Fuzzy Lattice Reasoning (FLR) Classification Using Similarity Measures

Al Cripps[1] and Nghiep Nguyen[2]

[1] Middle Tennessee State University, Murfreesboro, TN 37132, USA
Dept. of Computer Science
acripps@mtsu.edu
[2] Dept. of Economics and Finance
nguyen@mtsu.edu

Summary. In this work, we show that the underlying inclusion measure used by fuzzy lattice reasoning (FLR) classifiers can be extended to various similarity and distance measures often used in cluster analysis. We show that for the cosine similarity measures, we can weigh the contribution of each attribute found in the data set. Furthermore, we show that evolutionary algorithms such as genetic algorithms, tabu search, particle swarm optimization, and differential evolution can be used to weigh the importance of each attribute and that this weighting can provide additional improvements over simply using the similarity measure. We present experimental evidence that the proposed techniques imply significant improvements.

13.1 Introduction

This work provides a framework to extend fuzzy lattice reasoning (FLR) classifiers [13] to use similarity and distance measures. Furthermore we demonstrate experimentally the effectiveness of using similarity measures and evolutionary algorithms to improve fuzzy lattice classifiers in the Cleveland heart benchmark classification problem.

Stemming from the adaptive resonance theory (ART) neural networks [2, 10], fuzzy lattice reasoning bases both its learning and generalization on the computation of hyperboxes in space R^N. Note that other classifiers including ART, Min-Max neural networks and variations [2, 8, 10] compute hyperboxes in R^N. Nevertheless, a unique advantage of FLR is its applicability in a lattice data domain including the product $\mathsf{L} = \mathsf{L}_1 \times \ldots \times \mathsf{L}_N$ of N *constituent lattices* $\mathsf{L}_1, \ldots, \mathsf{L}_N$ [14].

A practical advantage of a lattice applicability is the capacity to deal with disparate types of data, e.g. vectors of numbers, fuzzy sets, symbols, graphs, etc. in applications [13, 14, 17, 18]. Note that with the proliferation of information technologies, the latter capacity may be useful in dealing as well with non-numeric data. Learning and generalization are effected in a lattice L

by computing lattice L intervals. Apart from the simplicity of the method, this work also shows that the computation of lattice intervals can imply significant improvements in classification problems.

The layout of this paper is as follows. Section 13.2 provides the fuzzy lattice classifier theoretic background. Section 13.3 explains principles of similarity and distance measures while Section 13.4 discusses evolutionary algorithms. Section 13.5 presents our theoretical classification improvements for the classifiers found in [6, 7, 14]. Section 13.6 investigates the characteristics of the weighted cosine similarity measure. Section 13.7 provides empirical results that demonstrate the improvement of our enhanced FLR classifiers when compared to other classification methods for the Cleveland heart data. Finally, section 13.8 summarizes the contribution of this work and delineates future work.

13.2 Fuzzy Lattice Theoretic Background

In this paper we employ the lattice theoretic notation introduced in [11, 13, 14]. More specifically, let L denote a mathematical lattice, then $\tau(\mathsf{L})$ denotes the set of lattice L intervals. For this paper, we deal exclusively with a complete lattice L, where O and I denote, respectively, the least and greatest elements in L. It follows that $\tau(\mathsf{L})$ is, also, a complete lattice [14]. Furthermore, $\alpha(\mathsf{L})$ denotes the set of *atoms* in a lattice L, the latter set $\alpha(\mathsf{L})$ includes all trivial intervals (singletons); hence $\alpha(\mathsf{L}) \subset \tau(\mathsf{L})$.

Note that the conventional space R^N of N-dimensional vectors is a product-lattice of N identical, totally-ordered lattices R. A trivial interval in R^N includes the set of N-dimensional points. Moreover, the unit N-dimensional hypercube I^N is a complete lattice, where I equals the closed interval $\mathsf{I} = [0, 1]$.

An *inclusion measure* is a map $\sigma : \mathsf{L} \times \mathsf{L} \rightarrow [0, 1]$, which satisfies the following conditions for $u, w, x \in \mathsf{L}$.

(C0) $\sigma(x, O) = 0, x \neq O$,

(C1) $\sigma(x, x) = 1$,

(C2) $w \wedge u < w \Rightarrow \sigma(w, u) < 1$, and

(C3) $u \leq w \Rightarrow \sigma(x, u) \leq \sigma(x, w)$ - Consistency Property

We remark that $\sigma(x, u)$ denotes the degree of inclusion of lattice element x to lattice element u; hence symbols $\sigma(x, u)$ and $\sigma(x \leq u)$ are used interchangeably.

A *positive valuation* v on a lattice L is a real function $v : \mathsf{L} \rightarrow \mathsf{R}$ which satisfies both (1) $v(x) + v(y) = v(x \wedge y) + v(x \vee y)$ and (2) $x < y$ implies $v(x) < v(y)$ for $x, y \in \mathsf{L}$. Given a positive valuation function in a lattice L, one can show that both of the following are inclusion measures: $k(u \leq w) = \frac{v(w)}{v(u \vee w)}$ and $s(u \leq w) = \frac{v(u \wedge w)}{v(u)}$ [11]. Furthermore note that for a positive valuation function v in a lattice L, a metric distance is given in L by $d(x, y) = v(x \vee y) - v(x \wedge y)$ [11]. The work in [14] has shown how a positive valuation function in a lattice L can be extended to the lattice $\tau(\mathsf{L})$ of intervals.

Given a category function $g : \alpha(\mathsf{L}) \to \mathcal{M}$ where $\mathcal{M}$ is a finite set of category labels, then the finite labeled training data set is the pairs $(a_i, g(a_i))$, $i = 1, \ldots, n$ where a_i is an atom in $\alpha(\mathsf{L})$ and $g(a_i)$ is the corresponding category label. For classification problems, the goal is to learn a valid approximation of category function $g : \alpha(\mathsf{L}) \to \mathcal{M}$ so as to achieve an acceptable generalization on the testing data.

Learning from the training data is effected by a fuzzy lattice algorithm by computing *fits*, the latter are intervals of lattice elements computed by the lattice-join of training data in the same category. A *contradiction* occurs when a training datum from one category is included in a fit of a different category. A fit is called *tightest* when the lattice-join with any training datum from the same category (and not already in the fit) causes a contradiction. We are most interested in those category functions that are tightest fits.

In [14], positive valuations of the form $v(x_1, \ldots, x_N) = c_1x_1 + \ldots + c_Nx_N$, where $c_i > 0$ are defined, but are only used for one simple data set, and in that case, a heuristic approach is used to find the constants $c_i > 0$. Almost exclusively in [6, 7, 14] a positive valuation function of the form $v(x_1, \ldots, x_N) = x_1 + \ldots + x_N$ is used, i.e., $c_i = 1$, $\forall i$. We will refer to positive valuations of the form $v(x_1, \ldots, x_N) = c_1x_1 + \ldots + c_Nx_N$, where $c_i > 0$ as *positive hyperplane* valuations and the valuation of the form $v(x_1, \ldots, x_N) = x_1 + \ldots + x_N$ as the *unit* valuation.

In [6, 7, 14] the following five tightest fit classifiers are defined: *FLR tightest fit* (*FLRtf*), *FLR first fit* (*FLRff*), *FLR maximal tightest fit* (*FLRmtf*), *FLR ordered tightest fit* (*FLRotf*), and *FLR selective fit* (*FLRsf*). In the sections that follow, we focus our attention upon these five classifiers plus the valuations and inclusion measures used by these classifiers.

