Construction of Convex Hull Classifiers in High Dimensions

Tetsuji Takahashi, Mineichi Kudo and Atsuyoshi Nakamura

Graduate School of Information Science and Technology
Hokkaido University
Kita-14, Nishi-9, Kita-ku, Sapporo 060-0814, JAPAN

Abstract

We propose an algorithm to approximate each class region by a small number of approximated convex hulls and to use these for classification. The classifier is one of non-kernel maximum margin classifiers. It keeps the maximum margin in the original feature space, unlike support vector machines with a kernel. The construction of an exact convex hull requires an exponential time in dimension, so we find an approximate convex hull (a polyhedron) instead, which is constructed in linear time in dimension. We also propose a model selection procedure to control the number of faces of convex hulls for avoiding over-fitting, in which a fast procedure is adopted to calculate an upper-bound of the leave-one-out error. In comparison with support vector machines, the proposed approach is shown to be comparable in performance but more natural in the extension to multi-class problems.

Keywords: Pattern Recognition, Convex Hull, Classifier Selection

1. Introduction

Determining class regions directly in a feature space and classifying a class-unknown sample to the class of which region is closest to the sample seems a promising approach to classification. However, the efficacy of this approach has not been established. In this paper, we address this issue, specifically for a set of convex hulls as a class region.

Usually, class regions are determined indirectly by discriminant functions or by an estimated decision boundary. Nevertheless, some classifiers are connected to certain types of regions. For example, a linear support vector machine (shortly, SVM) [1] is connected to the convex hulls of samples of two classes [2] and the nearest neighbor classifier is connected to the Voronoi diagram of samples [3]. Therefore, it is worth examining which type of regions is most effective for a wide range of classification problems and how such regions can be constructed. In this paper, we focus on a set of convex hulls that include training samples of a class maximally. Then, we assign a class label to a given test sample according to the distances of the sample to the estimated class regions. This approach [4] using convex hulls has shown performance comparable with that of SVM in low-dimensional data.

However, it is computationally difficult to construct the exact convex hull in high-dimensional data [5]. The algorithm by McBride *et al.* [6] also suffers from the same type of dimensionality problem. In this paper, therefore, we use a set of polyhedral convex sets that are constructed in a linear order of dimension.

Zero training errors are achieved by using a sufficient number of convex hulls in such a way that every training sample of a class is covered by at least one convex hull and any other sample belonging to the other classes is excluded, ensuring that no sample is shared by more than one class. Thus, according to Occam's razor [7], we should choose the simplest classifier as long as it attains the same degree of training error. In our case, we select the smallest number of convex hulls with the smallest number of faces.

There are three problems to be solved. First, construction of the convex hull of a given sample set is computationally difficult. The time and space complexities are exponential in dimension in the worst case. Second, more than one convex hull is generally needed to cover all training samples exhaustively. Indeed, as is well known, if each class sample set is covered by a single convex hull, then any pair of classes is linearly separable, and the reverse is also true. Finally, when we use convex hulls for classification, the distance

between a point and the boundary of each convex hull must be measured. However, the computational cost is quite high (e.g., see [8]).

In this paper, we first describe an algorithm for constructing approximate convex hulls that can cope with these three problems in a reasonable way and then we describe a model selection procedure for choosing simpler classifiers. The algorithm is based on the prototype version introduced in [8].

A similar idea is realized in [9] in which the distance between a given point and the convex hull of each class is used for class assignment. However, only a single convex hull is assigned for each class, and thus a non-linear kernel is necessary for solving non-linearly separable problems. Good selection of a kernel and its parameters thus becomes problematic as well as the case of SVM. The distance between a point and a convex hull is calculated by solving an SVM optimization problem by assuming the point as a singleton set. Since this calculation is made for each testing point, the time complexity is very high when the training sample set is large. Cevikap *et al.* [10] discussed linear manifolds spanned by nearest neighbors of a query sample and constructed an affine hull (a convex hull is contained in it) for classification, but the nonlinearzation also relies on a kernel. In contrast to these studies, the classifier developed here does not need a kernel.

2. Definitions of Convex Hull and Reflective Convex Hull

The convex hull $conv(S)$ of a given finite dataset $S = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ is defined as the intersection of all convex sets containing S. Here, $\boldsymbol{x}_i \in R^m$ $(i = 1, \ldots, n)$ is a point in an m-dimensional Euclidean space. Since S is finite, $C = conv(S)$ is a polyhedron with at most n vertices.

By ∂C, we denote the boundary of C and divide it into q-faces according to the dimensions. For example, 0-faces are the vertices of C and $(m-1)$-faces are the facets or hyper-planes. Let $V(C)$ be the set of vertices of C and let $F(C)$ be the set of facets of C. A convex hull of a finite set of S can be expressed by several ways. For example, the $\mathcal{V}$-representation of C is given by $C = \{\boldsymbol{y} = \sum c_{\boldsymbol{x}}\boldsymbol{x} | \sum c_{\boldsymbol{x}} = 1, c_{\boldsymbol{x}} \geq 0, \boldsymbol{x} \in V(C)\}$. Another expression is the $\mathcal{H}$-representation given by $C = \{\boldsymbol{y} | \langle \boldsymbol{w}, \boldsymbol{y} \rangle \leq c, \forall (\boldsymbol{w}, c) \in F(C)\}$, where $\langle \cdot, \cdot \rangle$ is the inner product and facet $(\boldsymbol{w}, c)$ is specified by a normal vector $\boldsymbol{w}$ $(||\boldsymbol{w}|| = 1)$ and a constant $c \in R$. We need both expressions to handle C efficiently in space complexity or in different goals. For example, an m-dimensional cube has $2m$ facets and 2^m vertices, and an m-dimensional simplex has $2m$ vertices and 2^m facets. For determining if a sample is included in C or not and for distance calculation to C, $\mathcal{H}$-representation has an advantage of $\mathcal{V}$-representation.

