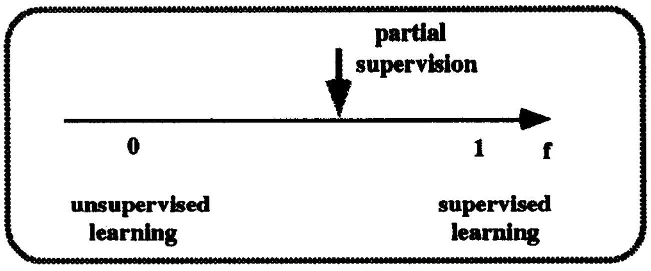
Implicitly - supervised fuzzy pattern recognition

[1]Kaoru Hirota [2]Witold Pedrycz

[1] Department of Control Systems Engineering, Hosei University,
Koganei-city, Tokyo 184 Japan
[2] Department of Electrical and Computer Engineering, University of Manitoba
Winnipeg, Canada R3T 2N2
pedrycz@eeserv.ee.umanitoba.ca

1. Introduction

The classic taxonomy of supervised and supervised learning commonly encountered in pattern recognition [1] [3] is a useful yet somewhat simplified categorization of many real-world classification problems. With the progress in applications of fuzzy sets techniques in this realm, some conceptually new domains have emerged. In this setting the issue of a fundamental dichotomy of supervised versus unsupervised learning need to be addressed as well. The examples of unsupervised learning viewed as fuzzy clustering cf. [1] [2] are usually contrasted with the notions of supervised learning (fuzzy classifiers) cf. [4]. The supervised learning is based on a training set with each pattern being labelled. For the unsupervised learning, this class assignment is not available. Those two are very distinct modes of learning. More realistically, there could also be many intermediate scenarios in which the training set is composed of a relatively small group of the labelled patterns and a vast majority of the unlabelled objects. This mixture of the patterns calls for the algorithms of the fuzzy clustering carried under partial supervision, the idea introduced in [5]. Depending upon the sizes of the respective populations of the labelled and unlabelled patterns the effect of partial supervision can be diminished or enhanced as shown in the diagram below (here "f" denotes a fraction of the labelled patterns occurring within the entire population). In an extreme situation, that occurs for f=0 or f=1,we come up with the classic unsupervised or supervised models of learning.



In this study we are interested in another aspect of partially supervised learning in which the information about classes is given in an implicit format. A convincing example arises in the realm of referential classification where the training set includes the patterns arranged in pairs along with their similarity levels (e.g., patterns x and y are λ - *similar* with $\lambda \in [0, 1]$ standing for this degree of similarity). This classification outcome is definitely less detailed than that achieved as a complete vector of membership values. The problem exhibiting this format of information about the classes will be referred to as an implicit (or implicitly-supervised) classification. Even though these two schemes fall under the same category of the supervised learning, the character of the available classification information is very distinct.

2. Problem formulation

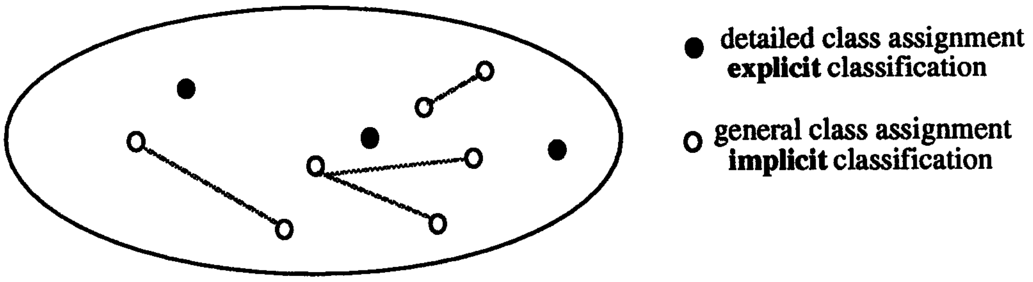
This section provides with a formal statement of the problem. Let the feature space be formed by n-dimensional vectors of the unit hypercube, say $x \in [0, 1]^n$. In its explicit form the learning set includes the classification results that are provided as the membership vectors of an m-dimensional classification space, $\omega \in [0, 1]^m$; each coordinate of ω is interpreted as a degree of membership in the corresponding class, $\omega_i \in [0, 1]$. The learning set, called $\mathfrak{D}$ - training set, appears as a collection of the feature - class assignment pairs,