13.3 Introduction to Similarity and Distance Measures

In section 13.5 we will use similarity and distance measures to extend the FLR classifiers found in [6, 7, 14]. In this section we provide an introduction to these measures. In general, *distance* measures numerically how unlike (different) two datum are where as *similarity* measures numerically how alike two datum are. For a similarity measure, the idea is that a higher value indicates greater similarity where as for a distance measure, the lower (positive) value indicates greater similarity. In concept, a similarity measure is the converse of a distance measure. A formal definition of distance measure is given below, however, we find no formal definition found for similarity measures in the literature. There are many popular measures defined in the literature (e.g. L_p norm, Squared chord, squared Chi-squared, Canberra measure, Czekanowski coefficient, cosine, and correlation coefficient) for which all can be thought of as calculating a correlation between two data items (one of which is usually a group centroid). Generally, for these measures it is necessary that each of the two data items be expressed by a real-valued array of feature attributes.

In section 13.5 we extend the FLR classifiers to use distance measures. For our experimental application, we are mostly interested in the cosine similarity measure often used in cluster analysis. For the cosine similarity measure, if two data items $\mathbf{x}$ and $\mathbf{y}$ have K attributes each, then $\mathbf{x}$ and $\mathbf{y}$ can be thought of as two vectors. The cosine similarity measure is then the cosine of the angle between the two vectors x and y and is described by the formula [21]

$$CS(\mathbf{x},\mathbf{y}) = \frac{\sum_1^K x_i\, y_i}{\sqrt{\sum_1^K x_i^2 \sum_1^K y_i^2}}$$

The *cosine distance metric* is defined as $CD(\mathbf{x},\mathbf{y}) = 1 - CS(\mathbf{x},\mathbf{y})$.

In section 13.5 we also use distance measures to extend the FLR classifiers. Typically, a distance measure falls into one of two groups: *metric* and *semi-metric*. To be classified as *metric*, a non-negative distance between two vectors x and y must obey: 1) $d(x,y) = 0 \Rightarrow x = y$, 2) $d(x,y) = d(y,x)$, 3) $d(x,x) = 0$, and 4) when considering three objects, x, y and z, $d(x,y) \leq d(x,z) + d(z,y)$. Distance measures that obey the last three rules, but fail to obey rule 1 are referred to as *semi-metric*. The L_p norm is a well known group of distance measures as defined by the following. The L_p norm between two data items $\mathbf{x}$ and $\mathbf{y}$ each of which has K attributes is given by the formula

$$L_p(\mathbf{x},\mathbf{y}) = \left[\sum_{i=1}^{K} |x_i - y_i|^p\right]^{1/p}$$

If $p = 1$ then this distance is known as the *Manhattan* distance; if $p = 2$ then the formula produces the well-known Euclidean distance. If $p = \infty$, then the distance is called the Chebychev distance.

13.4 Introduction to Evolutionary Algorithms

Often times when a problem has no known algorithmic solution, i.e., not known to be a tractable problem, then some type of search technique can be used to arrive at a reasonable solution. In section 13.5 we are interested in finding coefficients for hyperplane valuations associated with FLR classifiers for which no known algorithmic solution exists. In section 13.7 we employ a group of search techniques generally categorized as evolutionary algorithms to find reasonable coefficients for hyperplane valuations associated with our experimental data set. In general, evolutionary algorithms are based upon concepts found in the study of population genetics. More specifically, the algorithms in this category maintain a population (set) of candidate solutions (individuals) to a problem. The fitness of a candidate solution is determined by evaluating how well it does on the problem, and the most fit candidate solution has some type of effect on the overall makeup of the candidate population. Thus, the population of candidate solutions evolves over time, hopefully resulting in improved solutions for the problem.

In section 13.7 we employ four different Evolutionary Algorithms: Genetic Algorithms [23], Tabu Search [9], Particle Swarm Optimization [15], and Differential Evolution [19] to find coefficients associated with hyperplane valuations for FLR classifiers. The remainder of this section is devoted to an introductory description of the four search techniques.

Genetic Algorithms [23] were introduced in the 1970's by John Holland. For this search technique, an initial population of candidate solutions is established (either randomly or with some known constraints), on which a genetic algorithm performs the following four steps until some stopping criterion is met: 1. Establish the fitness of each candidate solution in the population from time t; 2. Clone individuals from population t for a new population for time $t+1$ using the fitness distribution of the population from time t; 3. Perform crossover in population $t+1$; 4. Perform mutation in population $t+1$. The set of steps $1-4$ is generally called a generation. In many cases, the stopping criteria is merely a set number of generations. The crossover (step 3) combines two chromosomes from the parents to produce a new offspring chromosome with the desired affect being the offspring is better than the parents. Mutation (step 4) is a random change of a parameter value somewhere in the population, i.e., an alteration of one or more gene values in a chromosome. While it is possible to use a high mutation rate, this effectively produces random search. Most researchers use a very low mutation rate. Steps 2 and 3 are a bit more complicated than the other steps.

In step 2, individuals from population t are selected to appear in population $t+1$ based on their fitness relative to the rest of the population. The selection is generally either roulette wheel selection (where the entire set of fitnesses is used to establish the composition of the roulette wheel), or else the selection is a binary tournament selection (where pairs of individuals are chosen randomly and the one with the best fitness is copied into the new population).

In step 3, a percentage of the population is chosen in pairs to create new candidate solutions. The chosen pairs are called parents, and the new individuals are called children. Children are created by choosing some parameters from one parent and other parameters from the other parent. Typically the children replace the parents in the population. There are three common forms of crossover: one point, two point, and uniform. One point crossover randomly selects a crossover point and everything to the left of that point comes from one parent, while everything to the right of that point comes from the other parent. Two point crossover randomly selects two points and everything between the two points comes from one parent, while everything outside the two points comes from the other parent. Uniform crossover randomly chooses the parent for each parameter.

Tabu Search [9] is generally attributed to Fred Glover and like Genetic Algorithms originated in the 1970's. Unlike Genetic Algorithms, Tabu Search works with a population of a single candidate solution and can be considered a neighborhood search method. In addition to the single candidate solution population, there is a list of individuals that are neighbors of the candidate solution and a historical memory of previously encountered candidate solutions. The historical memory is called the Tabu list (there can be more than one Tabu list). The first individual may be generated randomly or based on some type of constraint. Once the individual's fitness is determined, the Tabu search algorithm will perform the following four steps until some stopping criterion is met: 1. create a neighborhood of individuals for the current candidate solution each of which is slightly different than the current candidate solution; 2. determine which of the newly created individuals has the best fitness; 3. if the new best fitness is better than the current candidate solution's fitness, then keep the new individual and place the old individual on the Tabu list; 4. if the new fitness is worse than the previous fitness but it is not on the Tabu list and it is only slightly worse than the current candidate solution, then keep the new individual and place the old individual on the Tabu list. As with Genetic Algorithms, Tabu Search has several parameters that must be set: the size of the neighborhood, the size of the Tabu list, the number of Tabu lists, the definition of "slightly worse", and the stopping criterion.

Particle Swarm Optimization [15], which was introduced by Russell C. Eberhart and James Kennedy in the 1990's, is both a global and a neighborhood optimization technique. The concept comes from observing the behavior of groups of organisms in nature, e.g., schools of fish and flocks of birds. Individuals within the groups have a position and a velocity which change over time relative to other individuals in the group. With this behavior in mind, Particle Swarm Optimization maintains a population of candidate solutions each of which has a position, x (i.e. a point in the search space) and a velocity v (i.e. the distance to move from the current position in the next time step). After initializing the population, the algorithm performs the following five steps until some stopping criterion is met: 1. Calculate each candidate solution's new position $x = x + v$; 2. Calculate the fitnesses of the new positions; 3. If a new fitness is better than the best fitness the individual has seen, χ, then $\chi = x$; 4. If a new fitness is better than the best fitness χ_g the swarm has seen then $\chi_g = x$; 5. Calculate the velocities for the new individuals using the formula $v = wv + c_1r_1(\chi - x) + c_2r_2(\chi_g - x)$.

Differential Evolution [19] was developed by Kenneth Price in the 1990's. Differential Evolution is very similar to genetic algorithms without the cloning step or the mutation step. Instead, the algorithm performs the following four steps for each individual (target) in the population: 1. Choose two random population members and calculate their weighted sum; 2. Add the calculated answer to a third randomly chosen individual; 3. Create a child by performing crossover with the answer from step 2 and the target individual; 4. If the child's fitness is better than the target's fitness, then replace the target with the child.

