Clustering-based Nearest Neighbor Searching

Ping Ling
College of Computer Science and Technology, Jiangsu Normal University, Xuzhou, 221116, China
Email: lingicehan@yahoo.cn

Xiangsheng Rong
Training Department, Air Force Logistics of P. L. A, Xuzhou 221000, China
Email: rxs12@126.com

Yongquan Dong
College of Computer Science and Technology, Jiangsu Normal University, Xuzhou, 221116, China

***Abstract*—This paper proposes a Clustering-based Nearest Neighbor Search algorithm (CNNS) for high dimensional data. Different from existing approaches that are based on rigid-grid partition to develop data access structure, CNNS creates indexing structures according to data inherent distribution, with help of a progressive-styled clustering operation. The grids produced in this way adapt to data natural contours. CNNS is characterized with dataset reduction and dimension reduction. And parameterization heuristics are given to bring computation ease to CNNS. Empirical evidence on real datasets demonstrates the fine performance of CNNS.**

***Index Terms*—clustering-based method, nearest neighbor searching, grid partion, parameterization heuristics**

I. Introduction

Nearest neighbor (NN) searching is a fundamental problem in wide applications like data compression, data mining, information retrieval, image and video databases, pattern recognition and statistics analysis. This problem is stated as: given N points in a n dimensional space, find the nearest neighbor for a query point. For low dimensional data, it can be solved exactly. While in high dimensional case, the exact answer is shadowed by the course of dimensionality, that is, either the query time or the space required is exponential in n. In fact, for large enough n, the modified searching methods provide little improvement over the linear scanning method. Thus efforts are transferred to finding an approximate solution, that is, an approximate NN whose distance from the query point is at most $(1+\varepsilon)$ times its distance from the nearest neighbor [1].

A number of NN approaches for high dimensional data were proposed. Most of them share the same spirit of combining dimension reduction and data space partition. Dimension reduction aims to overcome the bottleneck of dimensionality by using data new representations while keeping a high portion of information. There are linear algebraic method Karhunen Loeve transformation (KLT) [2, 3] and mathematical transforms like DFT [4], DCT [5], DWT [6]. They are applied to general dynamic data, time-series, image or document data. Data partition creates the indexing structure. Diverse partition methods can be reduced to the definition of hashing functions. The designed hash function is expected to ensure that data that are close to each other have much higher probability of collision than those far apart. Various hash functions produce different shapes of hashing bucket. VA-file [7] produces rectangle-shaped bucket to group data. Another famous method family is Locality Sensitive Hashing (LSH) [8]. LSH employs random selection to finish dimension reduction; employs repeating random projection technique to achieve data partition. Based on this idea, interval-shaped [9], cell-shaped [10], ball-shaped [11] appeared in literatures. These partition procedures are often described by tree structure, say, Kdb-tree [12], hb-tree [13], R-tree [14], R*-tree [15], ss-tree [16], TV-tree [17] and X-tree [18].

These existing methods are characterized with rigid grid, ignoring inherently populated regions that naturally can act as buckets and can be helpful to locate data according to similarity. This paper focuses on these shortcomings by constructing buckets that coincide with the contours of inherently populated regions, which is achieved by clustering procedure. To construct data access structure, clusters are split into buckets of desired size, which is accomplished by a progressive-fashioned procedure. For a query q, it is indexed according to the nearest buckets, and then its NN is probed. For computation ease, before indexing, data dimensionality is reduced by spectrum analysis (SA); dataset is reduced by a self-tuning support vector clustering approach. And this paper proposes a SVD-based SA algorithm (SVDSA) to derive all data's new representations from reduced dataset. That brings CNNS the adaptation to high dimensional and large-sized dataset.

The paper is arranged as follows. Section 2 reviews some popular techniques of NN searching, basic support vector clustering (SVC) and SA. CNNS algorithm and its implementation details are given in Section 3, followed by SVDSA algorithm in Section 4. Section 5 records experiment results. Conclusion is in the last section.

II. Overview of Current Techniques

doi:10.4304/jcp.8.8.2085-2092

A. VA-file and LSH Family

VA-file approach is a partition-based approach. It divides data space into 2^t rectangular cells where t is the total number of bits specified by user. Each rectangular-cell has a bit representation of length t to approximate data points falling into the cell. The VA-file is actually an array of bit vector approximations based on the quantization of the original vectors. To search NN in a VA-file, vector approximations are scanned firstly to find rectangular candidates, and candidates are traversed to find true NN.