$$(x_1, \omega_1) \quad (x_2, \omega_2) , \dots , \quad (x_M, \omega_M)$$

On the other hand, the implicit format of the available classification results implies that the m-dimensional classification space is usually reduced to a unit interval the values of whose represent the referential characteristics of the patterns. For the matching (similarity) property we are concerned with the pairs of the patterns x_k, x_k' and their associated similarity levels; altogether these constitute a so-called $\mathfrak{R}$ - training set,

$$\left((x_1, x_1'), \mathrm{sim}_1\right) \quad \left((x_2, x_2'), \mathrm{sim}_2\right) \quad \dots \quad \left((x_N, x_N'), \mathrm{sim}_N\right)$$

where $\mathrm{sim}_i \in [0, 1]$ describes a similarity between the corresponding pairs of the patterns. Essentially, each sim_i could be regarded as an aggregation of the class membership ω_i and ω_i', namely $\psi(\omega_i , \omega_i')$, with $\psi : [0, 1]^m \rightarrow [0, 1]$ being treated as the referential transformation. Graphically, one can portray the results of the explicit and implicit classification as shown in Fig.1.



● detailed class assignment **explicit** classification
o general class assignment **implicit** classification

Fig.1. Explicit and implicit class assignment

One can also assume (that is intuitively appealing) that $M << N$ as the assignment of the detailed class membership values is far more demanding than the scalar appointment done at the higher level of some general properties of the classes (such as e.g., similarity, difference, etc.). The same family of the training examples may also include instances coming from the $\mathfrak{D}$- as well as $\mathfrak{R}$ - training set.

3. The General Architecture

The overall architecture of the classifier, Fig.2, consists of the two main functional blocks. The first one is responsible for the direct classification ($\mathfrak{D}$ - training set). It is followed by the module of the referential classification that is developed using the $\mathfrak{R}$ - training set.

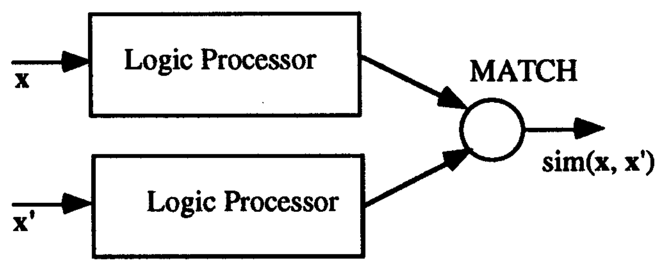


Fig.2. General architecture of a classifier

The direct classification is realized through a logic processor (LP) that constitute a fuzzy neural network with a single hidden layer, cf. [6][7], Fig.3.

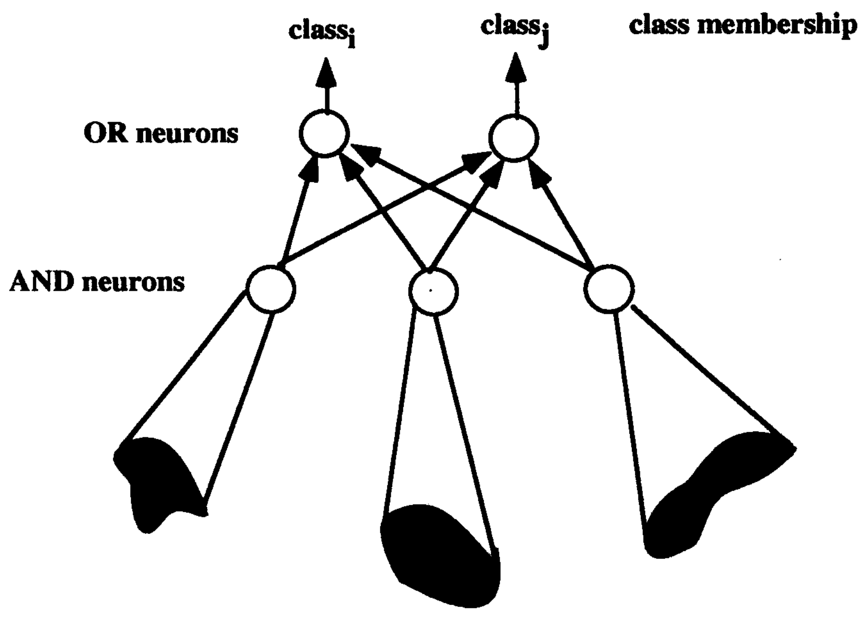


Fig.3. Topology of a Logic Processor

The hidden layer includes the AND neurons whose role is to develop combinations (conjunctions) of the features x_1, x_2 ... x_n deemed necessary in the classification problem. The regions covering the patterns belonging to the same class and produced by the hidden layer are aggregated disjunctively (ORwise) at the output layer.

