AI*IA 2003

Fusion of Multiple Pattern Classifiers

PART III

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):

Abstract-level outputs: each classifier outputs a unique class label for each input pattern

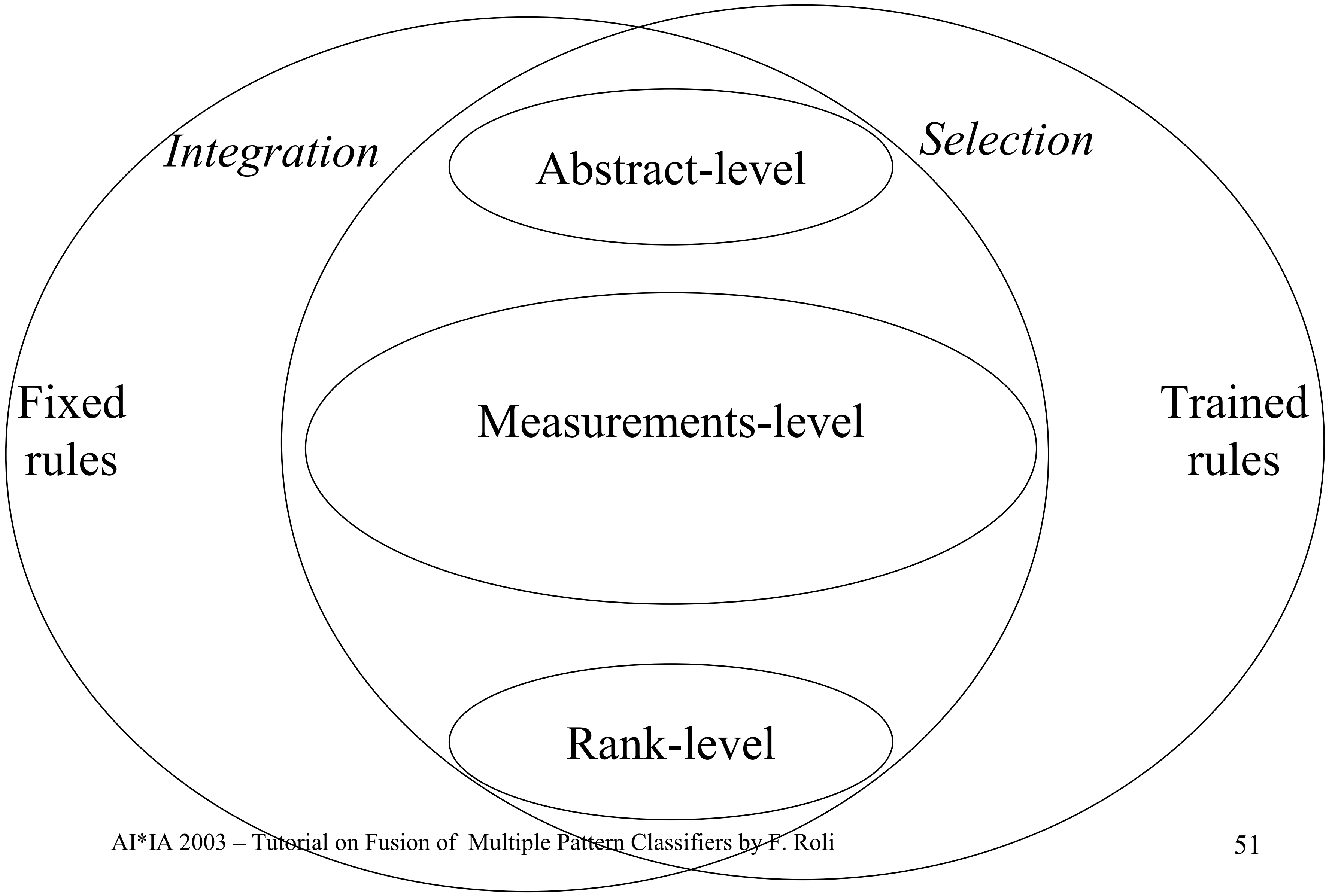
Rank-level outputs: each classifier outputs a list of possible classes, with ranking, for each input pattern