LSH family is also based on partition, but the grid yielded from partition is of diverse shapes. Interval-shaped LSH (iLSH) [9] defines the hash map from R^n to R^1, that is, points are projected to line and that line is partitioned into several intervals. Another fashion, cell-shaped LSH (cLSH) [10] builds grids by C pairs of random numbers (d, v_d), where d is an integer between 1 and n and v_d is a value within the range of the data along the d^{th} coordinate. The pair (d, v_d) partitions data according to an inequality: $x_{id} < v_d$. x_{id} is the coordinate of x_i in selected d^{th} dimension. Based on whether meeting that inequality or not, each x_i in each partition yields a C-length Boolean vector to state true or false state under C random embeddings. Points with the same Boolean vector are grouped into the same cell. After P partitions, each point belongs to P cells simultaneously: $CE_1 \ldots CE_P$. Recently ball-shaped LSH (bLSH) [11] appeared defines ball as the underlying geometric shape. It starts from an initial ball to generate other balls by shifting the original one. The number and size of ball are specified theoretically.

B. SVC

Classical SVC aims to find the cluster contours [19] by optimizing following target function:

$$\max_{\gamma} \sum_i \gamma_i K(x_i, x_i) - \sum_{i,j} \gamma_i \gamma_j K(x_i, x_j) \qquad (1)$$

$$\text{s.t.} \sum_i \gamma_i = 1,\ 0 \le \gamma_i \le C_{svc}.$$

Therein $x_i \in \Re^n$, $\Re^n$ is the input space, and C_{svc} is the penalty parameter to tradeoff error and clustering accuracy. Kernel function takes Gaussian fashion: $k(x_i, x_j) = \exp(-\| x_i - x_j \|^2 / \sigma^2)$. Points with $\xi_i = 0$ and $0 < \gamma_i < C_{svc}$ are mapped to the surface of sphere, referred as non-bounded Support Vector (nbSV). They describe cluster contours. Those points with $\xi_i > 0$, and $\gamma_i = C_{svc}$ are located outside the hyper sphere, and are called bounded Support Vector (bSV). Cluster assignment is done based on the deduction of an adjacent matrix of point pairs.

C.. SA

SA [20] procedure is used to obtain discriminant directions underlying data distribution. SA obtains data spectral projections by eigen-decomposing affinity matrix, and then groups data spectrums with a simple method. It assigns point the same label as its spectrum projection. Its main operation is based on the eigen-decomposition on pairwise matrix H, where H is the normalized version of data affinity matrix. Select top p eigenvectors and form spectral embedding matrix S by stacking p eigenvectors in columns. Rows of S are data spectral coordinates. Based on them, further partition or clustering task can be achieved in a simple fashion.

III. CNNS Algorithm

A. Idea

The idea of CNNS is described in below steps:

1) Tuning-scaled SVC runs to generate data representatives $\{DR_i\}$;

2) SA runs to group $\{DR_i\}$ in spectrum space;

3) SVDSA runs to obtain all data's spectrums;

4) Label data according to its nearest DR;

5) For each cluster, PK-means performs progressively to form hashing bucket;

6) Index the query.

Dataset is reduced by a modified SVC, whose Kernel function width is tuned adaptively. Then the resulted $\{DR_i\}$ are mapped to spectral space, where they are grouped. New representations of all data are derived from $\{DR_i\}$ by SVDSA procedure. With the nearest-labeling rule, data are clustered. Clusters provide natural boundaries that basic grids should observe, that is, a bucket is not expected to cover two clusters. Therefore buckets are formulated within each cluster respectively. We do the K-means clustering in a progressive way, to split a cluster into sub-clusters, and these sub-clusters act as buckets. The splitting runs until each bucket is equipped with the appreciate size.

In query time, q is indexed by its nearest bucket:

$$label(q) = \min_j \{\|s(q) - s(b_j)\|\}$$

Where $s(\cdot)$ denotes the spectrum representation of argument, b_i is the center of bucket B_i.

B. Self-tuning SVC

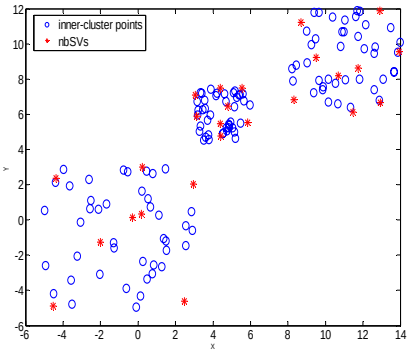

Figure 1. SVC clustering result

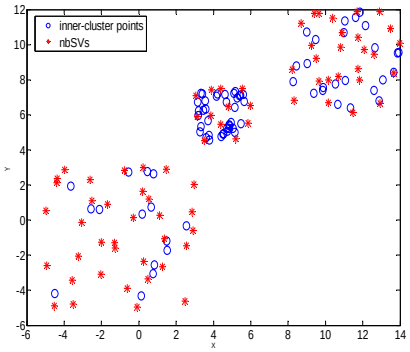

Figure 2. TSVC clustering result

This section introduces the strategy to tune the scale parameter of Gaussian Kernel. The idea is to adjust the scale parameter according to the local distribution information around individual data point. That will make the algorithm adapt to diverse dataset.

