# A Fuzzy Classifier Network with Ellipsoidal Epanechnikovs

by

Carl G. Looney*
Computer Science Department/171
University of Nevada
Reno, NV 89557
looney@cs.unr.edu

**Abstract**. *Our new fuzzy classifier network can be trained with a set of labeled feature vectors without adjusting any weights. Further, it checks for consistency of the labels for the various feature vectors. It is a trade-off between two extreme cases. First, if a circular Gaussian is centered on each labeled feature vector, then a large number of Gaussians may be required but the accuracy is high in the absence of outliers. At the other extreme is the use of a single radial Gaussian for each class that is centered on the average vector of the class. This is very efficient, but the class may have a noncircular shape and outliers can bias the location, both of which affect the accuracy. We extend Epanechnikov functions to be multivariate ellipsoidal fuzzy set membership functions. To develop methods that are efficient and yet accurate, we center an ellipsoidal Gaussian or Epanechnikov on the weighted fuzzy average vector of each class. These fuzzy centers are immune to outliers. The ellipsoidal shapes are rotated, dilated and contracted along the principal axes by the inverse covariance matrices for the respective classes. The inverses of these symmetric matrices are computed efficiently, which completes the training. A test vector to be classified is put through all ellipsoidal fuzzy set membership functions for the respective classes and the maximum value (fuzzy truth) determines the winner. Tests are done on the iris and other data sets.*

**Keywords**. fuzzy classifiers, networks, Gaussians, Epanechnikovs, ellipsoids, covariance

---

* This work was supported by Army Research Office Contract No. DAAD19-99-1-0089

# A Fuzzy Classifier Network with Ellipsoidal Epanechnikovs

by

Carl G. Looney
Computer Science Department/171
University of Nevada
Reno, NV 89557
U.S.A.

## 1. Introduction

Classification methods include linear discriminant and statistical methods (see [10]), multiple layered perceptrons (see [19]), crisp clustering (see [10]), radial basis function neural networks (see [4, 7, 17, 21]), probabilistic neural networks (see [2, 8, 25, 26, 28]), crisp and fuzzy rule-based systems [13, 15, 29], crisp and fuzzy competitive learning algorithms [9], syntactic methods (e.g., see [19]), nearest neighbor and k-nearest neighbors (see [10]), and genetic algorithms [11, 12, 24], among others [6, 14].

Much recent research has been in the areas of radial basis function neural networks (RBFNNs) [17] and fuzzy classifiers [1, 20, 27, 29]. The RBFNNs are in fact fuzzy classifiers when the RBFs are considered as fuzzy set membership functions with the output weighted sums of fuzzy truths being a form of defuzzification. The Gaussians here are *radial* Gaussians in that they have the same functional value for all vectors that are the same distance r from the Gaussian center (a *radial basis function*). Ellipsoidal functions are discussed in [17] for neural networks and used in [1] for fuzzy classification.

We show that probabilistic neural networks (PNNs) are also fuzzy classifiers when the mixed Gaussian probability density functions are considered to be fuzzy set membership functions. Thus fuzzy classifiers are a rather general paradigm. It is natural to attempt to combine various models into a general fuzzy classifier to find general properties, and to seek to extract a simple model that is efficient and effective. Such is the purpose here.

Section 2 describes the concept of a generalized fuzzy classifier with basic properties while leaving specific output functions to take various forms within the general setting. PNNs are shown here to be fuzzy classifiers when we reformulate them as fuzzy neural networks. This section also describes the extreme cases of: i) using every training (labeled) feature vector as the center of a radial Gaussian; or ii) using a single radial Gaussian centered on the average vector of each class. Section 3 presents a simple and efficient network model that employs for each class an ellipsoidal Gaussian with an inverse covariance matrix. It is centered on a robust fuzzy average vector for that class. It also extends Epanechnikov functions to ellipsoids that uses the inverse covariance matrix. Our approach uses special fuzzy centers that are immune to outliers and the extended Epanechnikovs that are more efficient to compute than Gaussians.

Section 4 lists the results of computer runs on synthetic data designed to test the methodology, on the difficult iris data and on a subset of the noisy Wisconsin breast cancer data. It makes some comparisons and shows graphs to illustrate our simple new ellipsoidal fuzzy classifier networks. The fifth section makes some analysis and conclusions about the new fuzzy classifiers.

## 2. Generalized Fuzzy Classifier Networks

**2.1 The General Architecture**. A *generalized fuzzy classifier network* (GFCN) is a network paradigm whose architecture has N input nodes at the *input layer* for inputting an N-dimensional feature vector $\mathbf{x} = (x_1,...,x_N)$. The GFCN has no hidden layer. There are K *decision* nodes in the *output layer* that represent the K classes. Each input node connects to all K decision nodes, so that each k-th decision node receives an entire feature vector and processes it to make a decision as to the certainty of its membership in the k-th class. The output of the k-th decision node is the fuzzy truth of the input vector's membership in the k-th class. The decision making process in each decision node is not defined in a general sense because it may take various forms.

Figure 1 displays the simple GFCN architecture that has two layers and processes N-dimensional feature vectors into K classes. The set of training feature vectors is used by all decision processes. In the simplest cases, when a test vector is input, the process computes the number of nearest neighbors in each class and there is a (possibly weighted) vote for its class of membership. Another case uses a fuzzy set membership function for the k-th class at the k-th output node so that the maximum fuzzy truth is the winner.

[Figure 1 goes here]

**Figure 1. A generalized fuzzy classifier network.**

**2.2 Probabilistic and Fuzzy Neural Network Classifiers**. We can reformulate the PNN [25, 26] (see [2] for an interesting application) into our related *fuzzy neural network* (FNN) model. For illustrative purposes we take K = 2 classes so that there are two decision making nodes. Given the set of labeled training feature vectors $\{\mathbf{x}^{(q)}: q = 1,...,Q\}$ of dimension N we partition the feature vectors into two sets for the respective two classes as designated by the labels. We build a probability density function (pdf) for the first class by centering a radial Gaussian on each training vector in that class. Each has the same standard deviation $\sigma$.

Figure 2 displays the PNN for K = 2 classes. The Gaussian centered on each training feature vector $\mathbf{x}^{(q)}$ in the first group is represented by a node in the hidden layer. The outputs of all of these Gaussians go to the decision node for Class 1 in the output layer, where they are summed by the decision process and scaled to form a *mixed Gaussian* [26, 28] pdf.

[Figure 2 goes here]

**Figure 2. The PNN and FNN architecture.**

The pdf for the second class is defined similarly. The sums for the two class outputs are

$$f_1 = [1/((2\kappa)^{N/2}\sigma^N)]\kappa_{u=1,P} \exp[-(\mathbf{x} - \mathbf{x}^{(u)})^t(\mathbf{x} - \mathbf{x}^{(u)})/(2\sigma^2)] \quad (1a)$$

$$f_2 = [1/((2\kappa)^{N/2}\sigma^N)]\kappa_{v=1,Q} \exp[-(\mathbf{x} - \mathbf{x}^{(v)})^t(\mathbf{x} - \mathbf{x}^{(v)})/(2\sigma^2)] \quad (1b)$$

$$\exp[-(\mathbf{x} - \mathbf{x}^{(u)})^t(\mathbf{x} - \mathbf{x}^{(u)})/(2\sigma^2)] = \exp[-\|\mathbf{x} - \mathbf{x}^{(u)}\|/(2\sigma^2)] \quad (2)$$

The P vectors in $\{\mathbf{x}^{(u)}\}$ in Eqn. (1a) are training vectors in Class 1 and the Q vectors in $\{\mathbf{x}^{(v)}\}$ in Eqn. (1b) are Class 2. All vectors are N-dimensional. While $\mathbf{x}^{(u)}$ could be the center of a subclass of Class 1, it could also

be the single center of that entire class. The PNN is completely trained upon defining these functions. When any test vector **x** is put into the input nodes for classification, it feeds into each Gaussian represented by a node at the hidden layer. For the general case of K classes there is a group of hidden nodes for each class and the output lines from each group feed only to the decision node for the particular class where they are summed into its output value $f_k$ . There are K pdf values put out from the K decision nodes and each output $f_k$ is a value of the mixed Gaussians (*Parzen window*). The highest value determines the classification. Figure 3 shows the (summed) mixed Gaussians for a class of 2-dimensional feature vectors.

