An Online Classifier for Enhancing the Accuracy of Multimedia Data Retrieval

Ding-Ying Chiu
Department of Computer Science
National Tsing Hua University
dr908312@cs.nthu.edu.tw

Arbee L. P. Chen[+]
Department of Computer Science
National Chengchi University
alpchen@cs.nccu.edu.tw

Abstract

The accuracy of multimedia data retrieval can be enhanced by a data classification and feedback mechanism. It is known that constructing a classifier for the multimedia data in high dimensional feature space is time-consuming. For supporting user feedbacks immediately, in this paper we study how to efficiently construct the classifier. Our main idea is to speed up the classifier construction process by employing an indexing strategy. The RCE-network classifier is good for this purpose due to its high accuracy and simple construction process. A new RCE-network construction algorithm which overcomes the defects of the existing algorithms was proposed. Moreover, a pruning method with dimension-independent pruning ability was used to efficiently construct the classifier in the high dimensional feature space. Compared with several existing classification methods, the experiment results show that our method significantly promotes the construction efficiency of the classifier for its online uses.

1. Introduction

With the growing amount of multimedia contents, many multimedia retrieval systems are developed to meet users' needs. In these systems, many techniques are developed to extract features to represent various types of multimedia data. The selected features form a high dimensional feature space in which a point represents a multimedia datum. For efficiently retrieving multimedia data from this feature space, many matching methods and index structures have been developed. Moreover, the techniques of data classification [1][4][7] [9][12][16][17][26] and feedback [12][13][15][21] of the users are used to improve the retrieval

[+]To whom all the correspondence should be sent.

accuracy.

In [12], for retrieving the images to answer users' queries, a binary classifier *SVM* (*Support Vector Machine*) [4][7] is used. It consists of two steps. In the first step, some random sample images are chosen from the database and the user selects the images which satisfy the query from the sample images. In the second step, based on the feedback, all images in the database are classified into two groups: one group satisfies the user's query and the other does not. The former group will become a new database to run the first step. The two steps alternate until the user finds "enough" images which satisfy the query.

The classification task consists of two steps. The first step is the *training step*. In this step, a classifier is constructed using training data. The second step is the *classifying step*. In this step, each test datum is classified into a class by the classifier. In general, the training step is more time-consuming than the classifying step and many researchers [7][14][19] attempt to reduce the cost of the training step. Moreover, in recent years, for more precisely representing the multimedia data, some high dimensional feature models are used. For example, in [10], an image is represented as 1,740 wavelet coefficients. In this case, when the feedback data violate the current data classification, the classifier needs to re-classify the data, which will be very time-consuming in the high dimensional feature space. Therefore, how to build an efficient classification mechanism with high accuracy for the large training data in a high dimensional feature space is a very important subject for research.

In the past, two strategies are often used to address the efficiency problem. The first strategy is *feature selection* and the second strategy is *sampling*. In the feature selection strategy, some suitable features are selected for representing the data to reduce the number of dimensions of the feature space. In [10], the authors design a method to select suitable features for SVM. The main idea of this method is to estimate the importance of each feature. A feature is of low importance if the data classification without considering this feature changes within a given threshold. Based on this idea, a formula is designed to estimate the importance of each feature. In the experiment, the performance of this method is better than that of the other feature selection methods. Moreover, if all the features are used, the result is better than that of the feature selection methods. That is, the feature selection strategy causes accuracy degradation of the classifier.

The concept of the sampling strategy is to pick "appropriate" data such that the classification result of the sampling data is similar to that of the whole data set. Take SVM as an example and see Figure 1. The goal of SVM is to compute a *separating boundary* (the solid line in Figure 1)

such that the data of the two classes can be distinctly separated. A datum is called *support vector* if its distance to the separating boundary is smaller than that of the other data in the same class. Figure 1 shows four support vectors for the separating boundary. Since the data other than the support vectors only cause limited influence for deciding the separating boundary, approaches [19][29] were proposed to sample data which hopefully contain the support vectors to decide the separating boundary. In [29], a method was proposed to prune the data which are "at places far from the boundary." The method recursively samples data to compute the separating boundary. In each iteration, it uses the sampling data to compute a boundary and selects the data which are closer to the boundary as new sampling data for the next iteration. A *hierarchical cluster tree* is used as an index to speed up the sample selection. The authors claim that all support vectors can be found by this method. However, in [19], the authors argue that the above method can only be used for some data distributions because the distance to the boundary is difficult to compute. To address this problem, a *k*-nearest-neighbors based method is proposed. The main idea of this method is that if a datum is close to the boundary, many of its neighbors may have different class labels. Therefore, a datum with many neighbors with different class labels is selected as a sample. This method can be used for any data distribution. However, some support vectors may not be found by this method. For example, given two data sets with different class labels, if the two data sets are remote from each other, no neighbor with a different class label can be found. If some support vectors are not found, the accuracy of the classifier will be degraded.

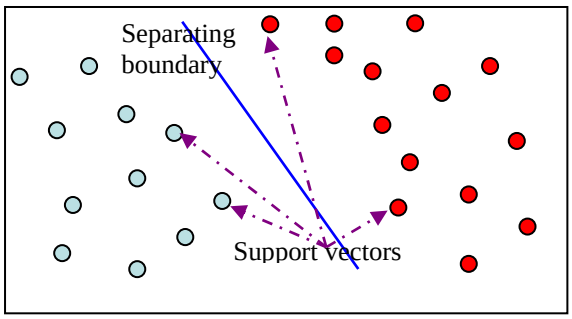


Figure 1. Separating boundary and support vectors

In this paper, we propose an approach to speed up the classifier construction process without degrading the classification accuracy by employing an indexing strategy in the process. For reaching our goal, we select a suitable classifier and analyze each step of its construction to use the indexing strategy. We observe that some queries are needed to be executed in the construction process of the selected classifier, and propose an index method for efficient query processing. Our approach can also employ the feature selection and sampling strategies to further

speed up the classifier construction process.

The remainder of this paper is organized as follows. The classifier selection is discussed in Section 2. Moreover, a new construction algorithm for the selected classifier is also presented in this section. In Section 3, we discuss the index methods and select one to speed up the construction process. Some properties are also presented to enhance the index method for its use in a high dimensional feature space. In Section 4, our method is compared with previous works using real data and synthetic data. Finally, we conclude this paper with future research directions in Section 5.

2. Classifier construction

In this section, we first describe an accurate classifier called *RCE-network* (*restricted coulomb energy network*) [9][17][22] as our selected classifier. In Section 2.2, we introduce some existing construction algorithms of the RCE-network, and point out their shortcomings. Then, in Section 2.3 we propose a new construction algorithm for the RCE-network to overcome the shortcomings of the existing construction algorithms. Finally in Section 2.4, we propose a method to solve a problem of the RCE-network in the classifying step.

2.1. Classifier selection

In [16], many classifiers are introduced such as neural network, SVM, RCE-network, etc. In this subsection, we discuss the advantages and disadvantages of these classifiers and select a suitable classifier for our purpose based on three conditions. The first condition is that the classifier must have a high accuracy. Second, the construction of the classifier must be efficient. Third, the classifier must easily work with an index method. In the following, we use these three conditions to select a suitable classifier.

The neural network methods satisfy the first condition but do not satisfy the second condition. The *convergence property* [16][26] can be used for illustration. In the neural network methods, the error function is often defined as the sum of the differences between the output class labels and the correct class labels for training data. Many neural network methods such as the back-propagation algorithm [14][16][26] must be executed in many iterations to reduce the error. The convergence property says that the error gradually converges to zero. The neural network methods were proved to have the convergence property [16]. However, the research [14][16][26] show the convergence of the neural network methods is slow. It means constructing a neural network is time consuming. Therefore, the neural network methods satisfy the first condition but

do not satisfy the second condition.

Next, we consider SVM. From the research [4][7], it was proved that SVM has the convergence property and converges fast. Therefore, SVM satisfies the first and second conditions. Before examining the third condition, we briefly introduce SVM in the following.

SVM is a binary classifier, i.e. it can only be used to distinguish between two classes. Given the training data (x_1, y_1), (x_2, y_2), …, (x_n, y_n), where y_i is the class label of the object x_i, which can only be 1 (positive data) or -1 (negative data). The goal of SVM is to decide a function [7]:

$$f(x) = w \cdot x + b \qquad (1)$$

such that the following expression is satisfied:

$$\begin{cases} w * x_i + b \geq 1, & if \quad y_i = 1 \\ w * x_i + b \leq -1, & if \quad y_i = -1 \end{cases} \qquad (2)$$

In Formula (1), w is a coefficient vector and b is a bias of $f(x)$. The meaning of Formula (2) is that all training data can be classified by $f(x)$. Obviously, $f(x)$ is a separating boundary of the two classes. Nevertheless, the two classes can be separated by many boundaries, and we must choose the most appropriate boundary from them. The main concept of SVM is to produce a separating boundary which has the maximum margin between the two classes. The concept of the maximum margin is shown in Figure 2, where d^+ denotes the minimum distance from the separating boundary to the nearest positive data; d^- denotes the minimum distance from the separating boundary to the nearest negative data. The maximum margin is defined as the maximum value of the sum of d^- and d^+. Note that the support vectors are the nearest data to the separating boundary. The advantage of the separating boundary with the maximum margin (the boldface line) can be easily observed from Figure 2. The diamond point, which is close to the positive data, will be assigned to the negative data due to a bad separating boundary (the dotted line) which has a value of the sum of d^- and d^+ less than the maximum margin. On the contrary, the diamond point will be correctly assigned to the positive data utilizing the separating boundary with the maximum margin. In [7], the problem of finding the separating boundary with the maximum margin is formulated as an optimization problem with some constraints, and the *Lagrange multiplier method* was used to solve this problem.