Classical SVC generates SVs describing cluster contours. Self-tuning SVC (TSVC) proposed in this paper produces more SVs of meaning to serve as data representatives by tuning the scale parameter of Kernel function data-dependently. Tuning idea is to set individual scale factor for each point, that is, for x, its scale factor is $\sigma_x = \|x - x_r\|$. To measure Kernel affinity between x and y, their scale factors are combined, leading to the new Gaussian Kernel:

$$k(x, y) = \exp(-\frac{\|x-y\|^2}{\sigma_x \cdot \sigma_y}) = \exp(-\frac{\|x-y\|^2}{\|x-x_r\| \cdot \|y-y_r\|}) \qquad (2)$$

r is specified as the max gap in the list of distances from x to other points: $r = \max_j \{ d(x, x_j) - d(x, x_{j-1}) \}$, where $d(x, x_j)$ is the row of Euclidean distance matrix and it is sorted in an ascending order. The result of SVC and TSVC is plotted in Figure 1 and 2, where $C_{svc} = 0.8$. In Fig. 1, SVs are located on cluster boundaries. In Fig. 2, SVs are located on both cluster boundaries and the important positions within clusters where sharp changes of density happen. These SVs describe a sketch of dataset.

C. PK-means

Different from classical K-means method that doesn't know the number of clusters and specifies the initialization in a random way, the algorithm of this paper propose PK-means to fulfill clustering task.

The key of PK-means is that it is an iteration procedure. That is, in each iteration run of PK-means, a bucket is constructed to help consequent clustering process. As to the termination of PK-means's iteration, this algorithm considers to ensure the population of each bucket is below the upper bound for the access effectiveness. Since the bucket is obtained by splitting clusters, so to do splitting we first duplicate its cancroid. And then we perturb the copy randomly to form the centroid of the new clusters. Perform standard K-means with the number of clusters, K, parameter being 2, with the aim to give the 2-split result. Split current buckets iteratively. As long as there is a bucket whose size is higher than the upper bound, let K=K+1, and then run K-means with the new K. Denote I^{th} cluster as C_I, and the resulted buckets as the set $B = \{B_i\}$ with corresponding centers $b = \{b_i\}$; denote NB as the number of buckets.

Conclude above analysis into the PK-means algorithm. So the steps of PK-means are:

```
PK-means (C_I, b_I)
{
 B_I = C_I;
 NB = 1;
 If (|C_I| > ub)
   {
    flag = 0;
    NB++;
    b_NB = b_I + δ;
    While (not flag)
        {
         [B, b] = K-means(C_I, NB, b);
          flag = 1;
          i = 1;
          While (i <= NB & flag)
              If (|B_i| > ub)
               {
                flag = 0;
                b_NB+I = b_i + δ;
                NB++;
               }
              Else
               i++;
          }
    }
 Return B;
}
```

Therein δ is a random offset.

Note that Standard K-means procedure returns the optimal clustering result with parameters coming form three aspects. They are parameters of C_I, cluster number NB and initial centers b. The calling parameter b_I corresponds to the mean of C_I. ub is the upper bound on bucket size. It is specified by following steps:

1) For each DR_i, sort the spectral distance values of it to other data x_j in the ascending order: $\{\|s(DR_i) - s(x_j)\|\}$;

2) Find the max gap between two adjacent distance values of this list, as shown in formula (9).

3) Compute $ub = \lambda(\text{average}\{gap(i)\})$.

Here the idea is to employ the average size of neighbor as the parameter. Coefficient $\lambda >= 2$. According to experimental results of below section, this paper sets this coefficient $\lambda = 2$.

Obviously $gap(i)$ is important in above parameterization steps. Here for point x_i, we firstly find the inherent dense region around it, which is reflected by the maximum gap between two adjacent distance values. Such a gap value reveals the size of neighborhood centered at x_i. Then we rescale this size with some distance value. So the setting of $gap(i)$ is:

$$gap(i) = \max_j \{\frac{\|s(DR_i)-s(x_{j+1})\|-\|s(DR_i)-s(x_j)\|}{\|s(DR_i)-s(x_{j+1})\|}\} \qquad (3)$$

gap(*i*) is viewed as the estimate of DR_i's neighborhood size.

