different" errors
- For linear combiners, Turner and Ghosh (1996) showed that averaging outputs of individual classifiers with unbiased and uncorrelated errors can improve the performance of the best individual classifier and, for infinite number of classifiers, provide the optimal Bayes classifier



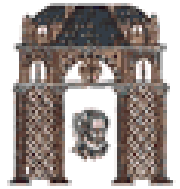
Definitions

- A multiple classifier system (MCS) is a structured way to combine (exploit) the outputs of individual classifiers
- Two classifiers are diverse, if they make different errors on a new object
- MCS can be thought as (different names):
  - Multiple expert systems
  - Committees of experts
  - Mixtures of experts
  - Classifier ensembles
  - Composite classifier systems



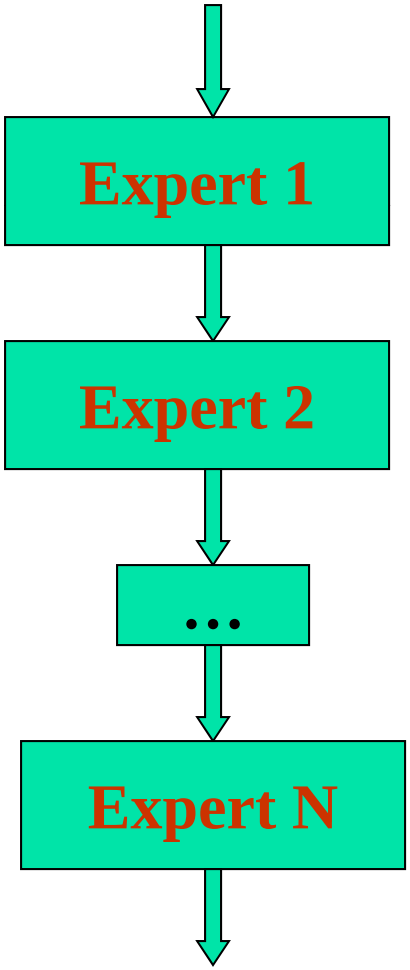
Basic concepts

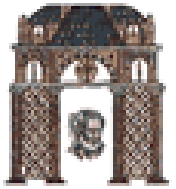
- Multiple Classifier Systems (MCS) can be characterized by:
  - The Architecture
  - Fixed/Trained Combination strategy
  - Others
- A necessary condition for the approach to be useful is that member classifiers should have a substantial level of disagreement, i.e., they make error independently with respect to one another (diversity)
  - Combining identical classifiers is useless!