13.5 Classification Enhancements

This section describes enhancements to the inclusion measure used in [6, 7, 14]. Recall that each of the classifiers described and applied in [6, 7, 14] is based upon an inclusion measure given by the function $\sigma(u \leq w) = \frac{v(w)}{v(u \vee w)}$, where v is a positive valuation function. We will refer to this inclusion measure as the σ-inclusion measure. Generally, for the experimental results found in [6, 7, 14] the unit valuation function of the form $v(x_1, \ldots, x_N) = x_1 + \ldots + x_N$ is used. As noted in [14], we can also *weigh* each of the attributes $x_1, \ldots, x_N$ via a positive hyperplane valuation function $v(x_1, \ldots, x_N) = c_1x_1 + \ldots + c_Nx_N$, where $c_i > 0$.

In the following, we investigate the effect of using the combination of hyperplane valuations and the σ-inclusion measure. First, consider a point $C = (c_{1,} \ldots, c_N)$ where each $c_i > 0$ represents the coefficients used for the positive hyperplane valuation, i.e. $v(x_1, \ldots, x_N) = c_1x_1 + \ldots + c_Nx_N$. Since the point C is in R^N, we can translate C into polar coordinates as follows:

Let $\rho = \sqrt{c_1^2 + \ldots + c_N^2}$, $\theta_1, \ldots, \theta_{N-2}$ be the polar angles and let θ_{N-1} be the azimuthal angle, then

$c_1 = \rho\ cos\theta_1$
$c_2 = \rho\ sin\theta_1\ cos\theta_2$
$c_3 = \rho\ sin\theta_1\ sin\theta_2\ cos\theta_3$
$\ldots$
$c_{N-1} = \rho\ sin\theta_1\ sin\theta_2 \ldots sin\theta_{N-2}\ cos\theta_{N-1}$
$c_N = \rho\ sin\theta_1\ sin\theta_2 \ldots sin\theta_{N-2}\ sin\theta_{N-1}$

Consider the evaluation of the positive hyperplane valuation function $v(x)$,

$$\begin{aligned} v(x) = v(x_1, \ldots, x_N) &= c_1x_1 + \ldots + c_Nx_N \\ &= \rho\ cos\theta_1 x_1 + \rho\ sin\theta_1\ cos\theta_2 x_2 + \ldots + \rho\ sin\theta_1 \ldots sin\theta_{N-1} x_N \\ &= \rho(cos\theta_1 x_1 + sin\theta_1\ cos\theta_2 x_2 + \ldots + sin\theta_1 \ldots sin\theta_{N-1} x_N) \\ &= \rho(b_1x_1 + b_2x_2 + \ldots + b_Nx_N) \\ &= \rho\ v'(x), \end{aligned}$$

where the point $B = (b_1, b_2, \ldots, b_N)$ is the point on the (unit) hypersphere corresponding to the point C in R^N, i.e. $1 = \sqrt{b_1^2 + \ldots + b_N^2}$ with $b_i > 0$. Hence for the σ-inclusion measure with hyperplane valuation is given by

$$\sigma(u \leq w) = \frac{v(w)}{v(u \vee w)} = \frac{\rho v'(w)}{\rho v'(u \vee w)} = \frac{v'(w)}{v'(u \vee w)}$$

Thus, the σ-inclusion measure defined in [14] using hyperplane valuations is equivalent to using (unit) *hypersphere* valuations. In particular, this means that we can limit our search space from the first quadrant of R^N to the (unit) hypersphere in the first quadrant of R^N when trying to find reasonable coefficients for a hyperplane valuation.

Next, we consider the relationship of the following three entities: 1) σ-inclusion measure, 2) the distance (metric) $d(u, w) = v(u \vee w) - v(u \wedge w)$ defined in [3] where v is a valuation, and 3) the $L_1(u, w)$ norm. For simplicity, we use the unit hypercube I^N in R^N, the unit valuation, and a point, say w in I^N; we also use *complement coding* (in regards to the set A below) to represent a data vector as detailed in [14]. With these conditions, we have the following three sets being identical: $A = \{u \in \mathsf{I}^N : \sigma(u \leq w) = \frac{N}{N+m}\}$, $B = \{u \in \mathsf{I}^N : d(u,w) = m\}$, and $C = \{u \in \mathsf{I}^N : L_1(u,w) = m\}$. Using the relationship between A, B, and C plus the fact that in I^N, $\sigma(u \leq w) = \frac{N}{N+m} = \frac{v(w)}{v(w)+L_1(u,w)}$ we can see that the inclusion measure defined in [14] is based upon the L_1 norm. Furthermore, this relationship provides a general technique to define a measure based upon distance measures, i.e. $\delta(u,w) \equiv \frac{v(w)}{v(w)+d(u,w)}$, where d is a distance measure. Likewise, if s is a similarity measure such that $0 \leq s(u,w) \leq 1$ then we can define a measure as follows: $\delta(u,w) \equiv \frac{v(w)}{v(w)+(1-s(u,w))}$. We will refer to measures defined in this way (either using a distance measure or similarity measure) as δ-measures.

Now consider the training data given in Fig. 13.1 which consists of 12 training data and 3 test data. We are interested in a solution with 100% accuracy for the test data that uses the fewest number of hyperboxes possible. Using the geometric interpretations described in [14] where the data are grouped via hyperboxes, we quickly conclude there is no solution using the unit valuation that is 100% accurate for the test data. In other words, one cannot place point g in a training data rectangle without that rectangle also including point 7 or 8. This can be verified via experiments as well. To solve this problem, we turn our attention to hyperplane valuations and σ-inclusion measure. Since we have shown that we may limit our search for coefficients of the hyper-

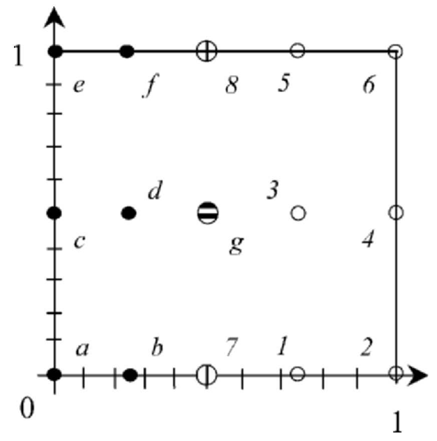

Fig. 13.1. Fifteen training data are presented in the order a(0,0), b(.25,0), c(0,.5), d(.25,.5), e(0,1), f(.25,1) (category "•"), and *1*(.75,0), *2*(1,1), *3*(.75,.5), *4*(1,.5), *5*(.75,1), *6*(1,1) (category "o"). Three test data are g(.5,.5) from category "•" and *7*(.5,0), *8*(.5,1) from category "o". The problem is to find a tightest fit category function that uses the fewest hyperboxes and has 100% test data accuracy

plane valuation to the first quadrant hypersphere, an exhaustive search is not unreasonable, i.e. we can restrict our search to the circumference of the unit circle in the first quadrant. Via experiments, we found that none of the five classifiers *FLRtf*, *FLRff*, *FLRmtf*, *FLRotf*, and*FLRsf* can group the training data with 100% test data accuracy using σ-inclusion measure with hyperplane valuation. Actually for the given data found in Fig. 13.1, each of the five classifiers *FLRtf*, *FLRff*, *FLRmtf*, *FLRotf*, and *FLRsf* provide the same test data accuracy results (one data point incorrect using two hyperboxes) whether a unit or hyperplane valuation is used.