For completeness, below steps are details of K-means procedure:

```
K-means (C_I, K, μ)
{ Δ = ∞;
  flag = 0;
  While (not flag)
        {
          For j = 1: K
                μ_j = (1/N_j) Σ_{x_i ∈ [g_{j-1}, g_j]} x_i ;
          For j = 2: K
                g_{j-1} = (μ_{j-1} + μ_j)/2 ;
          Δ' = Σ_{j=1}^{K} Σ_{x_i ∈ [g_{j-1}, g_j]} (x_i − μ_j)^2 ;
          If (Δ'/Δ < ε)
                flag = 1;
          Else
                Δ = Δ';
        }
   Return (g, μ);
}
```

Therein N_j is the size of interval of $[g_{j-1}, g_j]$, and $g = \{g_j \ldots\}$, $\mu = \{\mu_j \ldots\}$ to express interval cut points and interval centroids.

IV. SVDSA

Set |DRs| = *M* and |X| = *N*, with X being dataset. Form pair-wise Kernel affinity matrix $H_{M\times N}$ between DRs and X. We normalize *H* by rescaling its each entry using corresponding sum-of-row and sum-of-column. Let R = diag $(r_1 \ldots r_M)$ with $r_i = \Sigma_{j=1}^{N} H_{ij}$, and C = diag $(c_1 \ldots c_N)$ with $c_i = \Sigma_{i=1}^{M} H_{ij}$. *H* is normalized as:

$$H^* = (R^{-1/2}) \cdot H \cdot (C^{-1/2}) \tag{4}$$

Then SVD is conducted on H^* in the way:

$$H^* = U\Lambda V = (U\Lambda^{1/2})(V\Lambda^{1/2})^T = Q \cdot J^T \tag{5}$$

Therein $U^T \bullet U$ and $V^T \bullet V$ are identity matrices of corresponding sizes and Λ is a diagonal matrix of singular values. The vectors corresponding to the largest *m* singular values consist of the low-rank approximation of *H*. Columns of *J* are spectrum coordinates of all data. Dimensionality of spectrum coordinates, *m*, is specified by the max gap of the singular value list that is sorted in the descending order. Using SVD instead of Eigen Value Decomposition, complexity drops from $O(N^3)$ to $O(M^2 \times N)$.

Then K-means is performed on DRs' new representations and the rest data are labeled according to its nearest DR. <add how to compute spectrums of query>

Add Nystrom method and literature [21], to the query *x**, view it as the new data, and we update the eigen value and eigen vector according to Nystrom method:

$$\lambda_i^* = \frac{N+1}{N} \lambda_i \tag{6}$$

$$\mu_i^* = \sqrt{\frac{N}{N+1}} \frac{1}{\lambda_i} K_{N+1,N} \mu_i \tag{7}$$

There λ_i^* and λ_i are the new and original eigen values; μ_i^* and μ_i are the new and original eigen vectors. Based on new eigen vector, query is equipped with its spectrum coordinates.

V. Experimental Results

Firstly we combine the *T*SVC, *S*VDSA and *K*-means to form a clustering method, TSK. We apply it on a skewed dataset to check its ability to formulate appreciate cluster boundaries. TSK is compared with other clustering approaches: pure K-means, NJW and standard SVC. NJW is an appealing spectrum clustering method. It shows fine behaviors in multi applications. It maps data to a spectrum space and on the spectrum coordinates of original data points it conducts the clustering process. To datasets with special distribution shapes, NJW and its variants sometimes obtain good results.

Figure 3 gives the illustration of experimental dataset. Figure 4 shows the result of K-means.

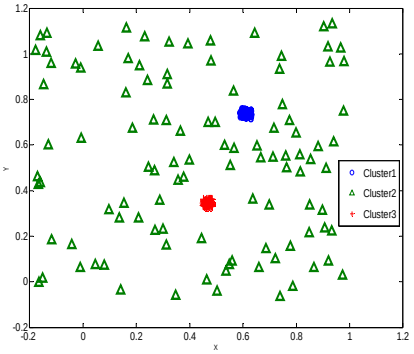

Figure 3. Dataset

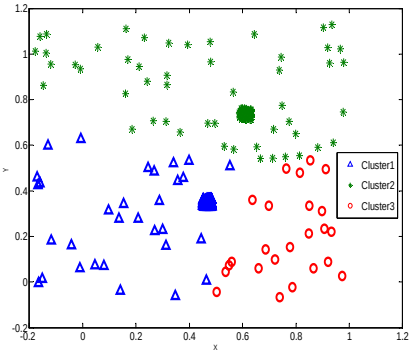

Figure 4. K-means Result

It is clear that clustering result of Figure 4 is undesired. The reason of K-means's failure lies in its hard metric plus inflexible partition. It can only address the datasets with Gaussian-type distribution. In times of datasets with random distribution, K-means often presents more errors. Furthermore it has to consume more cost to find the

proper number of clusters if the number of clusters is unknown.

