Training of a Fuzzy Classifier with Polyhedral Regions

Tomoo Takigawa*, Takafumi Shimozaki**, and Shigeo Abe*
* Graduate School of Science and Technology, Kobe University, Kobe, JAPAN
E-mail: abe@eedept.kobe-u.ac.jp
** IBM Japan, Ltd., Osaka, JAPAN

Abstract

In this paper, we discuss a fuzzy classifier with polyhedral regions. First, we generate an initial convex hull with the maximum dimension using the data, included in a class, in the general positions. Next, we modify the convex hull using one training datum at a time by the dynamic convex hull generation method. Finally, for each convex hull we define a membership function using the minimum operator and tune the slopes of the membership functions using the training data. We demonstrate the effectiveness of our method using two benchmark data sets.

1 Introduction

Class regions are usually approximated by hyperboxes [1]. But since performance of fuzzy classifiers is determined by how class regions are approximated, to improve generalization ability, ellipsoids [2, 3] or polyhedrons [4] are used. In approximating class regions by polyhedrons, in [4], first the multilayer neural network is trained and the separating hyperplanes are extracted from the trained network and class regions are approximated by shifting the separating hyperplanes. In [5], a class region is approximated by a convex hull that is the minimum convex region that includes the training data belonging to the class. But this is intended for clustering.

In this paper we approximate class regions by convex hulls, define a membership function for each convex hull, and tune the membership functions. We generate a convex hull, using the dynamic convex hull generation method developed for generating the Lyapunov function [6]. To use the dynamic convex hull generation method, we need to generate an initial convex hull whose dimension is the same with that of the convex hull including the training data for that class. Then starting from the initial convex hull, we expand the convex hull adding a training datum one at a time.

In the following, in Section 2, we explain the method for generating the initial convex hull and the dynamic convex hull generation method. In addition, we explain the method for defining the membership function for the convex hull. In Section 3, we compared the classification performance of the proposed method with that of the fuzzy classifier with ellipsoidal regions using the iris data [7] and thyroid data [8].

2 Polyhedral Fuzzy Rules

In this section we generate a convex hull, using the dynamic convex hull generation method [6] developed for generating the Lyapunov function. To use the dynamic convex hull generation method, we need to generate an initial convex hull whose dimension is the same with that of the convex hull including the training data for that class. Then starting from the initial convex hull, we expand the convex hull adding a training datum one at a time. If the training datum is in the convex hull, we do nothing. But if it is outside of the convex hull, we modify the facets, which are the surfaces of the convex hull, in front of the training datum.

In the following, in Section 2.1, we explain a general procedure for generating a convex hull, and in Section 2.2 we discuss a procedure for generating an initial convex hull. In Section 2.3 we explain the dynamic convex hull generation method. In Section 2.4, we explain how to define a membership function for a convex hull.

2.1 Generation of a Convex Hull

We generate a convex hull using N d-dimensional input vectors $\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^{N-1}$. To avoid confusion, we do not affix class labels to the input vectors. Let the convex hull generated by l input vectors $\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^{l-1}$ be $P(l)$:

$$P(l) = \mathrm{conv}\{\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^{l-1}\}, \qquad (1)$$

where conv $\{\cdot\}$ denotes the convex hull generated by the set of points in $\{\cdot\}$. Namely, we call the linear combination of the set consisting of l points $\{\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^{l-1}\}$:

$$P(l) = \Big\{\mathbf{x} \mid \mathbf{x} = \lambda_0 \mathbf{p}^0 + \cdots + \lambda_{l-1}\mathbf{p}^{l-1}, \\ \lambda_0 + \cdots + \lambda_{l-1} = 1, \lambda_k \geq 0, k \in \{0, 1, \cdots, l-1\}\Big\} \tag{2}$$

a convex hull or a convex polyhedron. If the dimension of $P(l)$ is j, we denote the convex hull as $P^j(l)$ or simply denote it as P^j.

Let the dimension of the convex hull $P(N)$ generated by points $\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^{N-1}$ be d_c $(\leq d)$.