In an effort to find better inclusion measures, we have conducted numerous experiments on different data sets using different δ-measures. We tried well known distance and/or similarity measures such as the L_p norms, cosine, squared chord, squared Chi-squared, Canberra measure, Czekanowski coefficient, and correlation coefficient. However, none of these δ-measures consistently performed as well as the σ-inclusion measure defined in [14] when using a unit valuation nor are these δ-measures able to solve the problem noted in Fig. 13.1 *when using a unit valuation.*

In order to solve the problem in Fig. 13.1, we now turn our focus to a δ-measure that uses weighted attributes. As previously noted, the σ-inclusion measure using a hyperplane valuation is equivalent to a σ-inclusion measure $\frac{v'(w)}{v'(u \vee w)}$ where the coefficients for v' are taken from the unit hypersphere. This equivalency is not true, however, of all δ-measures. Consider the weighted (attribute) cosine similarity measure

$$ACS(u,w) = \frac{\sum\limits_i c_i u_i w_i}{\sqrt{(\sum\limits_i c_i u_i^2)(\sum\limits_i c_i w_i^2)}}$$

and the following four δ measures with ACD representing the attribute cosine distance metric

$\delta 1(u,w) = \frac{v(w)}{v(w)+ACD(u,w)}$,

$\delta 2(u,w) = \frac{v(w)}{v(w)+ACD(u \vee w,w)}$,

$\delta 3(u,w) = \frac{v(w)}{v(w \vee u)+ACD(u \vee w,w)}$, and

$\delta 4(u,w) = \frac{v(w)-(1-ACD(u \wedge w,w))}{v(w)}$.

Note that for each measure, constant vector $C = (c_1, \ldots, c_N)$ is used in both the valuation function $v(x)$ and the attribute cosine distance metric ACD.

We now reconsider the problem described in Fig. 13.1 and conduct experiments using the five classifiers *FLRtf*, *FLRff*, *FLRmtf*, *FLRotf*, and *FLRsf* using each of the four weighted cosine δ-measures: $\delta 1$, $\delta 2$, $\delta 3$, and $\delta 4$. For each experiment, no FLR training parameters (such as epsilon for improved accuracy) are used. Hence, this requires the coefficients used in the weighted cosine δ-measure to completely compensate for all FLR parameters. Thus for our experiments, each FLR classifier is simply used to compute the number of hyperboxes needed by the training data, and in return, the number of correctly classified test data. The goal in each experiment is to find coefficients for the weighted cosine δ-measure such that the number of hyperboxes is the fewest possible and 100% test data accuracy. Recall for the weighted cosine δ-measure, our search space is not restricted to the unit circle. Hence an exhaustive search of the first quadrant in R^2 is used. When using the four weighted cosine δ-measures, each of the five classifiers are able to solve the problem in Fig. 13.1 using three hyperboxes and 100% test data accuracy. The graph of the solution space for the classifier *FLRotf* using the $\delta 2$-measure is given in Fig. 13.2. Table 13.1 provides a single solution (a pair of coefficients) for each of the modified five classifiers (there are infinitely many solutions) using the $\delta 2$-measure. As we previously stated, none of the five classifiers *FLRtf*, *FLRff, FLRmtf, FLRotf*, and *FLRsf* can solve the problem given in Fig. 13.1 using the original inclusion measure given in [14] and hyperplane valuations. It is only when one combines both the weighted cosine δ-measure and hyperplane valuations are the modified five classifiers *FLRtf*, *FLRff, FLRmtf, FLRotf,* and *FLRsf* able to solve the problem in Fig. 13.1. Via experiments, we found that the $\delta 2$-measure is most successful (of the four $\delta 1$, $\delta 2$, $\delta 3$, and $\delta 4$) in solving the problem in Fig. 13.1 and problems similar to those given in Fig. 13.8, Fig. 13.9, and Fig. 13.10 below. Because of the success of $\delta 2$, future unqualified references to *weighted cosine δ-measure* are assumed to be a reference to the $\delta 2$ definition only.

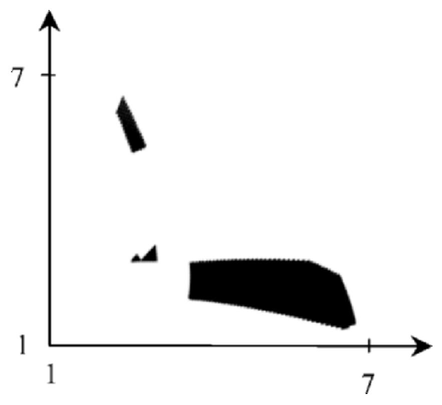

Fig. 13.2. FLRotf weighted cosine solution for the problem described in Fig. 13.1

Table 13.1. Coefficients for weighted cosine δ measure associated with problem described in Fig. 13.1

FLR classifier	C1 coefficients	C2 coefficients
FLRtf	6.391859	2.853784
FLRff	2.488825	5.349365
FLRmtf	6.391859	2.853784
FLRotf	2.761679	2.893634
FLRsf	2.761679	2.893634

Let's revisit the definition of an inclusion measure given above, namely σ is a map $\sigma : \mathsf{L} \times \mathsf{L} \rightarrow [0,1]$, which satisfies the following conditions

(C0) $\sigma(x,O) = 0, x \neq O$,

(C1) $\sigma(x,x) = 1$,

(C2) $w \wedge u < w \Rightarrow \sigma(w,u) < 1$, and

(C3) $u \leq w \Rightarrow \sigma(x,u) \leq \sigma(x,w)$ - Consistency Property

and our definition of the weighted cosine δ-measure given above. First, consider conditions (C0) and (C1), the unit hypercube, and the inclusion measure defined by $\sigma(u,w) = \frac{v(w)}{v(u \vee w)}$ where v is a positive valuation. Under these circumstances, in order to satisfy condition (C1), $\sigma(O,O)$ is defined to be 1 [11]. This definition is needed since one has division by zero whenever $v(u \vee w)$ is zero in $\frac{v(w)}{v(u \vee w)}$. Note that σ is not continous at O since as u approaches O, $\sigma(u,O)$ is zero and $\sigma(u,u)$ is one. For the weighted cosine δ-measure, a similar problem exists in that one has a division by zero whenever $ACD(u \vee w, w)$ is undefined or equivalently whenever $v(u \vee w) = 0$ or $v(w) = 0$ or equivalently whenever $v(u) = 0$ or $v(w) = 0$. If one defines $ACD(u,w)$ to be some value geater than zero, say 1, whenever $v(u) = 0$ or $v(w) = 0$, then another problem occurs when one evaluates $\delta(O,O)$, i.e. the value of $\delta(O,O)$ evaluates to 0 and in conflict with condition (C1). One solution is to define $ACD(u,w)$ to be zero whenever either $v(u) = 0$ or $v(w) = 0$ and define $\delta(O,O) = 1$. Hence, by defining special case values, we can satisfy conditions (C0) and (C1). An alternative would be to modify (C1) to read as $\sigma(x,x) = 1$, for all $x \neq O$ and to define $ACD(u,w) = 1$ whenever $v(u) = 0$ or $v(w) = 0$. Now, let's consider condition (C2). In general, condition (C2) is not true even with the restriction to positive valuations and special definitions for ACD. As an example, consider the unit hypercube I^N in R^N and two points (vectors) u and w. Now if the vectors u and w have the same direction (when viewed as vectors) and $u < w$ in I^N, then $w \wedge u = u < w$ and $\delta(w,u) = \frac{v(u)}{v(u)+1-ACS(w \vee u,u)} = \frac{v(u)}{v(u)+1-1} = 1$. To show that $ACS(w \vee u, u)$ is one, consider the following: 1) since w and u have the same direction and $w \wedge u = u < w$, $w \vee u = w$ has the same direction as u, 2) since $ACS(w \vee u, u)$ is the cosine of the angle formed between the vector u and $w \vee u$, the $ACS(w \vee u, u)$ is just the cosine of the angle zero since u and $w \vee u$ have the same direction. This is in direct conflict with condition (C2). An alternative for (C2) is $w \wedge u \leq w \Rightarrow \sigma(w,u) \leq 1$. In regards to condition (C3), the weighted cosine δ measure does not meet this condition. To demonstrate this, consider the unit hypercube I^N in R^N and three points (vectors) x, u and w with 1) the vectors x and u having the same direction, 2) $u < x$ and $u < w$ in I^N, 3) $u \neq O$, and 4) u and w do not have the same direction (see Fig. 13.3 for an example in I^2). Then