Figure 5 and Figure 6 give the results of NJW under two Kernel scales. These two scales are two representatives of a series of scale parameters. Actually we tried many scales and find experimental results are poor. It is due to the mismatch between fixed scales and the skewed distribution of experimental dataset.

SVC also can't present good result, as shown in Figure 7. Figure 8 describes the DRs produced by TSVC. It finds resulted DRs are located within three clusters. In consequent clustering process, they are grouped correctly by SVDSA. That correct clustering results are guaranteed by the idea of SVDSA. And these results also indicate TSK does hold the good adaptation to datasets with various distributions; this adaptation, of course, is brought by scale tuning strategy.

From these figures, the validation of TSK and SVDSA can be verified.

To make a further insight of TSK process, we run it on some real and benchmark datasets that are taken from UCI Repository of Machine Learning Databases [22]. These datasets are: Breast Cancer (BC), Diabetes, Vote, Thyroid, Heart and Waveform. Table 1 compares the errors of the above methods and another clustering algorithm: Girolami method [23]. Girolami is a more sophisticated Kernel-based expectation-maximization method.

From results of Table I, the performance of the proposed TSK can be verified.

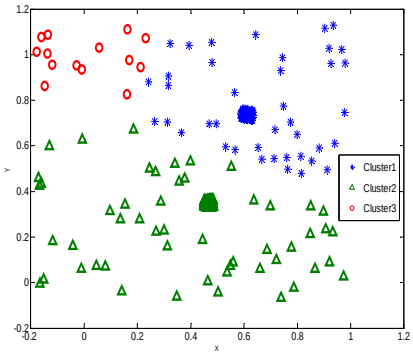

Figure 5. NJW: 1/σ2=19.9

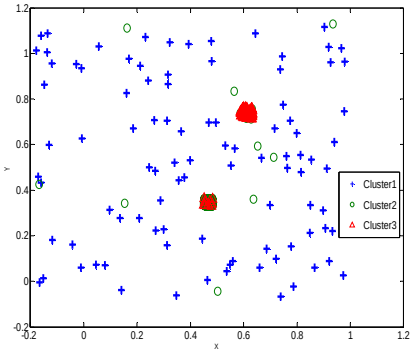

Figure 6. NJW: 1/σ2=2.79

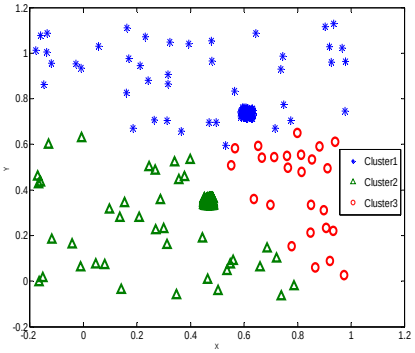

Figure 7. SVC: 1/σ2=0.5612

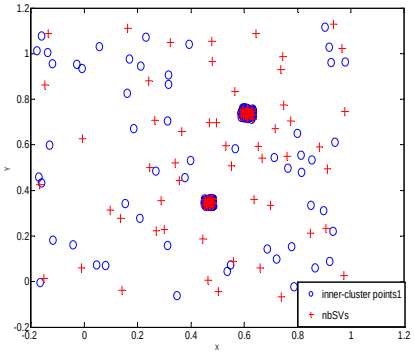

Figure 8. DRs of TSVC

After the usual comparison on clustering performance, in below experiments, two evaluations are used to check the quality of indexing approaches.

Firstly the quality of the indexing approaches is reflected by the quality of query's neighborhood. For some query q, we use the average affinity of q to its neighbors of the nearest bucket as the first evaluation. Assume q is indexed by B_i bucket; then the average of q's affinity within the bucket B_i is defined as the average affinity between q and its neighbors coming from B_i. In details, such an average affinity is defined in an exponential way:

$$Af = \frac{1}{B_i} \Sigma_{x \in B_i} \exp(- \| q - x \|^2) \quad (8)$$

TABLE I.