Unfortunately, we know that the number of facets can be of order $n^{\lfloor \frac{m}{2} \rfloor}$ [5]. Therefore, we propose to use an approximated convex hull that has a reasonable number of facets. To do this, we consider another expression of a convex hull. We use *support functions* [11] for the new expression, which is similar to $\mathcal{H}$-representation but needs an infinite number of half-spaces. A *support function* of a unit vector $\boldsymbol{w}$ ($\|\boldsymbol{w}\| = 1$) is given by

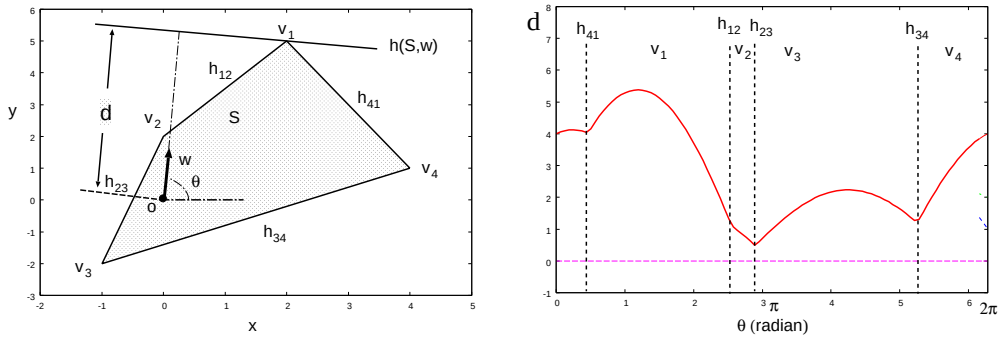
$$H(S, \boldsymbol{w}) \triangleq \sup\{\langle \boldsymbol{x}, \boldsymbol{w} \rangle | \boldsymbol{x} \in S\},$$

where "sup" denotes the supremum. With the set W_0 of all possible unit vectors, the convex hull C is defined as

$$C = conv(S, W_0) \triangleq \bigcap_{\boldsymbol{w} \in W_0} \{\boldsymbol{y} | \langle \boldsymbol{y}, \boldsymbol{w} \rangle \leq H(S, \boldsymbol{w})\}.$$

Here, $h(S, \boldsymbol{w}) = \{\boldsymbol{y} | \langle \boldsymbol{y}, \boldsymbol{w} \rangle = H(S, \boldsymbol{w})\}$ is called a *support plane*. The convex hull is an area that is surrounded by support planes $h(S, \boldsymbol{w})$ for all $\boldsymbol{w} \in W_0$. An example of the convex hull constructed by support planes is shown in Fig. 1. As an example of the support plane, $h(S, \boldsymbol{w})$ with angle θ is also shown.

We notice that a finite subset $W \subset W_0$ gives an approximate convex hull $conv(S, W)$ and thus a good selection of W derives a good approximation. Of course, instead of W, we can use the vectors corresponding to $F(C)$. Then



(a) (b)

Figure 1: (a) The convex hull of a set S, including points $v_1, v_2, \ldots, v_4$, represented by support planes. Only one support plane $h(S, \boldsymbol{w})$ is shown for a directional vector $\boldsymbol{w}$ with angle θ. As the value of θ increases, starting from this angle about $\theta = 85°$, support planes will find edge h_{12} first and then vertex v_2, edge h_{23}, and so on. The behavior is shown in (b) as a graph of support function $d = H(S, \boldsymbol{w})$ with the changing angle θ of $\boldsymbol{w}$.

the exact convex hull is obtained. However, as described before, the number of facets can grow exponentially in dimension. Therefore, we use a constant number of facets by letting $K = |W|$ be constant.

Next, let us consider separating a finite set S from another finite set T by the convex hull of S when they are linearly or non-linearly separated. A support plane h of S might locate both S and T in the same side of the two half-spaces separated by it. Apparently, such support planes are useless for separating S from T. For separation of S from T, all we need is *reflective*

support planes, which are support planes separating S from T perfectly or partly. A *reflective convex hull*, $C_r = conv(S, W_r)$, is the polyhedral convex set specified by the (infinite) set W_r of all unit vectors generating reflective support planes. Then the reflective convex hull C_r is formally defined by

$$C_r = conv(S, W_r) = \bigcap_{\boldsymbol{w} \in W_r} \{\boldsymbol{y} | \langle \boldsymbol{y}, \boldsymbol{w} \rangle \leq H(S, \boldsymbol{w})\}.$$

From the definition, $C = conv(S, W_0) \subseteq C_r = conv(S, W_r)$ since $W_r \subseteq W_0$. Intuitively speaking, the reflective convex hull of S is the polyhedral convex set of S whose faces reflect rays emitted from points in T. An example of the reflective convex hull is shown in Fig. 2. Note that usually a reflective convex hull, unlike the convex hull that encloses the samples of a class, is open on the side opposite the decision boundary.