Our FNNis also modeled by Figure 2, but it does not scale the mixed Gaussians because their output values are fuzzy truths.The summed outputs could exceed unity, however, which would violate the axioms of fuzzy logic. Therefore we standardized the outputs by dividing them by the sum of all outputs.

[Figure 3 goes here]

**Figure 3. A mixed Gaussian function for 2-D vectors.**

Rule-based fuzzy classifiers use the N feature values of an input feature vector $\mathbf{x} = (x_1,...,x_N)$ coming into the k-th decision node to activate fuzzy set membership functions that make up rule antecedents. Each feature $x_n$ is put through a fuzzy set membership function (FSMF) for an attribute $F_{n,k}$ of the feature to belong to Class k. The fuzzy rules take the following form.

IF (($x_1$ is $F_{1,k}$ ) AND ... AND ($x_N$ is $F_{N,k}$)) THEN (**x** is CLASS_k)

The minimum fuzzy truth of all conjuncted antecedents (to the left of "THEN") in a rule is transferred to the consequent part on the right-hand side. If more than one rule has the same consequent (**x** is CLASS_k), then the fuzzy truth of the consequent is the maximum of the fuzzy truths imparted to it by rules. If rules impart fuzzy truths to more than one consequent (to more than one class), the consequent with the maximum fuzzy truth designates the class.

Rule-based systems make decisions based on Cartesian products of the FSMFs for the features, as shown in Figure 4 for two features. This can be avoided with extra rules, but the number of rules may become quite large, which introduces the possibility of inconsistency among them. Radial basis functions restrict rules to circular disk regions of the feature space and so RBFNNs are equivalent to certain fuzzy rule-based systems. RBFNNs can also use ellipsoidal basis functions (see [17]).

[Figure 4 goes here]

**Figure 4. Cartesian products of fuzzy set membership functions.**

**2.3 How Many Gaussians?** Given a set of feature vectors, a clustering process can be used to yield a set of classes from which to label the vectors. These can first be labeled with target outputs to designate the classes and then used to design a fuzzy classifier. For each class we build a fuzzy set membership function in some way to represent that class. The two extreme cases for building such functions are: i) use every feature vector in a class as a center of a Gaussian and sum the Gaussians; ii) use a single prototype (the average in some sense) of each class on which to center a Gaussian.

The first case can be very accurate if the noise level is low and there are no outliers, but even so, it may require hundreds of Gaussians to be evaluated and summed for classification of a single input feature vector. It can also be adversely affected by outliers that become the centers of Gaussians and then incorrectly classify. The other extreme case is very fast but can not yield the high accuracy of the first extreme case, although it is less affected by noise by virtue of the smoothing in the averaging. But it suffers from inaccuracy when the class does not have a circular shape or when outliers are present.

Table 1 shows some concepts between the two extremes. *Concept 1* uses mixed Gaussians but is not always the most accurate [28] when the number of Gaussians becomes large. We expect that more training vectors leads to more outliers and hence more incorrectly placed radial Gaussians. *Concept 6* is the fastest but it is the least accurate. In the next section we develop *Concept 5a* that is similar to *Concept 5* in Table 1, but which is immune to outliers and thus more accurate while retaining the advantage of speed. We then employ Epanechnikov functions, which are bell shaped functions that are much faster to compute than Gaussians. We extend these to ellipsoidal FSMFs to cover noncircular classes and call this *Concept 5b*.

**Table 1. The Extreme and Intermediate Concepts for Classification.**

| **Concept** | **Speed** | **Accuracy** |
|---|---|---|
| 1. A radial Gaussian centered on every training vector | Slowest | High in absence of outliers |
| 2. A radial Gaussian centered on training vectors that are not close together (prune some of the feature vectors that are too close to any others) | Fast | High in absence of outliers |
| 3. A radial Gaussian centered on the center of each of a number of subclasses where the subclasses are associated with their class | Fast | High in absence of outliers |
| 4. Ellipsoidal Gaussians centered on multiple feature vectors in each class as in 2 or 3 above (compute covariance matrices from the sample) | Fast | Very high in absence of outliers |
| 5. A single ellipsoidal Gaussian centered on each class with sample inverse covariance matrix | Very Fast | High without outliers |
| 6. A single radial Gaussian centered on each class with sample variance | Fastest | Lowest |

Figure 5 shows two classes of noncircular shapes formed by putting a radial Gaussian on each training feature vector. The $\sigma$ values shown vary according to half the distance to the nearest other feature vector for sufficient overlapping. The Gaussians can be summed as pdf's or used as fuzzy set membership functions for the classes. Figure 6 shows three classes where each can be contained under a single ellipsoidal Gaussian (Concept 5 of Table 1). Concept 4 uses multiple ellipsoids for each class, which can be done by means of centers of subclasses, but each ellipsoid requires an inverse covariance matrix.

[Figure 5 goes here]

**Figure 5. Two classes formed by multiple radial Gaussians.**

[Figure 6 goes here]

**Figure 6. Three classes containable under ellipsoidal Gaussians.**

## 3. The New Ellipsoidal Fuzzy Classifier Network

**3.1 The Modified Weighted Fuzzy Expected Value**. Our first major improvement is on *Concept 5* with the result being *Concept 5a*. It uses a single ellipsoidal Gaussian for each class, but is made immune to outliers via a special fuzzy center that is not the average nor the median. We first planned to employ an $\kappa$-trimmed mean [5] which would help, but then we found a type of center that is a more typical value in that it is located among the more densely situated training feature vectors. Thus we select our *modified weighted fuzzy expected value* (MWFEV) [18] that is an improved version of the original weighted fuzzy expected value [23]. Our Gaussian is a canonical fuzzy set membership function whereas the original two-sided decaying exponential is not.

Figure 7 compares the average, the median and the MWFEV on a simple data set in two dimensions. There are 5 vectors designated by solid dots, of which one is an outlier (the bottom right). The location of the MWFEV is inside the more densely situated vectors, while the mean and median are not.

[Figure 7 goes here]

**Figure 7. The MWFEV versus the mean and median.**

To obtain the MWFEV of a set of vectors, we must find the MWFEV componentwise of the vectors. Thus it suffices to show the algorithm for a set of real values $\{x_1,...,x_P\}$ that are a single component of the training vectors. We assume a Gaussian fuzzy set membership function that is centered on the hypothetical, but unknown, MWFEV $\kappa$. This FSMF represents the linguistic variable "*CLOSE_TO_CENTER*" that gives a fuzzy truth (weight) to each value in the set that is close to the fuzzy center. Starting from the arithmetic mean as an initial approximation, we use Picard iterations that converge rapidly (4 or 5 iterations are usually sufficient).

---

Step 1. Compute arithmetic mean $\kappa^{(0)}$ of $x_1,...,x_P$ as initial approximation, compute $\sigma$, put r = 0

Step 2. Compute preliminary and standardized weights on each r-th iteration

$$W_p = \exp[-(x_p - \kappa^{(r)})^2 /(2\sigma^2)], \quad p = 1,...,P \tag{3}$$

$$w_p = W_p /(W_1 + ... + W_P), \quad p = 1,...,P \tag{4}$$

Step 3. Compute the (r+1)-st MWFEV approximation

$$\kappa^{(r+1)} = \mathbf{K}_{(p=1,P)}\, w_p x_p \tag{5}$$

Step 4. If stop_criterion is true then stop, else increment r and go to Step 2

---

**3.2 Concept 5a: Single Ellipsoidal Gaussians with Noise Immunity**. Given a set of Q labeled training feature vectors $\{\mathbf{x}^{(q)}: q = 1,...,Q\}$, where $\mathbf{x}^{(q)} = (x_1^{(q)},...,x_N^{(q)})$, we set up a trained fuzzy classifier network and then use it to recognize any test feature vector **x**. The ellipsoidal multivariate Gaussian FSMF centered on $\mathbf{x}^{(k)}$ takes the form

$$f_k = \exp[-(1/2)(\mathbf{x} - \mathbf{x}^{(k)})^t C^{-1}(\mathbf{x} - \mathbf{x}^{(k)})] \quad (6)$$

where $\mathbf{x}^{(k)}$ is the center of Class k (refer to Equation 1). We drop the scaling $[1/((2\kappa)^{N/2}|C|^{1/2})]$ that makes it a pdf so it becomes an FSMF. The following steps give a high level description of the algorithm.