ERROR COMPARISON ON REAL DATASETS (%)

Dataset	K-means	Girolami	SVC	NJW	TSK
BC	3.81	2.93	5.17	3.22	3.17
Diabetes	18.6	14.8	16.8	15.2	14.25
Vote	7.92	4.62	6.11	4.43	4.28
Thyroid	6.06	5.83	6.25	5.01	5.26
Heart	7.23	6.57	7.06	6.5	6.42
Waveform	3.51	3.11	3.72	2.82	3.07

The second evaluation is the ratio of false hits in q's nearest bucket. False hits are errors among neighbors, say, points having different label with q. That ratio tells how well the clustering approach finds relevant data even to the extent that it skips some correct answers. Both two

evaluation criteria well reveal the quality of neighborhood.

Now we run CNNS on News Group dataset [24]. News Groups dataset is a text records dataset, which contains about 20,000 articles. These articles are divided into 20 newsgroups according to their topics. These groups' topics are:

NG1: alt.atheism;
NG2: comp.graphics;
NG3: comp.os.ms.windows.misc;
NG4: comp.sys.ibm.pc.hardware;
NG5: comp.sys.mac.hardware;
NG6: comp.windows.x;
NG7: misc.forsale;
NG8: rec.autos;
NG9: rec.motorcycles;
NG10: rec.sport.baseball;
NG11: rec.sport.hockey;
NG12: sci.crypt;
NG13: sci.electronics;
NG14: sci.med;
NG15: sci.space;
NG16: soc.religion.christian;
NG17: talk.politics.guns;
NG18: talk.politics.mideast;
NG19: talk.politics.misc;
NG20: talk.religion.misc.

Before doing the clustering experiments, the preprocessing work is to apply the usual *tf.idf* weighting schema to express documents.

That is, set vector $x_i = (x_{i1}, x_{i2}, ..., x_{im})'$ for each document.

Here $x_{ij} = t_{ij} \cdot \log \frac{n}{df_j}$, t_{ij} is the appearance frequency of word f_j in document d_i, n is the number of documents, and df_j is the number of documents that contain word f_j. We delete words that appear too few times and normalize each document vector to have unit Euclidean length.

We choose some classes to form experiment subsets, where 10% data randomly selected serve as queries. Besides CNNS, the mentioned methods run for comparison. Parameters involved are specified according to their designers. Performance of them is described in Table 1, where two values in a blank are false hits ratio and *Af* respectively.

From Table II, it is clear that CNNS achieves the top place in 4 of 6 cases, and follows closely the optimal result in other 2 cases. In the first three subsets, where data classes are distinct relatively, CNNS is competitive with bLSH. Although in {N1, N2, N7, N8} and {N7, N8, N12, N16, N17} CNNS's performance is not the best, its bucket quality is better than other methods. In the last three subsets, where class boundaries are blurry, CNNS's advantage is obvious over its peers, which indicates the effect of clustering procedure.

Among three LSH-based approaches, bLSH does a better job on average, followed by cLSH, and then iLSH. bLSH buckets are ball-shaped, which better approximates the neighborhood shape. iLSH presents a moderate performance due to its weak hashing power that summarize data features to a scalar. As mentioned before, their different hashing embeddings specify different shapes of basic geometric grid, and these fixed-shaped grids might only exhibit their unique advantage in particular data distribution. While the shape of CNNS buckets depends on the natural contour of data itself, with adaptation to difficult datasets. That fosters CNNS's success. Note that CNNS and bLSH produce lowest *Af* values, because buckets of CNNS capture the natural boundary of class and contain true neighbors. VA-file produces buckets with similar shape with cLSH, so they present competitive results.

TABLE II.

ACCURACIES OF KNN BASED ON NEIGHBORHOOD FORMULATION METHODS (%)

Data	VA-file	iLSH	cLSH	bLSH	CNNS
N1, N2, N7, N8	84.4% 0.81	85.3% 0.85	87.2 0.83	88.3% 0.88	88.2% 0.89
N6, N7, N8	83.2% 0.79	82.8% 0.75	83.4% 0.81	84.0% 0.88	84.0% 0.86
N7, N8, N12, N16, N17	83% 0.8	83.1% 0.82	85.7% 0.86	85.2% 0.84	85.6% 0.84
N2, N3, N4	64.8% 0.59	63.7% 0.57	66% 0.6	66.8% 0.65	69.2% 0.66
N4, N5, N6	61.9% 0.51	60.7 0.49	62.0% 0.51	62.1% 0.53	64.6% 0.53
N12, N13, N14, N15	69% 0.65	65% 0.62	69.1% 0.65	70.6% 0.67	72.4% 0.68

Finally more datasets are used for experiments: Waveform [22], Musk [25] and Breast Cancer (BC). Musk has two versions Musk1 and Musk2. They record 476 and 6598 conformations for musk molecules and non-musk molecules. BC has two databases: breast-cancer (named as BC1) and breast-w (named as BC2), to describe the malignant and benign cases with 33 attributes and 30 attributes respectively.