2.2 Generation of the Initial Convex Hull

Let the first $d+1$ points $\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^d$ among N points $\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^{N-1}$ be linearly independent. Then the initial convex hull can be generated by P^d = conv $\{\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^d\}$. The $(d-1)$-dimensional convex hulls that form the surface of P^d are generated by deleting one point from $\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^d$:

$$\begin{array}{rcl} F_1 & = & \mathrm{conv}\{\mathbf{p}^1, \mathbf{p}^2, \cdots, \mathbf{p}^d\}, \\ F_2 & = & \mathrm{conv}\{\mathbf{p}^0, \mathbf{p}^2, \cdots, \mathbf{p}^d\}, \\ & \cdots & \\ F_{d+1} & = & \mathrm{conv}\{\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^{d-1}\}. \end{array} \tag{3}$$

We call them the facets of P^d and we denote the set as $\mathcal{F}_{d-1}(P^d)$. The k-dimensional convex hulls that are intersections of P^d and the tangent hyperplane are called the k-dimensional faces and we denote the set as $\mathcal{F}_k(P^d)$.

If $d+1$ points which are arbitrary chosen from N points are all linearly dependent, we cannot generate d-dimensional initial convex hull. To use the dynamic convex hull generation method, first we need to determine the rank of the N d-dimensional points. Instead of calculating the rank, we generate the initial convex hull reading points one at a time as follows.

1. **Generation of a one-dimensional convex hull** Let $\mathbf{p}^0$ be a reference point, and read $\mathbf{p}^{i_1} (i_1 \neq 0)$. Calculate the one-dimensional distance between two points $\mathbf{p}^0$ and $\mathbf{p}^{i_1}$ and if the distance for the d_1th $(0 \leq d_1 \leq d)$ input axis is not zero, a one-dimensional convex hull is generated. Store $\mathbf{p}^{i_1}$ and m_1, and go to Step 3.
2. If all the one-dimensional distances between $\mathbf{p}^0$ and $\mathbf{p}^{i_1}$ are zero, the two points are identical. Thus read another point $\mathbf{p}^{i_x}$ and repeat Step 1 until a one-dimensional convex hull is generated.
3. **Generation of an n-dimensional convex hull assuming that the $(n-1)$-dimensional convex hull is generated** For the d_kth $(k = 1, \ldots, n-1)$ input axes used to generate the $(n-1)$-dimensional convex hull, calculate the following vectors:

$$\mathbf{q}^k = \mathbf{p}^{i_k} - \mathbf{p}^0, \tag{4}$$

   where $\mathbf{p}^{i_k} = [p_{d_1}^{i_k}, \cdots, p_{d_{n-1}}^{i_k}]^t, \mathbf{p}^0 = [p_{d_1}^0, \cdots, p_{d_{n-1}}^0]^t$. Namely, $\mathbf{p}^{i_k}$ and $\mathbf{p}^0$ are $(n-1)$-dimensional vectors whose $n-1$ elements are chosen from $\mathbf{p}^{i_k}$ and $\mathbf{p}^0$, respectively.
4. Using $\mathbf{q}^k$, calculate the vector orthogonal to the $(n-1)$-dimensional convex hull:

$$\mathbf{a} = \mathbf{q}^1 \times \cdots \times \mathbf{q}^{n-1}, \tag{5}$$

   where $\times$ is the outer product operator and

$$\mathbf{a} = [a_{d_1}, \cdots, a_{d_{n-1}}]^t. \tag{6}$$

5. Calculate the distance of $\mathbf{p}^{i_n} (i_n \neq i_k, k = 1, \ldots, n-1)$ from the $(n-1)$-dimensional convex hull:

$$d(\mathbf{p}^{i_n}) = \frac{|a_{d_1} p_{d_1}^{i_n} + \cdots + a_{d_{n-1}} p_{d_{n-1}}^{i_n} - b|}{\sqrt{a_{d_1}^2 + \cdots + a_{d_{n-1}}^2}}, \tag{7}$$