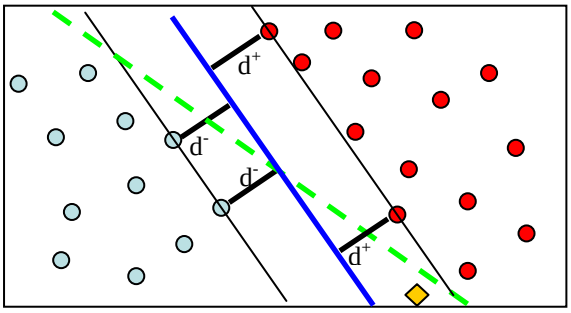


Figure 2. Maximal margin

An important problem of SVM is that the two classes cannot be separated by a separating boundary in many cases. In this situation, the main idea is to transform all training data to another space such that the data of the two classes can be separated in the new space. However, the transformation function between different spaces is difficult to obtain. Therefore, in [7], a kernel method is used to replace the transformation function to estimate the separating boundary in the new space. In [18], an important property is mentioned, which says all data are mapped to the surface of a unit hyper-sphere in the new space. By this property, in [18], when SVM is constructed, an index of the new space can be built by the kernel of SVM to retrieval the top-*k* answers of a query. Unfortunately, since the new space is influenced by the chosen kernel, the index must be rebuilt when the kernel changes.

Based on the above discussion, the two spaces should be considered at the same time when SVM is constructed. However, in the SVM construction process, when the kernel is changed, the new space will be also changed. Therefore, it is hard to build the index structure in the SVM construction process. Therefore, SVM is not suitable to work with an indexing strategy.

Consider the three conditions, the RCE-network [9][17][22] is adopted in our method. In [22], fast convergence of the RCE-network is guaranteed. Therefore, the first condition and second condition are satisfied. Another advantage of the RCE-network is that only simple arithmetic is used when constructing an RCE-network. Therefore, it is easy to work with an indexing strategy.

2.2. The RCE-network and its construction algorithms

In this subsection, we introduce the concepts of the RCE-network and two corresponding construction algorithms [17][22]. The two algorithms will be compared with our algorithm in Section 4.

2.2.1 The structure of the RCE-network

The RCE-network uses circles to cover training data with the following constraints:

(a) For each datum, it must be covered by a circle.

(b) The training data which are covered by a circle must be in the same class.

The structure of the RCE-network is shown in Figure 3. It is divided into three layers: the *input* layer, the *hidden* layer and the *output* layer. Each node of the output layer indicates a class, such that the number of the nodes of the output layer is equal to the number of the classes. Each node of the hidden layer denotes a circle which covers data in a class. From the second constraint, each node of the hidden layer has only one edge to link to the output layer because each circle only covers data in a class. Each node of the input layer represents one dimension of the feature space.

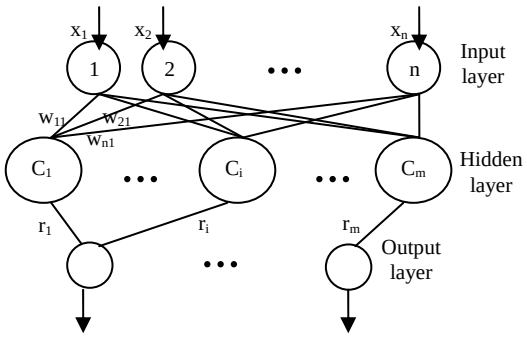


Figure 3. The structure of the RCE-network

The weight of the edge between an input node *j* and a hidden node C_i shows the value of the *j-th* dimension of the center of C_i. Take Figure 3 as an example. The coordinate of the center of the circle C_1 is (w_{11}, w_{21}, ..., w_{n1}). The weight of the edge between a hidden node C_i and an output node denotes the radius of the circle C_i.

When a test datum X with coordinate (x_1, x_2, ..., x_n) is to be classified, the RCE-network computes the distance between X and each C_i using the Euclidean distance $\sqrt{\sum_{k=1}^{n} | x_k - w_{ki} |^2}$ as the formula. If the distance is smaller than or equal to r_i, then X falls into C_i and the output node which is linked by C_i will output one. At the same time, the other nodes of the output layer will output zero. If all the outputs are zero (X does not fall into any circle) or more than one node outputs 1 (X falls into more than one circle), X cannot be correctly classified.

2.2.2 Two existing construction algorithms of the RCE-network

When constructing an RCE-network, the main problem is to efficiently decide proper circles to cover all training data. Two algorithms [17][22] for constructing the RCE-network are introduced in the following. For easy presentation, when given a training datum p, the symbols associated with p are summarized as follows:

Table 1. Symbol table

Symbol	Meaning
$SAME_p$	The set of data whose class is the same as that of p.
$DIFF_p$	The set of data whose classes are different from that of p.
$DIS(a,b)$	The distance between datum a and datum b.
n_p	The nearest datum in $DIFF_p$ to p.
f_p	The farthest datum in $SAME_p$ to p, satisfying $DIS(p,f_p) < DIS(p,n_p)$.

The Rajan's algorithm

For fast satisfying the constraint (a) in Section 2.2.1, when a circle is produced by the Rajan's algorithm [22], it is given a large radius such that the circle can cover many training data. However, the circle with the large radius may cover the training data which have different class labels. Therefore, for satisfying the constraint (b) in Section 2.2.1, the radius of a circle needs to be shrunk.

Based on the above discussion, in addition to the training data, the Rajan's algorithm needs more parameters, i.e. an initial radius φ and a radius reduction rate δ as the inputs. The value of δ is between 0 and 1. The Rajan's algorithm is shown in Figure 4. For each datum p, the algorithm checks whether it is included in any circle. If not, Step 2 will produce a circle whose center is p and its radius is set to φ. If p is already included in a circle C and the class label of the center of C is the same as the class label of p, the algorithm will do nothing. Otherwise, Step 3 will shrink the radius of each circle which covers p and the class label of the center of the circle is different from the class label of p. This algorithm will scan all training data repeatedly until all the circles are not changed.

Input: training data, initial radius φ, radius reduction rate δ ($0 \le \delta < 1$)

Output: An RCE-network

1. For each datum p, execute Steps 2~3.
2. If p is not included in any circle, produce a circle whose center is p and the radius is φ. Go to Step 1.
3. Suppose p is included by a circle C and the center of C is b. If the class label of p is different from the class label of b, then compute $DIS(p, b)$ and set the radius of C to $DIS(p, b)*\delta$.
4. Repeat Steps 1~3 until all the circles are not changed.

Figure 4. The Rajan's algorithm

For each scan, some data which are included in a circle may *escape* from the circle due to radius shrinking. In the next scan, the algorithm produces some circles for these escaped data. Therefore, the main shortcoming of the Rajan's algorithm is that it needs to scan the data many times.

The Wu's algorithm

For satisfying the constraint (a) in Section 2.2.1, the Wu's algorithm [17] produces a circle for each training datum. Moreover, for satisfying the constraint (b) in Section 2.2.1, when given a datum p, the Wu's algorithm finds n_p first. Then the radius of the circle whose center is p is set to $DIS(p, n_p)*\delta$ such that the circle ca not cover any datum in $DIFF_p$.

The algorithm is shown in Figure 5. The Wu's algorithm needs the radius reduction rate δ as the input. The value of δ is between 0 and 1. For each datum p, in Step 2, the algorithm finds n_p and $DIS(p, n_p)$ which are defined in Table 1. In Step 3, a circle is produced for p, and its radius is set to $DIS(p, n_p)*\delta$. Obviously, based on Step 2 and Step 3, all the class labels of the training data in the produced circle are the same, i.e. the radius of all the circles which are produced by the Wu's algorithm will not be shrunk.

Input: training data, radius reduction rate δ ($0 \le \delta < 1$)

Output: An RCE-network

1. For each datum p, execute Steps 2~3.
2. Find n_p and $DIS(p, n_p)$.
3. Produce a circle whose center is p and radius is $DIS(p, n_p)*\delta$.

Figure 5. The Wu's algorithm

In this algorithm, the main computational cost comes from Step 2. Moreover, the algorithm will produce a large number of circles because each datum will be used to produce a circle by the algorithm. The efficiency of the classifier is therefore hampered.

2.3. Our construction algorithm of the RCE-network

The same as the Wu's algorithm, for satisfying the constraint (b) in Section 2.2.1, when given a datum p as the center of the produced circle C, our algorithm finds n_p first. If the radius of C is smaller than $DIS(p, n_p)$, C will not cover any datum in $DIFF_p$. Moreover, for efficiently satisfying the constraint (a), we choose an appropriate radius such that C can cover each datum p' which satisfies the condition that $DIS(p, p')$ is smaller than $DIS(p, n_p)$. Therefore, the radius of C can be set to $DIS(p, f_p)$.

Based on the above discussion, the main steps of our algorithm are shown in Figure 6. In Step 1 and Step 2, we check the relationship between each datum and each circle. In Step 3, if $DIS(b, p) < DIS(b, n_p)$, where b is the center of a circle C, then p is covered by C. In Step 4, if $DIS(b, p)$ is larger than the current radius of C, the radius of C is changed to $DIS(b, p)$. Since Step 4 comes from Step 3, the radius of C must be smaller than $DIS(b, n_p)$. For satisfying constraint (a) in Section 2.2.1, if a datum p which is not covered by any circle is found in Step 5, Step 6 produces a circle for it. For satisfying the constraint (b) in Section 2.2.1, the radius of the circle is set to zero and the $DIS(p, n_p)$ is recorded to limit the radius of this circle. For avoiding the drawbacks of the previous methods discussed in Section 2.2.2 and fast satisfying the constraint (a) in Section 2.2.1, when producing a circle, the circle must cover as many training data as possible in one scan. Our algorithm called the *Radius Expansion* algorithm (RE) expands the radius in Step 4 to effectively reduce the number of circles. Notice that the RE algorithm does not need the parameters of radius reduction rate and initial radius.