Measurement-level outputs: 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

Methods for fusing multiple classifiers



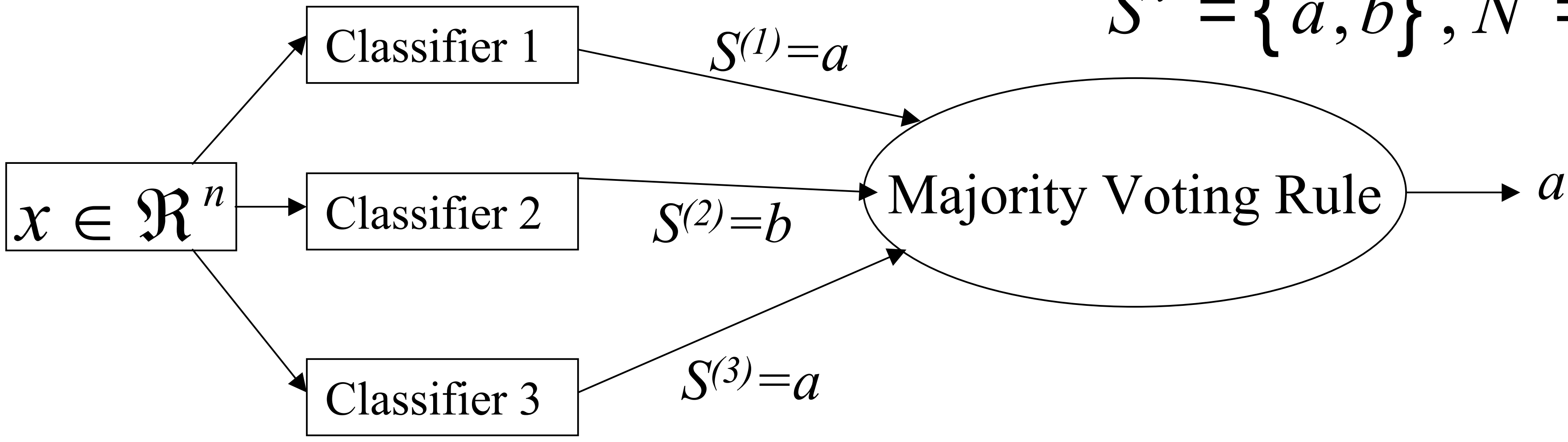
Fixed Rules at the abstract-level

The Majority Voting Rule

Let us consider the *N* abstract ("crisp") classifiers outputs $S^{(1)}, \ldots, S^{(N)}$ associated to the pattern *x*.

Majority Rule: *Class label* $\mathbf{c_i}$ *is assigned to the pattern x if* $\mathbf{c_i}$ *is the most frequent label in the crisp classifiers outputs*.

$$S^i = \{a, b\}, N = 3$$



Majority Vote Rule

Usually *N* is odd.

The frequency of the winner class must be at least

If the ***N*** classifiers make ***independent*** errors and they have the same error probability *e*<0.5, *then* it can be shown that the error ***E*** of the majority voting rule is monotonically decreasing in *N* (Hansen and Salamon, 1990):

$$\lim_{N \to \infty} \sum_{k > N/2}^{N} \binom{N}{k} e^{k} (1 - e)^{N-k} = 0$$

➢Clearly, performances of majority vote quickly decreases for **dependent** classifiers

Majority Vote Rule vs. Classifiers Dependency: An Example

- 3 Classifiers
- Two class task

Classifiers C1 and C3

exhibit error correlations

$$P(1|0,1,0) \gtrless P(0|0,1,0)$$

Classifier	Abstract	Outputs	Majority Class	True Class
C1	**C2**	**C3**		
0	0	0	0	0
0	0	1	0	0
0	1	0	0	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	0
1	1	0	1	1
1	1	1	1	1

Trainable Rules at the abstract-level

Rules based on the Bayes Approach

This kind of trained fusion rules are based on the Bayes approach.

Pattern x is assigned to the class c_i if its posterior probability:

$$P(c_i \mid S^{(1)}, S^{(2)}, \ldots, S^{(N)}) > P(c_j \mid S^{(1)}, S^{(2)}, \ldots, S^{(N)}) \ \forall\, j \neq i$$

$$S^{\,i} \in \left\{ c_{1,} c_2, \ldots, c_k \right\}$$

Fusion rules based on the Bayes Formula try to estimate, by an ***independent*** validation set, these *posterior probabilities*

➢ This fusion rule coincides with the multinomial statistical classifier, that is, the optimal statistical decision rule for discrete-valued feature vectors (Raudys and Roli, 2003)

Behaviour Knowledge Space (BKS)

•In the BKS method, every possible combination of abstract-level classifiers outputs is regarded as a cell in a look-up table.

•Each cell contains the number of samples of the validation set characterized by a particular value of class labels.