The referential part of the classifier treats the vectors of class membership as its inputs and returns the values of their referential computations. The simplest version of this module can be envisioned as a single m-input MATCH neuron as in Fig.2. Its extended version is just a referential (matching) processor, cf. [6], where the levels of matching among the classes are additionally processed before generating the overall similarity level between the classes.

4. The Design of the Classifier

Depending whether we deal with the patterns from the $\mathfrak{D}$ or $\mathfrak{R}$ - training family, different modules of the classifier are developed. The fully labelled patterns are used to train the logic processor. The formulation of the problem is fairly standard: given is a learning set $(x_1, \omega_1), (x_2, \omega_2), \ldots, (x_M, \omega_M)$ Construct the logic processor (including both its architecture and the connections) minimizing the given performance index Q,

$$\min_{LP} Q$$

where

$$Q = \sum_{k=1}^{M} (\omega_k - LP(x_k, \textbf{connections}))^T (\omega_k - LP(x_k, \textbf{connections}))$$

The structural learning usually involves additions and/or removals of the individual processing units (units). The parametric learning is driven by the gradient of the above performance index, $-\frac{\partial Q}{\partial \textbf{connections}}$.
It is important to note here that the cardinality of the labelled patterns ($\mathfrak{B}$ - training set) is usually very low in comparison to the overall training set. Therefore one should not be concerned too much about a plain memorization carried out by the processor; let us stress that the primary objective is to minimize Q even at an expense of a significant expansion of the logic processor. While at this phase the vast portion of the learning activities take place, some fine tuning can be done later in conjunction with the learning the referential part of the classifier. For the referential phase of learning (viz. the learning exploiting the elements in the $\mathfrak{R}$ training set), the two identical copies of the previously constructed logic processor are used in the configuration shown in Fig.4. The training set used there comprises of the triples $((x_k, x'_k), sim_k)$, k = 1, 2, ... N, constituting the dominant part of the entire training set, M << N, while sim_k denotes a degree of similarity reported for x_k and x'_k (generally speaking, the discussed learning scheme works well for any other referential operation). The patterns x_k as well as x'_k are propagated through the logic processors and produce two vectors of class assignment, say ω_k and ω'_k. These are presented at the inputs of the matching neuron. The new training set comes in the form of the ordered triples $(\omega_k, \omega'_k, sim_k)$, k = 1, 2, ..., N. The parameters of the MATCH neuron are updated based on a squared error between sim_k and the output of the matching neuron MATCH $(\omega_k; \omega'_k, w)$. The performance index Q is defined as the sum taken over the $\mathfrak{R}$-training set,

$$Q = \sum_{k=1}^{N} \left(sim_k - \text{MATCH}(\omega_k; \omega'_k, w)\right)^2$$

The gradient-driven scheme of updating is obvious,

$$w(\text{new}) = w - \xi \frac{\partial Q}{\partial w}$$

$\xi \in [0,1]$. In an on-line learning mode the modifications of w occur after presenting an individual input-output triple of data, (ω, ω', sim) (the subscript "k", as irrelevant in this scheme, has been left out)

$$\frac{\partial Q}{\partial w} = -2\left(sim - \text{MATCH}(\omega, \omega', w)\right) \frac{\partial \text{MATCH}(\omega, \omega', w)}{\partial w}$$

Its scalar notation leads to the expression,

$$\frac{\partial\, \mathrm{MATCH}(\omega, \omega', w)}{\partial w_i} = \frac{\partial Q}{\partial w_i} \left\{ \mathop{\mathrm{T}}_{j=1}^{m} \left((\omega_j \equiv \omega_j') \, s w_j \right) \right\} = \frac{\partial Q}{\partial w_i} \left\{ A t \left[(\omega_i \equiv \omega_i') \, s w_i \right] \right\}$$

where T and s denote some t- and s-norms, respectively, and ≡ stands for the equality index, cf. [6]. Furthermore,

$$A = \mathop{\mathrm{T}}_{j \neq 1}^{m} \left[(\omega_j \equiv \omega_j') \, s w_j \right] .$$