We can also define the *margin* $M(S, T, \boldsymbol{w})$ between S and T in direction $\boldsymbol{w}$ as

$$M(S, T, \boldsymbol{w}) \triangleq -H(T, -\boldsymbol{w}) - H(S, \boldsymbol{w}).$$

Note that when S and T are linearly separable, there exists a support plane specified by a unit vector $\boldsymbol{w}$ with positive margin $M(S, T, \boldsymbol{w}) = M(T, S, -\boldsymbol{w})$. Now a reflective support plane can be defined formally as a

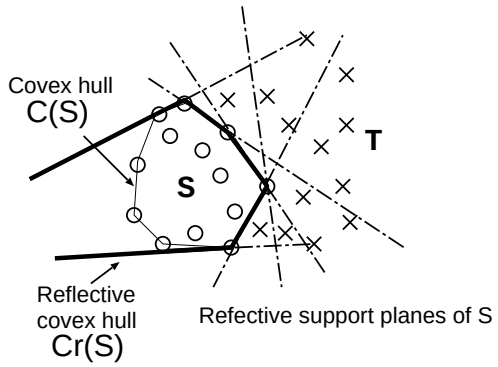


Figure 2: Reflective convex hull of the set of positive samples S against the set of negative samples T. A few reflective support planes are shown.

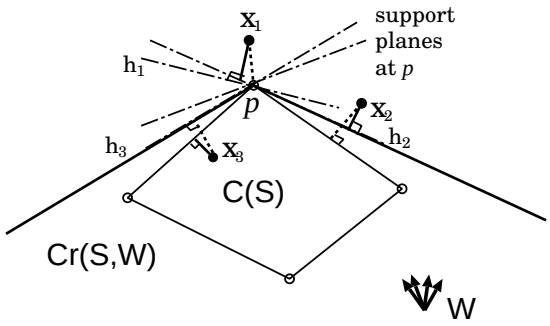


Figure 3: Distances $D(\boldsymbol{x}, \partial conv(S, W))$ from $\boldsymbol{x}$ to an approximated convex hull $conv(S, W)$ with W are shown by solid lines and the exact distances $D(\boldsymbol{x}, \partial conv(S))$ to the exact convex hull are shown by dashed lines.

support plane $h_r(S, \boldsymbol{w})$ with $\boldsymbol{w}$ satisfying

$$H(T, \boldsymbol{w}) - H(S, \boldsymbol{w}) > 0.$$

The (signed) distance between a point $\boldsymbol{x}$ and the nearest boundary of a convex hull $conv(S, W_0)$ is given by

$$D(\boldsymbol{x}, \partial conv(S, W_0)) = \sup_{\boldsymbol{w} \in W_0} \{M(S, \{\boldsymbol{x}\}, \boldsymbol{w})\}.$$

Here D takes a positive value for $\boldsymbol{x}$ outside $conv(S, W_0)$ and a negative value for $\boldsymbol{x}$ strictly inside $conv(S, W_0)$. The closer $\boldsymbol{x}$ is to $\partial conv(S, W_0)$,

the smaller the absolute value is. Note that the general calculation problem of $D(\boldsymbol{x}, \partial conv(S, W_0))$ is known to be NP-hard [12, 13]. In our case, since we will use a finite set $W \subset W_r$, we can calculate the distance $D(\boldsymbol{x}, \partial conv(S, W))$ in a linear order of $|W|$ as

$$D(\boldsymbol{x}, \partial conv(S, W)) = \max_{\boldsymbol{w} \in W} \{M(S, \{\boldsymbol{x}\}, \boldsymbol{w})\}. \tag{1}$$

An example is shown in Fig. 3. The method for calculating the distance $D(\boldsymbol{x}, \partial conv(S, W))$, where $conv(S, W)$ is the approximate convex hull specified by W, is shown in Fig. 3. When $\boldsymbol{x}$ is outside $conv(S, W)$ such as $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, then the distance takes a positive value; otherwise, e.g., $\boldsymbol{x}_3$, the distance takes a negative value. It is noted that the distance calculation can be made by Eq.(1) regardless of whether the point is inside or outside $conv(S, W)$. The difference from the true distance becomes smaller if we use a large set of W. For example, the nearest point of $\boldsymbol{x}_1$ on $\partial conv(S, W)$ is $\boldsymbol{p}$, but the distance calculated by (1) is the distance to the support plane h_1.

3. Approximation of a class region by reflective convex hulls

In this section, we introduce an algorithm to find a set of approximated reflective convex hulls for a class.

3.1. Algorithm

The following algorithm is applied class by class.