•Reject option by a threshold is used to limit error due to "ambiguous" cells

Classifiers outputs $S^{(1)}$, $S^{(2)}$, $S^{(3)}$

Class	0,0,0	0,0,1	**0,1,0**	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
0	100	50	**76**	89	54	78	87	5
1	8	88	**17**	95	20	90	95	100

$$P(c=0 \mid S^{(1)}=0, S^{(2)}=1, S^{(3)}=0) = \frac{76}{76+17} = 0.82 \geq th$$

BKS Small-Sample Size Drawback

- If k is the number of classes and N is the number of the combined classifiers, BKS requires to estimate $\boldsymbol{k^N}$ posterior probabilities.

- K and N are two critical parameter for the BKS rule, because *the number of posterior probabilities to estimate increases very quickly*.
- If this number is too high, we can have serious problems, because the number of training sample is often small.

$\Downarrow$

BKS rule suffers a lot from the small sample size problem

BKS Improvements

•In order to avoid the small sample size problem:

-we can try to reduce the number of the parameters to estimate (Xu et al., 1992 ; Kang, and Lee, 1999).

For example, under the assumption of classifier independence given the class (Xu et al., 1992):

$$P(S^{(1)},...,S^{(N)} \mid c_i) = \prod_j P(S^{(j)} \mid c_i) \propto \prod_j P(c_i \mid S^{(j)}) \cdot P(S^{(j)})$$

$$\boxed{bel(i) = P(c_i \mid S^{(1)},...,S^{(N)}) = \eta \prod_j P(c_i \mid S^{(j)})}$$

-We can use noise injection to increase sample size of training set (Roli, Raudys, Marcialis, 2002).

BKS Improvements

•If the number of classifiers is small, BKS cells can become "large" and contain vectors of different classes, that is, *ambiguous* cells can exist.

Classifiers outputs $S^{(1)}$, $S^{(2)}$, $S^{(3)}$

Class	0,0,0	0,0,1	**0,1,0**	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
0	100	50	**51**	89	54	78	87	5
1	8	88	**50**	95	20	90	95	100

Raudys and Roli (2003) proposed a method to address this issue

Remarks on Abstract Level Fusers

- Abstract level methods are the most general fusion rules

•They can be applied to any ensemble of classifiers, even to classifiers of different types

•The majority voting rule is the simplest combining method

–This allows theoretical analyses (Lam and Suen, 1997)

•When prior performance is not considered, the requirements of time and memory are negligible

•As we proceed from simple rules to adaptive (weighted voting) and trained (BKS, Bayesian rule) the demands on time and memory quickly increase

•Trained rules impose heavy demands on the quality and size of data set

Rank-level Fusion Methods

Some classifiers provide class “scores”, or some sort of class probabilities

This information can be used to “*rank*” each class.

$$\boxed{\text{Classifier}} \longrightarrow \begin{pmatrix} p_{c_1} = 0.10 \\ p_{c_2} = 0.75 \\ p_{c_3} = 0.15 \end{pmatrix} \longrightarrow \begin{pmatrix} r_{c_1} = 1 \\ r_{c_2} = 3 \\ r_{c_3} = 2 \end{pmatrix}$$

In general, if $\Omega = \{c_1, \ldots, c_k\}$ is the set of classes, these classifiers can provide an “ordered” (ranked) list of class labels.

The Borda Count Method: an example

Let N=3 and k=4. $\Omega = \{ a, b, c, d \}$.

For a given pattern, the ranked outputs of the three classifiers are as follows:

Rank value	**Classifier 1**	**Classifier 2**	**Classifier 3**
4	*c*	*a*	*b*
3	*b*	*b*	*a*
2	*d*	*d*	*c*
1	*a*	*c*	*d*

The Borda Count Method: an example

So, we have:

$$r_a = r_a^{(1)} + r_a^{(2)} + r_a^{(3)} = 1 + 4 + 3 = 8$$

$$r_b = r_b^{(1)} + r_b^{(2)} + r_b^{(3)} = 3 + 3 + 4 = 10$$

$$r_a = r_c^{(1)} + r_c^{(2)} + r_c^{(3)} = 4 + 1 + 2 = 7$$

$$r_a = r_d^{(1)} + r_d^{(2)} + r_d^{(3)} = 2 + 2 + 1 = 5$$

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

Remarks on Rank level Methods

- Advantages 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 sizeable lexicon

•Advantages over measurement level:

–rankings can be preferred to soft outputs to avoid lack of consistency when using different classifiers