The last section of experiments is to combine the proposed algorithm CNNS kNN to form an assembling classifier CNNSk. The aim of doing experiments on such an assembling classifier is to observe the generalization ability and clustering performance CNNS.

In experiments, CNNSk is compared with more popular classifiers. They are: classical kNN, the clustering method that takes distance information as clustering criterion; SVM_{11} [26], the assembling classifier consisting of classical SVM with the final decision being decided in the voting strategy; Machete [27], a recursive partitioning procedure, in which the input variables are used for splitting at each step is the one that maximizes the estimated local relevance; DANN, an adaptive nearest neighbor classifier [28]; Scythe [27], a method that do the generalization of Machete method; and Adamenn, an algorithm that is focused on the adaptive nearest neighbor approach, and this algorithm works in many practical applications [30].

In below experiments, the key spirit of last four methods is to develop weighted metrics through Dimension Derivation (DD) technique. Therefore in this paper they are called DD methods or DD-based methods.

Below they are combined with kNN to finish classification, and the resulted classification accuracy is compared with CNNSk.

In experiments, except CNNSk, other clustering methods have no parameterization strategy of neighborhood. So the neighborhood size of kNN and four DD-based methods are parameterized by 20-fold cross-validation. That cross-validation is the common method to parameterize algorithm parameters. And this parameterization process of course consumes more cost. The super parameters of SVM_{11} are also set by cross-validation. Sample 10% data at random as testing data. Table III describes the average of classification accuracies of these methods on UCI datasets.

TABLE III.

CLASSIFICATION ACCURACIES COMPARISON (%)

Data	Waveform	Musk1	Musk2	BC1	BC2
kNN	75.1	88.4	62.8	70.2	93.1
SVM_{11}	83.4	93.6	68.5	76.8	96.8
C4.5	77.2	91	68.5	74.1	95.3
DANN	84.8	94.2	67.3	75.1	96.1
Machete	79.7	94.9	66.1	75.8	96.2
Scythe	84.9	96	69.8	74.3	97.3
Adamenn	85.6	97	71.2	77.1	97.7
CNNSk	85.7	97	72	77.2	98.2

It is easy to find from Table III that the advantage of four DD methods and CNNSk over the first three methods is clear. For kNN, it works poorly due to its employment of Euclidean metric. Such a metric tends to lead errors of classes memberships when the dataset is of special distribution shape. And such a metric is not specialized to high dimensional data, so it consequently bring many errors. SVM_{11} is also trained in Euclidean space, while it presents higher accuracy thanks for its non-linear decision interface. C4.5 yields moderate results.

Then take a look at four methods based on DD technique. Find that among four DD methods, Adamenn works best on average; it investigates dimension relevance from probability distribution and presents a wisely weighted metric, thus spanning qualified neighborhood. But this method has to parameterize six parameters. No matter parameterization through searching strategy in the grid parameter space or the cross-validation, or some specified parameterization methods, huge computation will be consumed. Such expensive cost in tuning six parameters overshadows its advantages. Scythe follows it closely. Machete follows Scythe due to its greedy spirit. DANN approximates the weighted *Chi-squared* distance, which might fail in datasets of non-Gaussian distribution. That is, DANN lacks the adaptation to datasets with diverse distributions. Now look at CNNS. Clearly CNNS does best in 2 of 5 cases. And CNNS achieves the second place in 3 of 5 cases. CNNS's behaviors are not good as Adamenn. And the later gives optimal result in 3 of 5 cases, but if we take the fact that CNNS has computation ease in parameterization and steady performance into consideration, it knows that CNNS is a welcome method in practice.

From above experiments, it is concluded that if both performance and cost are taken into consideration, CNNS is a fine choice.

VI. CONCLUSION

This paper presents a NN search algorithm based on clustering (CNNS). CNNS organizes data with buckets, which are split into natural sub-clusters, namely, basic grids. The NN found by CNNS comes from the bucket that the query is indexed to. Indexing processing and NN searching are conducted in the spectrum space, and the splitting of clusters is finished by a progressive fashioned clustering method. CNNS is equipped with two components for computation ease: dataset reduction and dimension reduction. The first target is achieved by a tuning-scaled SVC, and the second target is fulfilled by the SVDSA procedure. Empirical evidence on real datasets demonstrates the performance of CNNS in producing qualified NN and neighborhood. Experimental results also verify the quality of two components. In empirical results, it finds the buckets produced by CNNS are more adaptable to data distribution than methods based on rigid partition.