$\delta(x,u) = \frac{v(u)}{v(u)+1-ACS(x \vee u,u)} = \frac{v(u)}{v(u)+1-ACS(x,u)} = \frac{v(u)}{v(u)+1-1} = 1$, and

$\delta(x,w) = \frac{v(w)}{v(w)+1-ACS(x \vee w,w)} = \frac{v(w)}{v(w)+1-m} < 1$,

where i) $ACS(x,u)$ is one since the angle between x and u is zero, ii) $ACS(x \vee w, w)$ is some constant m, $0 < m < 1$, i.e. the angle between $x \vee w$ and w, θ,

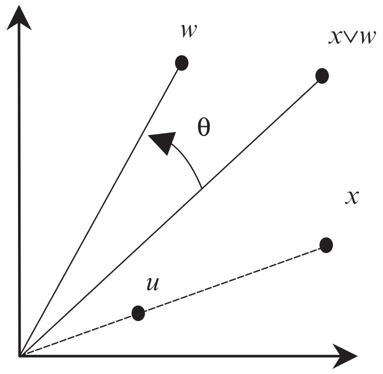

Fig. 13.3. For the given points, we have both $\delta(x,u) = 1$ and $\delta(x,w) < 1$

is between 0 and $\pi/2$, exclusive. Thus the weighted cosine δ measure fails to meet condition (C3) and no alternative for (C3) is offered. Although no proof is given here, note that $\delta 3$ as defined above is an inclusion measure.

13.6 Investigating Further the Weighted Cosine Measure

Let's further consider the *FLRotf* solution noted in Fig. 13.2 and the weighted cosine δ-measure. All of the coefficient solutions given in Fig. 13.2 give the same set of hyperboxes, i.e. if we consider a specific coefficient solution, say $(5, 2)$, and apply the *FLRotf* classifier, then the resulting solution consists of exactly three hyperboxes: one hyperbox containing the single point $(.75, 0)$, one hyperbox containing the single point $(.75, 1)$, and one hyperbox containing the single point $(.25, .5)$. We will refer to these hyperboxes as A, B, and C respectively. See Fig. 13.4. Is this solution a tightest fit solution, i.e. can either of the hyperboxes A, B, and C be expanded to include more training data without incurring a contradiction? Although visually it appears that one can expand each of the hyperboxes, the fit is indeed (based upon the computations) a tightest fit.

Although we cannot graph the weighted cosine δ-measure for the above problem $\{z : z = \delta(u, w)$, where u and w are members of $\mathsf{I}^2\times$ (dual of) $\mathsf{I}^2\}$ since the input range is 8 dimensions, we can, however, graphically view the how the weighted cosine δ-measure behaves for specific coefficients. To do this, we consider a pair of coefficients from each of the blocks in the solution space identified in Fig. 13.2: namely the coefficient pairs $(5, 2)$, $(2.75, 3)$, and $(2.3, 5.75)$ giving one pair from each region. To graphically view the measure, we use three data sets: the training and test sets associated with Fig. 13.1 and a third data set (validation) defined to be $\{(x, y, 1) : (x, y) \in \mathsf{I}^2$ and 1 represents the category for the pair$\}$. The training and test sets are used to obtain the hyperboxes identified in Fig. 13.4 (initial training) while the third data set is

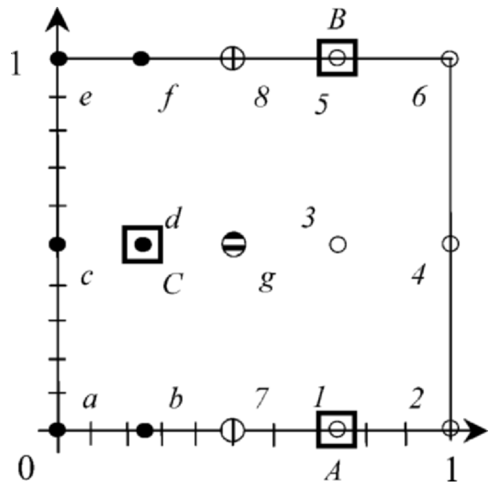

Fig. 13.4. Hyperbox A contains the single point $(.75, 0)$, hyperbox B contains the single point $(.75, 1)$, and hyperbox C contains the single point $(.25, .5)$

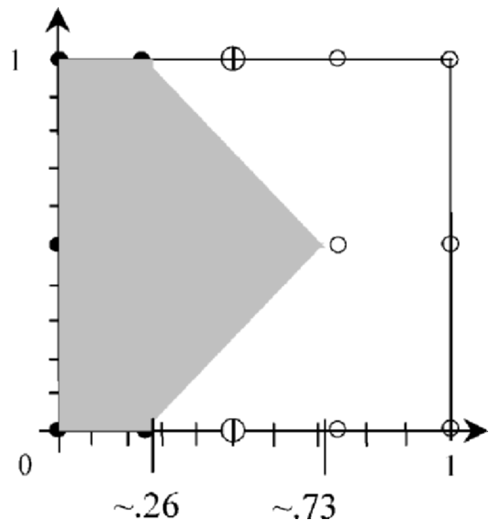

Fig. 13.5. Coefficients $(2.75, 3)$ are employed in this figure

used to locate the boundary between the region correctly classified and the region misclassified. We shall use this technique for multiple experiments that follow. Note that we not using the validation set in the traditional sense, but only to gather information about the boundary of the category regions given since we are no longer dealing with traditional hyperboxes. Hence, for the problem stated in Fig. 13.1 and coefficient solution space given in Fig. 13.2, the category regions for each pair of coefficients are given in Fig. 13.5, Fig. 13.6, and Fig. 13.7. For each the three figures, the category "•" is the shaded

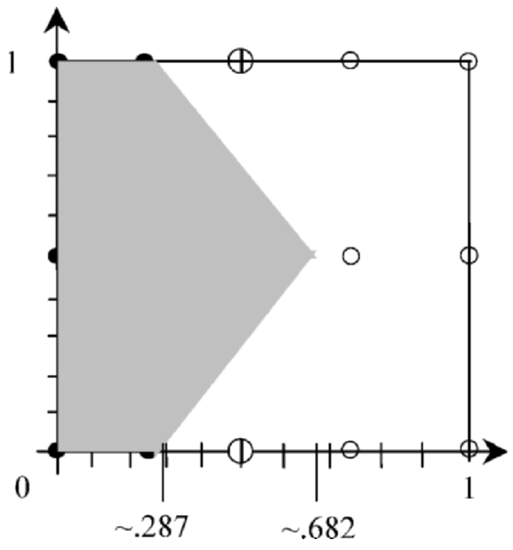

Fig. 13.6. Coefficients $(2.3, 5.75)$ are employed in this figure

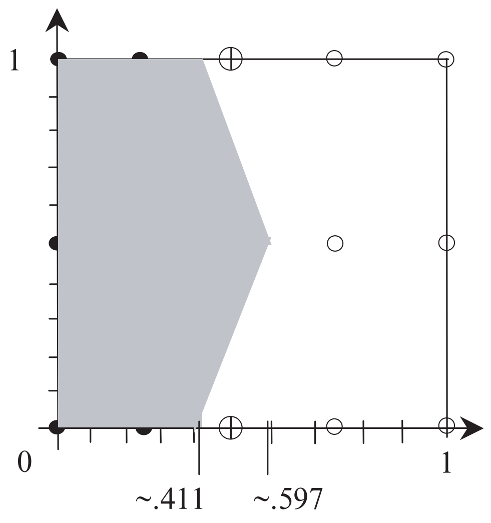

Fig. 13.7. Coefficients $(5, 2)$ are employed in this figure

portion while the category "o" is represented by the unshaded region. As demonstrated in Fig. 13.5, Fig. 13.6, and Fig. 13.7, the shaded triangular region grows and shrinks based upon the coefficients used in the weighted cosine δ-measure. For the coefficients $(1, 1)$, the triangular region disappears with the category "•" being the rectangular region $[0, .5] \times [0, 1]$ and category "o" being the remaining area. This results in two misclassifications for the test data. In other words, without the "weighting" the cosine δ-measure achieves the same results as the traditional *FLNotf* classifier (for this problem). It is also interesting to note that for other coefficient pairs such as $(8, 2)$, the triangular region protrudes to the left instead of to the right, i.e. a mirror image of the ones given in Fig. 13.5, Fig. 13.6, and Fig. 13.7.