1. Let S be the positive sample set of a target class and T be the negative sample set of other classes. Let $\mathcal{C} = W = \emptyset$. Let L be an upper bound of the number of convex hulls and K be the number of normal vectors, that is, the number of facets of a convex hull.
2. Find random K pairs of $\boldsymbol{x}$ ($\in S$) and $\boldsymbol{y}$ ($\in T$) and put $\boldsymbol{w} = \frac{\boldsymbol{y}-\boldsymbol{x}}{\|\boldsymbol{y}-\boldsymbol{x}\|}$ in set W.
3. Repeat L times the following steps 4–5.
4. Let $U = \emptyset$. According to a random presentation order of positive samples, add positive samples $\boldsymbol{x}$ to U as long as $conv(U \cup \{\boldsymbol{x}\}, W) \cap T = \emptyset$.
5. Add the obtained $conv(U, W)$ into $\mathcal{C}$, unless it is already in $\mathcal{C}$.
6. Select a subset of $\mathcal{C}$ by a greedy set cover procedure for all positive samples. That is, the largest member of $\mathcal{C}$ is chosen first and then the member including the largest number of uncovered samples is chosen, and so on.

By this procedure, we have at most L approximated convex hulls that include samples of one class only. It should be noted that each convex hull

includes the positive samples maximally in Step 4.

Let us analyze the time complexity first. The dominant step is Step 4 in which we add a positive sample $\boldsymbol{x} \in S$ to the current subset U in order to expand the previous convex hull to $conv(U \cup \{\boldsymbol{x}\}, W)$. This needs $O(K)$ steps. Then we check if this convex hull is still exclusive, that is, if it does not include any negative sample $\boldsymbol{y} \in T$. To do this, we need $O(Kn^-)$ steps for n^- negative samples ($n^- = |T|$). Since we need to scan every positive sample, $O(Kn^+n^-)$ is necessary in Step 4 to obtain one exclusive and maximal convex hull for n^+ positive samples ($n^+ = |S|$). In Step 3, we repeat these procedures L times, so that the total order is $O(LKn^+n^-)$. Here the time complexity with respect to dimensionality m is omitted, but it is linear because we can obtain all necessary values by the inner product between two vectors in an m-dimensional space. Regarding L as a constant and regarding n^+ and n^- as the same complexity as the total sample number n, we have $O(Kn^2m)$. It is also noted that the complexity to measure the distance of a query sample to one convex hull with K facets is $O(Km)$. In summary, this algorithm needs $O(Kn^2m)$ for training and $O(Km)$ for testing. Since the worst-case complexity of SVM procedure with p support vectors is $O(n^3m)$ ($O(nmp)$ with SMO solver) for training and $O(mp)$ for testing [14],

the proposed method is comparable with SVM.

The space complexity is the memory amount required to hold all the convex hulls. It is $O(LK + mK)$, because one convex hull is expressed by K values to specify the distances from the origin in K directions, and each direction is specified by an m-dimensional vector.

A controllable nature of complexity is a characteristic of the proposed algorithm. By decreasing the value of K we can reduce the time and space complexities at the expense of approximation precision to the exact convex hull. As will be discussed later, approaching to the exact convex hull is not necessarily recommended. We need to choose the best model in sample-limited situations.

One problem may arise when noisy samples are included in the training data. It is easily understood that only one negative sample greatly breaks down a convex hull covering many positive samples if it appears inside the convex hull. Any kind of region-based algorithm shares the same problem. In the proposed method, we cope with this problem by a simple technique. We carry out the above algorithm twice over all classes. The first one is carried out in order to find noisy samples and the second one is carried out in order to find class regions. After the first round, we regard the samples that are

included in only small convex hulls as *noisy samples* and remove them for the second round. The second round is carried out without such noisy samples. To distinguish noisy samples from normal samples, we use a threshold α. If a convex hull is smaller than α in the *size*, that is, if the ratio of the number of samples included in the convex hull to the number of positive samples is less than α ($= 1\%$ in the following experiments), all samples included in it are judged as noisy.

To emphasize the number of facets, we call an approximated reflective convex hull with K directional unit vectors a *K-directional approximated reflective convex hull* (in short, K-ARCH). A convex hull might have less than K facets, but we use this terminology whenever $|W| = K$. The number of vectors K is controllable, so the (time) complexity can be decreased if we use K-ARCHs instead of the exact (reflective) convex hulls. Roughly speaking, as K increases, the corresponding K-ARCH approaches the true reflective convex hull. Class assignment of an unknown sample is carried out on the basis of distance to the nearest boundary of K-ARCHs.

Extension to multi-class classification is straightforward. Changing the role of positive samples and negative samples class by class, we obtain c sets of L K-ARCHs for c classes. Since the classification of a query sample is

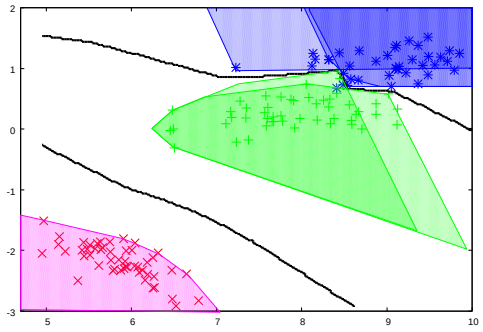


Figure 4: Approximated class regions by K-directional reflective convex hulls ($K = 50$). The dataset is 3-class 2-dimensional K-L expanded `iris` dataset. The decision boundary is also shown.