---

Step 1. Use the labels to separate the training vectors into K sets (one for each class)

Step 2. Compute the MWFEV for each of the K sets

Step 3. Compute the sample NxN covariance matrix C for each of the K sets and compute the inverse matrices $C^{-1}$

Step 4. Define an ellipsoidal Gaussian fuzzy set membership function centered on the MWFEV of each set [this completes the training]

Step 5. For any input vector **x**, put **x** through each ellipsoidal Gaussian, standardize the outputs and then determine the class number for the maximum value

---

Figure 8 presents ellipsoidal Gaussians for two classes of 2-dimensional feature vectors. The elliptical axes are rotated to the principal components for the functions and these have different principal components.

[Figure 8 goes here]

**Figure 8. Ellipsoidal Gaussians for 2-D classes**.

**3.3 Concept 5b: Single Ellipsoidal Epanechnikov Functions**. Gaussians approach zero away from their centers, but never become zero. An *Epanechnikov* function is zero outside of a closed ball and yet has a bell shape on the ball (but without the flare-out at the bottom). These are defined as functions on the reals [22].

$$E(r) = \max\{1 - r^2, 0\} \quad (7a)$$

At r = 0, E(r) takes its maximum value of 1, but as r goes to 1, the function value goes to 0 and remains 0 for all r such that |r| > 1. To use these *circular* Epanechnikov functions on normed vector spaces, we modify them to be

$$E(\mathbf{x}) = \max\{1 - (1/2)\|\mathbf{x} - \mathbf{c}\|^2, 0\} \quad (7b)$$

where **c** is the center of the unit ball. Because we need balls of arbitrary sizes, we use the function

$$E(\mathbf{x}) = \max\{1 - [1/(2\kappa^2)]\|\mathbf{x} - \mathbf{c}\|^2, 0\} \quad (7c)$$

To suit our needs here we extend the Epanechnikov functions to be nonzero on ellipsoidal balls rather than only on circular balls. We do this as is shown in the next equation by using the *Mahalanobis* distance (see [19]) in place of the Euclidean distance.

$$E(\mathbf{x}) = \max \{1 - (1/2)(\mathbf{x} - \mathbf{c})^t C^{-1}(\mathbf{x} - \mathbf{c}),\ 0\} \tag{8a}$$

Here C is the covariance matrix whose inverse rotates the ellipse to the axes of its principal components. These functions are computed much more quickly than are Gaussians because the exponential $e^x$ requires an approximation by a cubic polynomial divided by a cubic polynomial for $0 < x < 0.5$ and multiplication of the parts. For example

$$\exp[-3.62] = 1 / \{(e)(e)(e)(\exp[0.5])(\exp[0.12])\}$$

where e is a stored value, exp[0.5] is the stored square root of e and the rightmost part is approximated by the cubic-cubic rational function. With hundreds or possibly thousands Gaussians being computed, this slows the process.

Figure 9 displays two *extended ellipsoidal Epanechnikov* functions rotated on the major and minor axes. Their centers $\mathbf{c}^{(1)}$ and $\mathbf{c}^{(2)}$ are the MWFEVs of the classes that make these FSMFs immune to outliers. Thus we should gain accuracy as well as speed over previous circular functions. Figure 10 defines our *special extended ellipsoidal Epanechnikovs* on the same data where the argument contains a squared part as shown below.

$$E(\mathbf{x}) = \max \{1 - (1/2)[(\mathbf{x} - \mathbf{c})^t C^{-1}(\mathbf{x} - \mathbf{c})]^2,\ 0\} \tag{8b}$$

[Figure 9 goes here]

**Figure 9. Extended Epanechnikov functions on the plane**.

[Figure 10 goes here]

**Figure 10. Special extended Epanechnikovs on the plane.**

During the computer runs we discovered that the Epanechnikovs require a *covariance factor* $\kappa^2$ that multiplies the covariance matrix to increase the magnitudes, or equivalently, $1/\kappa^2$ multiplies the inverse covariance matrix. This is because of the steep drop off of these functions and also because the covariance values are averages so that some vectors will be located outside of the ellipsoidal boundary and thus will evaluate to zero with an Epanechnikov but not with a Gaussian. The factor $1/\kappa^2 < 1.0$ compensates for this by enlargening the ellipsoidal ball. Equations (8a,b) then become

$$E(\mathbf{x}) = \max \{1 - [1/(2\kappa^2)][(\mathbf{x} - \mathbf{c})^t C^{-1}(\mathbf{x} - \mathbf{c})],\ 0\} \tag{8c}$$

$$E(\mathbf{x}) = \max \{1 - [1/(2\kappa^2)][(\mathbf{x} - \mathbf{c})^t C^{-1}(\mathbf{x} - \mathbf{c})]^2,\ 0\} \tag{8d}$$

## 4. Classification Runs and Results

**4.1 Training**. We use all of the feature vectors in a class to determine the class prototype and covariance matrix and then use the same feature vectors to test. It is not the individual vectors that are so important here but their aggregate properties (rotation of axes, widths along each axis, shape, etc.). If we are using extended Epanechnikovs then we must test with the feature vectors and adjust the covariance factor until all classes are learned properly. After training, tests must be made to insure that all good feature vectors can be classified correctly, although far outliers or mislabeled vectors will not be. This acts as a consistency check on the training feature vectors.

**4.2 The Iris Data Set**. This famous test data set contains 150 feature vectors that are 4-dimensional to represent the 4 features petal width, petal length, sepal width and sepal length. These vectors were labeled by Anderson [3] into 3 classes, or species (*Sestosa*, *Versicolor* and *Virginicus*) of 50 each, but they are known to be noisy and nonseparable [16]. To demonstrate this, Figure 11 shows the MWFEV of each feature for each class. The actual feature values in the data set force us to conclude that a very small number of the vectors were misclassified by Anderson, which is the usual case when humans estimate memberships of classes instead of using a rigorous clustering algorithm. If we were training a neural network by adjusting weights to minimize the output mean-square error, then the incorrectly labeled vectors would be learned by the network and it would therefore be faulty.

The analysis of Figure 11 shows that features 3 and 4 are the best separators, while feature 2 does not separate sufficiently strongly to overcome a moderate signal-to-noise ratio for the observations. The iris data set is appended to this paper (it also contains 2-dimensional labels for the classes that are robust in that the Hamming distance between the different labels is 2 for each pair of labels). These labels are for training neural networks.

[Figure 11 goes here]

**Figure 11. The MWFEV centers of the four iris features**.

**4.3 Results on the Iris Data**. We use all 150 feature vectors to train via Concept 5a (Gaussian ellipsoids on MWFEV centers) and also via Concept 5b (extended Epanechnikovs with MWFEV centers). The algorithm for Concept 5b is similar to that for Concept 5a and so is not repeated here, but considerable compensation was needed due to the sharp drop off of Epanechnikovs near the boundary of the ellipsoidal supporting ball. Using the sample covariance, the bell cut off the outlying points in each class, so we adjusted the inverse covariance matrix by multiplying it by a scalar less than 1.0.

Table 2 shows the results of running the algorithms on the individual iris feature vectors after defining the functions on their aggregate shape properties. The vectors are numbered from 1 to 150. The ones that were misclassified (found to be inconsistent) were examined to see how well they fit the MWFEV centers of their classes. As shown in Table 2, feature vectors numbered 62, 68 and 101 should be labeled as Class 3 rather than Class 2.

The bottom part of Table 2 shows the results for extended Epanechnikovs. On the first run three vectors were not classified because they were outside of the ellipsoidal ball where the functions were greater than 0. When $\kappa$ was increased to 4 the results were the same as for Gaussians. Thus Epanechnikovs can perform like

Gaussians but require less computation and are zero except on a finite region. We used Equation (8c) because it declines more rapidly and can differentiate better between values of overlapping functions on the same vector.