MCS Architecture/Topology

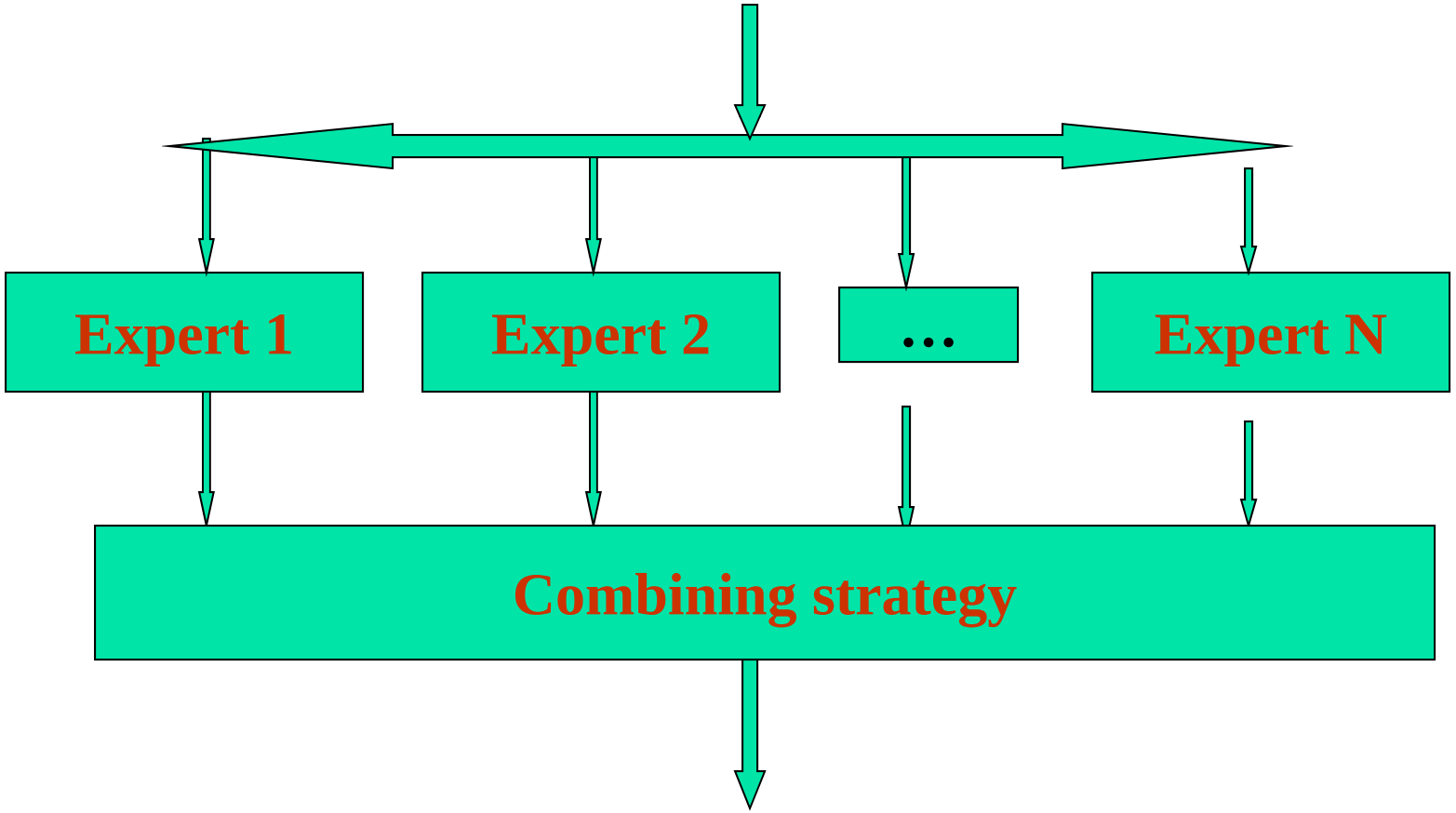
- Serial

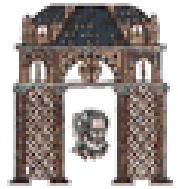




MCS Architecture/Topology

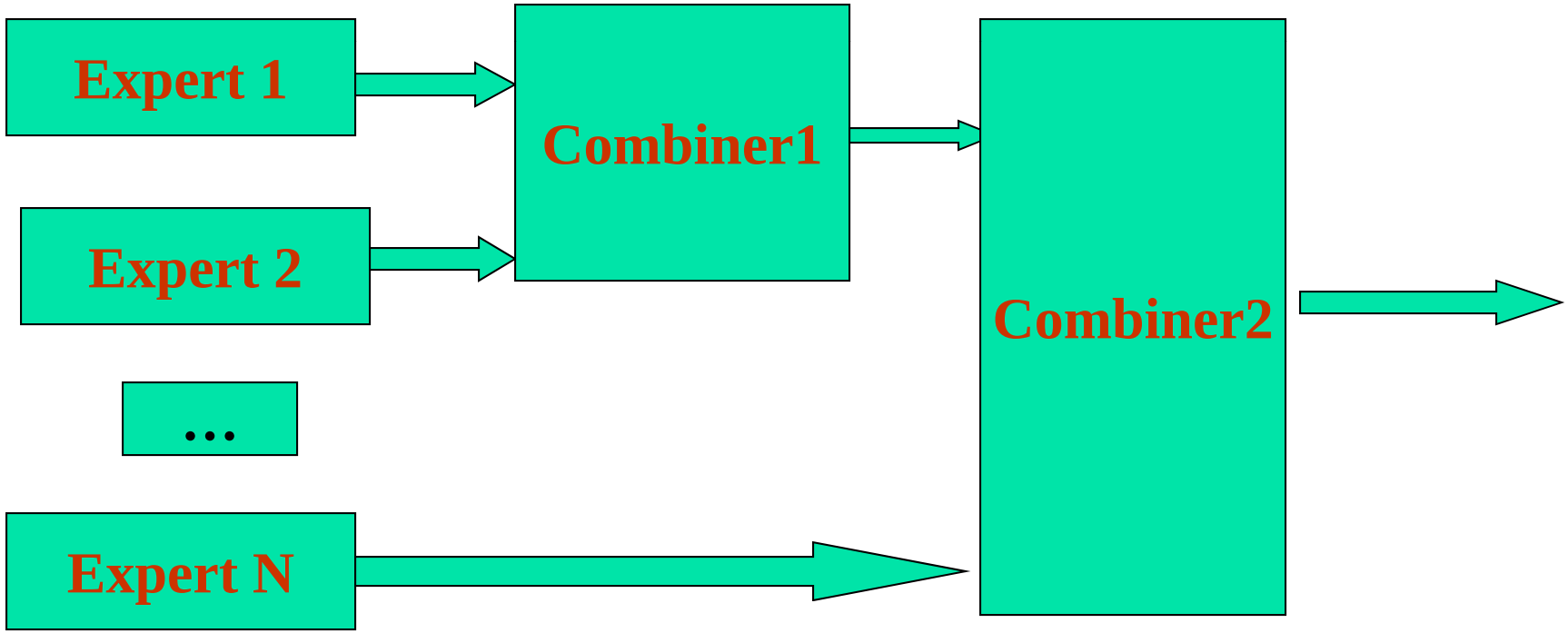
- Parallel

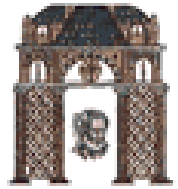




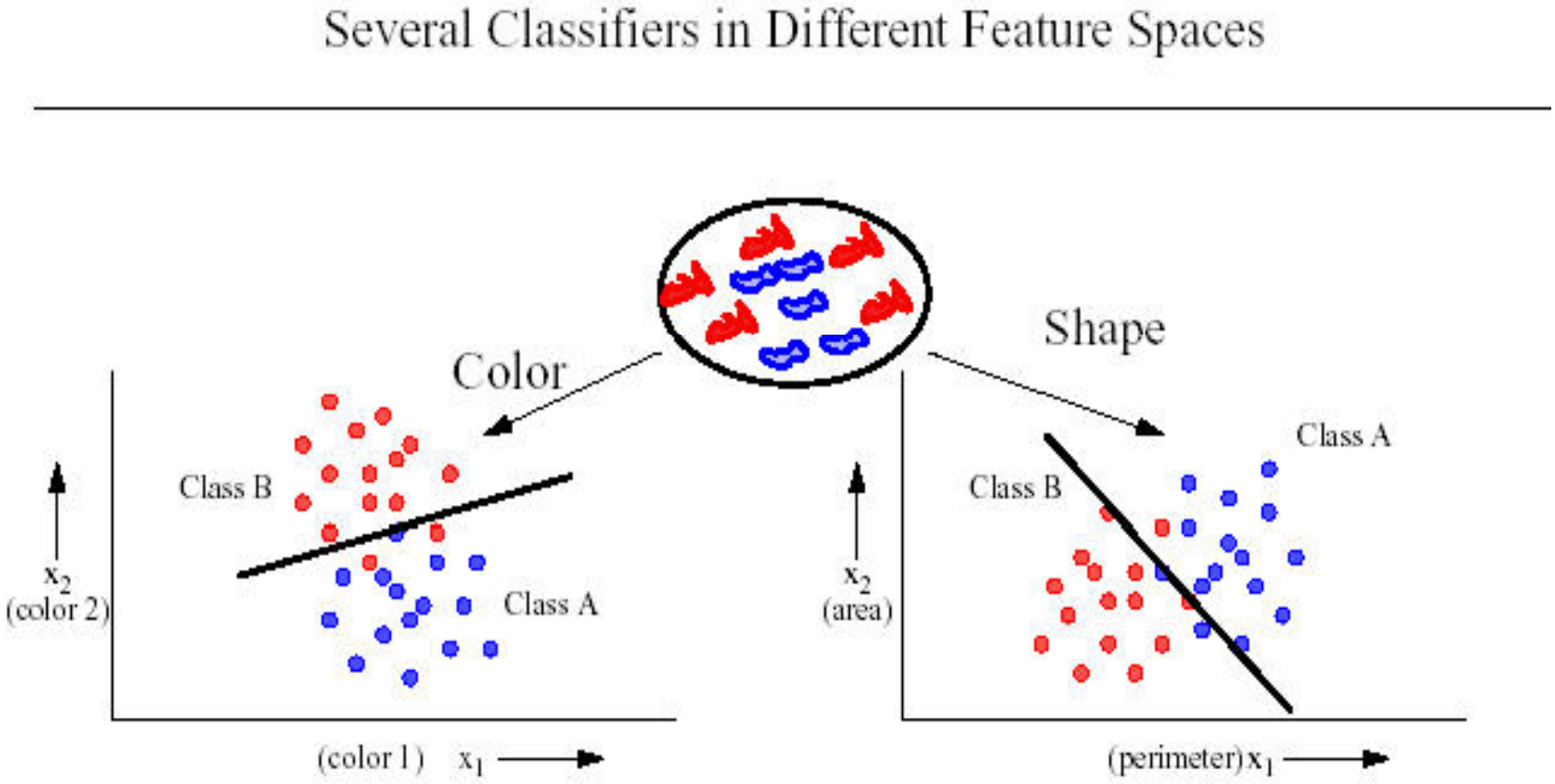
MCS Architecture/Topology

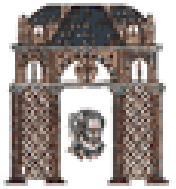
- Hybrid



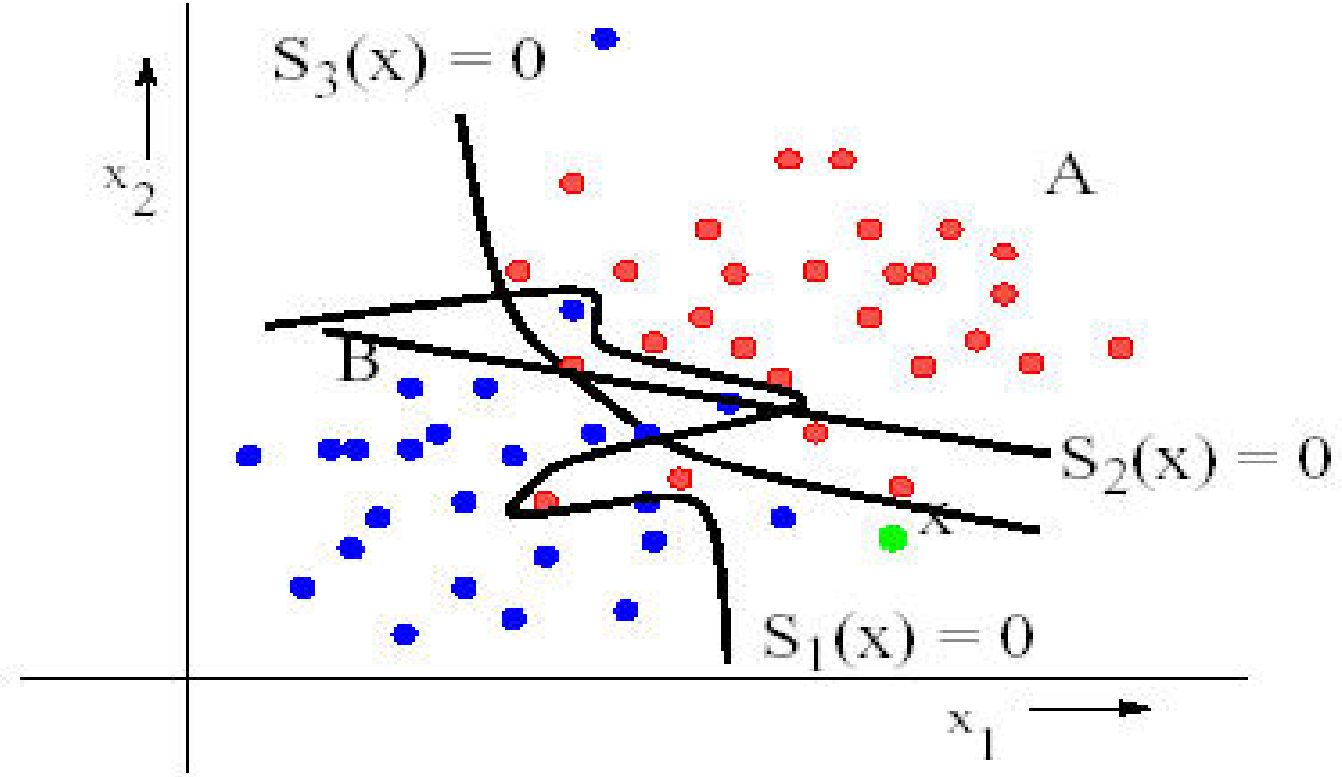


Multiple Classifiers Sources?

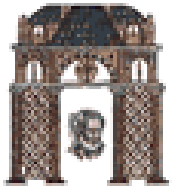




Multiple Classifiers Sources?