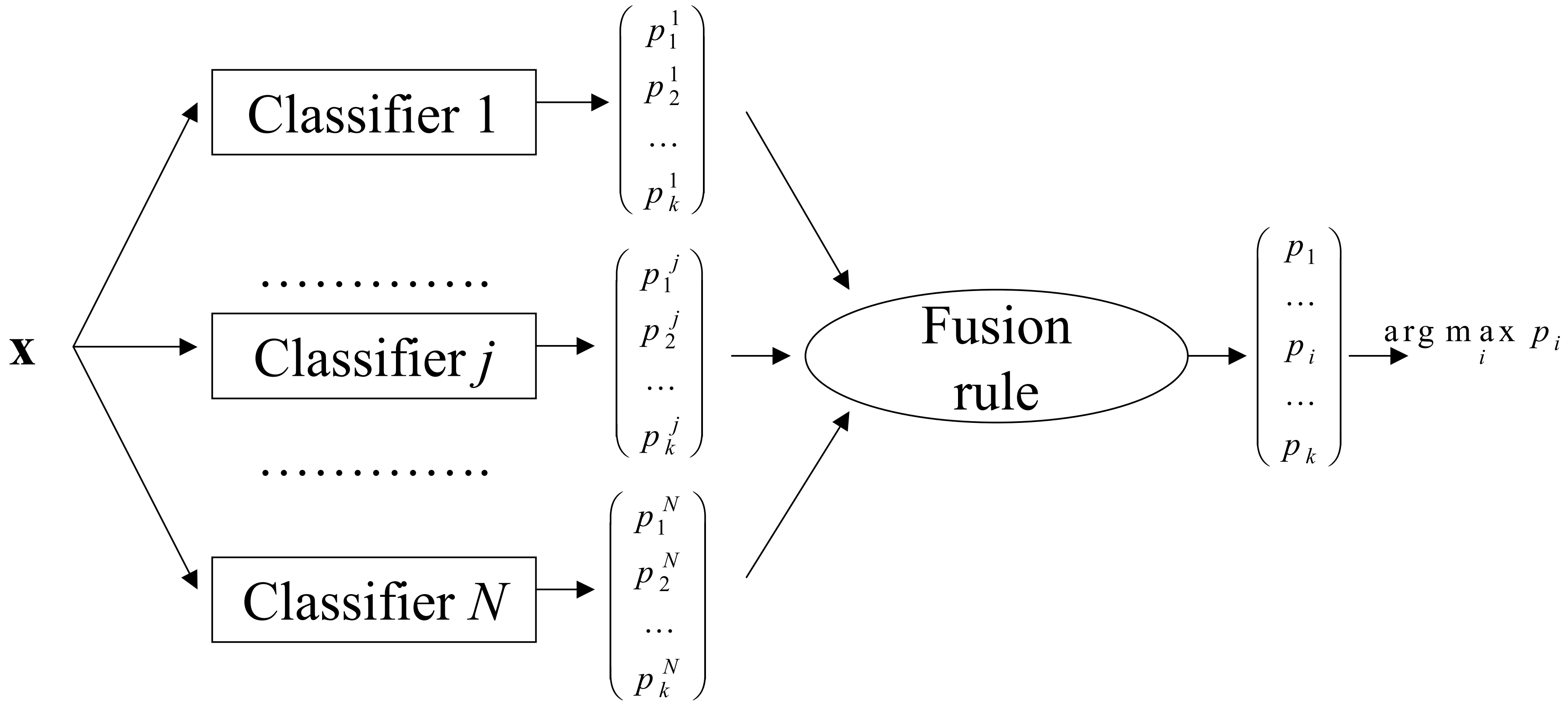
–rankings can be preferred to soft outputs to simplify the combiner design

•Drawbacks:

–Rank-level methods are not supported by clear theoretical underpinnings

–Results depend on the scale of numbers assigned to the choices

Measurement-level Fusion Methods



x
Classifier 1
Classifier j
Classifier N
....
....
$p_1^1, p_2^1, \dots, p_k^1$
$p_1^j, p_2^j, \dots, p_k^j$
$p_1^N, p_2^N, \dots, p_k^N$
Fusion rule
$p_1, \dots, p_i, \dots, p_k$
$\arg\max_i p_i$

➢Normalization of classifiers outputs is not a trivial task when combining classifiers with different output ranges and different output types (e.g., distances vs. membership values).