Input: training data

Output: An RCE-network

1. For each datum p, execute Step 2.
2. For each circle C, if the class label of its center, b, is the same as that of p, execute Step 3.
3. If $DIS(b, p) < DIS(b, n_p)$, then p is covered by C and execute Step 4. Otherwise, execute Step 2.
4. If $DIS(b, p)$ is larger than the radius of C, set the radius of C as $DIS(b, p)$. Execute Step 1.
5. If p is not covered by any circle, execute Step 6.
6. Find n_p and $DIS(p, n_p)$ and produce a circle whose center is p and radius is zero. Execute Step 7.
7. If each datum is covered by a circle, the algorithm terminates. Otherwise, execute Step 1.

Figure 6. The RE algorithm

Observing the algorithm, Step 5 needs to do a range query and Step 6 needs to do a nearest-neighbor query for finding n_p. An indexing strategy can be employed to speed up these query processing.

2.4 The classification problem of the RCE-network

From Section 2.2.1, the test datum cannot be correctly classified if the outputs of the RCE-network are all zero or if more than one output is 1. This characteristic can be useful in

some applications while in most applications requiring precise classification the RCE-network cannot work well. The following two applications illustrate this characteristic. In [17], the RCE-network is used on human face recognition. In this application, when a picture cannot be recognized, it will be treated as an invader. However, in other applications, each test datum needs to be classified into a class. In handwritten digits recognition, each datum must be a digit and should be classified into its corresponding class. The accuracy is the ratio of the number of data which are correctly classified into their corresponding classes to the number of the data which are classified. Based on our experiment, the accuracy of the RCE-network is 97.26% (the accuracy of SVM is 94.46%). However, there are 10.33% data that cannot be classified by the RCE-network.

For addressing this problem, when a test datum cannot be classified, the idea of the *KNN* classifier can be adopted to adjust the classifying mechanism of the RCE-network. Different from the general KNN classifier which uses all training data to do the classification, the adjusted classifying mechanism only uses the centers of all the circles as a summary to do the classification. Based on the adjusted classifying mechanism, each test datum can be classified into a class. From our experiment the accuracy of the adjusted classifying mechanism of the RCE-network reaches 95.43% when K is set to 7. Although the accuracy of the adjusted classifying mechanism drops from the original accuracy of the RCE-network (97.26%), all the test data can be classified into a class. Moreover, the accuracy of the adjusted classifying mechanism of the RCE-network is better than the accuracy of SVM. The adjusted classifying mechanism is used in our experiments because all the test data of our experiments need to be assigned a class label.

3. Finding a suitable index method

In the RE algorithm, if a circle is produced, a nearest-neighbor query (step 6 in Figure 6) should be executed to find n_p. Therefore, we need to find a suitable index method for efficient query processing in a high dimensional feature space. In Section 3.1, we compare existing index methods to select one of them, and in Section 3.2, we point out the drawbacks of the selected index method to be used in a high dimensional feature space. In Section 3.3, we propose a new pruning method which utilizes the selected index method and two properties to reduce the cost of query processing in the high dimensional feature space.

3.1. Index method selection

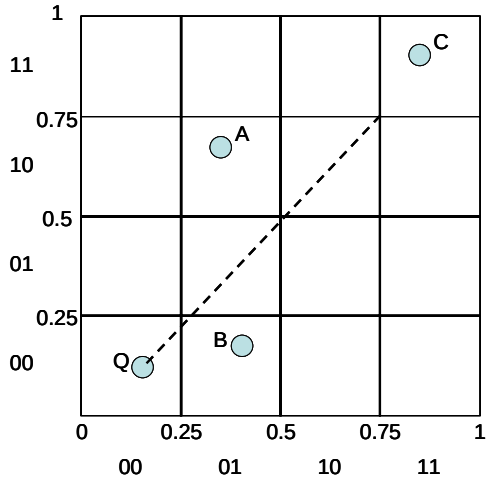
In [18], the index methods are classified into two categories. One is the *coordinate-based* index methods [8][20][23][25][30], and the other is the *distance-based* index methods [2][3][5][27]. In [30], the coordinate-based index methods are further divided into *space foundation* [20][23] and *data foundation* [8][25][30]. We compare these index methods to select a suitable index method for a high dimensional feature space.

The main idea of the coordinate-based index methods is to separate the space into several parts. The major difference between the space foundation and data foundation is the separating criterion. The separating criterion of space foundation is to separate the space such that the size of each subspace or the number of data in each subspace is similar. On the contrary, the data foundation index methods use the rectangle or circle to cover the data. Moreover, these rectangles or circles usually form a hierarchical structure. The space foundation index methods include the *K-D-tree* [11], *K-D-B-tree* [23], and *Bkd-tree* [20]. The data foundation index methods include the *R-tree* [11], *SS-tree* [25], and *cluster tree* [30].

In [24] the authors propose several properties to show that the coordinate-based index methods are not efficient in the high dimensional environment. An important conclusion is that for every space foundation and data foundation index method “there is a dimensionality d such that, on average, all blocks are accessed if the number of dimensions exceeds d” (excerpts from [24]). A data structure *VA-file* is proposed to avoid the large number of disk accesses. As Figure 7 shows, the VA-file separates the space into many blocks and uses a bit vector to represent a block. The bit vector implies the position of the block represented in the high dimensional feature space. For example, the bit vector 1111 represents the block whose lower left corner coordinate is (0.75, 0.75). For each datum, the bit vector of the block which contains the datum is recorded by the VA-file. Therefore, the distance low bound of the query to each datum can be estimated. For example, in Figure 7, the distance low bound of the query q to datum C is 0.829. Moreover, the size of the bit vectors is smaller than the size of the coordinates of the data. Therefore, the VA-file can be loaded into main memory to estimate the distance. In [5], the authors add the polar coordinate information to enhance the pruning ability of the VA-file such that the distance estimation can be more precise.

Although the VA-file reduces the cost of disk accesses in the distance low bound computation, it is useless in our classification task since the training data often can be loaded to the main

memory. In our experiments, all seven real datasets can be loaded to the main memory.



node	coordinate	VA-file
A	0.35, 0.65	01 10
B	0.4, 0.2	01 00
C	0.87, 0.88	11 11
Q	0.2, 0.13	00 00

Figure 7. A VA-file

Formula (3) is the Euclidian distance between two given data q and p in an n dimensional feature space.

$$DIS(q, p) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{3}$$

The square of the distance can be computed by the followed method. First, set the square of the distance to be zero. Then, add the square of the difference one by one between q and p from the *1*-st dimension to the n-th dimension. That is,

$$(DIS(q, p))^2 = (q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2 \tag{4}$$

In [28], the authors proposed a method called *Branch-and-bound ON Decomposed data* (BOND) to avoid scanning all the dimensions to compute the distance for a KNN query. In an n dimensional feature space, BOND scans the first m dimensions to compute a partial distance first. The lower bound and the upper bound of the distance are then computed by the partial distance and the estimate distance of the remaining n-m dimensions. The lower bound and the upper bound of the distance can be used to reduce the search space. This method has a significant drawback, that is, it is difficult to choose a good m.

Next, we introduce the distance-based index methods. The distance-based index methods include *M-tree* [6], and *multiple reference points* (*MRP*) methods [2][3][27]. The distance-based index methods utilize the triangle inequality to prune the distant data. Take the multiple reference points method as an example. First, some data are chosen as the reference points. Then,

the distance between each datum and each reference point is computed and recorded. For any two data p, q and a reference point r, the distance between p and q is larger than or equal to $|pr-qr|$ (triangle inequality), where pr denotes the distance between p and r and qr denotes the distance between q and r. Therefore, given a query q and an error parameter ε, if we want to find all the data b such that $|qb| \leq \varepsilon$, we can prune those data p which satisfies $|pr-qr| > \varepsilon$ for a reference point r. In the method, pr and qr are pre-computed such that the pruning only needs a subtraction. This pruning skill is *dimension-independent* [2], which means the number of operations needed is independent of the number of dimensions. The dimension-independent characteristic is important in some applications. For example, the number of dimensions of the kernel space of SVM may be infinite. In [18], the authors utilize a dimension-independent method to efficiently find the top-k answers of SVM. Since the number of the dimensions of the feature space can be large in many applications, we design a dimension-independent method to find an approximate nearest neighbor for a given query q using the MRP method. The approximate nearest neighbor is then used to reduce the computation of the distance between q and the other data.

3.2. The MRP method and its drawbacks

We adopt the MRP method [2][3][27] as our index structure to improve the efficiency of the RCE-network construction. In Section 3.2.1, we introduce the MRP method and its index structure. In Section 3.2.2, we indicate the drawbacks of the MRP method which can be observed from our experiments.

3.2.1 The MRP method

Two actions must be done to use the MRP method. First, some points are chosen as the reference points. Second, a data structure is used to record the distance between each datum and each reference point. In [27], *B^+-tree* is chosen to record the information. As Figure 8 shows, a pointer array is used to point to the location of each item in the B^+-tree.

Example 3.1:

Take Table 2 as an example, point g and point h are chosen as the reference points. Figure 8 shows the B^+-tree structures of the reference points g and h. The B^+-tree of reference point g stores the distances between g and each other point.

Table 2. The database

Point	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*
Coordinates	(5,5)	(7,8)	(7,5)	(7,6)	(6,6)	(3,7)	(3,8)	(11,11)
Class	1	1	1	1	1	2	2	3

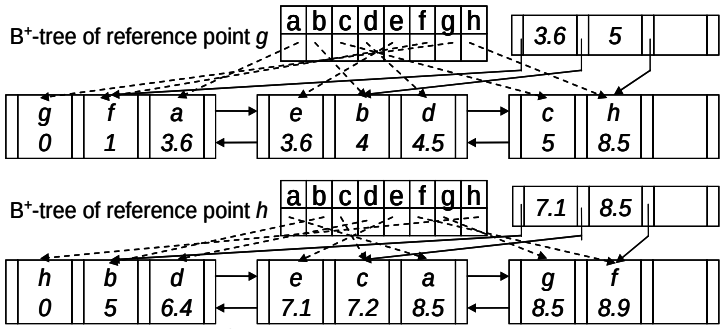


Figure 8. The B^+-tree structure of the reference points

Given a query q and several reference points, if a datum p is close to q, for each reference point r, $DIS(q, r)$ should be close to $DIS(p, r)$. Therefore, based on the B^+-tree structure, the data whose $DIS(p, r)$ are close to $DIS(q, r)$ can be found efficiently.