Next, we investigate how the weighted cosine δ-measure adapts to a slightly different test set (the training set remains as found in Fig. 13.1). Namely, the test data set: $g(.5, .75)$, $h(.5, .25)$ from category "•" and $7(.5, 0)$, $8(.5, 1)$, $9(.5, .5)$ from category "o". We can perform an exhaustive search and determine that in deed the weighted cosine δ-measure can solve this modified problem as well. For one solution to this problem using the weighted cosine δ-measure with coefficients $(1.71, 2.1)$, see Fig. 13.8. In Fig. 13.8, the curve line near the vertical line $x = 0.5$ represents the boundary between the two categories "•" and "o" and is generated by the weighted cosine δ-measure using coefficients $(1.71, 2.1)$. When using the given coefficients, the *FLNotf* with weighted cosine δ-measure is able to train with 100% accuracy and 100% correct for the test set. Also in Fig. 13.8, the resulting trained hyperboxes are labeled A, B, C, D, E, and F, i.e. there are 6 hyperboxes. Again, this solution is a tightest fit. Visually, it appears that one could form one hyperbox containing both the hyperbox A and hyperbox B, but in doing so, the single box would cause miss classifications along the vertical line at $x = 0.5$.

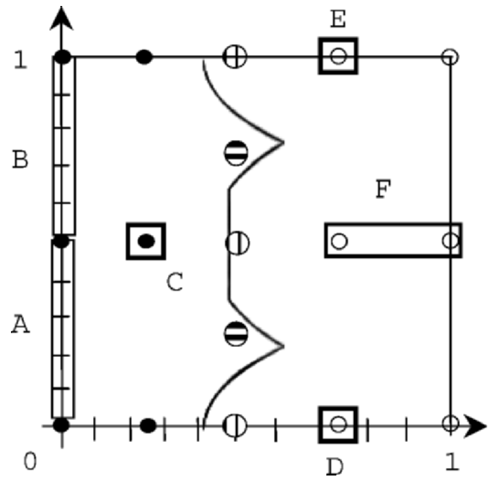

Fig. 13.8. The training set consists of points $a(0, 0)$, $b(.25, 0)$, $c(0, .5)$, $d(.25, .5)$, $e(0, 1)$, $f(.25, 1)$ (category "•") and $1(.75, 0)$, $2(1, 1)$, $3(.75, .5)$, $4(1, .5)$, $5(.75, 1)$, $6(1, 1)$ (category "o") plus the test data set: $g(.5, .75)$, $h(.5, .25)$ from category "•" and $7(.5, 0)$, $8(.5, 1)$, $9(.5, .5)$ from category "o". To produce the curve boundary between the two categories, the weighted cosine δ-measure uses coefficients $(1.71, 2.1)$. The six hyperboxes A, B, C, D, E, and F represent the training results for *FLNotf* and form a tightest fit

Next, we experiment by tilting the series of test points found near $x = .5$ for Fig. 13.8 while the training set remains the same. We tilt the test points to the left, i.e. negative slope, with a pivot point at $(.5, .5)$. Via experimentation, we determine that *FLNotf* is successful in achieving 100% accuracy for the test data with a slope range from vertical down to -7 (actually somewhere between -6 and -7) with each line passing through the pivot point $(.5, .5)$. Thus for a slope of -7, the given line is $y = -7x + 4$ and corresponding points $g(.4643, .75)$, $h(.5357, .25)$ from category "•" and $7(.5714, 0)$, $8(.4246, 1)$, $9(.5, .5)$ from category "o" are given in Fig. 13.9. Although the *FLNotf* achieves success in adapting to the tilting, it does so by modifying the curvature of the boundary (see Fig. 13.9) and not by tilting of the curve. Hence the axis remains vertical. Again for these experiments, an exhaustive search is first done to determine possible coefficients for the weighted cosine δ-measure followed by the use of a validation set to determine the boundary between the correctly classified sets.

Finally, we investigate how the weighted cosine δ-measure adapts to a slightly different training and test set. Namely, a training set of $a(0, 0)$, $b(.25, 0)$, $c(0, .5)$, $d(.25, .5)$, $e(0, 1)$, and $f(.25, 1)$ in category "•" and $1(.75, 0)$, $2(1, 1)$, $3(.75, 0.5)$, $4(1, 0.5)$, $5(.75, 1)$, and $6(1, 1)$ in category "o" plus the test data set: $(.5, 0.125)$, $(.5, 0.375)$, $(.5, 0.625)$, $(.5, 0.875)$ from category "•" and $(.5, 0)$, $(.5, 0.25)$, $(.5, 0.5)$, $(.5, 0.75)$, and $(.5, 1)$ from category "o". We again perform an exhaustive search and determine that in deed the weighted cosine

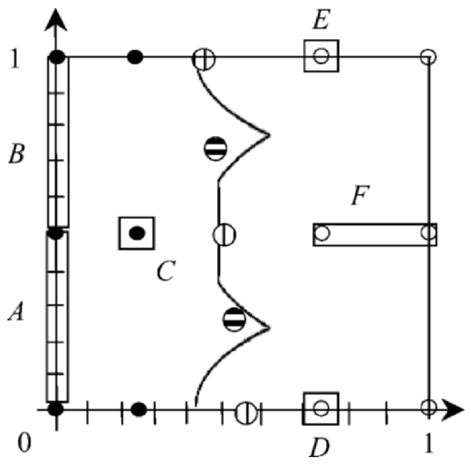

Fig. 13.9. The training set consists of points $a(0, 0)$, $b(.25, 0)$, $c(0, .5)$, $d(.25, .5)$, $e(0, 1)$, $f(.25, 1)$ (category "•") and $1(.75, 0)$, $2(1, 1)$, $3(.75, .5)$, $4(1, .5)$, $5(.75, 1)$, $6(1, 1)$ (category "o") plus the test data set: $g(.4643, .75)$, $h(.5357, .25)$ from category "•" and $7(.5714, 0)$, $8(.4246, 1)$, $9(.5, .5)$ from category "o". In this case the test data are tilted to the left and are obtained from the line $y = -7x+4$. To produce the curve boundary between the two categories, the weighted cosine δ-measure uses coefficients $(0.600012, 2.42178)$. Six hyperboxes represent the training results for *FLNotf* (there are two hyperboxes along the y axis). These form a tightest fit

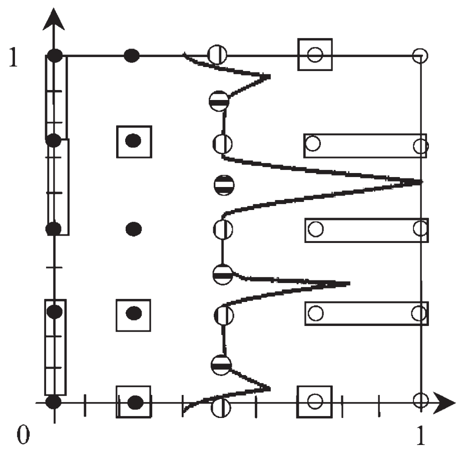

Fig. 13.10. The training set consists of points $a(0,0)$, $b(.25,0)$, $c(0,.5)$, $d(.25,.5)$, $e(0,1)$, $f(.25,1)$ (category "•") and $1(.75,0)$, $2(1,1)$, $3(.75,.5)$, $4(1,.5)$, $5(.75,1)$, $6(1,1)$ (category "o") plus the test data set: $(0.125, 0.5)$, $(0.375, 0.5)$, $(0.625, 0.5)$, $(0.875, 0.5)$ from category "•" and $(.5, 0)$, $(.5, 0.25)$, $(.5, 0.5)$, $(.5, 0.75)$, $(.5, 1)$ from category "o". To produce the curve boundary between the two category regions, the weighted cosine δ-measure uses coefficients $(0.004, 0.255272)$. When using the given coefficients, the *FLNotf* with weighted cosine δ-measure is able to train with 100% accuracy and 100% correct for the test set. There are 11 hyperboxes in the training results for *FLNotf*