Linear Combiners

$$p_i^{ave}(x) = \sum_{k=1}^{N} w_k \, p_i^k(x)$$

•Simple and Weighted averaging of classifiers' outputs

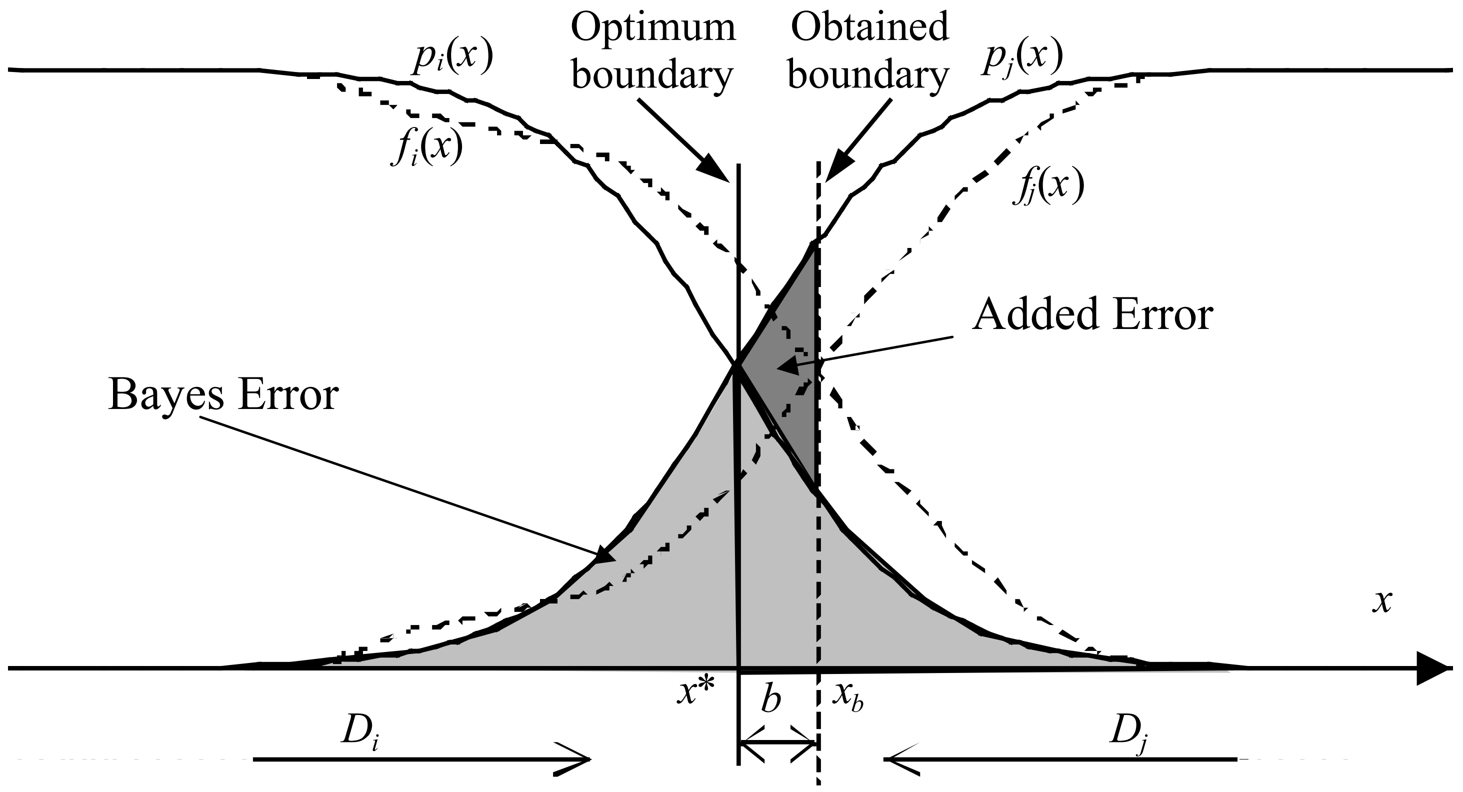
•Simple average is the optimal fuser for classifiers with the same accuracy and the same pair-wise correlations (Roli and Fumera)

•Weighted average is required for *imbalanced* classifiers, that is, classifiers with different accuracy and/or different pair-wise correlations (Roli and Fumera)

•Improvement of weighted average over simple average has been investigated theoretically and by experiments (Roli and Fumera)

Bias/Variance Analysis in Linear Combiners

- The "added" error over the Bayes one can be reduced by linear combinations of classifiers outputs (Tumer and Ghosh; Roli and Fumera)



Bias/Variance Analysis in Linear Combiners

Correlated and Unbiased Classifiers

•Added error of the simple average of *N* classifiers (Tumer and Ghosh):

$$E_{add}^{ave} = E_{add}\left(\frac{1+(N-1)\delta}{N}\right)$$

•The reduction of variance component of the added error achieved by simple averaging depends on the correlation factor δ between the estimation errors

➢Negatively correlated estimation errors allow to achieve a greater improvement than independent errors