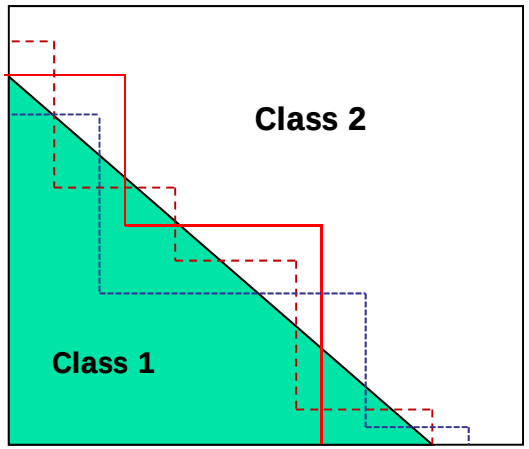


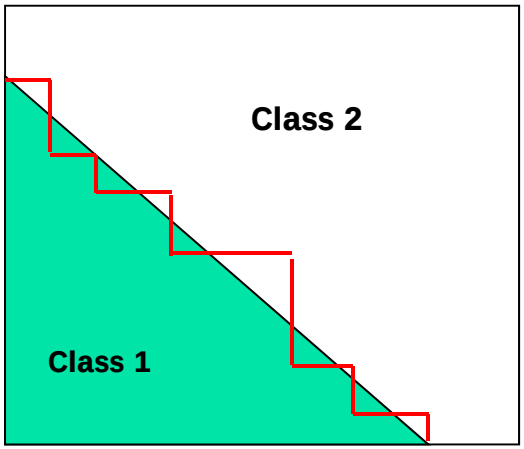
Same feature space, three classifiers demonstrate different performance

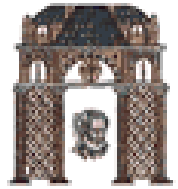


Averaging Results

- The diagonal decision boundary may be difficult for individual classifiers, but may be approximated by ensemble averaging.
- Decision boundaries constricted by decision trees → hyperplanes parallel to the coordinate axis – "staircases".
- By averaging a large number of "staircases" the diagonal boundary can be approximated with some accuracy.



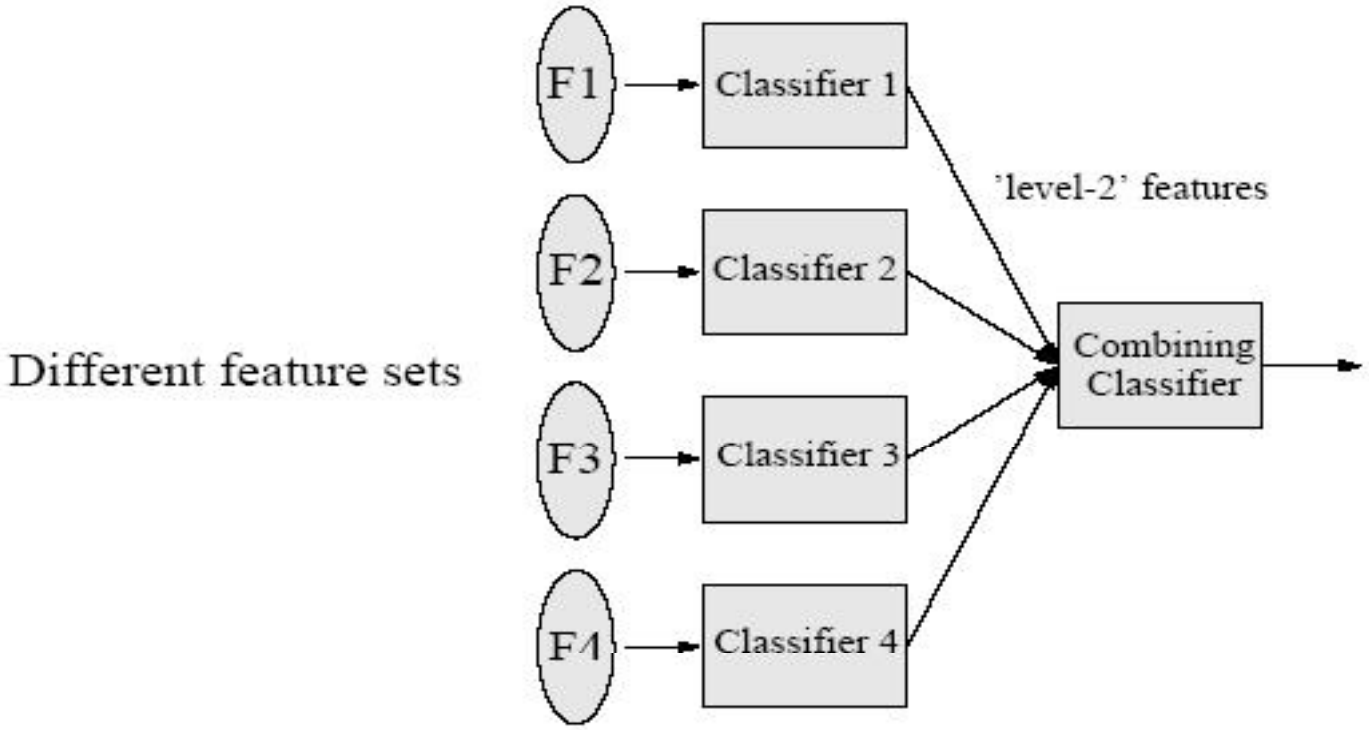




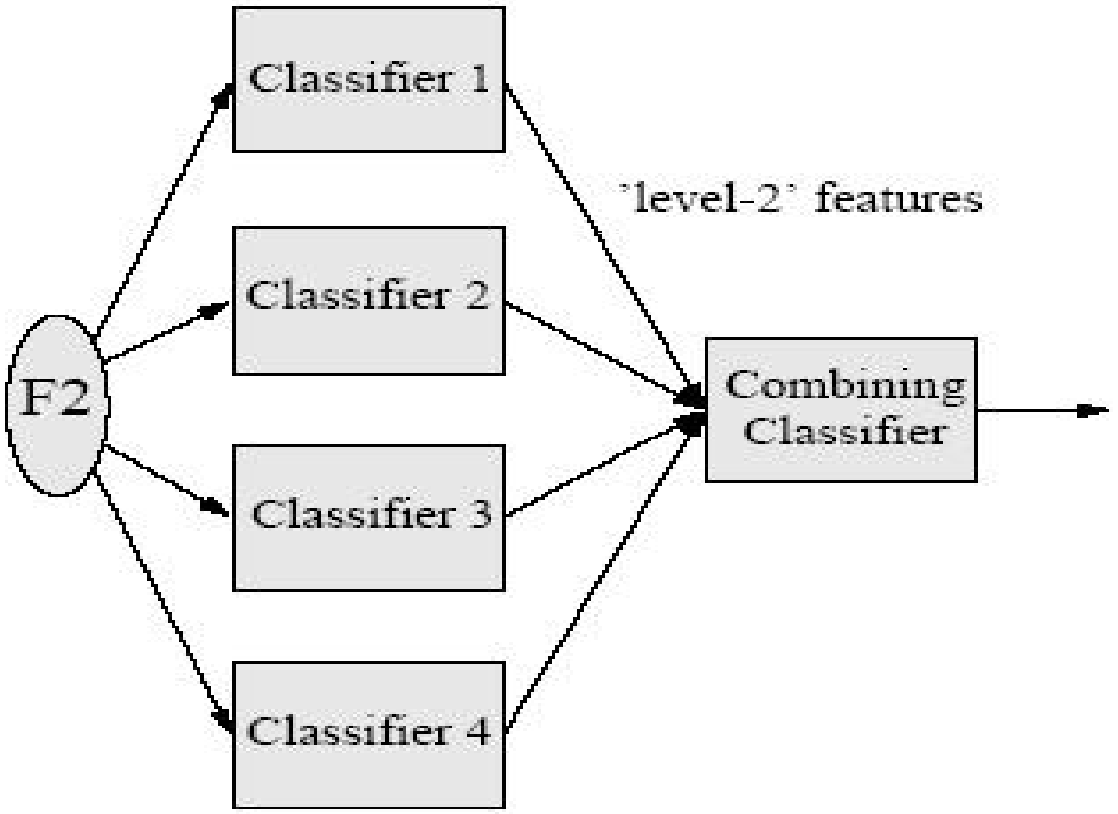
Multiple Classifiers Sources?