δ-measure can solve this modified problem. For one solution to this problem using the weighted cosine δ-measure with coefficients $(0.004, 0.255272)$, see Fig. 13.10. In Fig. 13.10, the curve line near the vertical line $x = 0.5$ represents the boundary between the solution regions for the two categories "•" and "o". When using the given coefficients, the *FLNotf* with weighted cosine δ-measure is able to train with 100% accuracy and 100% correct for the test set. As noted in Fig. 13.10, the resulting 11 trained hyperboxes are given. It is somewhat disappointing that this boundary region is not symmetrical, i.e. the spikes near $y = 0.375$ and $y = 0.625$ are not the same. This observation tends to imply that if the following two data points were added to the training set: $(.75, 0.625)$, $(1, 0.625)$, then certainly this solution would not work. When these additional two points are added, we are not able via experimentation using an exhaustive search for the weighted cosine δ-measure coefficients determine a solution that is 100% accurate for the test data.

13.7 Experiments and Results

The general learning capacity of the classifiers *FLRff*, *FLRotf*, and *FLRsf* with the weighted cosine δ-measure is demonstrated in this section by considering the Cleveland heart benchmark classification problem from the UCI repository of machine learning data sets [4].

The Cleveland heart data set involves numeric data of various sizes including missing attribute values. A missing attribute value is dealt with by replacing it with the least element in the corresponding constituent lattice as explained in [14]. Thus each data vector is represented in the lattice R^N. *Complement coding* is used to represent a data vector, in particular instead of $(x_1, \ldots, x_N)$ the vector $(1 - x_1, x_1, \ldots, 1 - x_N, x_N)$ is used [12, 14].

For the Cleveland heart data, the problem is to diagnose heart disease in a patient from a 14-attribute vector. This benchmark consists of 303 data vectors. The severity of heart disease is indicated by an integer ranging from 0 (no heart disease) to 4. By collapsing the classes into two, i.e. absence versus presence of heart disease, the problem becomes a "2-categories" problem as opposed to the original "5-categories" problem. Because no training and testing data sets are given explicitly, a *keep-250-in* series of 100 experiments is carried out such that in each experiment 250 randomly selected data are used for training and the remaining 53 data are left out for testing. Also, since we are wanting to compare the weighted cosine δ-measure results with existing published results for other classifiers. Since our comparison is to published works that did not use a validation set in their experiments, neither will our experiments use a validation set. Again, we are primarily interested in showing the malleability of the weighted cosine δ-measure via our experiments.

Table 13.2 shows the results by various methods from the literature in the 2-categories problem. Table 13.3 shows results for the 5-category problem using various FLR classifiers as reported in [6, 7, 14]. In Table 13.3, the average and the corresponding standard deviation are shown for both the classification accuracy and the number of rules in a series of experiments. The same 100 randomly generated data sets are used in all 5-category problems listed in Table 13.3.

For our experiments involving the Cleveland heart data, the same 100 randomly generated data sets are used in all 5-category problems. For each of the three classifiers *FLRff*, *FLRotf*, and *FLRsf*, four evolutionary algorithms are employed to find reasonable coefficients for the weighted cosine δ-measure. All total, we conducted 1200 experiments (100 data sets with three classifiers

Table 13.2. Performance of various methods from the literature in classifying Cleveland's 2-categories benchmark "heart problem"

Pattern classification method	Classification accuracy (%)
Probability analysis	79.0
Conceptual clustering (CLASSIT)	78.9
ARTMAP-IC (10 voters)	78.0
Discriminant analysis	77.0
Instance based prediction (NTgrowth)	77.0
Instance based prediction (C4)	74.8
Fuzzy ARTMAP (10 voters)	74.0
KNN (10 neighbors)	67.0

Table 13.3. Performance of classifiers *FLRtf*, *FLRff*, *FLRotf*, *FLRmtf*, and *FLRsf* in the Cleveland's benchmark "5-categories" heart problem. Results are from [7]

FLR classifier	Classification Accuracy Average (%)	std. deviation	Number of rules Average (%)	std. deviation
FLRff	66.74	4.96	53.47	10.30
FLRotf	66.60	5.52	51.18	11.00
FLRsf	66.49	5.59	49.47	9.80
FLRmtf	65.38	5.00	61.24	7.00
FLRtf	56.74	7.23	86.81	4.67

and four evolutionary algorithms per classifier). For each of the four evolutionary algorithms used, the same parameters are used on all 100 data sets and for each classifier. Each FLR classifier is merely used to determine the fitness of an evolutionary algorithm's solution as noted in Sect. 13.4. We use the Genetic Algorithm Utility Library (GAUL) software [1] for genetic algorithms, tabu search, and differential evolution portion of our experiments. For particle swarm optimization, we adapted software originally written by Clerc [5].

For our experiments, we modify each FLR classifier *FLRff*, *FLRotf*, and *FLRsf* to use the weighted cosine δ-measure in place of the traditional σ-inclusion measure. For the problem given in Fig. 13.1, it is reasonable to use an exhaustive search to find the hyperplane valuation for the weighted cosine since the search space is the first quadrant of R^2. However for larger problem sets (with more attributes) such as the Cleveland heart data, an exhaustive search is not realistic. For larger problem sets we employ evolutionary algorithms to find a hyperplane valuation for the weighted cosine δ-measure.

In particular, we use evolutionary algorithms to provide a set of coefficients for the weighted cosine δ-measure. In each experiment, the initial coefficient set is randomly generated. Each of the FLR classifiers are then used to determine the fitness of the coefficients provided by the evolutionary algorithm, i.e. to determine the number of test data correctly identified and the number of hyperboxes used. Using the fitness information provided by the FLR classifier, the evolutionary algorithm updates its candidate solution and in return provides an additional set of coefficients for the weighted cosine δ-measure. For each evolutionary algorithm and all data sets, the evolutionary algorithm parameters remain fixed.

The parameter settings for each evolutionary algorithm follows. In all cases, the stopping criterion for each evolutionary algorithm is a fixed number of generations. For the genetic algorithm, a set of coefficients for the weighted cosine δ-measure represent a single chromosome. We use a population size of 500 which is initially randomly generated/seeded within the range 3.0 to 44.0. The stopping criterion is set to 200 generations; our mutation rate is 0.001 (i.e. one out of every 1000 parameters in a population was altered). For selection we use binary tournament selection where the elitist parents survive from one generation to the next. For crossover, we use a two point crossover in which two crossover points are randomly selected, and we set the percentage of the population that undergo crossover at 60%. We experimented with multiple techniques of selection and mutation. For example, we experimented with directional adjustments to the chromosome similar to what one would do using a "hill climbing" technique, i.e. we iterated through the genes of a chromosome making small adjustments and testing for improvement. Unfortunately, none of techniques showed improvement over a mutation rate of 0.001 and binary tournament selection. For tabu search, we used a population size of 500 which was initially randomly generated within the range of 0.3 and 50. The stopping criterion was set to 300 generations. We used a single tabu list size of 400 with a neighborhood size of 100. For mutation, we randomly chose whether to swap or shift a list item. A member was tagged as tabu if it differed no more than 0.1 from other members. For differential evolution, we used a population size of 500 which was initially randomly generated within the range 0.3 and 50. The stopping criterion was set to 100 generations. We chose to use the strategy "best", crossover "exponential", perbations "1", weighting factor 0.5, and crossover 0.8. For particle swarm optimization, we used a swarm size of 500, confidence coefficient of 1.42694, 100 informers, and 15 executions as the stopping criterion.