Bias/Variance Analysis in Linear Combiners

Correlated and Biased Classifiers

•Added error of the simple average of *N* classifiers (Tumer and Ghosh):

$$E_{add}^{ave} = \frac{1}{s}\sigma^2\left(\frac{1+(N-1)\delta}{N}\right)+\frac{1}{2s}\left(\bar{\beta}_i - \bar{\beta}_j\right)^2$$

➢*Simple averaging is effective for reducing the variance component, but not for the bias component*

➢*So, individual classifiers with low biases should be preferred*

Analysis of bias/variance for weighted average can be found in (Fumera and Roli)

Product and Order Statistics Fusers

$$p_i = P^{-(N-1)}(\omega_i)\prod_{j=1}^{N} p_i^j, \quad i = 1,\ldots,k$$

–Product is obviously sensible to classifiers outputting probability estimates close to zero ("overconfident" classifiers)

Fusers based on order statistics operators are *max*, *min*, and *med*:

$$p_i^{\max}(\mathbf{x}) = p_i^{N:N}(\mathbf{x}), \quad i = 1,\ldots,k$$

$$p_i^{\min}(\mathbf{x}) = p_i^{1:N}(\mathbf{x}), \quad i = 1,\ldots,k$$

$$p_i^{\text{med}}(\mathbf{x}) = \begin{cases} \dfrac{p_i^{\frac{N}{2}:N}(\mathbf{x}) + p_i^{\frac{N+1}{2}:N}(\mathbf{x})}{2} & \text{if } N \text{ is even} \\ p_i^{\frac{N+1}{2}:N}(\mathbf{x}) & \text{if } N \text{ is odd} \end{cases}, \quad i = 1,\ldots,k$$

A Theoretical Framework

- A theoretical framework for the *product*, *min*, *med* and *max* fusers was established by J. Kittler et al. (1998)

•These rules can be formally derived assuming that the *N* individual classifiers use ***distinct*** feature vectors

$$p^j(\omega_i / X_1, X_2, \ldots\ldots, X_N)$$

–the product and min rules are derived under the hypothesis that classifiers are ***conditionally statistically independent***

–sum, max, median rules are derived under the further hypothesis that the ***a posteriori probabilities*** estimated by the classifiers ***do not deviate significantly from the class prior probabilities***

Fusers with Weights

A natural extension of many fixed rules is the introduction of weights to the outputs of classifiers (weighted average, weighted majority vote)

A simple criterion is to introduce weights proportional to the accuracy of each individual classifier

Weights related to classes can also be computed by extracting them from the confusion matrix of each classifier

- Methods for robust weights estimation are a matter of on-going research

“Stacked” Fusion

The k soft outputs of the N individual classifiers, $p_i^j(\mathbf{x})$, $i=1,\ldots,k$, $j=1,\ldots,N$, can be considered as features of a new classification problem (*classifiers output* feature space)

Classifiers can be regarded as feature extractors !

Another classifier can be used as fuser: this is the so-called “stacked” approach (D.H. Wolpert, 1992), or “meta-classification” (Giacinto and Roli, 1997), “brute-force” approach (L.I. Kuncheva, 2000)

•To train the metaclassifier, the outputs of the N individual classifiers on an **independent** validation set must be used

➢Experts’ Boasting Issue ! (Raudys, 2003)

➢An **independent** validation set is required

Pros and Cons of the "Stacked" approach

Pros:

•the meta-classifier can work in a "enriched" feature space

•No classifiers dependency model is assumed

Cons:

•The dimensionality of the output space increases very fast with the number of classes and classifiers.

•The meta-classifier should be trained with a data set different from the one used for the individual classifiers (Experts' Boasting Issue)

•***The space of classifiers outputs might not be well-behaved***

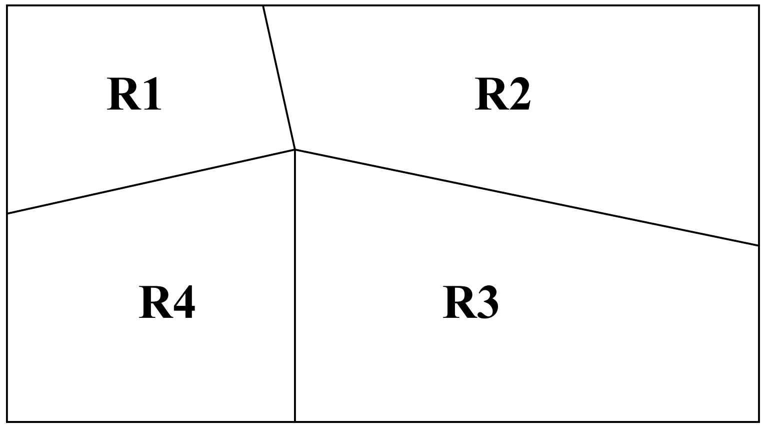
Classifier Selection

Goal: for each input pattern, select the classifier, if any, able to correctly classify it