**Table 2. Iris Data: Ellipsoidal Gaussians and Epanechnikovs with MWFEVs.**

| Algorithm | Class 1 | Class 2 | Class 3 | Notes |
|---|---|---|---|---|
| *Concept 5a* (ellipsoidal Gaussians) | 50 | 47 | 53 | 3 vectors (62, 68, 101) were labeled as Class 2 but classified as Class 3 |
| Inconsistent Vector No. | | | | Feature Values |
| 62 | | | | 5.90 3.20 4.80 1.80 |
| 68 | | | | 6.30 2.50 4.90 1.50 |
| 101 | | | | 6.00 2.70 5.10 1.60 |
| *Concept 5b* (ellips. Epanechnikovs) | 49* 50† | 46* 47† | 52* 53† | 3 vectors (62, 68, 101) were labeled as Class 2 but classified as Class 3. On the first run, one from each of Classes 1, 2 and 3 were not classified because they were outside of all ellipsoidal balls, but were correctly classified when the covariance factor $\kappa$ was increased to $\sqrt{20}$ on the second run |
| Inconsistent Vector No. (same as above) | | | | |

* first run † second run: adjusted $\kappa$

**4.3 Results on the Synthetic Data**. Table 3 shows the synthetic data that we manufactured to form two ellipsoids to determine how well the Gaussian and Epanechnikov ellipsoidal functions would fit them with appropriate rotations. Figures 8, 9 and 10 used this data. The classes are well separated and the ellipsoids contained them very well when the covariance factor was increased for the Epanechnikovs. Table 4 shows the results for both types of functions, where all of the Epanechnikovs use the form of Equation (8c).

**Table 3. Synthetic Data to Show Ellispoidal Rotation.**

| **Class 1** | | | **Class 2** | | |
|---|---|---|---|---|---|
| Features 1, 2 | | Label | Features 1, 2 | | Label |
| 0.1, | 0.5 | 0 | 0.5, | 0.5 | 1 |
| 0.1, | 0.4 | 0 | 0.6, | 0.5 | 1 |
| 0.1, | 0.3 | 0 | 0.6, | 0.6 | 1 |
| 0.2, | 0.3 | 0 | 0.6, | 0.7 | 1 |
| 0.2, | 0.2 | 0 | 0.7, | 0.7 | 1 |
| 0.2, | 0.4 | 0 | 0.7, | 0.8 | 1 |
| 0.3, | 0.3 | 0 | 0.8, | 0.6 | 1 |
| 0.3, | 0.4 | 0 | 0.8, | 0.7 | 1 |
| 0.3, | 0.2 | 0 | 0.8, | 0.8 | 1 |
| 0.3, | 0.1 | 0 | 0.9, | 0.9 | 1 |
| 0.4, | 0.2 | 0 | | | |
| 0.4, | 0.1 | 0 | | | |
| 0.5, | 0.1 | 0 | | | |

The Epanechnikovs needed a small covariance factor to learn the synthetic data, which is a small data set that is well separated and noise free. It learned well with $\kappa = \sqrt{5}$. However, the iris and Wisconsin breast cancer real world data required much larger values.

**Table 4. Results of Training on the Synthetic Data.**

| Algorithm | Class 1 | Class 2 | Notes |
|---|---|---|---|
| Concept 5a (ellipsoidal Gaussians) | 13 | 10 | All labeled vectors were correctly classified after defining the Gaussians |
| Concept 5b (ellipsoidal Epanechnikovs) | 13 | 10 | All labeled vectors were correctly classified after the Epanechnikovs were defined with adjustments to $\kappa = \sqrt{5}$ |

**4.4 Results on the Wisconsin Breast Cancer Data**. This data has 30 features that are quantified questionnaires completed from patient records. We selected 200 feature vectors at random, of which 121 were labeled as Class 1 and the remaining 79 were labeled as Class 2. Due to the nature of the data, the noise in the data has significant power (mean-square error). Thus this data is a difficult test. Each feature was standardized (normalized) before classification so that each feature has the same importance because some of the feature values are relatively small compared to the others (we don't have a priori knowledge of which are the most effective). The results are given in Table 4.

**Table 4. The Results of Ellipsoidal Tranining on the WBCD.**

| **Algorithm** | **Labeled Classes** | **Classified Classes** | **Inconsistent Vectors** |
|---|---|---|---|
| Ellips. Gaussians | 121, 79 | 106, 94 = 79 + 15* | 4, 24, 41, 57, 75, 78, 87, 88, 90, 98, 104, 105, 110, 113, 116 |
| Ellips. Epanechnikovs | 121, 79 | 109, 91<br>106, 94<br>106, 94 = 79 + 15* | Some vectors were out of the ellipsoids on first 2 runs; all were inside on last run ($\kappa = 8$) |

* all 79 vectors labeled in second class were consistent, 15 vectors from first class were inconsistent

## 5. Analysis and Conclusions

While neural networks can be trained to learn a set of labeled feature vectors, they necessarily learn the noise and incorrect labels. They have no way to check the consistency of the classes. Our new fuzzy classifiers are not only efficient and accurate, but they have a built-in mechanism that enforces consistency. They are immune to outliers because outliers affect the prototypes negligibly. Once the fuzzy centers are found and the fuzzy set membership functions defined, any inconsistent data in the training feature vectors are discovered by testing the training vectors.

In tests on the famous iris data, which neural networks can learn exactly as labeled, our FNNs that used Gaussians and Epanechnikovs both show that three feature vectors (out of 150) are not consistent and are mislabeled. We expect that there will remain controversy over the iris data, but we suggest that anyone with

doubts throw out these outliers, retrain and then reclassify these outliers to see where they belong. It should be noted that unlabeled data can be clustered with an effective clustering algorithm to assign labels and then put through our fuzzy classifier, which will then discover any inconsistencies.

We found that the Gaussians are very robust, but the Epanechnikovs require a sufficiently large covariance factor. It does not seem to matter if that factor is considerably larger than the minimum magnitude necessary to learn, but if it is less then there are some vectors outside of the functions so that all of their fuzzy truths of membership are zero.

## References


[1] S. Abe and R. Thawonmas, "A fuzzy classifier with ellipsoidal regions," *IEEE Trans. Fuzzy Systems*, vol. 5, no. 2, 358-368, 1997.

[2] C. Anagnostopoulos, J. Anagnostopoulos, D. Vergados, E. Kayafas, V. Loumos and G. Stassinopoulos, "A neural network and fuzzy logic system for face detection on RGB images," *Proc. ISCA Int. Conf. Computers and Their Applications*, 233-236, 2001.

[3] E. Anderson, "The iris of the Gaspe peninsula," *Bulletin of the American Iris Society*, vol. 59, 2-5, 1935.

[4] F. Anouar, F. Badran and S. Thira, "Probabilistic self-organizing map and radial basis function networks," *Neurocomputing* **20**, 83-96, 1998.

[5] J. B. Bedner and T. L. Watt, "Alpha-trimmed means and their relation to median filters," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 32, no. 1, 145-153, 1984.

[6] G. Bortolan and W. Pedrycz, "Fuzzy clustering preprocessor in neural classifiers," *Kybernetes*, 900-918, 1998.

[7] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, 321-355, 1988.

[8] K. Chen and H. Chi, "A method of combining multiple probabilistic classifiers through soft competition on different feature sets, *Neurocomputing* **20**, 227-252, 1998.

[9] F. L. Chung and T. Lee, "Fuzzy competive learning," *Neural Networks*, vol. 7, no. 3, 539-551, 1994.

[10] R. O. Duda, P. E. Hart and David Stork, "Pattern Classification," 2nd Edition, Wiley-Interscience, New York, 2001.

[11] S. Endoh, H. Kawakami and A. Ohuchi, "Multiple hierarchical classifier system with self GA invocation," *Nonlinear Analysis, Theory, Methods, and Applications*, vol. 30, no. 5, 3001-3010, 1997.