- Different feature spaces: face, voice fingerprint;
- Different training sets: Sampling;
- Different classifiers: K_NN, Neural Net, SVM;
- Different architectures: Neural net: layers, Units, transfer function;
- Different parameter values: K in K_NN, Kernel in SVM;
- Different initializations: Neural net

Combination based on different feature spaces

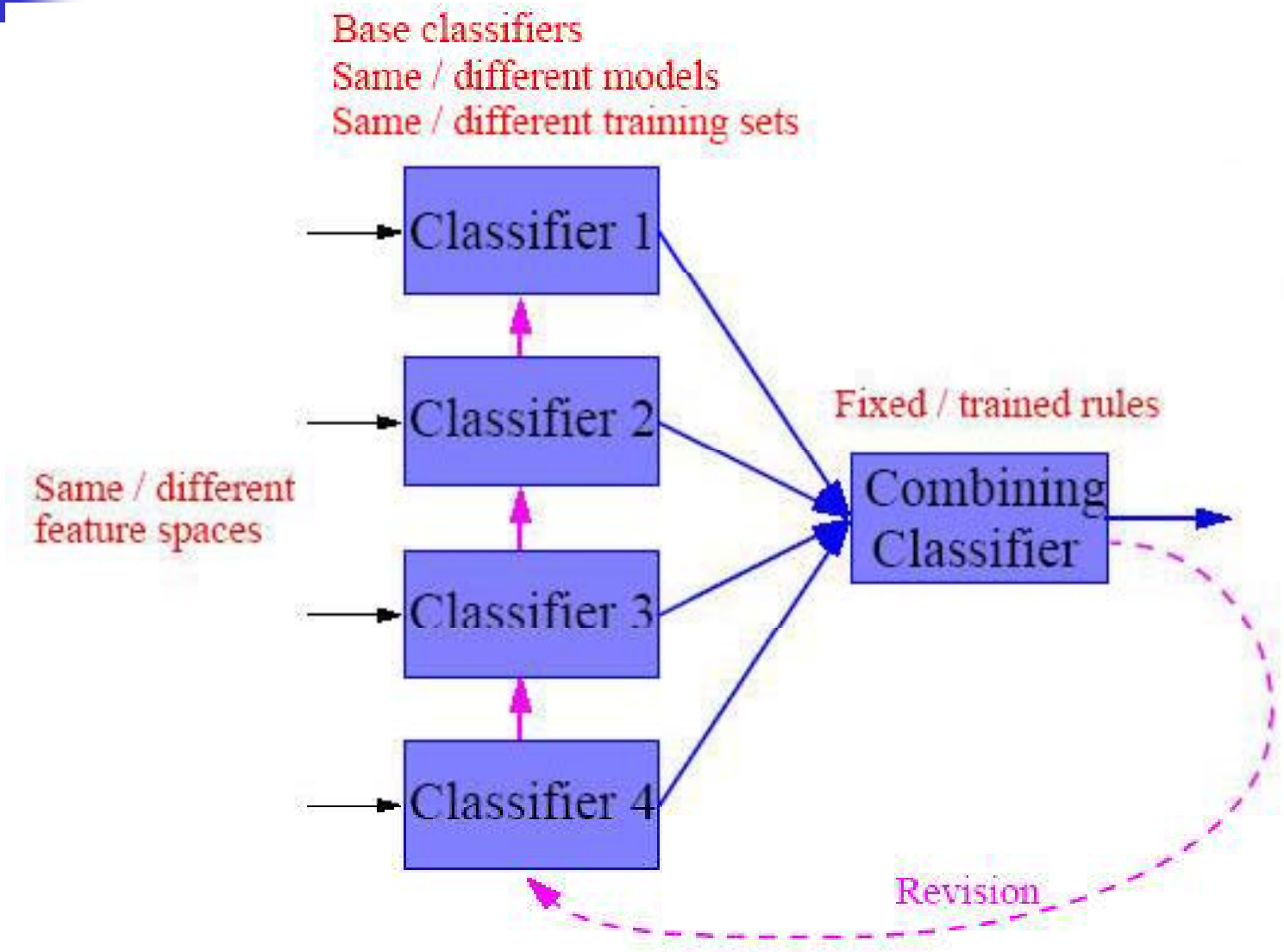


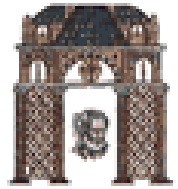
Combining based on a single space but different classifiers



Single feature set, different classifiers

Architecture of multiple classifier combination

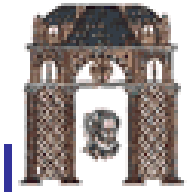




Fixed Combination Rules

- Product, Minimum
  - Independent feature spaces;
  - Different areas of expertise;
  - Error free posterior probability estimates
- Sum(Mean), Median, Majority Vote
  - Equal posterior-estimation distributions in same feature space;
  - Differently trained classifiers, but drawn from the same distribution
  - Bad if some classifiers(experts) are very good or very bad
- Maximum Rule
  - Trust the most confident classifier/expert;
  - Bad if some classifiers(experts) are badly trained.

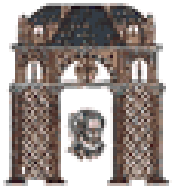
Ever optimal?



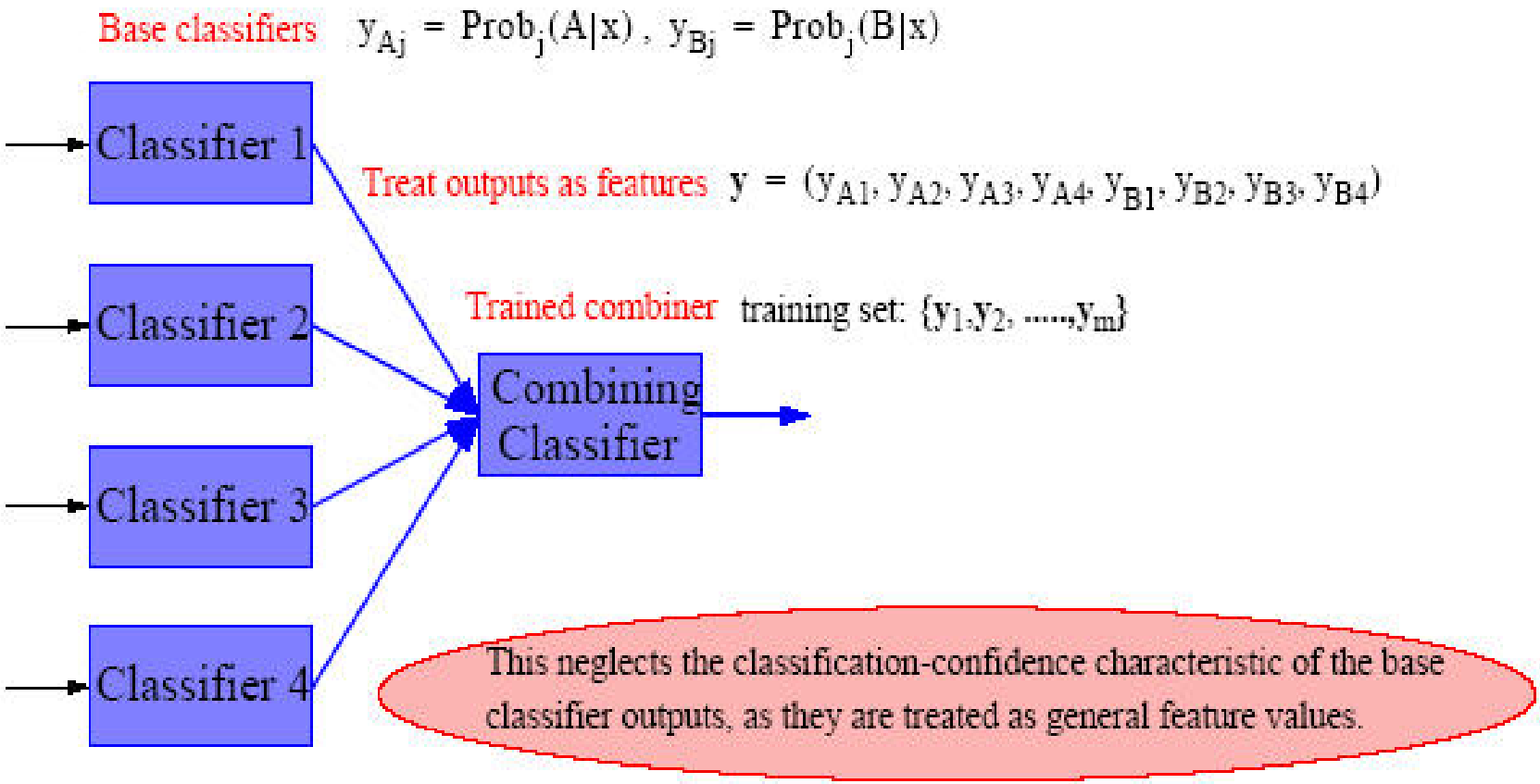
Fixed combining rules are sub-optimal

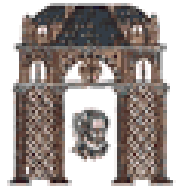
- Base classifiers are never really independent (Product)
- Base classifiers are never really equally imperfectly trained (sum,median,majority)
- Sensitivity to over-confident base classifiers(product, min,max)