The main idea of the pruning skill of the MRP method is as follows. Using the triangle inequality property, the distance lower bound and the distance upper bound between the query and each datum can be estimated. Then, the distance lower bound and the distance upper bound are used to decide whether a datum can be pruned [2]. The distance lower bound and the distance upper bound can be estimated as follows. Suppose that p, q, and r are three points in a space. Let r be the reference point. The triangle inequality property is as follows:

$$|DIS(p, r)-DIS(q, r)| \le DIS(p, q) \qquad (5)$$

$$|DIS(p, r)+DIS(q, r)| \ge DIS(p, q) \qquad (6)$$

Formula (5) indicates that the distance between p and q is larger than or equal to the difference of $DIS(p, q)$ and $DIS(q, r)$. Formula (6) indicates that the distance between p and q is smaller than or equal to the sum of $DIS(p, r)$ and $DIS(q, r)$. In the multiple reference points case, the triangle inequality property can be extended as follows [3]:

Property 3.1

Given the multiple reference points $\{r_1, r_2, \ldots, r_m\}$ and a query q. For a point p, the distance between p and q is bounded as follows:

$$\max_{1 \le i \le m} \{| DIS(q, r_i) - DIS(p, r_i) |\} \le DIS(p, q) \qquad (7)$$

$$\min_{1 \le i \le m} \{ | DIS(q, r_i) + DIS(p, r_i) | \} \ge DIS(p, q) \qquad (8)$$

Given a range query whose center is q and the radius h. Based on Property 3.1, we can prune each datum p whose maximum difference between $DIS(q, r_i)$ and $DIS(p, r_i)$ is larger than h, where $1 \le i \le m$.

3.2.2 The drawback of the MRP method

The advantage of the MRP method is that it has the dimension-independent characteristic. However, the characteristic cannot ensure that the MRP method works well in a high dimensional feature space. We use a simple experiment to verify the *pruning rate* of the MRP method. The pruning rate is the ratio of the number of the pruned data to the total number of the data. We vary the number of dimensions from 2 to 30 to create 29 feature spaces. For each feature space, we generate 10,000 uniform data in a unit cube. For each datum, its nearest neighbor is found using the MRP method. We test two strategies, in which Strategy A uses 10 reference points and Strategy B uses 20 reference points. In Figure 9, when the number of dimensions is larger than 18, the average pruning rate of Strategy A is lower than 10%. Moreover, we compare the average pruning rate between Strategy A and Strategy B. In Figure 9, the average pruning rate of Strategy B is better than that of Strategy A. However, the higher average pruning rate cannot ensure shorter execution time. Figure 10 shows that Strategy A is more efficient than Strategy B. Also, we can see that the difference in the execution time of the two strategies increases when the number of dimensions increases. The reason is as follows. Based on Formula (7) and Formula (8), the costs for estimating the distance lower bound and the distance upper bound grow with the number of reference points. For example, in Figure 9, when the number of dimensions is larger than 20, the average pruning rates of both strategies are lower than 10%. However, Strategy B incurs additional computation cost such that its execution time grows faster than that of Strategy A as shown in Figure 10.

The experiments show that the MRP method is useless when the number of dimensions is larger than 30 even though many reference points are used by the method. For overcoming the problem, in Section 3.3, we propose a method which is based on the MRP method and is useful on a high dimensional feature space whose number of dimensions can be larger than 30.

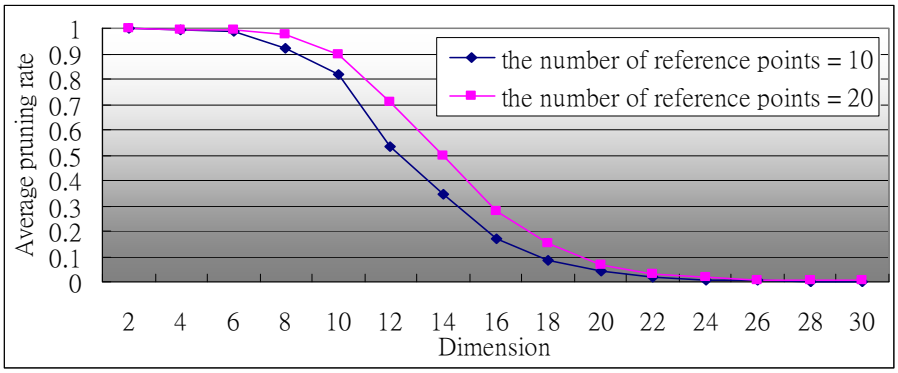


Figure 9. The average pruning rate of the MRP method

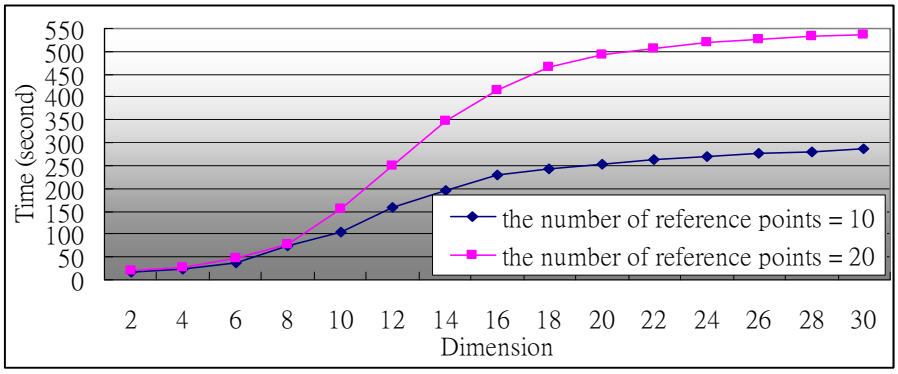


Figure 10. The run time of the MRP method

3.3. The dimension pruning method

In this section, we propose a pruning method which is based on the MRP method for efficient query processing. Based on the above discussion, an important characteristic for designing a high dimensional pruning method is dimension-independent. Our pruning method consists of two steps. In Section 3.3.1, we introduce the first step whose goal is to find an approximate nearest neighbor for the query q using the MRP method. We will show that this step is dimension-independent. In Section 3.3.2, the square of the distance between q and the approximate nearest neighbor is used as a threshold t in the second step to reduce the computation of the distance between q and each datum p. Based on Formula (4), the square of the distance can be computed dimension by dimension. It is called a *partial result* if it is the sum of the square of the difference from the 1-st dimension to the k-th dimension. $k < n$. If the partial result is larger than the threshold t, the datum p cannot become the answer of the query and therefore can be pruned. Moreover, we use some properties and experiments to show that the pruning method is efficient.

3.3.1 Finding an approximate nearest neighbor by multiple reference points

In this section, we first introduce a simple method which is called *random selection* to find the approximate nearest neighbor, which can be easily analyzed. Based on the information of the MRP method, a better method is proposed to find the approximate nearest neighbor.

When given a dataset of size N and a query q, the random selection method randomly selects λ data and computes the distance between q and each selected datum. The datum p which has the minimum $DIS(q, p)$ of all the selected data is set as the approximate nearest neighbor of q. Moreover, we can rank all the data by the distance between q and each selected datum. A datum with a smaller rank means it is closer to q. Therefore, before analyzing the random selection method, we define the *minimum rank* as follows.

Definition 3.1 Minimum Rank

Given a query q, a dataset of size N, and a method to find an approximate nearest neighbor of q by computing the minimum $DIS(q, p)$ from selected λ data. In the dataset, if p is the k-nearest neighbor of q, we say the minimum rank of the selected λ data is k.

Example 3.2:

Take Table 2 as an example, and let a be the query. Suppose we randomly select four data: b, d, g, and h. The $DIS(a, d)$ is smaller than the other three selected data and d is the 3-nearest neighbor of a in the database. Then, the minimum rank of the four selected data is 3.

Based on Definition 3.1, the result of the random selection method can be estimated by the following property.

Property 3.2

Given a query q and a dataset of size N. If the random selection method is adopted to select λ data to find an approximate nearest neighbor of q, the average minimum rank of the found approximate nearest neighbor is $(N+1)/(\lambda+1)$.

Proof:

The average minimum rank is computed by dividing the sum of the minimum ranks of all different selection cases by the total number of all different selection cases. The total number of all different selection cases is C_{λ}^{N}. In these selection cases, the number of the selection cases whose minimum rank of the selected data is 1 is $C_{\lambda-1}^{N-1}$. It means that the nearest neighbor datum is selected and the other λ-1 data can be selected from the remaining N-1 data. The number of the selection cases whose minimum rank of the selected data is 2 is $C_{\lambda-1}^{N-2}$. It means

that the 1-nearest neighbor datum cannot be selected and the 2-nearest neighbor datum is selected and the other λ-1 data can be selected from the remaining N-2 data. Based on the same reason, the number of the selection cases whose minimum rank of the selected data is 3 is $C_{\lambda-1}^{N-3}$, the number of the selection cases whose minimum rank of the selected data is 4 is $C_{\lambda-1}^{N-4}$, and so on. Therefore, the formula of the average minimum rank is as follow:

$$\left(\sum_{i=1}^{N-(\lambda-1)} i \cdot C_{\lambda-1}^{N-i}\right) \div C_{\lambda}^{N} = (N+1)/(\lambda+1) \qquad (9) \qquad \blacksquare$$

The details of Formula (9) can be seen in the Appendix.

From Formula (9), we can see that the random selection method is dimension-independent because the variables are not influenced by the number of dimensions.

Now, we introduce a greedy method which is based on the information of the MRP method to select better data, such that the average minimum rank of the selected data can be smaller than that of the randomly selected data. The method is called *MRP-based selection*. In the following, we first introduce the B^+-tree structure of the MRP method, as shown in Figure 11. For each reference point, a B^+-tree is built and each leaf node of the B^+-tree records a datum and the distance between the reference point and the datum. The B^+-tree is sorted by the distance. Therefore, the first leaf node of each B^+-tree records the reference point itself because the distance between the reference point and itself is zero. For example, in the B^+-tree of reference point j, the first node is j. When given a nearest neighbor query q and q is at the third node of the B^+-tree of reference point j, it means that q is the 2-nearest neighbor of j.