Once the learning of the referential part is over but the value of the performance index is still not acceptable, two conceptually distinct remedial steps could be sought:

- an expansion of the referential module of the classifier by replacing the single MATCH neuron by the referential processor [6] [7].
- an incremental retraining the logic processor already constructed with the aid of the $\mathfrak{D}$ - training set. A special caution should be exercised, though, as this part of the classifier has been already trained and too radical modifications of its connections done at this stage could easily wash away the previous structure (due to the relatively small size of the $\mathfrak{D}$ - training set). Hence the learning should be vigilantly monitored. The corresponding learning schemes are shown in Fig.4.

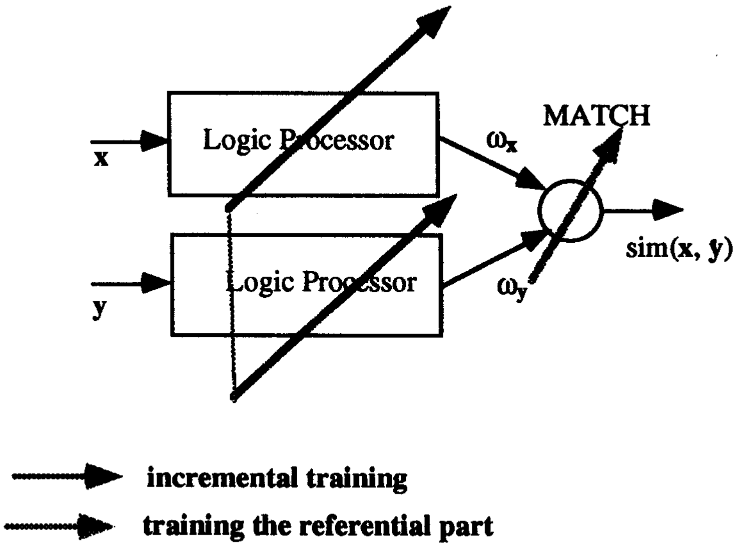


Fig.4. Learning modes in the classification scheme

The connections of these two logic processors are updated simultaneously so that they are always maintained equal. Let **x**, **y**, and sim be provided. In view of this learning policy, the derivative of Q reads as

$$\frac{\partial Q}{\partial\ \text{connection}} = \frac{\partial Q}{\partial \omega_x} \frac{\partial \omega_x}{\partial\ \text{connection}} + \frac{\partial Q}{\partial \omega_y} \frac{\partial \omega_y}{\partial\ \text{connection}}$$

(note that we have started with the two identical copies of the logic processor).

5. Conclusions

We have introduced the new class of pattern classifiers operating under explicit-implicit supervision. The architecture of the classifier along with the learning policies are developed in a way that allows us to operate under a variety of combinations of implicitly and explicitly labelled patterns.

The usefulness of the discussed classifier extends far beyond the domain of pattern recognition; one can easily formulate problems of multi-objective decision-making into the way they fit the studied referential framework.

Acknowledgements

Support from the Natural Sciences and Engineering Research Council of Canada and MICRONET is fully acknowledged.

6. References

1. J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, 1981, N. York.
2. J.C.Bezdek, S.K.Pal, Fuzzy Models for Pattern Recognition, IEEE Press, 1992, N. York.
3. R. Duda, P. Hart, Pattern Classification and Scene Analysis, J.Wiley, 1973, N. York.
4. J.M. Keller, D.J. Hunt, Incorporating fuzzy membership functions into the perceptron algorithm, IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-7, 1985, 693-699.
5. W.Pedrycz, Algorithms of fuzzy clustering with partial supervision, Pattern Recognition Letters, 3, 1985, 13-20.
6. W. Pedrycz, Fuzzy Control and Fuzzy Systems, 2nd edition, Research Studies Press/J.Wiley, 1993, Taunton, N. York.
7. W.Pedrycz, A. F. Rocha, Fuzzy-set based models of neurons and knowledge-based networks, IEEE Trans. on Fuzzy Systems, 1, 1993, 254-266.