Fixed combining rules are never optimal



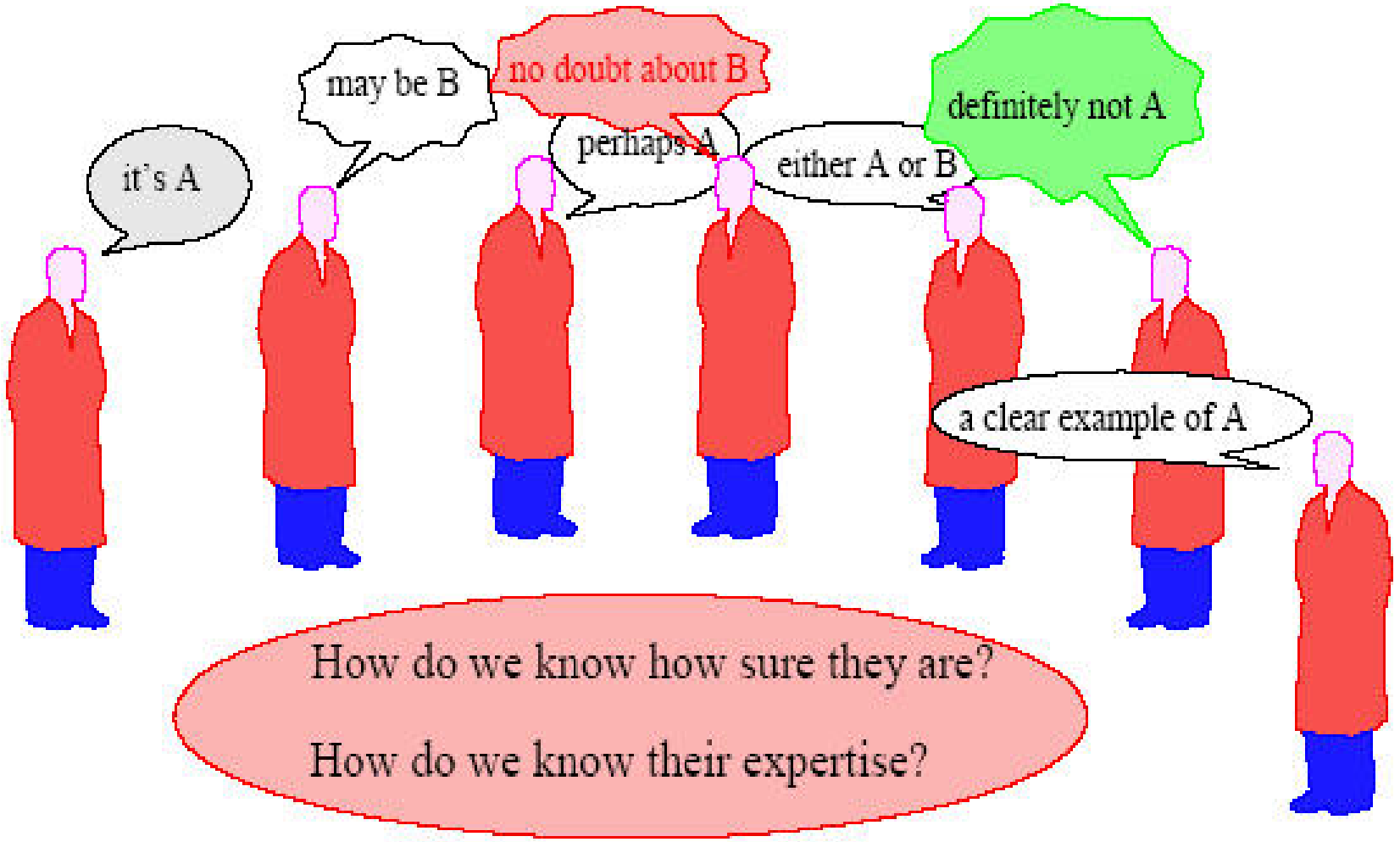
Trained combiner

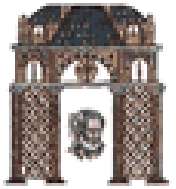




Remarks on fixed and trained combination strategies

- Fixed rules (i.e. Majority rule)
  - Simplicity
  - Low memory and time requirements
  - Well-suited for ensembles of classifiers with independent/low correlated errors and similar performances
- Trained rules
  - Flexibility: potentially better performances than fixed rules
  - Trained rules are claimed to be more suitable than fixed ones for classifiers correlated or exhibiting different performances
  - High memory and time requirements


it's A
may be B
no doubt about B
perhaps A
either A or B
definitely not A
a clear example of A
How do we know how sure they are?
How do we know their expertise?

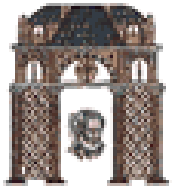


Methods for fusing multiple classifiers

- Methods for fusing multiple classifiers can be classified according to the type of information produced by the individual classifiers (Xu et al., 1992)
  - The abstract level output: a classifier only outputs a unique label for each input pattern;
  - The rank level output: each classifier outputs a list of possible classes, with ranking, for each input pattern
  - The measurement level output: each classifier outputs class "confidence" levels for each input pattern

  For each of the above categories, methods can be further subdivided into:

  Integration vs **Selection** rules and **Fixed** rules vs **trained** rules



Examples of Fixed Rules

The Product Rule

Assign $\underline{x}$ to the class ω_i :

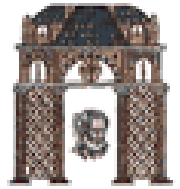
$$i = \arg\max_k \prod_{j=1}^{L} P_j(\omega_k \mid \underline{x})$$

where $P_j(\omega_k \mid \underline{x})$ is the respective posterior probability of the j^{th} classifier.

The Sum Rule

Assign $\underline{x}$ to the class ω_i :

$$i = \arg\max_k \sum_{j=1}^{L} P_j(\omega_k \mid \underline{x})$$



Example of Fixed Rules

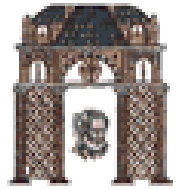
The majority voting rule

Assign to the class for which there is a consensus or when at least of the classifiers agree on the class label of where:

$$\ell_c = \begin{cases} \frac{L}{2}+1, \text{L even} \\ \frac{L+1}{2}, \text{L odd} \end{cases}$$

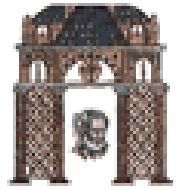
otherwise the decision is rejection, that is no decision is taken.

Thus, correct decision is made if the majority of the classifiers agree on the correct label, and wrong decision if the majority agrees in the wrong label.

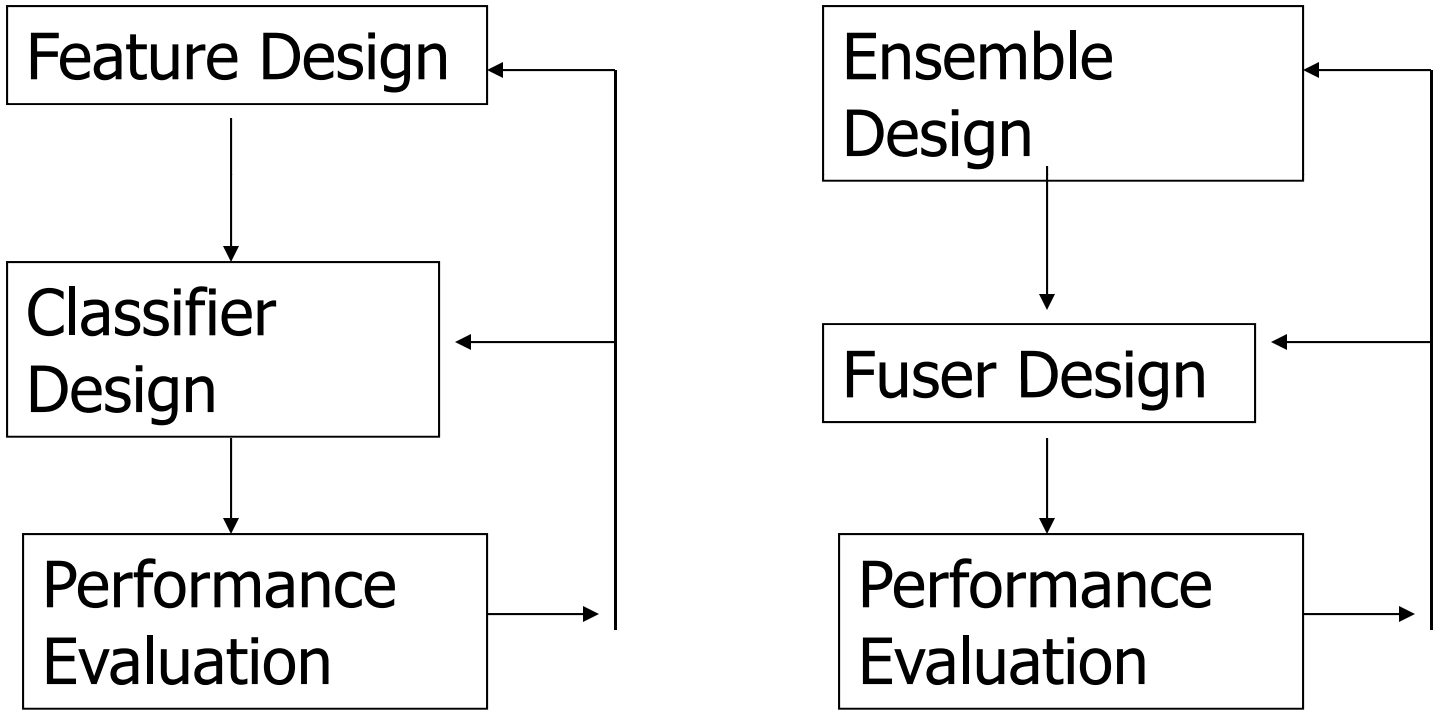


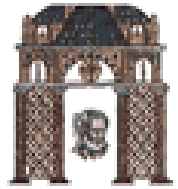
Fuser ("Combination" rule)