The main idea of MRP-based selection is described as follows. If a datum p is close to q, for each reference point r, $DIS(q, r)$ should be close to $DIS(q, p)$. Based on the idea and the B^+-trees, MRP-based selection first selects those data whose locations are close to q in each B^+-tree. For each reference point, we first find the location of q in its B^+-tree. Then, we select the data from both sides of the location. For example, in Figure 11, we start from the B^+-tree of reference point i. Because the location of q in the B^+-tree is i_3, the first two selected data are i_2 and i_4. The next two selected data are j_1 and j_3. After l_5 and l_6 are selected, if more data are needed, i_1 and i_5 are selected. Here, we do not care the order of reference points selection because the importance of each reference point is the same. The selection process will terminate when the number of selected data reaches a given value λ. For each selected datum p, we compute $DIS(q, p)$. When

the selection process stops, p which has the minimum DIS(q, p) is set as the approximate nearest neighbor of q. Since the information of the dimensions is not considered in the selection process, MRP-based selection is dimension-independent.

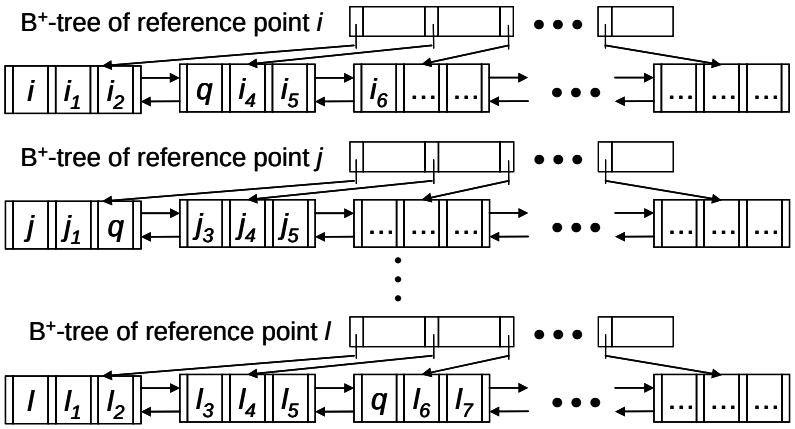


Figure 11. The B^+-tree structure

The following experiment shows that the performance of MRP-based selection is better than that of the random selection method. Three real datasets *USPS*, *MNIST*, and *LETTER* (the three datasets can be obtained from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/) are used in the experiment. The number of dimensions of *USPS* is 256, the number of dimensions of *MNIST* is 784, and the number of dimensions of *LETTER* is 16. Since the number of the dimensions of the three datasets is different, we fix the number of the data of the three datasets for the experiment. We pick 7291 data from *MNIST* and *LETTER* respectively because *USPS* only has 7291 data. In these datasets, we execute an approximate 1-nearest neighbor query for each datum. The average minimum rank of MRP-based selection is shown in Figure 12. The x-axis denotes the number of the selected data of MRP-based selection. It is distinct that our method is dimension-independent since the average minimum ranks of the three datasets whose dimensions are very different are similar when the number of the selected data is 200. Moreover, based on Formula (9), the average minimum rank for the random selection method on 7291 data and 200 selected data is 36.27. The average minimum rank for MRP-based selection on *USPS* under the same setting is 11.47. The result shows that MRP-based selection is better than the random selection method in a high dimensional feature space.

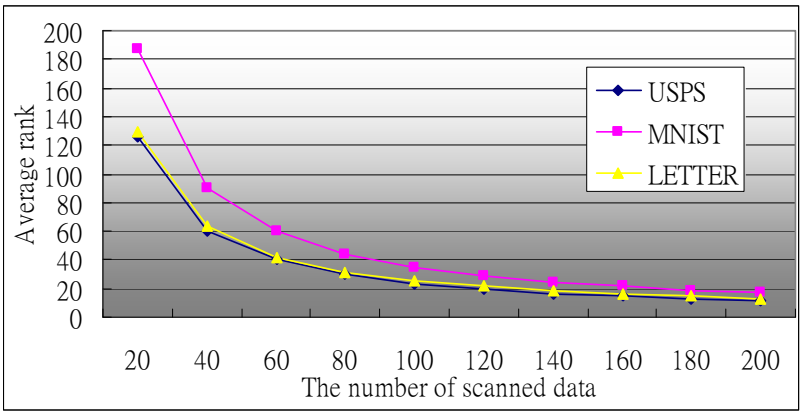


Figure 12. The average minimum rank

3.3.2 The dimension pruning

The approximate nearest neighbor can be used to prune the data which cannot be an answer in the nearest neighbor query. The main idea of the pruning method is described as follows. Given a query q and its approximate nearest neighbor a. For each datum p, when we compute the distance between q and p, the Euclidian distance which is shown in Formula (3) is used. Our pruning method use $(DIS(q, a))^2$ as a constraint. When $\sum_{i=1}^{k} (q_i - p_i)^2 > (DIS(q, a))^2$, where $1 \le k \le n$, p cannot be the nearest neighbor of q since $DIS(q, a)$ is smaller than $DIS(q, p)$. The smallest k is called *the Number of the Computed Dimensions* (*NCD*) of p. Given a data set X, the *Average Number of the Computed Dimensions* (*ANCD*) of X is the sum of the *NCD's* of the data in X divided by the number of the data in X. Although the pruning method is simple, it is useful in a high dimensional feature space to be shown in the following.

In an n dimensional feature space, given a query q and a data set $X=\{x_i | x_i$ is a datum in the space and $DIS(q, x_i)$ is $d\}$. Assume the data in X are uniformly distributed on the surface of a hyper-sphere whose center is q and radius is d. An array with n buckets is used to record the sum of $(q_j-x_{ij})^2$ for all x_i, where j indicates the j-th dimension, which is stored at the j-th bucket. Since the data of X are uniformly distributed on the surface of a hyper-sphere whose center is q and radius is d, the value of each bucket is $t*(d^2/n)$ and the average value of each bucket is $(t*(d^2/n))/t = d^2/n$. Take Figure 13 as an example. There are 5 data which are distributed on the surface of 6 dimensions sphere. We use an array of six buckets to record the sum of $(q_j-x_{ij})^2$ for all x_i for q_j respectively, where $1 \le j \le 6$. The value of each bucket is close to $5*(10^2/6)=83.33$ and the average value of each bucket is close to $10^2/6=16.67$. Therefore, we say that the *average*

value of each dimension is d^2/n because the average value of each dimension is close to d^2/n if the data distribution is close to uniform distribution.

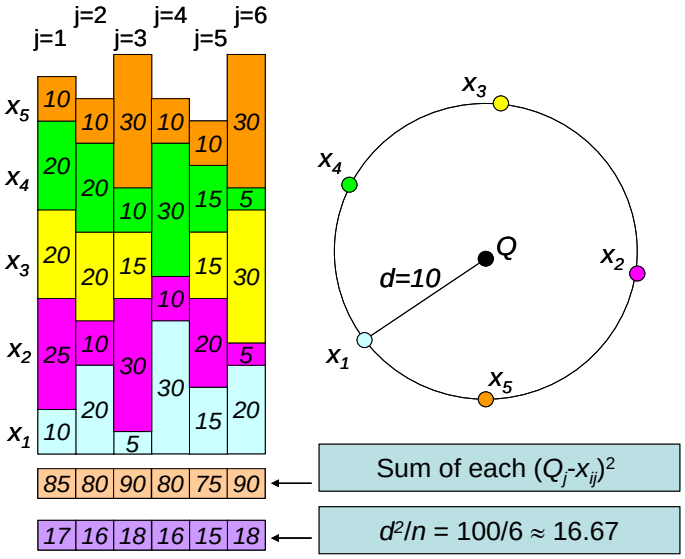


Figure 13. An example of average value of each dimension

Property 3.3

In an n dimensional feature space, given a query q and its approximate nearest neighbor a. Suppose that X and Y are two data sets: X={x| x is a datum in the feature space and $DIS(q, x)$ is d} and Y={y| y is a datum in the feature space and $DIS(q, y)$ is $r*d$}. Moreover, the data of X are uniformly distributed on the surface of a hyper-sphere whose center is q and radius is d. The data of Y are uniformly distributed on the surface of a hyper-sphere whose center is q and radius is $r*d$. If the ANCD of X is m, $m < n$, then the ANCD of Y is $m/(r^2)$.

Proof:

Since the data of X are uniformly distributed on the surface of a hyper-sphere whose center is q and radius is d, the average value of each dimension is d^2/n. Since the ANCD of X is m and the average value of each dimension is d^2/n, we can derive the following formula:

$$m \cdot \frac{d^2}{n} > DIS(q, a) \qquad (10)$$

The data of Y are uniformly distributed on the surface of a hyper-sphere whose center is q and radius is $r*d$, the average value of each dimension is $\frac{r^2 d^2}{n}$. Suppose the ANCD of Y is g, we can derive the following formula:

$$g \cdot \frac{r^2 d^2}{n} > DIS\ (q, a)\ . \qquad (11)$$

By Formula (10) and Formula (11), we have $g \cdot r^2 = m$. Therefore, the ANCD of Y is $g = \frac{m}{r^2}$. ■

Property 3.3 shows that the ANCD will be reduced with the square of the distance between the data and query. We use synthetic data to show Property 3.3. In a 100 dimensional feature space, we generate a query q and its approximate nearest neighbor a, $DIS(q, a)$=90. Then, we produce the data on the surface of five hyper-spheres. The center of the five hyper-spheres is q and their radii range from 100 to 500. For each hyper-sphere, 100,000 data are produced whose distribution is uniform on its surface. Figure 14 shows the pruning rate of the dimension pruning method in each dimension, i.e., the ratio of the number of the pruned data in the dimension to the total number of the data. The ANCD of the hyper-sphere whose radius is 100 is about 82 because the peak of the blue line is about 82. Based on the same reason, the ANCD of the hyper-sphere whose radius is 200 is about 21. The ANCD of the hyper-sphere whose radius is 300 is about 9. The ANCD of the hyper-sphere whose radius is 400 is about 5. The ANCD of the hyper-sphere whose radius is 500 is about 3. By Property 3.3, if the ANCD of the hyper-sphere whose radius is 100 is about 82, then the ANCD of the hyper-sphere whose radius is 500 is 82/(5^2)=3.28. The experiment result is close to the claim stated in Property 3.3.