   where b is calculated by substituting $\mathbf{p}^0$ into the equation of the hyperplane including the $(n-1)$-dimensional convex hull:

$$b = a_{d_1} p_{d_1}^0 + \cdots + a_{d_{n-1}} p_{d_{n-1}}^0. \tag{8}$$

   If $d(\mathbf{p}^{i_n}) \neq 0$, the convex hull is expandable, and go to Step 6. Otherwise, go to Step 7.
6. Store $\mathbf{p}^{i_n}$ and d_n to generate the n-dimensional convex hull and go to Step 8.
7. Repeat Steps 4 and 5 for the input axes other than m_k $(k = 1, \ldots, n-1)$. If the convex hull is not expandable for all the dimensions, do the Steps 4 and 5 for the remaining points.
8. If for all the points and the remaining input axes the convex hull is not expandable, terminate the algorithm and store the final dimension as d_c $(d_c \leq d)$.

2.3 Dynamic Convex Hull Generation

Let the dimension of the convex hull $P(N)$ generated by N d-dimensional points $\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^{N-1}$ be d_c, and the first $d_c + 1$ points $\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^{d_c}$ be linearly independent. Then the initial convex hull is given by $P^{m_c} = \mathrm{conv}(\mathbf{p}^0, \mathbf{p}^1, \cdots, \mathbf{p}^{d_c})$. To generate $P(N)$ by the dynamic convex hull generation starting from the initial convex hull P^{d_c}, first we generate the convex hull by adding $\mathbf{P}^{d_c+1}$ to P^{d_c}. Namely,

$$P(d_c+2) = \mathrm{conv}\left\{P^{d_c}, \mathbf{p}^{d_c+1}\right\}. \quad (9)$$

In general, for $i = 2, \ldots, N - d_c - i$ we modify the convex hull $P(d_c + i - 1)$ by

$$P(d_c+i) = \mathrm{conv}\left\{P(d_c+i-1), \mathbf{p}^{d_c+i-1}\right\} \quad \text{for} \quad i = 2, \cdots, N - d_c - i. \quad (10)$$

Here if P^{d_c+1} is included in $P(d_c+2)$,

$$P(d_c+2) = P^{d_c} \quad (11)$$

and thus we need not modify the convex hull. But if P^{d_c+1} is not included in P^{d_c}, we need to modify P^{d_c}.

In the following, according to [6], we explain how to generate $P(n) = \mathrm{conv}\{P(n-1), \mathbf{p}^n\}$ by adding $\mathbf{p}^n$ to the d_c-dimensional convex hull $P(n-1)$.

1. We calculate the vector orthogonal to facet $F_i \in \mathcal{F}_{d-1}(P(n-1))$, $\mathbf{a}_i$, by the outer product of two linearly independent vectors on the facet. Calculating the outer product of edge vectors $\mathbf{b}_i$ and $\mathbf{c}_i$ in facet F_i, we get

$$\mathbf{a}_i = \mathbf{b}_i \times \mathbf{c}_i. \quad (12)$$

We need to set $\mathbf{a}_i$ so that $\mathbf{a}_i$ is in the outer direction. To determine the direction of the orthogonal vector, we use the point $\mathbf{d}$ that is not on F_i and modify $\mathbf{a}_i$ as follows:

$$\mathbf{a}_i = \begin{cases} \mathbf{a}_i & \text{for} \quad \mathbf{a}_i^t \cdot \mathbf{d} < 0, \\ -\mathbf{a}_i & \text{for} \quad \mathbf{a}_i^t \cdot \mathbf{d} > 0. \end{cases} \quad (13)$$