[12] H. Ishibuchi, K. Nozaki, N. Yamamoto and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Systems*, vol. 3, no. 3, 260-270, 1995.

[13] I. Y. Iwakoshi, T. Furuhashi and Y. Uchikawa,, "A fuzzy classifier system for evolutionary learning of robot behaviors," *Applied Math. And Computation*, vol. 91, no. 1, 73-81, 1998.

[14] N. S. Iyer, A. Kandel and M. Schneider, "Feature-based fuzzy classification for interpretation of mammograms," *Fuzzy Sets and Systems*, vol. 114, 271-280, 2000.

[15] M. A. Kbir, H. Benkirane, K. Maalmi and R. Benslimane, "Hierarchical fuzzy partition for pattern classification with fuzzy if-then rules," *Pattern Recognition Letters*, vol. 21, 503-509, 2000.

[16] J. M. Keller, M. R. Gray and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE Trans. Systems, Man and Cybernetics*, vol. 15, no. 4, 580-585, 1985.

[17] Carl G. Looney, "Radial basis functional link nets and fuzzy reasoning," (in press) *Neurocomputing*.

[18] C. G. Looney and Y. L. Varol, "Fuzzy clustering for associating and fusion in multitarget tracking with multisensors," *Proc. ISIF Fusion'99*, vol. I, 255-261, July 1999.

[19] Carl G. Looney, *Pattern Recognition Using Neural Networks*, Oxford University Press, New York, 1997.

[20] H. Maturino-Lozoya, D. Munoz-Rodriguez, F. Jaimes-Romero and H. Tawfik, "Handoff algorithms based on fuzzy classifiers," *IEEE Trans. Vehicular Technology*, vol. 49, no. 6, 2286-2294, 2000.

[21] J. E. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, 281-294, 1989.

[22] I. Rojas, H. Pomares, J. Gonzales, J. L. Bernier, E. Ros, F. J. Pelayo and A. Prieto, "Analysis of the functional block involved in the design of radial basis function networks," *Neural Processing Letters*, vol. 12, 1-17, 2000.

[23] M. Schneider and M. Craig, "On the use of fuzzy sets in histogram equalization," *Fuzzy Sets & Systems*, vol. 45, 271-278, 1992.

[24] S. Sette and L. Boullart, "An implementation of genetic algorithms for rule based machine learning," *Engineering Applications of Artificial Intelligence*, vol. 13, no. 4, 381-390, 2000.

[25] D. F. Specht, "Probabilistic neural networks for classification, mapping or associative memory," *Proc. IEEE Int. Conf. Neural Networks*, vol. 1, 525-532, 1988.

[26] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 1, no. 3, 109-118, 1990.

[27] Manuel Valenzuela-Rendon, "Reinforcement learning in the fuzzy classifier system," *Expert Systems with Applications*, vol. 14, no. 1-2, 237-247, 1998.

[28] D. K. Wedding II and K. J. Cios, "Certainty factors versus Parzen windows as reliability measures in RBF networks," *Neurocomputing* **19**, 151-165, 1998.

[29] C.-C. Wong, C.-C. Chen and S.-L. Yeh, "K-means-based fuzzy classifier design," *IEEE Int. Conf. Fuzzy Systems*, vol. 1, 48-52, 2000.

**Appendix A. The 150* Iris Feature Vectors with 2-D Labels.**

| Vector | Label | Vector | Label | Vector | Label | Vector | Label |
|---|---|---|---|---|---|---|---|
| 5.1 3.5 1.4 0.2 | -1 0 | 7.0 3.2 4.7 1.4 | 0 1 | 6.3 3.3 6.0 2.5 | 1 -1 | 4.9 3.0 1.4 0.2 | -1 0 |
| 6.4 3.2 4.5 1.5 | 0 1 | 5.8 2.7 5.1 1.9 | 1 -1 | 4.7 3.2 1.3 0.2 | -1 0 | 6.9 3.1 4.9 1.5 | 0 1 |
| 7.1 3.0 5.9 2.1 | 1 -1 | 4.6 3.1 1.5 0.2 | -1 0 | 5.5 2.3 4.0 1.3 | 0 1 | 6.3 2.9 5.6 1.8 | 1 -1 |
| 5.0 3.6 1.4 0.2 | -1 0 | 6.5 2.8 4.6 1.5 | 0 1 | 6.5 3.0 5.8 2.2 | 1 -1 | 5.4 3.9 1.7 0.4 | -1 0 |