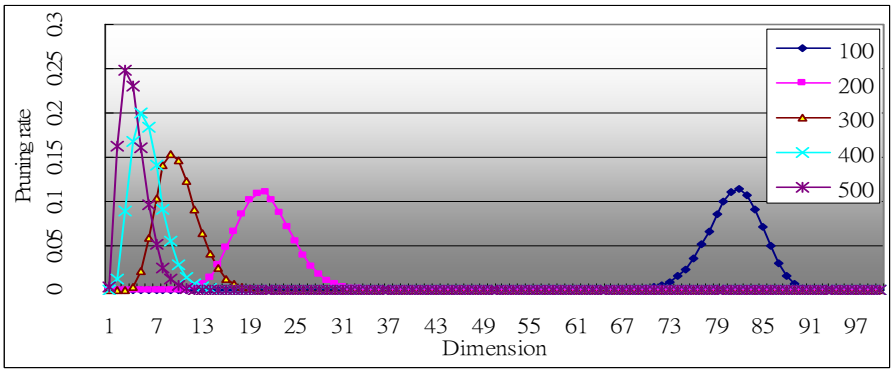


Figure 14. The average pruning rate for each dimension

Based on Property 3.3, we know that choosing a good parameter m for BOND [28], which is discussed in Section 3.1, is difficult because the data are pruned at different dimensions which are dependent on the distances between the query and the data.

Our pruning method can also be used to address the range query, as shown in Property 3.4.

Property 3.4

In an n dimensional feature space, given a query q and a radius d. Suppose that X and Y are two data sets: $X=\{x|\ x$ is a datum in the space and $DIS(q, x)$ is $d+\varepsilon\}$ and $Y=\{y|\ y$ is a datum in the space and $DIS(q, y)$ is $r*(d+\varepsilon)\}$, where $\varepsilon > 0$. Moreover, the data of X are uniformly distributed on the surface of a hyper-sphere whose center is q and radius $d+\varepsilon$. The data of Y are uniformly distributed on the surface of a hyper-sphere whose center is q and radius $r*(d+\varepsilon)$. If the ANCD of X is m, then the ANCD of Y is $m/(r^2)$.

Proof:

Since the data of X are uniformly distributed on the surface of a hyper-sphere whose center is q and radius is $d+\varepsilon$, m is smaller than n. Based on Property 3.3, Property 3.4 is proved. ■

In Property 3.4, if ε is very small, we know that our pruning method is also efficient in the range query.

Property 3.3 describes the pruning ability of the dimension pruning method when the distribution of data is uniform. However, the uniform distribution is not a good distribution for our dimension pruning method. If the distribution of the data is not uniform, we can analyze the variance of each dimension and change the computation order of the Euclidian distance. If the variance of a dimension is zero, the values of all the data in the dimension are the same, i.e., the average distance of the data in the dimension is zero. On the contrary, the large variance of a dimension implies that the average distance of each datum in the dimension is large. Therefore, we can adjust the computation order of the Euclidian distance such that the dimensions whose variances are large are computed first. The original dimension pruning method is called the *direct dimension pruning* and the adjusted dimension pruning method is called the *variance affected dimension pruning*. We use the experiments to show the pruning ability of these two dimension pruning methods in Section 4.2. Moreover, the variance affected dimension pruning method is used in all the experiments on real databases since the distributions of real databases are not uniform.

4. Experiments

In this section, SVM and three RCE-network construction algorithms are executed to observe their training times and accuracy. In the experiments, we implement the Wu's algorithm, the Rajan's algorithm, and the RE algorithms. We also compare three RCE-network algorithms with *LIBSVM*, which can be found in the following webpage:

http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html. The real data and synthetic data are both used to test the above algorithms. The experimental results of all the algorithms on the synthetic data are taken to observe the performance of each algorithm with various parameters. The experimental results of all the algorithms on the different real databases can reflect the performance of each algorithm in the real environment. The experiments are made upon the Intel Pentium 4 CPU 2.8GHz with 1024 MB main memory and Microsoft Windows XP Professional. In Section 4.1, we discuss the experiments on synthetic data and the experiments on real data will be discussed in Section 4.2.

4.1. The experiments on synthetic data

The data generator generates the synthetic data in a hype-cube. The parameters of the data generator are shown in Table 3. We simply introduce the main steps of the data generator. First, the data generator randomly produces the center *C* of each class. Then, for a center *C*, the shortest distance *S* between *C* and the centers of the other classes is computed. In order to avoid the generated data from being covered by the other classes, the radius of the class is set to 0.49**S*. The data generator randomly produces *N* data in a circle whose center is *C* and radius is 0.49**S*.

Table 3: The descriptions of major parameters

Parameter	**Description**
D	The number of dimensions
C	The number of classes
N	The number of nodes of each part

In the experiments, all programs are executed on three varied synthetic data. The parameter settings of the three varied synthetic data are shown in Table 4. First, we increase the number of data to obverse the variation of the training time. We fix the parameters *D* and *C*, and vary *N* such that the data size are generated from 5000 to 50000. The results are shown in Figure 15. The training time of all the algorithms grows linearly with data size except the Wu's algorithm. For each datum *q*, the Wu's algorithm must execute a *1*-nearest neighbor query for *q* to find the nearest datum whose class is different from that of *q*. Therefore, the training time of the Wu's algorithm is worse than that of the other algorithms.

Table 4: The parameter settings

	D	***C***	***N***	***Data size***
Varied *Data size*	*500*	*20*	*10~100*	*5000~50000*
Varied *Number of Dimensions*	*200~2000*	*20*	*20*	*10000*
Varied *Number of Classes*	*1000*	*100~1000*	*180~18*	*About 18000*

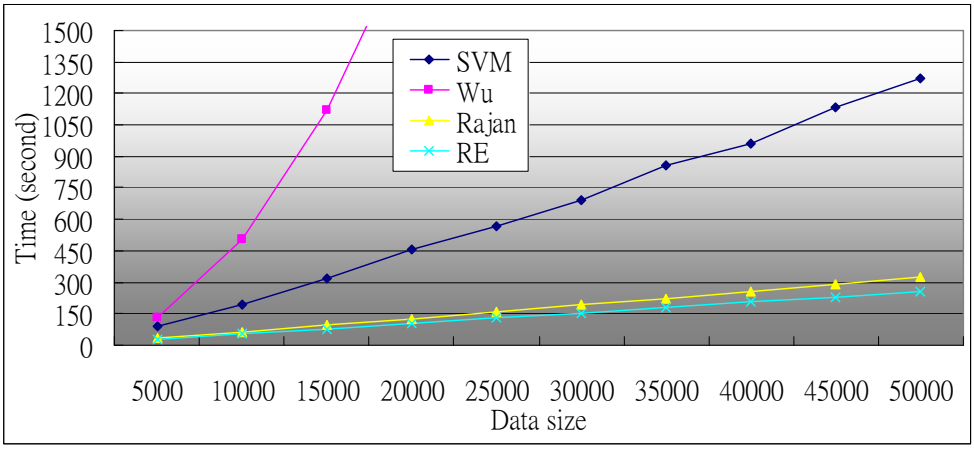


Figure 15. Varied data size

Second, we fix the parameters *C* and *N*, and vary *D*. The results are shown in Figure 16. The training time of all the algorithms grows linearly with the number of dimensions. Third, we fix the parameters *D* and *N*, and vary C. The results are shown in Figure 17. Almost all the training time of the algorithms grows linearly with the number of classes except the Wu's algorithm. Observe Figure 17, the Wu's algorithm is not influenced by the number of classes because the Wu's algorithm must produce a circle for each training datum no matter what the number of classes is.