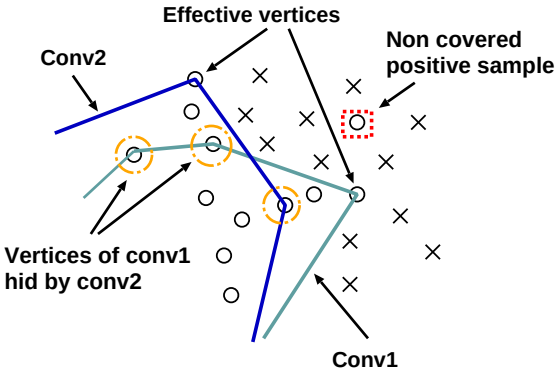


Figure 5: Effective vertices necessary for LOO upper bound. The circled vertices are not necessary.

made by the closest convex hull, the time complexity is $O(cLKm)$.

An example is shown in Fig. 4. Figure 4 shows the approximate class regions for 2-dimensional `iris` data consisting of two principal components. The original data is taken from a database [15]. The value of K is 50. There are five K-ARCHs covering all samples. In Fig. 4, we can confirm that 1) the approximated convex hulls are larger than the corresponding exact convex hulls, 2) the convex hulls are often unbounded, and 3) the decision boundary is taken to keep the maximum margin locally.

4. Model Selection

In reference [8], we used a fixed value of K for each dataset. It is expected to raise the performance of the proposed K-ARCH algorithm by choosing the optimal value of K. Unfortunately, it is not so easy to solve this optimization problem because the performance evaluation is theoretically difficult. Therefore, in this paper we propose a model selection procedure to find a suboptimal value of K. In the following, we will show the efficacy of this model selection procedure in the process speed and in the classification performance of the resultant classifier.

4.1. Estimation of generalization error

As a measure of testing error, we use the LOO (Leave-One-Out) error rate. Leaving one sample out, we construct a classifier from the remaining samples to test the left-out sample, and continue this procedure to estimate the classification performance. As is well known, the LOO rate is almost unbiased (Luntz and Brailovsky Theorem [16]), but it requires the building of n classifiers for n samples. Hence, we consider an upper bound of LOO that can be easily obtained without reconstruction of classifiers. Let ϵ_{LOO} be the LOO error rate, V be the set of vertices of all convex hulls and Z be

the set of samples that are outside all convex hulls. If a single convex hull is taken in each class, ϵ_{LOO} is bounded by the sum of $|V|$ and $|Z|$ divided by n, because removal of samples inside a convex hull does not affect the classifier design. However, in the case of more than one convex hull being taken in a class, a vertex of one convex hull can be covered by another. Figure 5 shows such a case. Such vertices can be safely ignored when counting for possible errors. We call vertices "*effective vertices*" if they are not covered by the other convex hulls. Let V_e be the set of effective vertices on the boundary of the approximated class region as the union of the convex hulls. Then we have an upper bound by

$$\epsilon_{LOO} \leq \frac{|V_e| + |Z|}{n}. \tag{2}$$

Here, in the numerator we count the number of samples that can change the classifier if one of them is left out of training.

We use the value on the right-hand side of (2). Obviously, there is a trade-off between $|V_e|$ and $|Z|$. The greater is the number of convex hulls, the larger is the size of $|V_e|$, while the smaller is the number of samples that are not included in any convex hull, the smaller is the size of $|Z|$. In addition, a large number of K increases the number of approximate convex hulls because a more acute angle is allowed in a convex hull with a large variety of direction

Table 1: The statistics of datasets.

Dataset	#classes	#attributes	#samples
balance-scale	3	4	625
diadetes	2	8	768
ecoli	8	8	336
glass	6	10	214
heart-statlog	2	13	270
ionosphere	2	34	351
iris	3	4	150
sonar	2	60	208
wine	3	13	178

vectors. We therefore use the right-hand side term of (2) as our criterion.

4.2. Experiments

To construct W of unit vectors, we used n_p positive samples and $n_p(c-1)$ negative samples, so that $K = n_p^2(c-1)$ unit vectors were chosen randomly, where c is the number of classes. In the following, we changed the value of n_p in $[1, 50]$, thus, K in $[c-1, 2500(c-1)]$.

We used 9 datasets taken from the UCI machine learning repository [15]. In Table 1, the number of classes, the number of attributes (the dimensionality of the feature space), and the number of samples are shown. It is noted that in some of them the dimensionality m is too large (34 in `ionosphere` and 60 in `sonar`) to find the exact convex hull of these sizes of the training sample set. We increased the value of K until K reached the maximum value. For each value of K, we repeated the algorithm 10 times to reduce the effect of other random factors. Among 10 trials with a fixed value of K, we chose the best case in which the LOO estimate (2) took the minimum value. The recognition rate was estimated by 10-fold cross validation. The loop number L of the algorithm (Steps 4 and 5) was set to $L = 20$. That is, the number of convex hulls was limited to 20 in each class. The K-ARCH algorithm was implemented in $C++$ and SVMTorch algorithm [17] was used for SVMs. Both algorithms were executed on a PC with Intel Core 2 Quad Q8200 2.33GHz CPU and 3GB RAM.