| 5.7 2.8 4.5 1.3 | 0 1 | 7.6 3.0 6.6 2.1 | 1 -1 | 4.6 3.4 1.4 0.3 | -1 0 | 6.3 3.3 4.7 1.6 | 0 1 |
| 4.9 2.5 4.5 1.7 | 1 -1 | 5.0 3.4 1.5 0.2 | -1 0 | 4.9 2.4 3.3 1.0 | 0 1 | 7.3 2.9 6.3 1.8 | 1 -1 |
| 4.4 2.9 1.4 0.2 | -1 0 | 6.6 2.9 4.6 1.3 | 0 1 | 6.7 2.5 5.8 1.8 | 1 -1 | 4.9 3.1 1.5 0.1 | -1 0 |
| 5.2 2.7 3.9 1.4 | 0 1 | 7.2 3.6 6.1 2.5 | 1 -1 | 5.4 3.7 1.5 0.2 | -1 0 | 5.0 2.0 3.5 1.0 | 0 1 |
| 6.5 3.2 5.1 2.0 | 1 -1 | 4.8 3.4 1.6 0.2 | -1 0 | 5.9 3.0 4.2 1.5 | 0 1 | 6.4 2.7 5.3 1.9 | 1 -1 |
| 4.8 3.0 1.4 0.1 | -1 0 | 6.0 2.2 4.0 1.0 | 0 1 | 6.8 3.0 5.5 2.1 | 1 -1 | 4.3 3.0 1.1 0.1 | -1 0 |
| 6.1 2.9 4.7 1.4 | 0 1 | 5.7 2.5 5.0 2.0 | 1 -1 | 5.8 4.0 1.2 0.2 | -1 0 | 5.6 2.9 3.6 1.3 | 0 1 |
| 5.8 2.8 5.1 2.4 | 1 -1 | 5.7 4.4 1.5 0.4 | -1 0 | 6.7 3.1 4.4 1.4 | 0 1 | 6.4 3.2 5.3 2.3 | 1 -1 |
| 5.4 3.9 1.3 0.4 | -1 0 | 5.6 3.0 4.5 1.5 | 0 1 | 6.5 3.0 5.5 1.8 | 1 -1 | 5.1 3.5 1.4 0.3 | -1 0 |
| 5.8 2.7 4.1 1.0 | 0 1 | 7.7 3.8 6.7 2.2 | 1 -1 | 5.7 3.8 1.7 0.3 | -1 0 | 6.2 2.2 4.5 1.5 | 0 1 |
| 7.7 2.6 6.9 2.3 | 1 -1 | 5.1 3.8 1.5 0.3 | -1 0 | 5.6 2.5 3.9 1.1 | 0 1 | 6.0 2.2 5.0 1.5 | 1 -1 |
| 5.4 3.4 1.7 0.2 | -1 0 | 5.9 3.2 4.8 1.8 | 0 1 | 6.9 3.2 5.7 2.3 | 1 -1 | 5.1 3.7 1.5 0.4 | -1 0 |
| 6.1 2.8 4.0 1.3 | 0 1 | 5.6 2.8 4.9 2.0 | 1 -1 | 4.6 3.6 1.0 0.2 | -1 0 | 6.3 2.5 4.9 1.5 | 0 1 |
| 7.7 2.8 6.7 2.0 | 1 -1 | 5.1 3.3 1.7 0.5 | -1 0 | 6.1 2.8 4.7 1.2 | 0 1 | 6.3 2.7 4.9 1.8 | 1 -1 |
| 4.8 3.4 1.9 0.2 | -1 0 | 6.4 2.9 4.3 1.3 | 0 1 | 6.7 3.3 5.7 2.1 | 1 -1 | 5.0 3.0 1.6 0.2 | -1 0 |
| 6.6 3.0 4.4 1.4 | 0 1 | 7.2 3.2 6.0 1.8 | 1 -1 | 5.0 3.4 1.6 0.4 | -1 0 | 6.8 2.8 4.8 1.4 | 0 1 |
| 6.2 2.8 4.8 1.8 | 1 -1 | 5.2 3.5 1.5 0.2 | -1 0 | 6.7 3.0 5.0 1.7 | 0 1 | 6.1 3.0 4.9 1.8 | 1 -1 |
| 5.2 3.4 1.4 0.2 | -1 0 | 6.0 2.9 4.5 1.5 | 0 1 | 6.4 2.8 5.6 2.1 | 1 -1 | 4.7 3.2 1.6 0.2 | -1 0 |
| 5.7 2.6 3.5 1.0 | 0 1 | 7.2 3.0 5.8 1.6 | 1 -1 | 4.8 3.1 1.6 0.2 | -1 0 | 5.5 2.4 3.8 1.1 | 0 1 |
| 7.4 2.8 6.1 1.9 | 1 -1 | 5.4 3.4 1.5 0.4 | -1 0 | 5.5 2.4 3.7 1.0 | 0 1 | 7.9 3.8 6.4 2.0 | 1 -1 |
| 5.2 4.1 1.5 0.1 | -1 0 | 5.8 2.7 3.9 1.2 | 0 1 | 6.4 2.8 5.6 2.2 | 1 -1 | 5.5 4.2 1.4 0.2 | -1 0 |
| 6.0 2.7 5.1 1.6 | 0 1 | 6.3 2.8 5.1 1.5 | 1 -1 | 4.9 3.1 1.5 0.2 | -1 0 | 5.4 3.0 4.5 1.5 | 0 1 |
| 6.1 2.6 5.6 1.4 | 1 -1 | 5.0 3.2 1.2 0.2 | -1 0 | 6.0 3.4 4.5 1.6 | 0 1 | 7.7 3.0 6.1 2.3 | 1 -1 |
| 5.5 3.5 1.3 0.2 | -1 0 | 6.7 3.1 4.7 1.5 | 0 1 | 6.3 3.4 5.6 2.4 | 1 -1 | 4.9 3.6 1.4 0.1 | -1 0 |
| 6.3 2.3 4.4 1.3 | 0 1 | 6.4 3.1 5.5 1.8 | 1 -1 | 4.4 3.0 1.3 0.2 | -1 0 | 5.6 3.0 4.1 1.3 | 0 1 |
| 6.0 3.0 4.8 1.8 | 1 -1 | 5.1 3.4 1.5 0.2 | -1 0 | 5.5 2.5 4.0 1.3 | 0 1 | 6.9 3.1 5.4 2.1 | 1 -1 |
| 5.0 3.5 1.3 0.3 | -1 0 | 5.5 2.6 4.4 1.2 | 0 1 | 6.7 3.1 5.6 2.4 | 1 -1 | 4.5 2.3 1.3 0.3 | -1 0 |
| 6.1 3.0 4.6 1.4 | 0 1 | 6.9 3.1 5.1 2.3 | 1 -1 | 4.4 3.2 1.3 0.2 | -1 0 | 5.8 2.6 4.0 1.2 | 0 1 |
| 5.8 2.7 5.1 1.9 | 1 -1 | 5.0 3.5 1.6 0.6 | -1 0 | 5.0 2.3 3.3 1.0 | 0 1 | 6.8 3.2 5.9 2.3 | 1 -1 |
| 5.1 3.8 1.9 0.4 | -1 0 | 5.6 2.7 4.2 1.3 | 0 1 | 6.7 3.3 5.7 2.5 | 1 -1 | 4.8 3.0 1.4 0.3 | -1 0 |
| 5.7 3.0 4.2 1.2 | 0 1 | 6.7 3.0 5.2 2.3 | 1 -1 | 5.1 3.8 1.6 0.2 | -1 0 | 5.7 2.9 4.2 1.3 | 0 1 |
| 6.3 2.5 5.0 1.9 | 1 -1 | 4.6 3.2 1.4 0.2 | -1 0 | 6.2 2.9 4.3 1.3 | 0 1 | 6.5 3.0 5.2 2.0 | 1 -1 |
| 5.3 3.7 1.5 0.2 | -1 0 | 5.1 2.5 3.0 1.1 | 0 1 | 6.2 3.4 5.4 2.3 | 1 -1 | 5.0 3.3 1.4 0.2 | -1 0 |
| 5.7 2.8 4.1 1.3 | 0 1 | 5.9 3.0 5.1 1.8 | 1 -1 | | | | |