- Two main categories of fuser:
- Integration (fusion) functions: for each pattern, all the classifiers contribute to the final decision. Integration assumes competitive classifiers
- Selection functions: for each pattern, just one classifier, or a subset, is responsible for the final decision. Selection assumes complementary classifiers
- Integration and Selection can be "merged" for designing the hybrid fuser
- Multiple functions for non-parallel architecture can be necessary



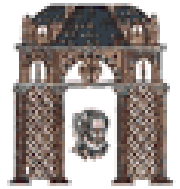
Analogy Between MCS and Single Classifier Design





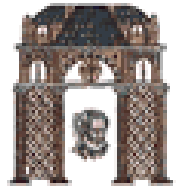
MCS Design

- The design of MCS involves two main phases: the design of the classifier ensemble, and the design of the fuser
- The design of the classifier ensemble is aimed to create a set of complementary/diverse classifiers
- The design of the combination function/fuser is aimed to create a fusion mechanism that can exploit the complementarity/diversity of classifiers and optimally combine them
- The two above design phases are obviously linked (Roli and Giacinto, 2002)



Methods for Constructing MCS

- The effectiveness of MCS relies on combining diverse/complentary classifiers
- Several approaches have been proposed to construct ensembles made up of complementary classifiers. Among the others:
  - Using problem and designer knowledge
  - Injecting randomness
  - Varying the classifier type, architecture, or parameters
  - Manipulating training data
  - Manipulating input features
  - Manipulating output features



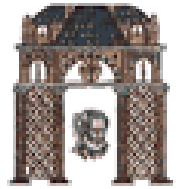
AdaBoost Algorithm

The Boosting Approach

➢The origins: Is it possible a weak learning algorithm (one that performs slightly better than a random guessing) to be boosted into a strong algorithm? (Villiant 1984).

➢The procedure to achieve it:

- Adopt a weak classifier known as the base classifier.
- Employing the base classifier, design a series of classifiers, in a hierarchical fashion, each time employing a different weighting of the training samples. Emphasis in the weighting is given on the hardest samples, i.e., the ones that keep "failing".
- Combine the hierarchically designed classifiers by a weighted average procedure.



AdaBoost Algorithm (cont.)

Construct an optimally designed classifier of the form:

$$f(\underline{x}) = \text{sign}\{F(\underline{x})\}$$

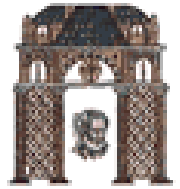
where:

$$F(\underline{x}) = \sum_{k=1}^{K} a_k \varphi(\underline{x}; \underline{\vartheta}_k)$$

Where $\varphi(\underline{x}; \underline{\vartheta}_k)$ denotes the <u>base classifier</u> that returns a binary class label:

$$\varphi(\underline{x}; \underline{\vartheta}_k) \in \{-1, 1\}$$

$\underline{\vartheta}$ is a parameter vector.



AdaBoost Algorithm (cont.)

- The essence of the method.

  Design the series of classifiers:

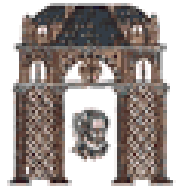
$$\varphi\left(\underline{x};\underline{\vartheta}_1\right),\ \varphi\left(\underline{x};\underline{\vartheta}_2\right),\ ...,\ \varphi\left(\underline{x};\underline{\vartheta}_k\right)$$

  The parameter vectors

$$\underline{\vartheta}_k,\ k=1,\ 2,\ ...,\ K$$

  are optimally computed so as:

  - To minimize the error rate on the training set.
  - Each time, the training samples are re-weighted so that the weight of each sample depends on its history. Hard samples that "insist" on failing to be predicted correctly, by the previously designed classifiers, are more heavily weighted.



AdaBoost Algorithm (cont.)

- Updating the weights for each sample $\underline{x}_i, i = 1, 2, \ldots, N$

$$w_i^{(m+1)} = \frac{w_i^m \exp\left(-y_i a_m \varphi\left(\underline{x}_i; \underline{\vartheta}_m\right)\right)}{Z_m}$$

- Z_m is a normalizing factor common for all samples.

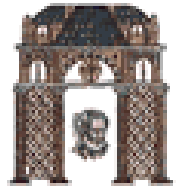
$$a_m = \frac{1}{2} \ln \frac{1 - P_m}{P_m}$$

- where P_m<0.5 (by assumption) is the error rate of the optimal classifier $\varphi\left(\underline{x}; \underline{\vartheta}_m\right)$ at stage m. Thus $\alpha_m > 0$.

- The term: $\exp\left(-y_i a_m \varphi\left(\underline{x}_i; \underline{\vartheta}_m\right)\right)$

- takes a large value if $y_i \varphi\left(\underline{x}_i; \underline{\vartheta}_m\right) < 0$ (wrong classification) and a small value in the case of correct classification $\{y_i \varphi\left(\underline{x}_i; \underline{\vartheta}_m\right) > 0\}$

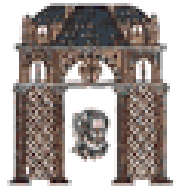
- The update equation is of a multiplicative nature. That is, successive large values of weights (hard samples) result in larger weight for the next iteration



AdaBoost Algorithm (cont.)

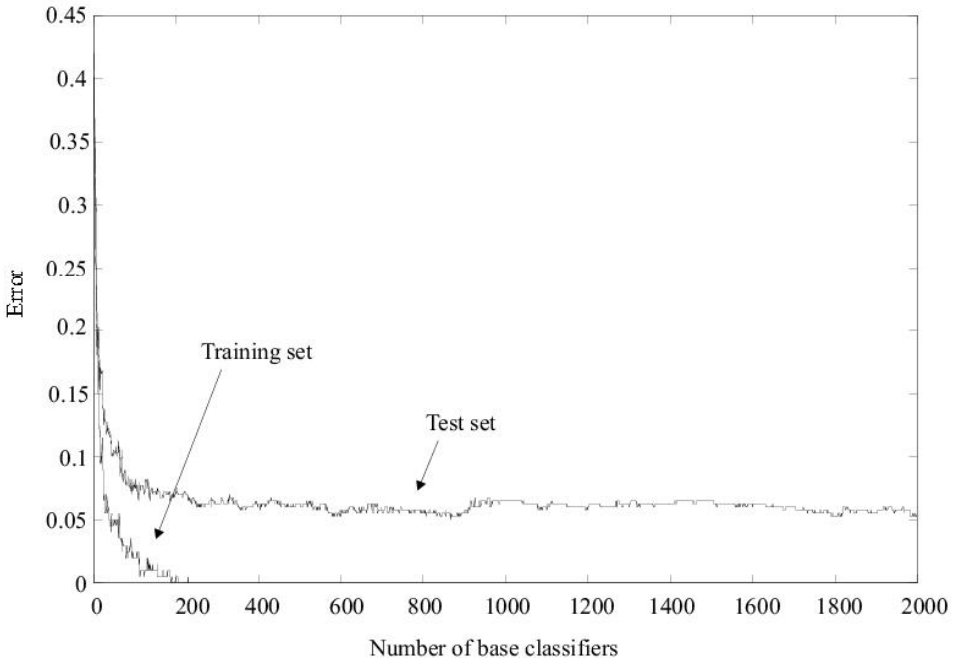
- Initialize: $w_i^{(1)} = \frac{1}{N},\ i = 1, 2 \ldots, N$
- Initialize: $m = 1$
- Repeat
  - Compute optimum $\boldsymbol{\theta}_m$ in $\phi(\cdot; \boldsymbol{\theta}_m)$ by minimizing P_m
  - Compute the optimum P_m
  - $\alpha_m = \frac{1}{2} \ln \frac{1-P_m}{P_m}$
  - $Z_m = 0.0$
  - For $i = 1$ to N
    - $w_i^{(m+1)} = w_i^{(m)} \exp\left(-y_i \alpha_m \phi(\boldsymbol{x}_i; \boldsymbol{\theta}_m)\right)$
    - $Z_m = Z_m + w_i^{(m+1)}$
  - End{For}
  - For $i = 1$ to N
    - $w_i^{(m+1)} = w_i^{(m+1)} / Z_m$
  - End {For}
  - $K = m$
  - $m = m + 1$
- Until a termination criterion is met.
- $f(\cdot) = \text{sign}(\sum_{k=1}^{K} \alpha_k \phi(\cdot, \theta_k))$