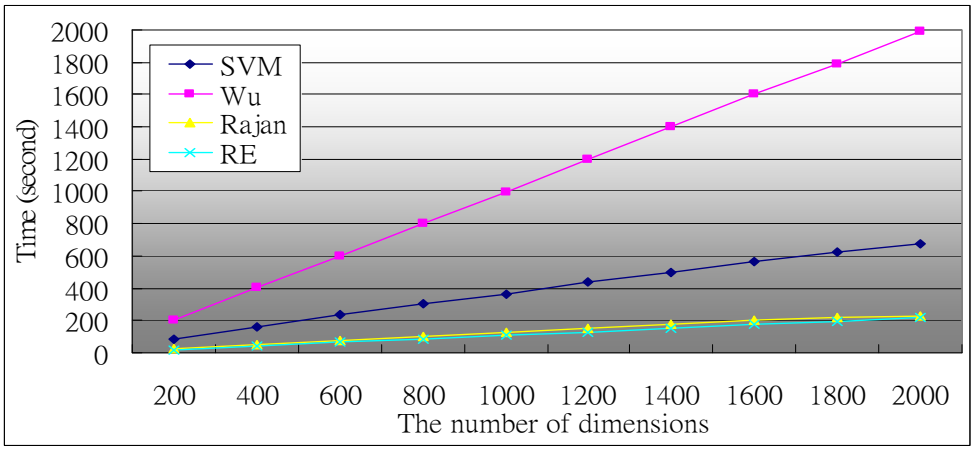


Figure 16. Varied the number of dimensions

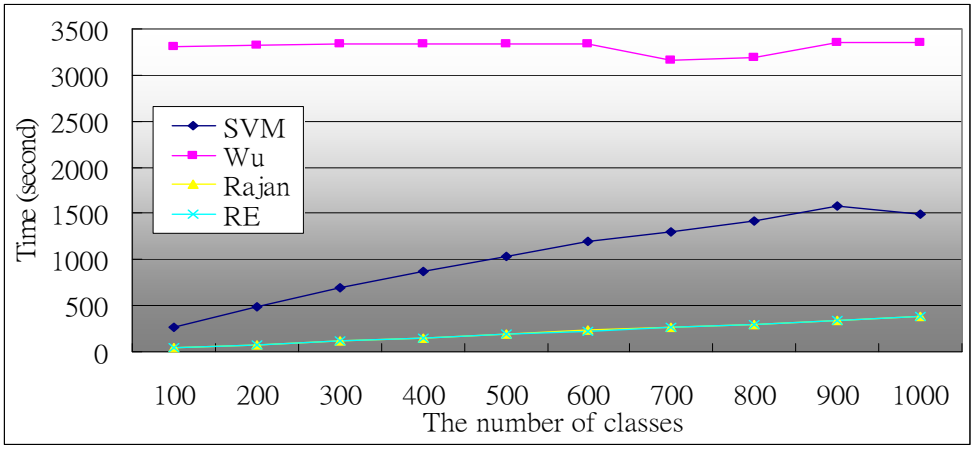


Figure 17. Varied the number of classes

4.2. The experiments on real data

Executing all algorithms on real data to test their performance is necessary because the distributions of the real data are very different from the synthetic data. The different distributions will cause the different performances of each algorithm. Seven real databases are used to test the training time and the accuracy of all the algorithms. The data are collected from different applications and the parameters of each database are listed in Table 5. For example, *LETTER* is a handwritten letter database. *MNIST* is a handwritten digital database. *ML FACE* and *UM FACE* are the human face databases. The first five data sets can be got from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. *ML FACE* can be got from http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/faces.html. *UM FACE* can be got from http://images.ee.umist.ac.uk/danny/database.html.

Table 5: The parameters of real data sets

Dataset	Class	Training size	Testing size	The total number of dimensions
SHUTTLE	7	43,500	14,500	9
LETTER	26	15,000	5,000	16
IJCNN	2	49,990	91,701	22
USPS	10	7,291	2,007	256
MNIST	10	60,000	10,000	780
ML FACE	20	564	60	15,360
UM FACE	20	912	100	32,400

First, we compare the pruning abilities of our pruning methods discussed in Section 3.3 on the real data. For each real database, we randomly pick 500 data to find the *1*-nearest neighbor for them respectively. The pruning results are shown in Table 6. The ANCD is defined in Section 3.3.2. For example, in *UM FACE*, it shows that after about 2562.92 dimensions are computed for

a datum, we know whether the datum is the answer, i.e., 92.1% dimensions need not be computed. In Table 6, the *average dimension pruning rate* (*ADPR*) shows the ratio of the ANCD to the total number of the dimensions. Moreover, in Section 3.3, the variance affected dimension pruning is presented. The ANCD of the variance affected dimension pruning and the ADPR of the variance affected dimension pruning are also shown in Table 6. The experiment shows that the variance affected dimension pruning is better than the direct dimension pruning in six real databases. The ADPR's of the variance affected dimension pruning method are larger than 87% in the seven real databases. Since the data pruning rate is the ratio of the number of data which are pruned to the total number of the data, in our viewpoint, the ADPR is equivalent to the data pruning rate. For example, a database has 300,000 data and the number of the dimensions of its feature space is 2,000. When given a *1*-nearest neighbor query, if 87% dimensions are not computed between the query and each datum, the computation cost is 300,000*260 because the query and each datum should compute 260 dimensions. If 87% data are pruned, the computation cost is 39,000*2,000 because the query and 13% data should compute 2,000 dimensions. Since 300,000*260 is equal to 39,000*2,000, the ADPR and the data pruning rate are equivalent. Therefore, the experiments show that the dimension pruning method is useful in the high dimensional feature space.

Table 6: The average pruning dimension for *1*-NN query

Database	***SHUTTLE***	***LETTER***	***IJCNN***	***USPS***	***MNIST***	***ML FACE***	***UM FACE***
The total number of dimensions	9	16	22	256	780	15360	32400
The ANCD of direct dimension pruning	1.16	2.32	1.12	51.41	45.7	5270.48	2562.92
The ADPR of direct dimension pruning	87.1%	85.5%	94.9%	79.9%	94.1%	65.7%	92.1%
The ANCD of variance affected dimension pruning	1.11	1.73	1.06	26.01	46.18	1041.5	1410.92
The ADPR of variance affected dimension pruning	87.7%	89.2%	95.2%	89.8%	94.1%	93.2%	95.6%

Second, we discuss the accuracy of each algorithm. The accuracy of each algorithm at each database is shown in Table 7. To address the classification problem of the RCE-network which is mentioned in Section 2.4, the *KNN* classifier is adopted to adjust the classifying mechanism of the RCE-network. In the experiments, *k* is 7. Compare the accuracy of the RE algorithm with

SVM on the seven databases, the RE algorithm only loses 1.44% accuracy on *USPS*. However, the accuracy of the RE algorithm is better than that of SVM 10.86% on *LETTER*.

Table 7: The accuracy of each algorithm on each database

Database / Algorithm	***SHUTTLE***	***LETTER***	***IJCNN***	***USPS***	***MNIST***	***ML FACE***	***UM FACE***
RE	99.16%	93.10%	94.90%	92.33%	95.43%	98.33%	99%
Rajan	99.74%	93.98%	95.69%	93.37%	96.40%	98.33%	99%
Wu	99.90%	95.80%	97.37%	95.32%	97.04%	98.33%	100%
SVM	97.61%	82.24%	92.79%	93.77%	94.46%	98.33%	94%

In Table 7, the accuracy of the Wu's algorithm are better then that of the Rajan's algorithm, and the accuracy of the Rajan's algorithm are better then that of the RE algorithm. However, the differences between the accuracy of the RE algorithm and that of the Wu's algorithm are all within 3% in the seven databases. The number of circles of the RCE-network can be used to explain why the accuracy of the RE algorithm is worse than that of the other two RCE-network construction algorithms. The number of circles indicates how many circles are used by the RCE-network construction algorithms to cover all the training data. The RCE-network can be considered as using the circles to represent the training data. In general, the representation capability is better when the number of circles is larger. In Table 8, the numbers of circles of the three RCE-network algorithms on each database are listed. Clearly, the number of circles of the Wu's algorithms is larger than that of the Rajan's algorithm and the number of circles of the Rajan's algorithms is larger than that of the RE algorithm. Therefore, the accuracy of the Wu's algorithm is better than that of the Rajan's algorithm and the RE algorithm. However, two drawbacks are caused by the large number of circles. First, the training time grows with the number of circles. Second, when a test datum will be classified, the large number of circles causes the huge computation cost to compute the distances between the center of all the circles and the test datum to decide the class label of the test datum. The first drawback can be observed from Table 9. The training time of the Rajan's algorithm and the Wu's algorithm are worse than that of the RE algorithm. The training time of the Rajan's algorithm is about 9.72 times of that of the RE algorithm on *IJCNN*. The training time of the Wu's algorithm is about 78.24 times of that of the RE algorithm on *SHUTTLE*.

Table 8: The number of circles of the three RCE-network algorithms on each database

Database / Algorithm	*SHUTTLE*	*LETTER*	*IJCNN*	*USPS*	*MNIST*	*ML FACE*	*UM FACE*
RE	192	2863	5375	849	7106	88	95
Rajan	230	3514	6281	1194	10791	98	115
Wu	43500	15000	49990	7291	60000	564	912

For comparing the training time of the three RCE-network construction algorithms with that of SVM, the results also can be seen in Table 9. In the best case, the training time of SVM is about 49.65 times of that of the RE algorithm on *SHUTTLE*. In the high dimensional database *UM FACE*, the training time of SVM is also about 9.81 times of that of the RE algorithm. Moreover, in the seven real databases, the performance of the RE algorithm is better than that of SVM.

Table 9: The training time (second) of each algorithm on each database

Database / Algorithm	*SHUTTLE*	*LETTER*	*IJCNN*	*USPS*	*MNIST*	*ML FACE*	*UM FACE*
RE	1.937	14.422	67.687	12.328	1377.531	10.328	37.406
Rajan	5.938	68.313	657.906	52.297	3753.938	24.828	89.861
Wu	151.547	51.313	155.281	136.25	7893.406	81.75	387.782
SVM	96.172	49.172	155.047	22.859	1380.063	65.375	366.86

In the synthetic data and real data, the training time and the accuracy of the RE algorithm are almost all better than that of SVM. To compare with the Wu's algorithm and the Rajan's algorithm, the differences between the accuracy of the RE algorithm and that of the two RCE-network construction algorithms are all within 3% in the seven databases. Moreover, in the best case of the seven real databases, the training time of the Wu's algorithm is about 78.24 times of that of the RE algorithm. Based on these experimental results, the performance of the RE algorithm is almost better than all the other algorithms.

5. Conclusion

In this paper, we propose a new RCE-network construction algorithm. Moreover, for reducing the computation cost in the high dimensional feature space, we also propose a new pruning method which is based on the MRP method. Two properties are used to explain why the pruning method works well in the high dimensional feature space. In the experiments, we compare the RE algorithm with SVM and two past RCE-network construction algorithms. The results show an outstanding performance of the RE algorithm in training time and accuracy.

In the following, we discuss how to improve the dimension pruning method and RE algorithm and how to use them in other situations.

(a) **Enhance the dimension pruning method considering disk accesses.** The dimension pruning method assumes all training data are loaded into the main memory for classification. However, in many applications, disk accesses are required. How to reduce the number of disk accesses using our pruning method is a problem to solve.

(b) **Utilize the dimension pruning method for other tasks.** The dimension pruning method is efficient in the high dimensional feature space. It can be used for other tasks such as clustering and similarity search. For example, in the clustering task, when given a new datum q, q should be assigned to a cluster. In general, q is assigned to a cluster whose center is closest to q. Therefore, the dimension pruning method can be used to address the nearest neighbor query.