4.3. Results

We compared the proposed K-ARCH algorithm with an SVM in which an RBF (Radial Basis Function) kernel with the default values of parameters (standard deviation $\sigma = 10.0$ and soft margin parameter $\gamma = 100.0$) was

Table 2: Recognition rates of SVM and K^*-ARCH where K^* is optimal in our model selection criterion and $|V_e|$ is the number of effective vertices.

Dataset	Classifier		#SV or $\|V_e\|$	
	SVM	K^*-ARCH (K^*)	SVM	K^*-ARCH
balance-scale	**93.2**	90.3 (98)	255.0	200.1
diabetes	64.1	**75.0** (2401)	1310.0	483.3
ecoli	79.8	**83.0** (252)	385.7	144.6
glass	**66.3**	63.6 (180)	336.9	138.0
heart-statlog	59.3	**63.7** (2401)	479.4	201.8
ionosphere	**94.0**	90.9 (196)	132.5	331.2
iris	**98.0**	95.3 (18)	54.0	20.2
sonar	77.4	**80.4** (121)	214.0	429.3
wine	72.5	**87.0** (3200)	447.2	67.7
average	78.3	**81.0**	401.6	224.0

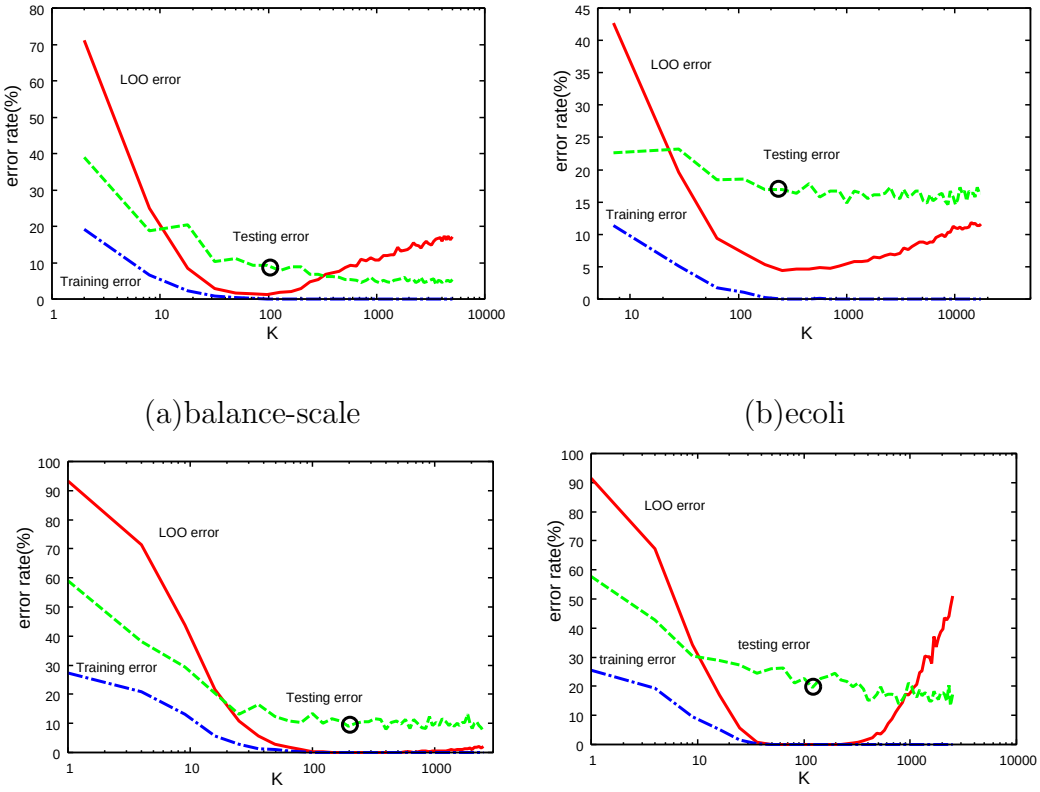


Figure 6: Error rate of K-ARCH as the number K of facets increases on four datasets. The three curves show the estimated LOO error (right-hand term of Ineq. (2)), the training error and testing error. The circled testing-error corresponds to the optimal value of K chosen by the minimum LOO error.

used [17]. For the K-ARCH algorithm, we used K^*-ARCHs for classification, where K^* is the value of K attaining the minimum LOO upper bound of (2). The results are shown in Table 2.

In Table 2, it is noted that K^*-ARCH outperforms SVM in more than half of the cases (5/9). Note that three problems of the remaining 4/9 cases are all easier or well-separated class problems. Indeed, in these three problems (`balance-scale`, `iris` and `ionosphere`), the recognition rates are over 90%. This might mean that K-ARCH tends to generate a slightly more complicated decision boundary compared with that of SVM. Note that a large number K^* is chosen for more difficult problems. This implies that K-ARCH formed a complex boundary. Note also that the number of (effective) vertices is often less than the number of support vectors. This means that K^*-ARCH often has higher sparsity than SVM.

We can see the details in some datasets in Fig. 6. From Fig. 6, we see that after reaching the optimal value K^*, the testing error is no longer significantly reduced. In general, a model selection criterion is expected to form a valley to simulate the testing error, but this is not the case. This implies that K-ARCH does not change its decision boundary even if the model becomes more complicated than necessary. We can interpret this phenomenon as

follows. Even if the facets increase more than necessary, they are limited in the location opposite to the decision boundary. Such a situation is shown in Fig 7. As can be seen in Fig 7, such a redundant non-reflective support plane can be generated by some noisy samples. In this sense, we have to be careful about the value of α used for the judgment of noisy samples. The curve of the testing error fluctuates a little as K increases. This is because small convex hulls with very acute angles can be generated when K is large.