Problem formulation

Two dimensional feature space

Partitioned in 4 regions



- Easy to see that selection outperforms individual classifiers if I am able to select the best classifier for each region
- Two critical issues: i)definition of regions; ii)selection algorithm

Classifier Selection

Two main approaches to design a classifier selection system:

Static vs. **Dynamic** Selection

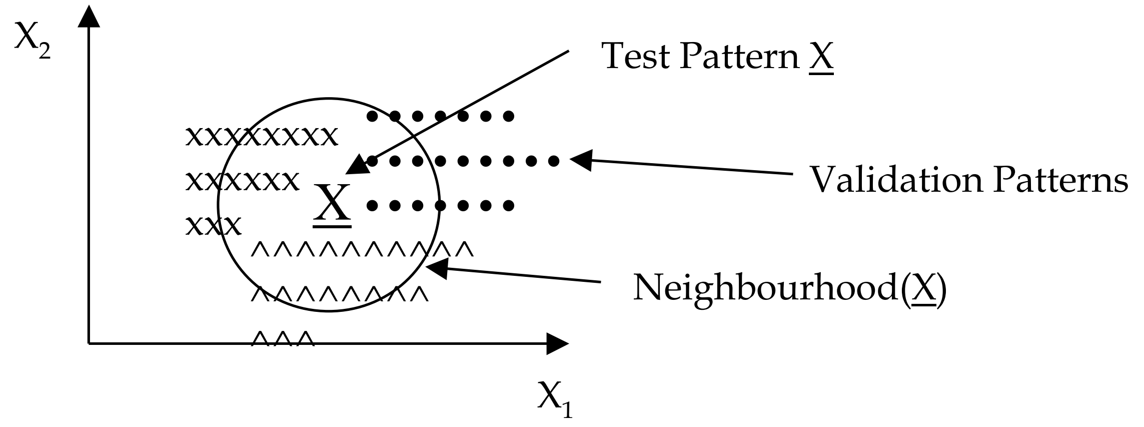
Static Selection

- Regions are defined before classification.
- For each region, a responsible classifier is identified
  - Regions can be defined in different ways:
    - Histogram method: Space partition in "bins"
    - Clustering algorithms

Classifier Selection

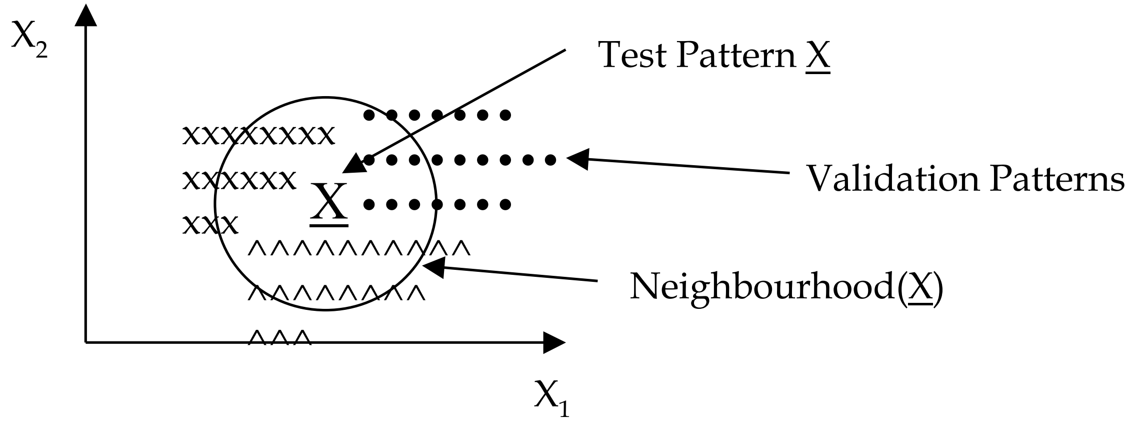
Dynamic Selection

SELECTION CONDITIONS based on estimates of classifiers accuracies in local regions of feature space surrounding an unknown test pattern X (Neighbourhood(X))



See works of Woods et al.; Giacinto and Roli

Selection Condition: An Example



➢Woods et al. (1997) simply computed classifier local accuracy as the percentage of correctly classified patterns in the neighbourhood