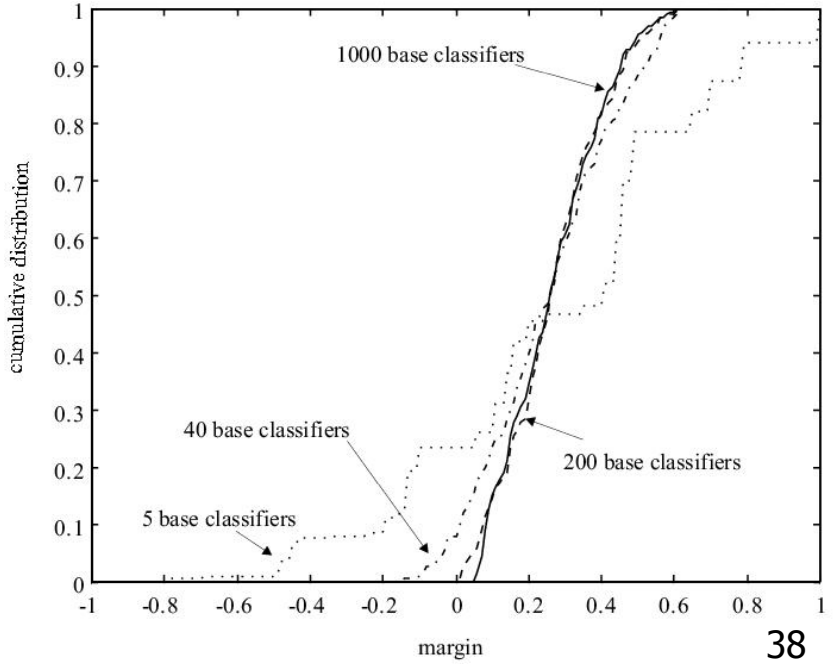
37

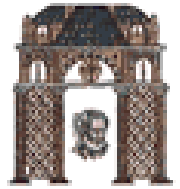


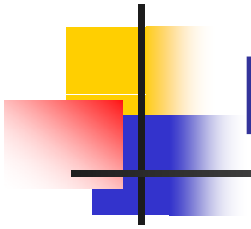
AdaBoost Algorithm (cont.)

- Remarks:
  - Training error rate tends to zero after a few iterations. The test error levels to some value.
  - AdaBoost is greedy in reducing the margin that samples leave from the decision surface.



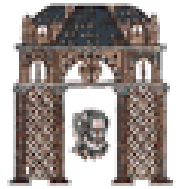






Part B

LEARNING to RANK



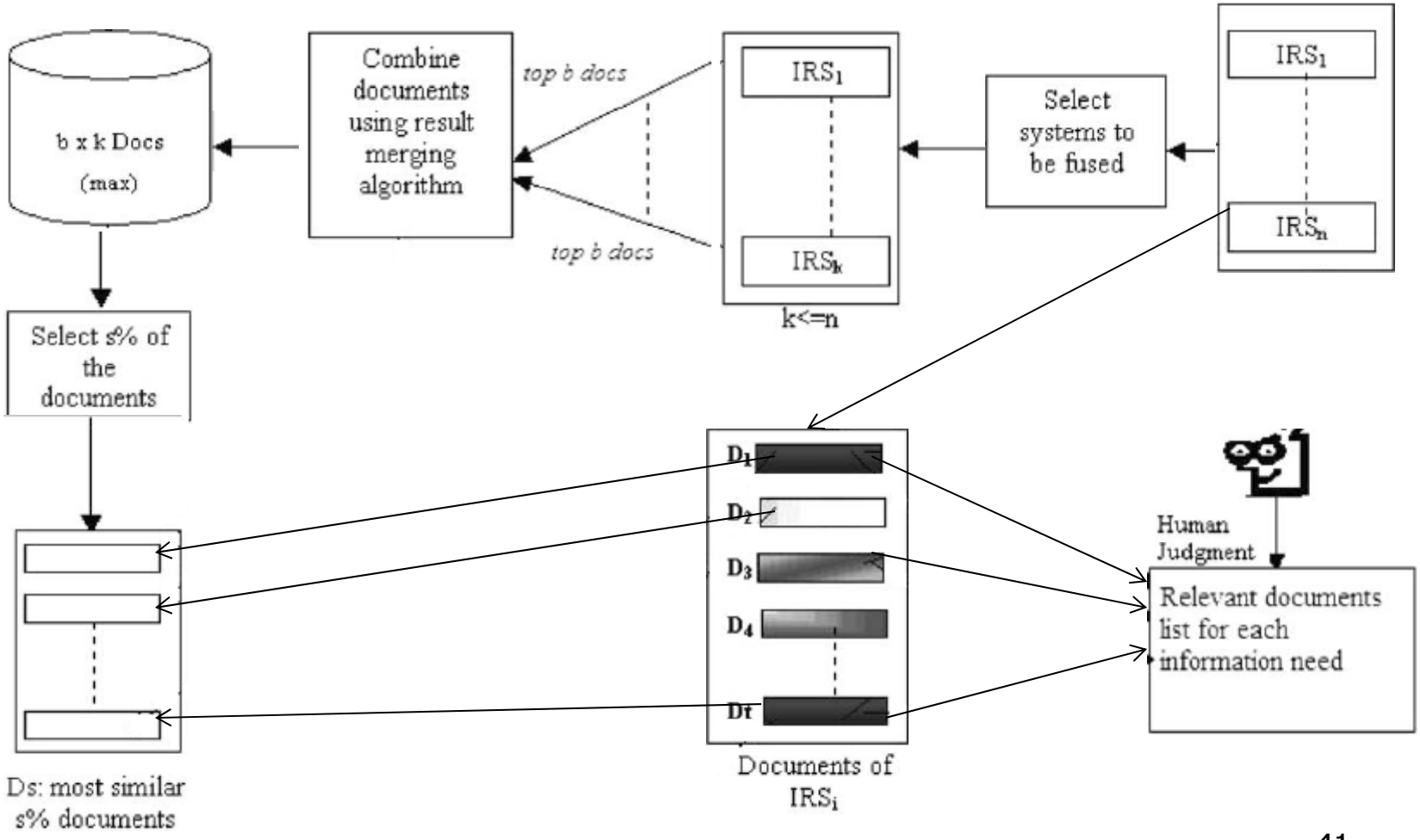
Rank-level Fusion Methods

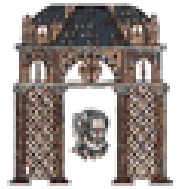
- Some classifiers provide class "scores", or some sort of class probabilities
- This information can be used to "rank" each class
- Pc1=0.10 Rc1=1
- Classifier -> Pc2=0.75 -> Rc2=3
- Pc3=0.15 Rc3=2
- In general if Ω={c1,…ck} is the set of classes, the classifiers can provide an "ordered" (ranked) list of class labels for each outcome

A. We will compare the classifiers (Ranking Classifiers)

B. We will merge (fuse) the ordered lists

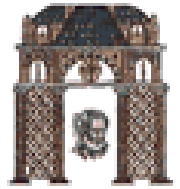
Ranking retrieval systems





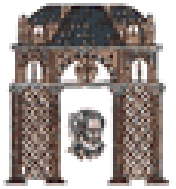
System selection methods

1. Best: certain percentage of top performing systems used
2. Normal: all systems to be ranked are used
3. Bias: certain percentage of systems that behave differently from the norm (majority of all systems) are used



More on bias concept

- A system is defined to be biased if its query responses are different from the norm, i.e., the majority of the documents returned by all systems.
- Biased systems improve data fusion
  - Eliminate ordinary systems from fusion
  - Better discrimination among documents and systems



Calculating bias of a system

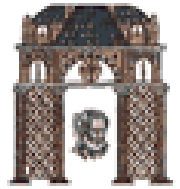
- Similarity value

$$s(v,w) = \frac{\sum v_i \cdot w_i}{\sqrt{\sum (v_i)^2 \cdot \sum (w_i)^2}}$$

v: vector of norm
w: vector of retrieval system

- Bias of a system

$$B(v,w) = 1 - s(v,w)$$



Example of calculating bias

Two classifiers, 7 classes (*a,b,c,d,e,f,g*) and 3 samples (queries) (x_1 , x_2 , x_3)
Each classifier replies with an order list of possible classes and we keep only the first k=4
i.e. Classifier A classifies sample x_2 in the classes with the ordered list (b,a,c,d).