2. We paint the facet F_i red, yellow, or blue according to whether the given point $\mathbf{p}^j (j > d_c)$ lies against the facet. Namely, we calculate the dot product of $\mathbf{a}_i$ and $\mathbf{q}_i^j$ that is a vector from an arbitrary point on the facet F_i to $\mathbf{p}^j$, and we paint the facet F_i with the color C_i:

$$C_i = \begin{cases} \text{Red} & \text{for} \quad \mathbf{a}_i^t \cdot \mathbf{q}_i^j > 0 \\ \text{Yellow} & \text{for} \quad \mathbf{a}_i^t \cdot \mathbf{q}_i^j = 0 \\ \text{Blue} & \text{for} \quad \mathbf{a}_i^t \cdot \mathbf{q}_i^j < 0 \end{cases} \quad (14)$$

3. After coloring facets in $F_i (\in \mathcal{F}_{d_c-1}(P(n)),\ i = 1, \cdots, d_c+1)$ in Step 2, we color the faces, included in the facet $\mathcal{F}_{d_c-2}(P(n))$, that are intersections of facets. The color is determined by mingling the colors of the facets as listed in Table 1.

Table 1: Face coloring according to facet colors

		Facet Color		
		Red	Yellow	Blue
Facet	Red	Red	Orange	Purple
	Yellow	Orange	Yellow	Green
Color	Blue	Purple	Green	Blue

4. From Step 2, according to the location of $\mathbf{p}^n$ against the facets, the set of facets of the d_c-dimensional convex hull $P(n-1)$, $\mathcal{F}_{d_c-1}(P(n-1))$, are divided into

$$\mathcal{F}_{d_c-1}(P(n-1)) = \mathcal{F}_{d_c-1}^B(P(n-1)) \cup \mathcal{F}_{d_c-1}^R(P(n-1)) \cup \mathcal{F}_{d_c-1}^Y(P(n-1)), \quad (15)$$

where the superscripts B, R, and Y denote that the associated facets are painted blue, red, and yellow, respectively. Now consider how to modify the facets. The facets included in $\mathcal{F}_{d_c-1}^B(P(n-1))$ are not seen from $\mathbf{p}^n$. And thus, we need not modify the facets. Namely,

$$\mathcal{F}_{d_c-1}^B(P(n)) = \mathcal{F}_{d_c-1}^B(P(n-1)). \quad (16)$$

Next, the facets included in $\mathcal{F}_{d_c-1}^R(P(n-1))$ are seen from $\mathbf{p}^n$. Thus, these facets need to be modified to be the convex hull $\mathcal{F}_{d_c-1}^R(P(n))$ including $\mathbf{p}^n$ and the faces that connect blue and red facets:

$$\hat{\mathcal{F}}_{d_c-1}^P(P(n)) = \{\mathrm{conv}\{E, \mathbf{p}^n\} \mid E \in \mathcal{F}_{d_c-2}^P(P(n-1))\}. \quad (17)$$

Finally, $\mathbf{p}^n$ is on the extension of the facets included in $\mathcal{F}_{d_c-1}^Y(P(n-1))$. Thus the modified set of facets, $\overline{\mathcal{F}}_{d_c-1}^Y(P(n))$, are given by

$$\overline{\mathcal{F}}_{d_c-1}^Y(P(n)) = \{\mathrm{conv}\{E, \mathbf{p}^n\} \mid E \in \mathcal{F}_{d_c-2}^G(P(n-1))\}. \quad (18)$$

Using (15) to (18), we modify the convex hull.

5. We set $n \leftarrow n+1$ and if $n \le N$ go back to Step 1. Otherwise, we stop the algorithm.

2.4 Definition of Membership Functions

For the convex hull belonging to class c we define the membership function as follows: the degree of a membership of $\mathbf{x}$ at the center of the convex hull is 1, and it decreases as $\mathbf{x}$ move away from the center (see Fig.1). Namely, we define the center of a convex hull by the center of the training data that are included in the convex hull:

$$\mathbf{c}_c = \frac{1}{|X_c|} \sum_{\mathbf{p} \in X_c} \mathbf{p}, \quad (19)$$

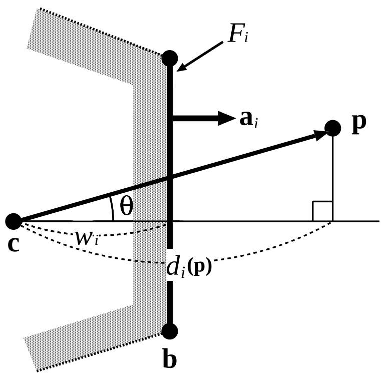

Figure 1: Membership function for the two-dimensional input

where X_c is the set of training data belonging to class c.