Table 13.4 shows the results for the classifiers *FLRff*, *FLRotf*, and *FLRsf* with the weighted cosine δ-measure for the 5-category problem. The use of the weighted cosine δ-measure shows considerable improvement (approximately 10%) in the classification accuracy for this problem as compared to the same classifiers for the same data sets using σ-inclusion measure. In fact, the 5-category results are comparable to most other methods when they are applied to the 2-category problem. See Table 13.2. When differential evolution is used to find the coefficients of the weighted cosine δ-measure, each of three classifiers performed better than the original classifier in every case, i.e. in each of the 100 data sets. Overall, the genetic algorithm and differential evolu-

Table 13.4. Both classification accuracy and no. rules statistics for *FLRff*, *FLRotf*, and *FLRsf* in the Cleveland "heart problem" using weighted cosine δ-measure

	FLRff					*FLRotf*					*FLRsf*				
	% clas accu			no. rules		% clas accu			no. rules		% clas accu			no. rules	
Method	av	sd	1-1*	av	sd	av	sd	1-1*	av	sd	av	sd	1-1*	av	sd
GA	75.2	2.7	100	49.0	3.6	75.3	2.5	99	49.7	3.6	75.3	2.6	99	48.9	3.5
Tabu	72.8	2.7	97	49.9	3.1	73.2	2.7	95	49.8	3.3	72.5	2.8	90	49.8	2.9
PSO	70.5	2.6	92	49.9	3.2	70.5	2.6	89	49.8	3.2	70.5	2.6	84	49.9	3.2
DE	75.5	2.3	100	50.8	3.8	75.2	2.5	100	51.2	4.3	75.5	2.3	100	50.9	3.8
Original	66.0	2.6	–	53.5	10.3	66.6	2.9	–	51.2	11.0	66.5	3.0	–	49.5	9.8

* For 100 data sets, a 1-to-1 comparison was made between an enhanced classifier and the original one. For example, entry 97 for Tabu and *FLRff* indicates that of the 100 data sets tested, the enhanced *FLRff* classifier outperformed the traditional *FLRff* on 97 out of 100 data sets.

tion search techniques provided better results than tabu and particle swarm optimization search techniques. Over fitting the training data is ruled out by construction, i.e. no FLR training parameters are used for the FLR classifiers and a fixed number of generations are used for each evolutionary algorithm — independent of the data set used.

13.8 Discussion and Conclusion

This work builds upon previous work found in [6, 7, 14] by modifying the definition of an inclusion measure used in those articles to a measure that we refer to as a δ measure. Although this measure is no longer an inclusion measure in the strictest sense, it can still be used in the traditional FLR tightest fit classifiers. We have also shown that the traditional σ-inclusion measure based upon hyperplane valuations (used in previous articles) is actually limited to the unit hypersphere valuation. It is also shown that the σ-inclusion measure is related to the L_1 norm (in I^N). We have also extended the σ-inclusion measure to a δ measure that is based upon distance and similarity measures. This extension opens a corridor between FLR and cluster analysis research, i.e. the similarity measures of cluster analysis can be used in FLR, and in return FLR classifiers can be more readily applied to cluster analysis. We have shown that FLR classifiers using the weighted cosine measure are able to solve problems for which σ-inclusion hyperplane measures could not. We have used the weighted cosine δ-measure in FLR classifiers plus evolutionary algorithms to find the weights associated with the cosine δ-measure. An application of three FLR classifiers using δ-measure to the Cleveland heart data 5-category problem shows significant improvements over alternative classification methods from the literature. In fact, our results for the 5-category problem are comparable to alternative classification methods reported in the literature for the 2-category problem.

Future work includes a comparison of the weighted cosine δ-measure and σ-inclusion measure on other data sets. Additional work needs to be done in search of similarity measures now used in cluster analysis that may be used as δ measures. Additional research needs to be done to determine if there is a more efficient way to find coefficients used in the weighted cosine δ-measure for a given problem set. Open questions: is there a methodology that can be used to find an optimal or near optimal set of coefficients for different problem sets; are there other measures that can generally perform better than the weighted cosine δ-measure.

References

1. Adcock S, Genetic Algorithm Utility Library, `http://gaul.sourceforge.net/`, version devel-0.1849
2. Anagnostopoulos GC, Georgiopoulos M (2001) New geometric concepts in fuzzy-ART and fuzzy-ARTMAP: category regions. In: Proc Intl J Conf Neural Networks (IJCNN) 1:32–37

3. Birkhoff G (1967) Lattice Theory. American Math Society, Colloquium Publications XXV
4. Blake CL, Merz CJ (1998) UCI repository of machine learning data-bases `http://www.ics.uci.edu/ mlearn/MLRepository.html` Univ California, Irvine
5. Clerc M, Basic particle swarm optimization software `http://clerc.maurice.free.fr/pso/PSO_basic/zo`
6. Cripps A, Kaburlasos VG, Nguyen N, Papadakis SE (2003) Improved experimental results using fuzzy lattice neurocomputing (FLN) classifiers. In: Proc Intl Conf Machine Learning, Models, Technologies and Applications (MLMTA) pp 161–166
7. Cripps A, Nguyen N, Kaburlasos VG (2003) Three improved fuzzy lattice neurocomputing (FLN) classifiers. In: Proc Intl J Conf Neural Networks (IJCNN) 3:1957–1962
8. Gabrys B (2002) Combining neuro-fuzzy classifiers for improved generalisation and reliability. In: Proc Intl J Conf Neural Networks (IJCNN) 3:2410–2415
9. Glover F, Laguna M (1997) Tabu Search. Kluwer, Norwell, MA
10. Granger E, Rubin MA, Grossberg S, Lavoie P (2002) Classification of incomplete data using the fuzzy ARTMAP neural network. In: Proc Intl J Conf Neural Networks (IJCNN) 6:35–40
11. Kaburlasos VG (2006) Towards a Unified Modeling and Knowledge Representation Based on Lattice Theory — Computational Intelligence and Soft Computing Applications, ser Studies in Computational Intelligence 27. Springer, Heidelberg, Germany
12. Kaburlasos VG (2007) Granular enhancement of fuzzy-ART/SOM neural classifiers based on lattice theory. This volume, Chapter 1
13. Kaburlasos VG, Athanasiadis IN, Mitkas PA (2007) Fuzzy lattice reasoning (FLR) classifier and its application for ambient ozone estimation. Intl J Approximate Reasoning 45(1):152–188
14. Kaburlasos VG, Petridis V (2000) Fuzzy lattice neurocomputing (FLN) models. Neural Networks 13(10):1145–1170
15. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proc IEEE Intl Conf Neural Networks IV:1942–1948
16. Mitchell TM (1997) Machine Learning. McGraw-Hill, New York, NY
17. Petridis V, Kaburlasos VG (1998) Fuzzy lattice neural network (FLNN): a hybrid model for learning. IEEE Trans Neural Networks 9(5):877–890
18. Petridis V, Kaburlasos VG (2001) Clustering and classification in structured data domains using fuzzy lattice neurocomputing (FLN). IEEE Trans Knowledge Data Engineering 13(2):245–260
19. Price K, Storn R, Lampinen J (2006) Differential Evolution — A Practical Approach to Global Optimization. Springer-Verlag, New York, NY
20. Quinlan JR (1992) C4.5: Programs for Machine Learning. Morgan Kaufman, San Mateo, CA
21. Rasmussen E (1992) Clustering algorithms. In: Information Retrieval: Data Structures and Algorithms, Frakes WB, Baeza-Yates R (eds) pp 419-442. Prentice Hall, Englewood Cliffs, NJ
22. Vapnik VN (1998) Statistical Learning Theory. Wiley, Chichester, UK
23. Vose MD (1999) The Simple Genetic Algorithm: Foundations and Theory. MIT Press, Cambridge, MA