* vectors are numbered row-wise (e.g., Vectors 1, 2, 3 and 4 are in the first row)

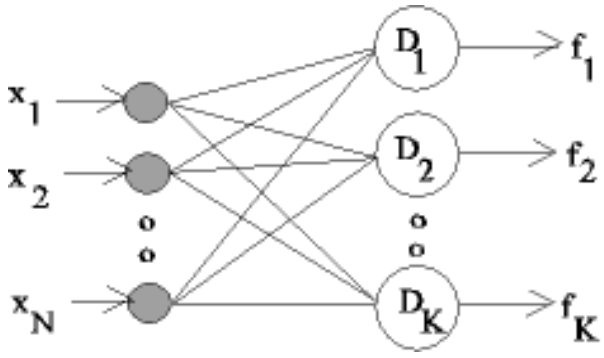


**Figure 1. A generalized fuzzy classifier network.**

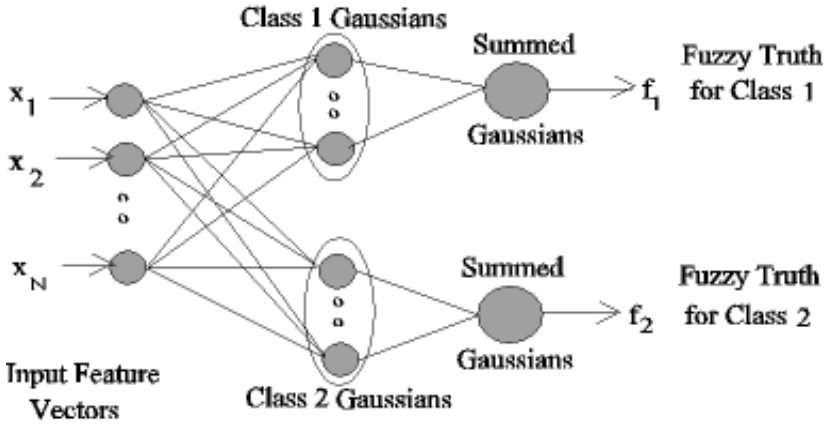


**Figure 2. The PNN and FNN architecture.**

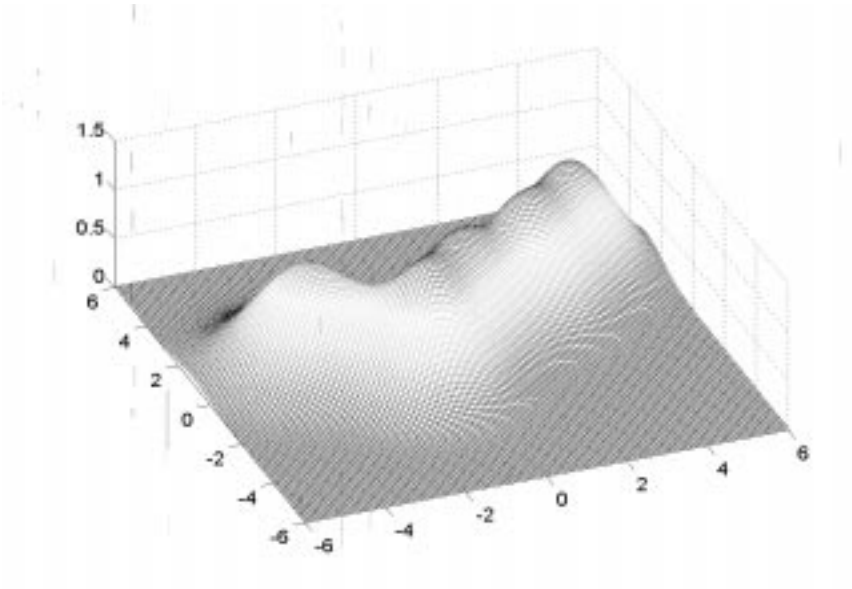


**Figure 3. A mixed Gaussian function for 2-D vectors.**

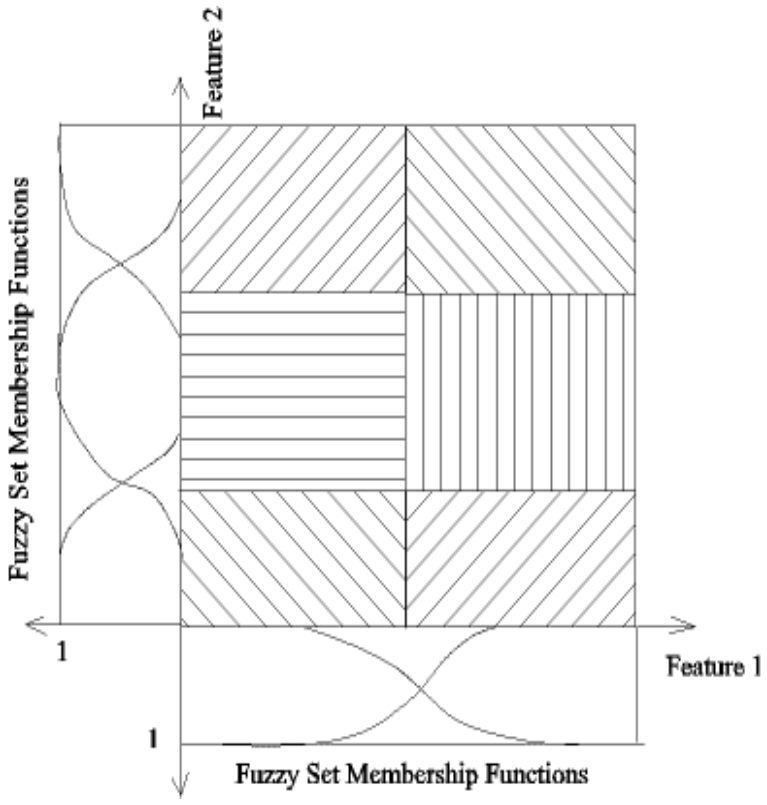


**Figure 4. Cartesian products of fuzzy set membership functions.**

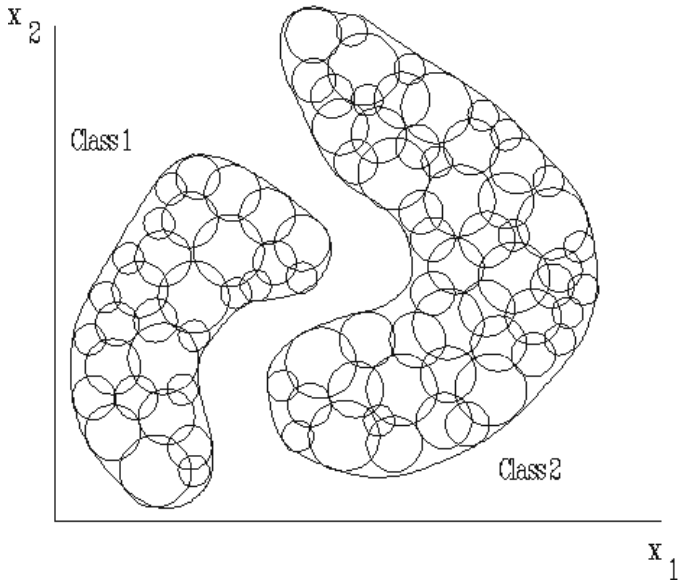


**Figure 5. Two classes formed by multiple radial Gaussian functions.**

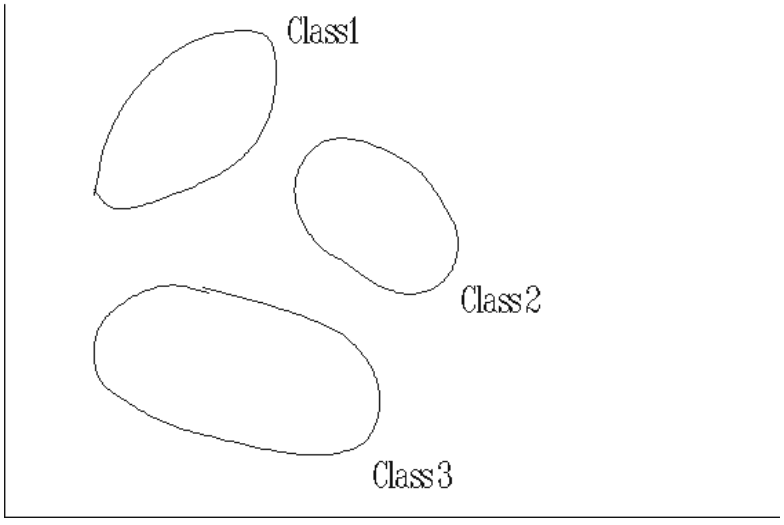