The membership function $m_{ci}(\mathbf{p})$ is given by

$$\begin{aligned} m_{ci}(\mathbf{p}) &= \exp\left(-h_{ci}(\mathbf{p})\right) \\ &= \exp\left(-\frac{d_{ci}(\mathbf{p})}{\alpha_c}\right), \end{aligned} \tag{20}$$

where $h_{ci}(\mathbf{p})$ is the tuned distance, α_c is a parameter to tune the slope of the membership function, and $d_{ci}(\mathbf{p})$ is the weighted distance in the outer orthogonal direction from a center $\mathbf{c}_c$ to $\mathbf{p}$:

$$d_{ci}(\mathbf{p}) = \begin{cases} \frac{\mathbf{a}_{ci}^t(\mathbf{p}-\mathbf{c}_c)}{|\mathbf{a}_{ci}|w_{ci}} & \text{for} \quad \mathbf{a}_{ci}^t(\mathbf{p}-\mathbf{c}_c) \geq 0, \\ 0 & \text{for} \quad \mathbf{a}_{ci}^t(\mathbf{p}-\mathbf{c}_c) < 0. \end{cases} \tag{21}$$

Here $\mathbf{a}_{ci}$ is the outer orthogonal vector of facet F_{ci} and w_{ci} is the distance from F_{ci} to $\mathbf{c}_c$.

We define the multi-dimensional membership function $m_c(\mathbf{p})$ using the minimum operator for $m_{ci}(\mathbf{p})$. Then $m_c(\mathbf{p})$ is given by

$$\begin{aligned} m_c(\mathbf{p}) &= \min_i m_{ci}(\mathbf{p}) \\ &= \exp\left(-h_c(\mathbf{p})\right), \end{aligned} \tag{22}$$

where

$$h_c(\mathbf{p}) = \max_i h_{ci}(\mathbf{p}). \tag{23}$$

Instead of using the membership function given by (22), we can use $h_c(\mathbf{p})$.

3 Tuning of Slopes

In this section we consider maximizing the recognition rate of the training data by changing α_c ($c = 1, \ldots, n$) where n is the number of classes. If we increase α_c, the degree of membership $m_c(\mathbf{x})$ (the tuned distance $h_c(\mathbf{x})$) increases (decreases), and if we decrease it, the degree of membership (the tuned distance) decreases (increases). Thus we can improve the recognition rate by iterate tuning α_c one at a time without allowing new misclassification. By tuning membership functions successively without allowing new misclassification, the recognition rate is improved monotonically and reaches a maximum. This is, however, a local maximum. Thus to avoid trapping into a local maximum, we allow new misclassification during tuning.

Now suppose we tune the tuning parameter α_c. Up to some value we can increase or decrease α_c without making the correctly classified datum $\mathbf{x}$ be misclassified. Thus we can calculate the upper bound $U_c(\mathbf{x})$ or lower bound L_c of α_c that causes no misclassification of $\mathbf{x}$. Now let $U_c(1)$ and $L_c(1)$ denote the upper and lower bounds that do not make the correctly classified data be misclassified, respectively. Likewise, $U_c(l)$ and $L_c(l)$ denote the upper and lower bounds in which $l-1$ correctly classified data are misclassified, respectively. Then, for instance, if we set a value in the interval $[U_c(1), U_c(2))$ to α_c, one correctly classified datum belonging to class i is misclassified, where $[a, b]$ and (a, b) denote the closed and open intervals, respectively.