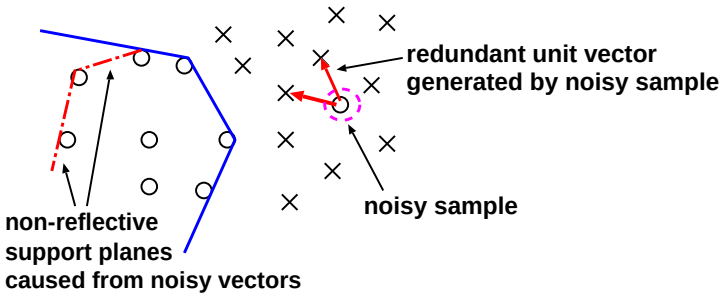


Figure 7: Redundant faces generated by noisy vectors.

5. Comparison with SVM

It is worth comparing the proposed K-ARCH algorithm with SVM to make clear the advantages and disadvantages of K-ARCH algorithm. They are summarized in Table 3.

Table 3: Comparison between the proposed K-ARCH and SVM. In below, n is the number of training samples and m is the dimensionality.

Evaluation item	Classifier	
	SVM	K-ARCH
Maximization of margin	in the kernel space	in the original space
Adaptation to nonlinearlity	kernel	multiple K-ARCHs
Extension to multi-class	one-against-other	natural
Classification performance	high	high
Training time	medium ($O(nm) - O(n^3m)$)	slow ($O(n^2m)$)
Testing time	fast ($O(m)$)	fast ($O(m)$)

The largest difference is the space where the margin is taken. An SVM uses a kernel to solve non-linearly separable problems. It is guaranteed to keep the maximum (linear) margin in the mapped space with the kernel, but the (non-linear) margin in the original feature space is not always maximized. On the other hand, the K-ARCH algorithm keeps (locally linear) maximum margins in the original space, though the margins are measured between closest pairs of K-ARCHs of two different classes. As a result, in some cases, the latter finds a better decision boundary than the former as shown in Fig. 8. Note that the margin is not maximized by SVM in Fig. 8.

Table 4: Execution time of K-ARCH and SVM in training and testing phases. The time (seconds) is the results of 10 trials of the same task.

Dataset	K-ARCH		SVM	
	Training	Testing	Training	Testing
balance	30.40	0.04	1.59	0.07
diabetes	25.03	0.02	1.71	0.22
ecoli	29.36	0.20	0.84	0.11
glass	7.83	0.06	0.95	0.05
heart	3.07	0.01	0.44	0.10
ionos	6.36	0.02	0.44	0.11
iris	1.38	0.00	0.41	0.05
sonar	2.73	0.01	0.50	0.10
wine	2.32	0.02	0.56	0.06

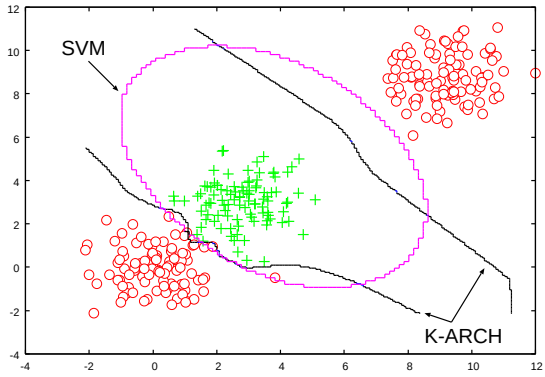


Figure 8: Decision boundaries by SVM with RBF and K-ARCH algorithms. Samples of class 1 are denoted by circles and samples of class 2 by crosses.

The K-ARCH algorithm is also advantageous to SVM in extension to multi-class problems. Typically one-against-other strategy is adopted to use two-class SVM for multi-class problems, though some multi-class SVMs have been studied (e.g., see [18]). It is known, however, that such an extension does not work in some cases [19]. On the contrary, K-ARCH algorithm can naturally deal with multi-class problems because it finds a set of K-ARCHs in each class and classify a sample to the class of the closest K-ARCH.

Note that we can see the example of Fig. 8 as a 3-class problem by regarding two clusters of one class (denoted by circles) as two different classes. Then the problem of inappropriate decision boundary of SVM still remains because of the one-against-other strategy.

In the usage of SVMs, the choice of a kernel and its parameters can be problematic. In this respect, K-ARCH is advantageous because the critical parameter is only the number K of facets.

In time complexity, SVM is superior to K-ARCH algorithm, especially in the training phase. In the testing phase, they are almost the same. The execution time on the nine datasets is shown in Table 4, from which we can confirm that K-ARCH is comparable to SVM and sometimes even faster than SVM in the testing time. It is also noted that parallelization of K-ARCH algorithm is easy due to its simple procedure.