**Figure 6. Three classes containable in single ellipsoidal Gaussian functions.**

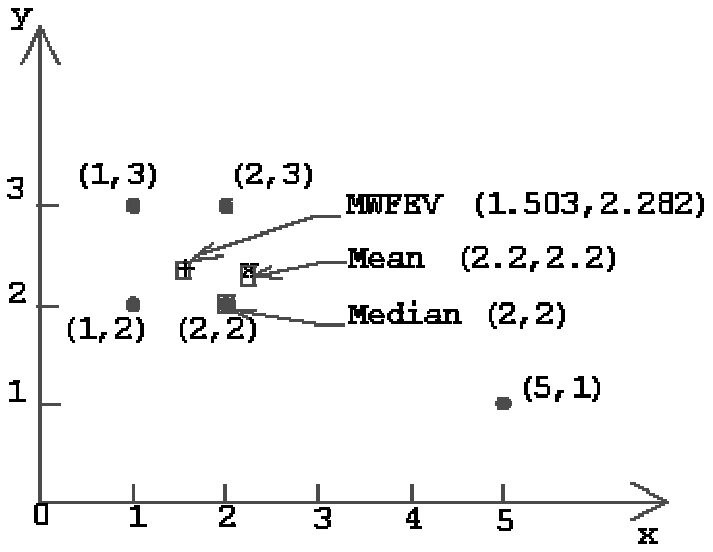


**Figure 7. The MWFEV versus the mean and median.**

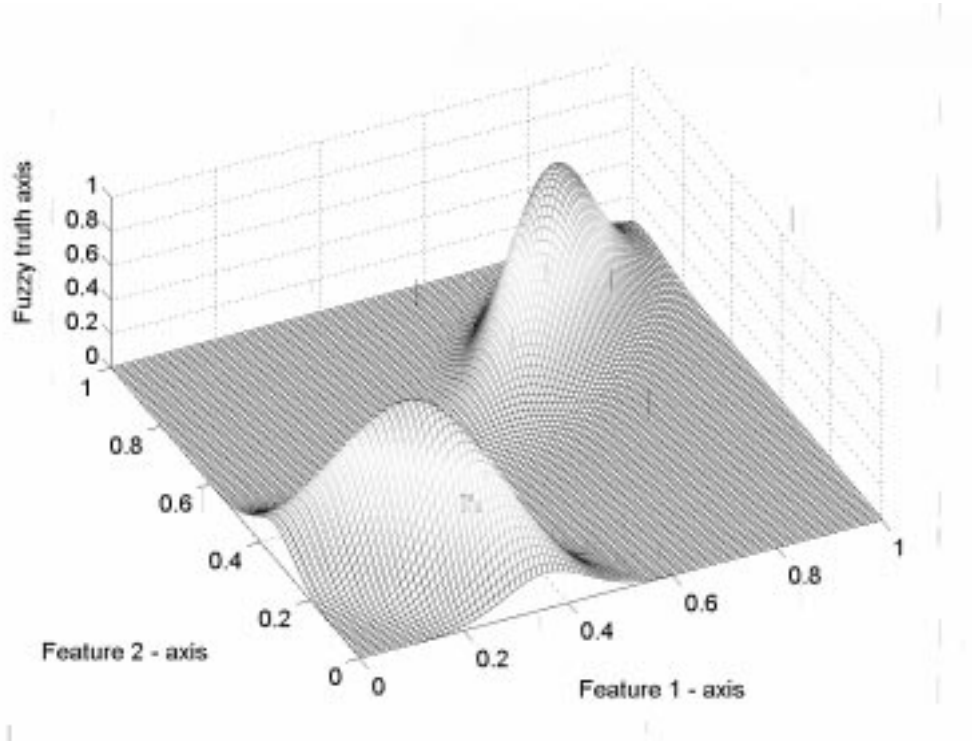


**Figure 8. Ellipsoidal Gaussians for 2-D classes.**

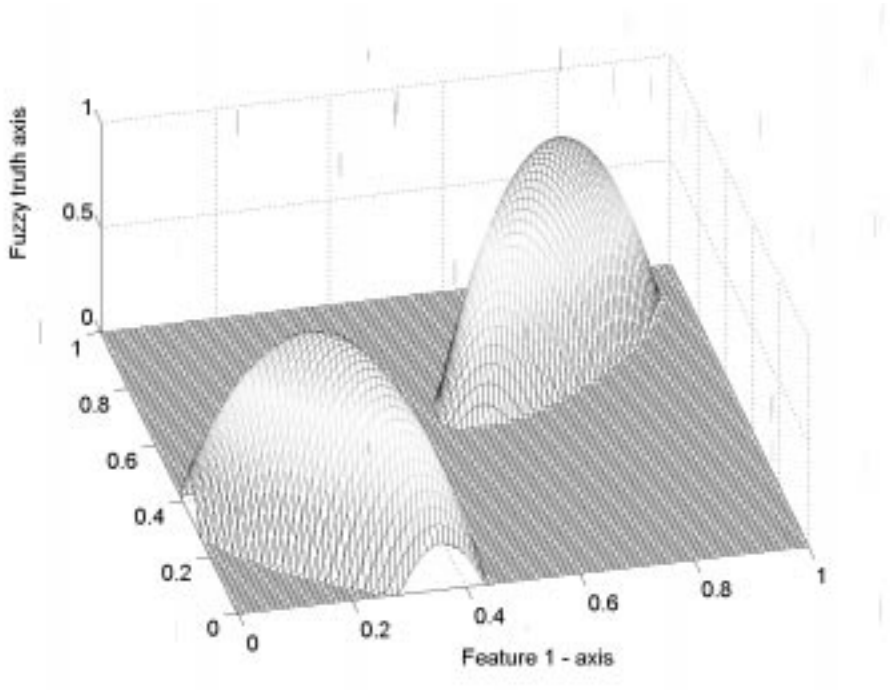


**Figure 9. Extended Epanechnikov functions on the plane.**

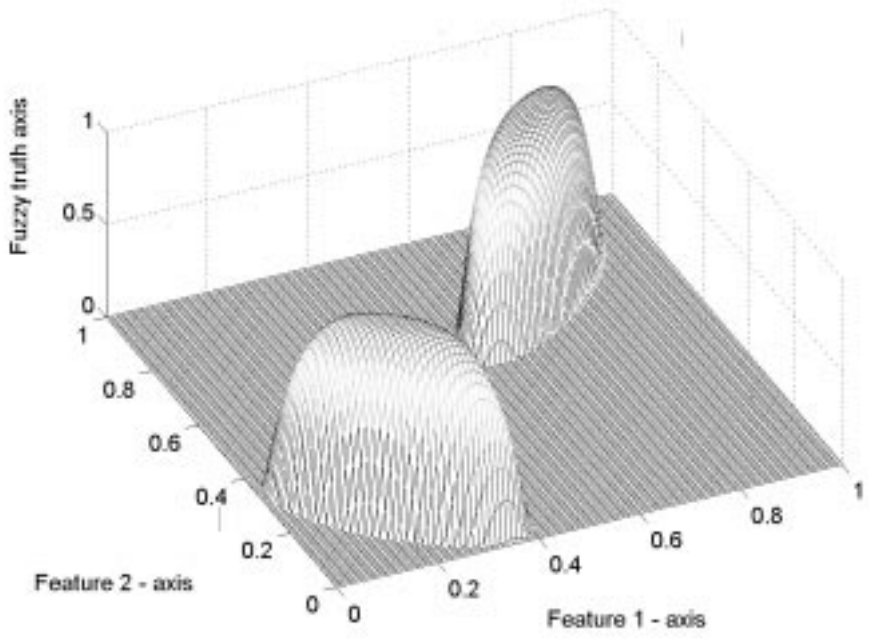


**Figure 10. Special extended Epanechnikovs on the plane.**

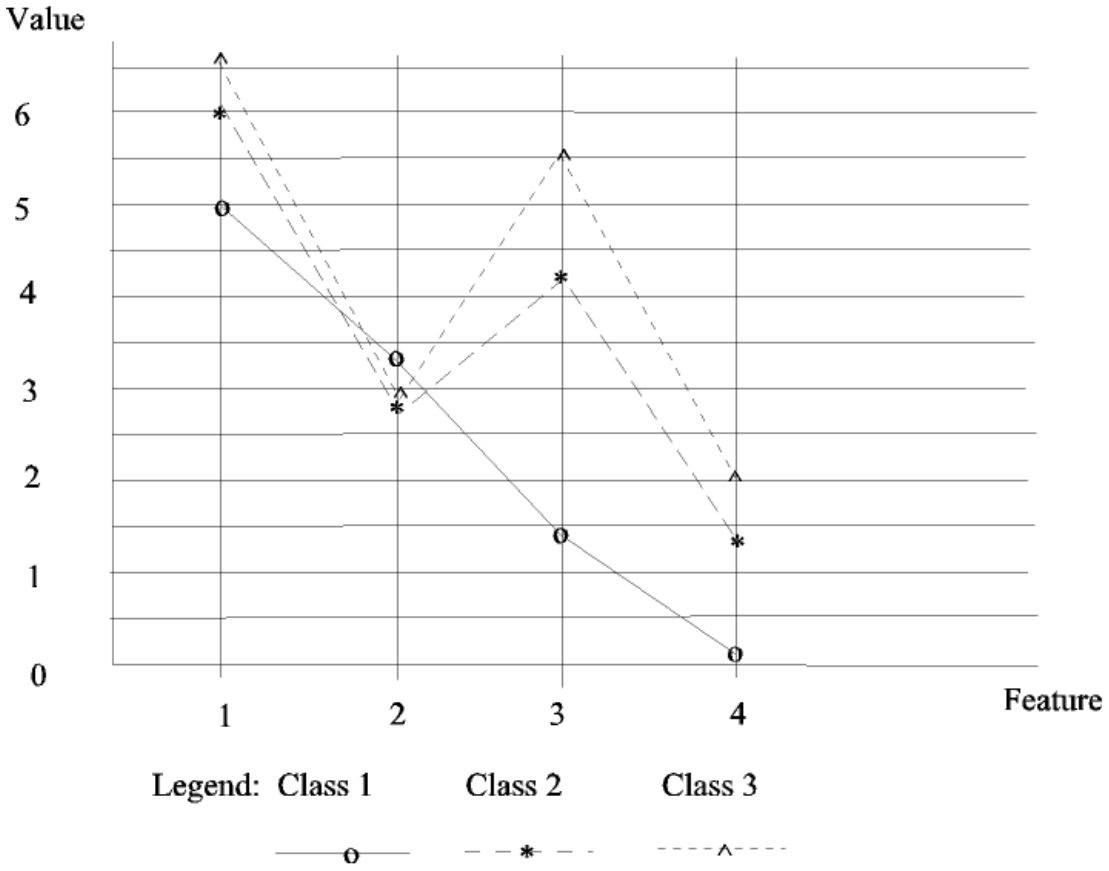


**Figure 11. The MWFEV centers of the four iris features.**