(c) **Utilize the RCE-network to solve the classification problem in a streaming environment.** The RCE-network produces the circles as the summary to represent all the training data. This idea can be adopted such that the circles can be a suitable summary to keep the huge number of streaming data. Moreover, RE algorithm can be adjusted to meet the classification requirement in the streaming environment. We are currently working on employing our RE algorithm and the pruning method to classify data in a streaming environment.

Reference

[1] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu, “A Framework for On-Demand Classification of Evolving Data Streams,” *IEEE Transactions on Knowledge and Data Engineering*, 18(5): 577-589, 2006.

[2] J. Barros, J. French, W. Martin, P. Kelly, and M. Cannon, “Using the Triangle Inequality to Reduce the Number of Comparisons Required for Similarity-based Retrieval,” *International Conference on Storage and Retrieval for Image and Video Databases*, pp. 392-403, 1996.

[3] A. Berman and L. Shapiro, “Efficient Image Retrieval with Multiple Distance Measures,” *International Conference on Storage and Retrieval for Image and Video Databases*, pp. 12-21, 1997.

[4] C.J.C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition,” *Data Mining and Knowledge Discovery*, 2(2): 121-167, 1998.

[5] G.H. Cha, X. Zhu, D. Petkovic, and C.W. Chung, “An Efficient Indexing Method for Nearest Neighbor Searches in High-Dimensional Image Database,” *IEEE Transactions on Multimedia*, 4(1):76-87, 2002.

[6] P. Ciaccia, M. Patella, and P. Zezula, “M-tree: An efficient access method for similarity search in metric spaces,” *Proc. of International Conf. on Very Large Data Bases*, pp. 426-435, 1997.

[7] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Machine Learning*, 20(3): 273-197, 1995.

[8] B. Cui, B.C. Ooi, J. Su, and K.L. Tan, “Indexing High-Dimensional Data for Efficient In-Memory Similarity Search,” *IEEE Transactions on Knowledge and Data Engineering*, 17(3): 339-353, 2005.

[9] A. Durg, W.V. Stoecker, J.P. Cookson, S.E. Umbaugh, and R.H. Moss, “Identification of Variegated Coloring in Skin Tumors: Neural Network vs. Rule-based Induction Methods,” *IEEE Engineering in Medicine and Biology Magazine*, 12(3): 71-74, 98, 1993.

[10] T. Evgeniou, M. Pontil, C. Papageorgiou, and T. Poggio, “Image Representation and Feature Selection for Mulitmedia Database Search,” *IEEE Transactions on Knowledge and Data Engineering*, 15(4): 911-920, 2003.

[11] V. Gaede and O. Günther, “Multidimensional Access Methods,” *ACM Computing Surveys*, 30(2): 170-231, 1998.

[12] K. Goh, E.Y. Chang, and K.T. Cheng, “SVM Binary Classifier Ensembles for Image

Classification," *ACM Conference on Information and Knowledge Management*, pp. 395-402, 2001.

[13] D.H. Kim and C.W. Chung, "Qcluster: Relevance Feedback Using Adaptive Clustering for Content-Based Image Retrieval," *ACM SIGMOD Conference*, pp. 599-610, 2003.

[14] J. K. Kruschke and J. R. Movellan, "Benefits of Gain: Speed Learning and Minimal Hidden Layers in Back-propagation Networks", *IEEE Transaction on systems, Man and Cybernetics*, 21(1): 273-280, 1991.

[15] Y. Lu, H. Zhang, L. Wenyin, and C. Hu, "Joint Semantics and Feature Based Image Retrieval Using Relevance Feedback," *IEEE Transactions on Multimedia*, 5(3): 339-347, 2003.

[16] T.M. Mitchell, "Artificial Neural Networks," *Machine Learning*, pp. 81-127, McGraw Hill, 1997.

[17] X. Mu, M. Artiklar, M.H. Hassoun, and P. Watta, "An RCE-based Associative Memory with Application to Human Face Recognition," *INNS-IEEE International Joint Conference on Neural Networks*, 4: 2552-2557, 2003.

[18] N. Panda and E.Y. Chang, "KDX: An Indexer for Support Vector Machines," *IEEE Transactions on Knowledge and Data Engineering*, 18(6): 748-763, 2006.

[19] N. Panda, E.Y. Chang, and G. Wu, "Concept Boundary Detection for Speeding up SVMs," *International Conference on Machine Learning*, pp. 681-688, 2006.

[20] O. Procopiuc, P.K. Agarwal, L. Arge, and J.S. Vitter, "Bkd-tree: A Dynamic Scalable Kd-tree," *International Symposium on Spatial and Temporal Databases*, pp. 46-65, 2003.

[21] F. Qian, M. Li, H.J. Zhang, W.Y. Ma, and B. Zhang, "Alternating Feature Spaces in Relevance Feedback," *Multimedia Tools and Applications*, 21(1): 35-54, 2003.

[22] V. Rajan, J. Ying, S. Chakrabarty, and K. Pattipati, "Machine Learning Algorithms for Fault Diagnosis in Analog Circuits," *IEEE Conference on Systems*, 2: 1874-1879, 1998.

[23] J.T. Robinson, "Physical Storage Structures: The K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indexes," *ACM SIGMOD Conference*, pp. 10-18, 1981.

[24] R. Weber, H.J. Schek, and S. Blott, "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Space," *Proc. of International Conf. on Very Large Data Bases*, 1998.

[25] D.A. White and R. Jain, "Similarity Indexing with the SS-tree," *IEEE Conference on Data Engineering*, pp. 516-523, 1996.

[26] B.M. Wilamowski, S. Iplikci, O. Kaynak, and M. Önder Efe, "An Algorithm for Fast

Convergence in Training Neural," *INNS-IEEE International Joint Conference on Neural Networks*, pp.1778-1782, 2001.

[27] C. Yu, B.C. Ooi, K.L. Tan, and H.V. Jagadish, "Indexing the Distance: An Efficient Method to KNN Processing," *Proc. of International Conf. on Very Large Data Bases*, pp. 421-430, 2001.

[28] A. P. Vries, N. Mamoulis, N. Nes, and M. Kersten, "Efficient k-NN Search on Vertically Decomposed Data," *ACM SIGMOD Conference*, pp. 322-333, 2002.

[29] H. Yu, J. Yang, J. Han, and X. Li, "Making SVMs Scalable to Large Data Sets using Hierarchical Cluster Indexing," *Data Mining and Knowledge Discovery*, 11(3): 295-321, 2005.

[30] D. Yu and A. Zhang, "ClusterTree: Integration of Cluster Representation and Nearest-neighbor Search for Large Data Sets with High Dimensions," *IEEE Transactions on Knowledge and Data Engineering*, 15(5): 1316-1337, 2003.

Appendix

The detailed proof of Formula (9) is as follows.

$$T = \frac{\sum_{i=1}^{N-(\lambda-1)} i \cdot C_{\lambda-1}^{N-i}}{C_{\lambda}^{N}}, \quad S = \sum_{i=0}^{N-\lambda+1} i \cdot C_{\lambda-1}^{N-i}$$

$$S = \sum_{i=0}^{N-\lambda+1} (N+1-(N+1-i)) C_{\lambda-1}^{N-i}$$

$$= \sum_{i=0}^{N-\lambda+1} (N+1) C_{\lambda-1}^{N-i} - \sum_{i=0}^{N-\lambda+1} (N+1-i) C_{\lambda-1}^{N-i}$$

$$\left(\begin{array}{l} \because \quad a C_{a}^{b} = b C_{a-1}^{b-1} \\ \therefore \quad \text{Let } a = \lambda, b = N+1-i \\ \qquad \Rightarrow (N+1-i) C_{\lambda-1}^{N-i} = \lambda \cdot C_{\lambda}^{N+1-i} \end{array} \right)$$

$$= (N+1) \cdot \sum_{i=0}^{N-\lambda+1} C_{\lambda-1}^{N-i} - \lambda \cdot \sum_{i=0}^{N-\lambda+1} C_{\lambda}^{N+1-i}$$

$$= (N+1) \cdot A - \lambda \cdot B$$

where $A = \sum_{i=0}^{N-\lambda+1} C_{\lambda-1}^{N-i}, \quad B = \sum_{i=0}^{N-\lambda+1} C_{\lambda}^{N+1-i}$

$$\left(\begin{array}{l} \because \quad \sum_{0 \le i \le N-\lambda+1} C_{\lambda}^{N+1-i} = \sum_{0 \le N+1-i \le N+1-i} C_{\lambda}^{N+1-(N+1-i)} \\ \qquad = \sum_{\lambda \le i \le N+1} C_{\lambda}^{i} \\ \qquad = \sum_{0 \le i \le N+1} C_{\lambda}^{i} \\ \therefore \quad B = \sum_{0 \le i \le N+1} C_{\lambda}^{i} = C_{\lambda+1}^{N+2} \end{array} \right)$$

$$S = (N+1) \cdot A - \lambda \cdot B$$

$$= (N+1) \cdot C_{\lambda}^{N+1} - \lambda \cdot C_{\lambda+1}^{N+2}$$

$$= (N+1 - \lambda \cdot \frac{N+2}{\lambda+1}) \cdot C_{\lambda}^{N+1}$$

$$= (\frac{N-\lambda+1}{\lambda+1}) \cdot C_{\lambda}^{N+1}$$

$$
\begin{aligned}
T &= S \Big/ C_{\lambda}^{N} \\
&= \left(\frac{N-\lambda+1}{\lambda+1} \cdot C_{\lambda}^{N+1}\right) \Big/ C_{\lambda}^{N} \\
&= \left(\frac{N-\lambda+1}{\lambda+1}\right) \cdot \frac{(N+1)!}{(N-\lambda+1)!\,\lambda!} \Big/ \frac{N!}{(N-\lambda)!\,\lambda!} \\
&= \frac{N+1}{\lambda+1}
\end{aligned}
$$