➢Giacinto and Roli (1997, etc) proposed some probabilistic measures:

$$\hat{P}(correct_j|\mathbf{X}^*) = \frac{\sum_{n=1}^{N} \hat{P}^{(j)}(\omega_i \mid \mathbf{X}_n \in \omega_i) \cdot W_n}{\sum_{n=1}^{N} W_n}$$

Some remarks on classifier selection

•Theoretically, “local” fusion rules can outperform “global” ones

•Selection can effectively handle correlated classifiers

•In practice, large and representative data sets are necessary to design a good selection system

•Further work is necessary to develop “robust” methods for estimating classifier local accuracy

➢Works on Adaptive Mixtures of Local Experts (M. Jordan, et al.), not discussed for the sake of brevity, can be regarded as methods for dynamic classifier selection

Final Remarks on Fixed vs. Trained Fusers

- Fixed rules
  - –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

➢*Heavy demands on the quality and size of the training set*

MCS DESIGN

Parts 2 and 3 showed that designer of MCS has a toolbox containing a large number of instruments for generating and fusing classifiers.

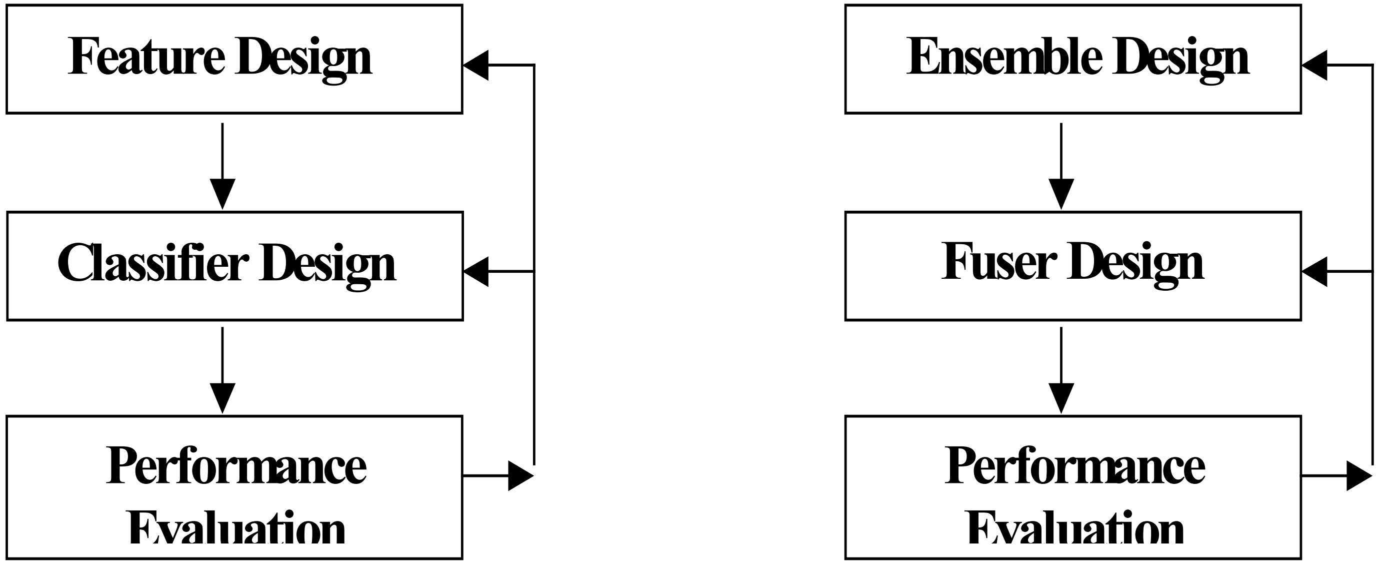
Two main design approaches have been defined so far

Coverage optimisation methods

Decision optimisation methods

- However, combinations of the two above methods and hybrid, ad hoc, methods are often used

MCS DESIGN



Coverage optimisation methods: a simple fuser is given without any design. The goal is to create a set of complementary classifiers that can be fused optimally

-Bagging, Random Subspace, etc.

Decision optimisation methods: a set of carefully designed and optimised classifiers is given and unchangeable, the goal is to optimise the fuser

MCS DESIGN

- Decision optimisation method to MCS design is often used when previously carefully designed classifiers are available, or valid problem and designer knowledge is available

- Coverage optimisation method makes sense when creating carefully designed, “strong”, classifiers is difficult, or time consuming

•Integration of the two basic approaches is often used

However, no design method guarantees to obtain the “optimal” ensemble for a given fuser or a given application (Roli and Giacinto, 2002)

- The best MCS can only be determined by performance evaluation.