6. Discussion

It is well known, in the case of two-class problems, that the hyper-plane of linear SVM is equivalent to the bisector between the closest points on the boundaries of the convex hulls, each of which encloses the training samples of one class, if the training sample sets are linearly separable [2]. Recently a similar relationship was reported even for soft-margin SVM using *reduced convex hulls* [20, 21]. Our classifier is almost identical to SVM when two classes are linearly separable (e.g., see Fig 4) in which one convex hull is sufficient for one class. An advantage of our approach is the capability of

handling non-linearly separable cases in which more than one convex hull is required for one class. Even in such cases, our K-ARCH algorithm maximizes the margin locally with a relatively smooth decision boundary (see Fig. 8). In this respect, our classifier is one of the maximum margin classifiers.

One drawback of the proposed algorithm is that it includes many random factors producing different classifiers. The direction vectors in W are randomly chosen and the set of K-ARCHs has a randomness because of the random algorithm. This randomness does not affect much to the resultant classifier, but needs an appropriate control to reduce its bad effect.

7. Conclusion

In this paper, a model selection procedure for a family of polyhedron classifiers has been proposed. The family is based on polyhedral class regions close to the convex hulls of some parts of training samples. In the polyhedron family here, complexity mainly comes from the number of facets and vertices of each polyhedral region. The selection procedure employs an upper bound of the LOO error for time reduction and showed satisfactory results in selecting a good model. However, more sophisticated, hopefully theoretically established, procedure to choose the optimal value of K is desired. Another

direction of study is to find another criterion instead of margin maximization. Once nonlinear margins are taken into consideration, it is not easy even to define the nonlinear margins appropriately. As long as considering the geometric margin between training samples of two classes, the largest margin is kept by the nearest neighbor rule. However, it is not the best classifier when only a limited number of samples is given for training. The authors are currently considering a way to widen the margin taken by the proposed algorithm. The result would be helpful for seeking a better criterion.

Acknowledgment

This work was partly supported by Grant-in-Aid for Scientific Research (C) (No. 10213216) of the Japan Society for the Promotion of Science.

References

[1] V. N. Vapnik, The nature of statistical learning theory, Springer, 1996.

[2] D. Zhou, B. Xiao, H. Zhou, R. Dai, Global geometry of svm classifiers, Institute of Automation, Chinese Academy of Sciences, Technical report, AI Lab.

[3] B. V. Dasarathy, Nearest neighbor(NN) norms: NN pattern classification techniques, IEEE Computer Society Press, Los Alamitos, 1991.

[4] M. Kudo, M. Shimbo, Approximation of class region by convex hulls, Technical report of IEICE. PRMU 100 (507) (2000) 1–6, (In Japanese).

[5] E. Jeff, New lower bounds for convex hull problems in odd dimensions, SIAM Journal of Computing 28 (4) (1999) 1198–1214.

[6] B. McBride, G. L. Peterson, Blind data classification using hyper-dimensional convex polytopes, in: Proceedings of the 17th International FLAIRS Conference, 2004, pp. 520–525.

[7] A. Blumer, A. Ehrenfeucht, D. Haussler, W. M.K., Occam's razor, Readings in machine learning (1990) 201–204.

[8] M. Kudo, I. Takigawa, A. Nakamura, Classification by reflective convex hulls, Proceedings of the 19th International Conference on Pattern Recognision (ICPR2008), Tampa, Florida, USA.

[9] G. I. Nalbantov, P. J. F. Groenen, J. C. Bioch, Nearest convex hull classification, Econometric Institute Report.

[10] H. Cevikalp, D. Larlus, M. Neamtu, B. Triggs, F. Jurie, Manifold based

local classifiers: Linear and nonlinear approaches, Journal of Signal Processing Systems (2008) 1–13.

[11] P. K. Ghosh, K. V. Kumar, Support function representation of convex bodies, its application in geometric computing, and some related representations, Computer Vision and Image Understanding 72 (3) (1998) 379–403.

[12] O. L. Mangasarian, Polyhedral boundary projection, SIAM Journal on Optimization, 9 (4) (1999) 1128–1134.

[13] P. Gritzmann, V. Klee, Computational complexity of inner and outer j-radii of polytopes in finite-dimensional normed spaces, Mathematical Programming 59 (1) (1993) 163–213.

[14] D. Decoste, B. Schölkopf, Training invariant support vector machines, Machine Learning 46 (1) (2002) 161–190.

[15] A. Asuncion, D. Newman, UCI machine learning repository (2007).

[16] A. Luntz, V. Brailovsky, On estimation of characters obtained in statistical procedure of recognition, Technicheskaya Kibernetica 3 (6).

[17] R. Collobert, S. Bengio, SVMTorch: support vector machines for large-scale regression problems, Journal of Machine Learning Research 1 (2001) 143–160.

[18] K. Crammer, Y. Singer, On the algorithmic implementation of multi-class kernel-based vector machines, The Journal of Machine Learning Research 2 (2002) 265–292.

[19] Y. Lee, Y. Lin, G. Wahba, Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data, Journal of the American Statistical Association 99 (2004) 67–81.

[20] S. Theodoridis, M. Mavroforakis, Reduced convex hulls: A geometric approach to support vector machines, IEEE Signal Processing Magazine 24 (3) (2007) 119–122.

[21] B. Goodrich, D. Albrecht, P. Tischer, Algorithms for the computation of reduced convex hulls, AI 2009: Advances in Artificial Intelligence (2009) 230–239.