Similarly, if we increase or decrease α_c, misclassified data may be correctly classified. Let $\beta_c(l)$ denote the upper bound of α_c that is smaller than $U_c(l)$ and that resolves misclassification. And $\gamma_c(l)$ denotes the lower bound of α_c that is larger than $L_c(l)$ and that resolves misclassification.

Then the next task is to find which interval among$(L_c(l), \gamma_c(l))$ and $(\beta_c(l), U_c(l))$ ($l = 1, \ldots$) gives the maximum recognition rate. To limit the search space, we introduce the maximum l, i.e., l_M. Let $(L_c(l), \gamma_c(l))$ be the interval that gives the maximum recognition rate of the training data among $(L_c(k), \gamma_c(k))$ and $(\beta_c(k), U_c(k))$ for $k = 1, \ldots, l_M$. Then even if we set any value in the interval to α_c, the recognition rate of the training data does not change but the recognition rate of the test data may change. To control the generalization ability, we set α_c as follows:

$$\alpha_c = \beta_c(l) + \delta(U_c(l) - \beta_c(l)) \tag{24}$$

for $(\beta_c(l), U_c(l))$, where δ satisfies $0 < \delta < 1$ and

$$\alpha_c = \gamma_c(l) - \delta(\gamma_c(l) - L_c(l)) \tag{25}$$

for $(L_c(l), \gamma_c(l))$.

4 Performance Evaluation

We compared the classification performance of the proposed method with that of the fuzzy classifier with ellipsoidal regions [3] using the iris data [7] and thyroid data [8].

Table 2 shows the number of inputs, classes, training data, and test data of the benchmark data sets.

Table 2: Feature of benchmark data

Data	Inputs	Classes	Train.	Test
Iris	4	3	75	75
Thyroid	21	3	3772	3428

The thyroid data included 16 discrete input variables among 21 input variables. And when we generated convex hulls using the thyroid data, a large number of facets were generated. Then to reduce the number of facets, we selected 5 relevant features, namely, 3rd, 17th, 18th, 19th, 21st features, from the 21 original features using the feature selection method discussed in [9].

Table 3 shows the results for the proposed method and the fuzzy classifier with ellipsoidal regions.

Table 3: Comparison of recognition rates (in %) for the benchmark data

Data	Initial	Final	Ellipsoid
Iris	97.33 (100)	97.33 (100)	98.67 (98.67)
Thyroid	98.10 (98.09)	98.22 (99.53)	97.29 (99.02)

In the table, "Initial" and "Final" columns show the recognition rates of the test (training) data when the membership functions were not tuned and tuned, respectively. "Ellipsoid" column shows the best recognition rates by the fuzzy classifier with ellipsoidal regions.

For the iris data, the recognition rate of the test data reached 100%, and tuning membership functions did not improve the recognition rate. But for the thyroid data, by tuning, the recognition rate of the training data improved by 1.44% and so did that of the test data by 0.12%.

Although the recognition rates of the iris data are almost the same for both methods, those of the thyroid data by the proposed method are better than those of the fuzzy classifier with ellipsoidal regions. Figures 2 and 3 shows the change of the recognition rates of the iris and thyroid data, respectively, as facets are generated. The horizontal axis denotes the number of facets and the vertical axis denotes the recognition rate. From the figures, it is seen that the recognition rates fluctuate as the facets are generated. This may be caused by generating facets by selecting the data that are far away from the convex hull.