$$A = \begin{bmatrix} a & b & c & d \\ b & a & c & d \\ a & b & c & e \end{bmatrix} \qquad B = \begin{bmatrix} b & f & c & e \\ b & c & f & g \\ c & f & g & e \end{bmatrix}$$

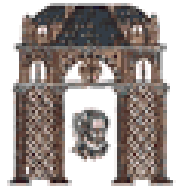
X_A=(3, 3, 3, 2, 1, 0, 0) X_B=(0, 2, 3, 0, 2, 3, 2)

2 classifiers: A and B
7 classes: a, b, c, d, e, f, g
3 samples
i^{th} row is the result for i^{th} query

norm vector → $X = X_A + X_B$ = (3, 5, 6, 2, 3, 3, 2)

$s(X_A,X)=49/[32][96]^{1/2} = 0.8841$
Bias(A)=1-0.8841=0.1159

$s(X_B,X)=47/[30][96]^{1/2} = 0.8758$
Bias(B)=1-0.8758=0.1242



Bias calculation with order

Order is important because users usually just look at the documents of higher rank.

$$A = \begin{bmatrix} a & b & c & d \\ b & a & c & d \\ a & b & c & e \end{bmatrix} \quad B = \begin{bmatrix} b & f & c & e \\ b & c & f & g \\ c & f & g & e \end{bmatrix}$$

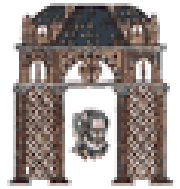
2 classifiers: A and B
7 classes: a, b, c, d, e, f, g
3 samples
i^{th} row is the result for i^{th} query

X_A=(10, 8, 4, 2, 1, 0, 0) X_B=(0, 8, 22/3, 0, 2, 8/3, 7/3)

Increment the frequency count of a document by m/i instead of 1 where m is number of positions and i position of the document.

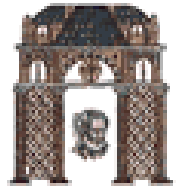
m=4
X_A=(a)=4/1+4/1+4/2=10, X_B=(c)=4/1+4/2+4/3=44/6

X_A=(10, 8, 4, 2, 1, 0, 0); X_B=(0, 8, 22/3, 0, 2, 8/3, 7/3)
Bias(A)=0.0087; Bias(B)=0.1226



Ranking the Lists

- We assume that each Classifier give an ordered list for the classes that a sample belongs.
- We create a new ordered list by combining all the outputs of the classifiers

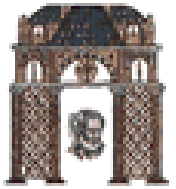


Rank position method

- Merge documents using only rank positions
- Rank score of document i (j: system index)

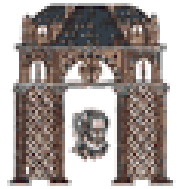
$$r(d_i) = \frac{1}{\sum_j 1/pos(d_{ij})}$$

- If a system j has not ranked document i at all, skip it.



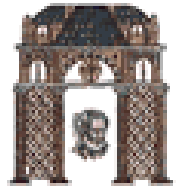
Rank position example

- 4 classifiers: A, B, C, D
  classes: a, b, c, d, e, f, g
- Query results (patterns):
  A={a,b,c,d}, B={a,d,b,e},
  C={c,a,f,e}, D={b,g,e,f}
- r(a)=1/(1+1+1/2)=0.4
  r(b)=1/(1/2+1/3+1)=0.52
- Final ranking of documents:
  (most relev) a > b > c > d > e > f > g *(least relev)*



Borda Count method

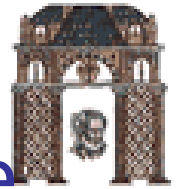
- Based on democratic election strategies.
- The highest ranked class in a classifier gets n Borda points and each subsequent gets one point less where n is the number of total retrieved classes by all classifiers.



The Borda Count Method: an example

- Let N=3 and k=4, Ω={a,b,c,d}
- For a given pattern, the ranked ouptuts of the three classifiers are as follows

Rank value	Classifier1	Classifer2	Classifier3
4	c	a	b
3	b	b	a
2	d	d	c
1	a	c	d



The Borda Count Methods: an example

- So we have

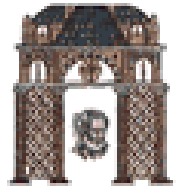
$r_a = r_a^1 + r_a^2 + r_a^3 = 1+4+3=8$

$r_b = r_b^1 + r_b^2 + r_b^3 = 3+3+4=10$

$r_c = r_c^1 + r_c^2 + r_c^3 = 4+1+2=7$

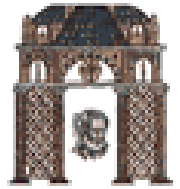
$r_d = r_d^1 + r_d^2 + r_d^3 = 2+2+1=5$

The winner-class is b because it has the maximum overall rank



Pairwise Comparison method (Condorcet method)

- It is also based on election strategy.
  - The winner is the class which beats all other classes in a pair wise comparison
  - In a system while counting votes, a document loses to all other retrieved documents if it is not retrieved by that system
  - *n * n matrix* is used for comparison, where n is the number of classes
  - **Each cell of matrix** compares a ***class(x)*** with ***class(y),*** and it contains three entries
    - Number of ***wins of x with y***,
    - number of ***loses of x with y***,
    - and ***number of ties with y***



Pairwise Comparison example

- **3 Classes**: a, b, c
  5 Classifiers, A, B, C, D, E
- **Query's Result Lists**
- A: {a,b,c}, B:{a,c,b} C, {a,b=c} D:{b,a} E: {c,a,b}

Pairwise comparison

	a	b	c
a	-	4, 1, 0	4, 1, 0
b	1, 4, 0	-	2, 2, 1
c	1, 4, 0	2, 2, 1	-

Pairwise winners

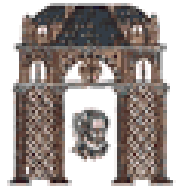
	Win	Lose	Tie
a	2	0	0
b	0	1	1
c	0	1	1

- **Final ranking of pseudo relevance judgments**
  a > b = c

(a,b) = Wins, loses, ties

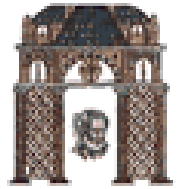
Remarks on Rank level Methods

- Advantage over abstract level (majority vote)
  - Ranking is suitable in problems with many classes, where the correct class may appear often near the top of the list, although not at the top
    - Example: word recognition with sizeablee lexicon
  - Advantages over measurement level:
    - Rankings can be preferred to soft outputs to avoid lack of conssitency when using different classifier
    - Rankins can be preferred to soft outputs to simplify the combiner design
  - Drawbacks:
    - Rank-level method are not supported by clear theorectical underpinnings
    - Results depend on the scale of numbers assigned to the choices



Open issues

- General combination strategies are only sub-optimal solutions to most applications;



References

PART A. Combining Classifiers

1. Dr K Sirlantzis “Diversity in Multiple Classifier Systems”, University of Kent; www.ee.kent.ac.uk;
2. F. Roli, Tutorial Fusion of Multiple Pattern Classifier”, University of Cagliari
3. Robert P.W.Duin, “The Combining Classifier: to Train or Not to Train?”, ICPR 2002, Pattern Recognition Group, Faculty of Applied Sciences;
4. L. Xu, A. Kryzak, C. V. Suen, “Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition”, IEEE Transactions on Systems, Man Cybernet, 22(3), 1992, pp. 418-435.
5. J. Kittle, M. Hatef, R. Duin and J. Matas, “On Combining Classifiers”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(3), March 1998, pp. 226-239.
6. D. Tax, M. Breukelen, R. Duin, J. Kittle, “Combining Multiple Classifiers by Averaging or by Multiplying?”, Patter Recognition, 33(2000), pp. 1475-1485.
7. L. I. Kuncheva, “A Theoretical Study on Six Classifier Fusion Strategies”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(2), 2002, pp. 281-286.
8. *L.Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, John Wiley & Sons, 2004*
9. *S. Theodoridis, K. Koutroubas, “Pattern Recognition”,* Fourth Edition, 2008, Academic Press Chapter 4.20

PART B. Ranking Ordered lists and Data Fusion

1. Automatic Ranking of Retrieval Systems using Data Fusion (Nuray,R & Can,F, IPM 2006)
2. Fusion of Effective Retrieval Strategies in the same Information Retrieval System (Beitzel et.al., JASIST 2004)
3. Learning a Ranking from Pairwise Preferences (Carterette et.al., SIGIR 2006)