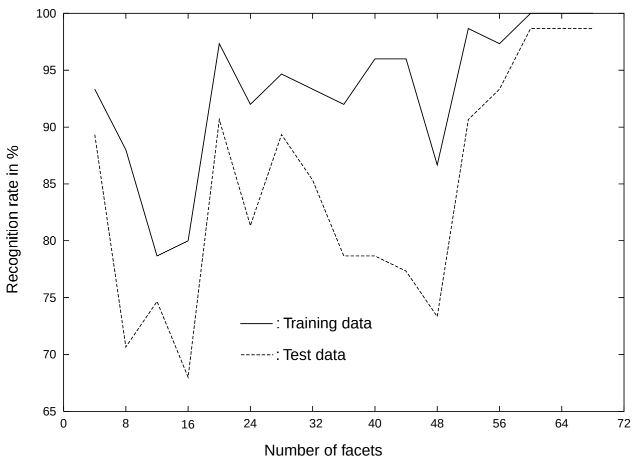

Figure 2: The recognition rates of the iris data

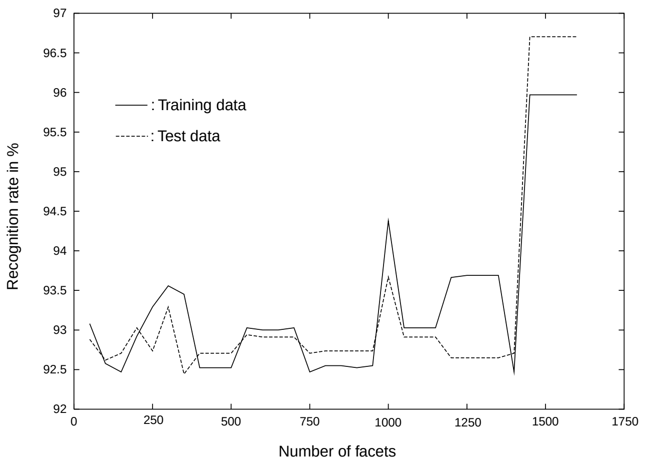

Figure 3: The recognition rates of the thyroid data

5 Conclusions

In this paper, we discussed the training method of a fuzzy classifier with polyhedral regions. Namely, first, we generate an initial convex hull with the maximum dimension using the data in the general positions. Next, we modify the convex hull using one training datum at a time by the dynamic convex hull generation method. Finally, for each convex hull we define a membership function using the minimum operator and tune the slopes of the membership functions using the training data. Using two benchmark data sets, we show that our method is comparable to or better than that of the fuzzy classifier with ellipsoidal regions.

Acknowledgements

We are grateful to Mr. P. M. Murphy and Mr. D. W. Aha of the University of California at Irvine for organizing the data bases including the thyroid data (ics.uci.edu: pub/machine-learning-databases) and to Professor Y. Ohta of Kobe University for providing references.

References

[1] S. Abe, "*Neural Networks and Fuzzy Systems: Theory and Applications*," Kluwer Academic, 1996.

[2] S. Abe and R. Thawonmas, "A Fuzzy Classifier with Ellipsoidal Regions," *IEEE Trans. Fuzzy Systems*, Vol. 5, No. 3, pp. 358-368, 1997.

[3] S. Abe, "Dynamic Cluster Generation for a Fuzzy Classifier with Ellipsoidal Regions" *IEEE Trans. System, Man, and Cybernetics–Part B*, Vol. 28, No. 6, pp. 869–875, 1998.

[4] F. Uebele, S. Abe, and M.-S. Lan, "A Neural Network-Based Fuzzy Classifier," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 25, No. 2, pp. 353–361, 1995.

[5] I. H. Suh and J. H. Kim, and F. C.-H, Rhee, "Convex-Set-Based Fuzzy Clustering," *IEEE Trans. Fuzzy Systems*, Vol. 7, No. 3, pp. 271–285, 1996.

[6] Y. Ohta, Y. Nagai, and L. Gong, "Beneath-Beyond Method and Construction of Lyapunov Functions," *Proc. International Symposium on Nonlinear Theory and its Applications (NOLTA'97)*, pp. 353–356, 1997.

[7] R. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, Vol. 7, Part II, pp. 179–188, 1936.

[8] S. M. Weiss and I. Kapouleas, "An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods," *Proc. 11th International Joint Conference on Artificial Intelligence*, pp. 781–787, 1989.

[9] R. Thawonmas and S. Abe, "A Nobel Approach to Feature Selection Based on Analysis of Class Regions," *IEEE Trans. Systems, Man, and Cybernetics-Part B*, Vol. 27, No. 2, pp. 196-207, 1997.