Future work direction is to probe the more informed method of neighborhood development. Especially it is an appealing trend to specify neighborhood based on local geometric information around data points. Since neighborhood also carries much local distribution information, seeking neighborhood's details from data point's local dense or sparse geometric properties is a good research direction.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 61105129 and 61100167, and the Natural Science Foundation of Jiangsu Province, China under Grant No. BK2011204, the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant No. 11KJB520019.

REFERENCES

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," Journal of the ACM, vol. 45 (6), pp. 891 – 923, 1998.

[2] Y. B Hua, W. Q Liu, "Generalized Karhunen-Loeve Transform," Signal Processing Letters, vol. 5(6), pp. 141-142, 1998.

[3] Y. Yamashita, H. Ogawa, "Relative Karhunen-Loeve Transform," IEEE Transactions on Signal Processing, 1996, vol. 44(2), pp. 371-378.

[4] J. Stanek, W. Kozminski, "Iterative algorithm of discrete Fourier Transform for processing randomly sampled NMR data sets. Journal of biomolecular NMR, 2010, vol. 47(1), pp. 65-67.

[5] W. Burger, M. J. Burge, "The Discrete Cosine Transform DCT," Principles of Digital Image Processing Undergraduate Topics in Computer Science, 2009, pp 1-8.

[6] M.K, Wali. M. Murugappan, R. B. Ahmad, B. S. Zheng, "Development of Discrete Wavelet Transform (DWT)

toolbox for signal processing applications," International Conference on Biomedical Engineering, 2012, pp. 211 – 216.
[7] R. Weber, H. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," Proceedings of the 24th International Conference on Very Large Data Bases, pp. 194-205. 1998.
[8] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing," Proceeding of International Conference on Very Large Data Bases, pp. 518–529, 1999.
[9] M. Datar, N. Immorlica, P. Indyk, V. S. Mirrokni, "Locality-Sensitive Hashing Scheme Based on p-Stable Distributions," Proceedings of the 20th Annual Symposium on Computational geometry. pp. 253-262, 2004.
[10] S. J. Gong, "A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering," Journal of Software, vol 5(7), 2010, pp. 745-752.
[11] A. Andoni, P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," Proceeding of 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 459-468, 2006.
[12] J. T. Robinson, "The K-D-B tree: A search structure for large multi-dimensional dynamic indexes," Proceedings of 1981. ACM SIGMOD International Conference on Management of Data (SIGMOD), 1981.
[13] D. Lomet, B. Salzberg, "The hB-tree: A multiattribute indexing method with good guaranteed performance," ACM Transactions on Database System, 1990. vol.15, pp.625-658.
[14] K. Ibrahim, F. Christos, "Hilbert R-Tree: An Improved R-Tree Using Fractals," Technical Report, Institute for Systems Research, University of Maryland. TR 93-19.
[15] N. Beckmann, H. P. Kriegel, R. Schneider, B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," ACM SIGMOD Record, vol. 19(2), 1990, pp. 322-331.
[16] L. F Yang, X. L. Huang, R. Lv, M. M. Kang, X. X. Yin, "Performance of SS-Tree with Slim-Down and Reinsertion," International Conference on Algorithm Measuring Technology and Mechatronics Automation, 2010.
[17] K. Lin, H. V. Jagadish, C. Faloutsos, "The TV-Tree: An Index Structure for High-Dimensional Data," The VLDB Journal -VLDB, 1994.
[18] S. Berchtold, D. A. Keim, H. P. Kriegel, "The X-tree: An Index Structure for High-Dimensional Data," Jounal of 22nd VLDB, pp.28-39, 1996.
[19] B. H., A., D. Horn, H. T. Siegelmann, "Support Vector Clustering," Journal of Machine Learning Research, 2001 pp. 125-137.
[20] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," Advances in Neural Information Processing Systems 14: Proceedings of the 2001. pp.849-856
[21] C. Williams, M. Seeger, "Using Nyström Method to Speed up Kernel Machines," Advances in Neural Information Processing Systems, 2001, vol. 13, pp. 682-688
[22] L. C., Y, "Comparison of Learning Algorithms for Handwritten Digit Recognition," Proceedings of the International Conference on Artificial Neural Networks, 1995, pp. 53-60.
[23] http://archive.ics.uci.edu/ml/
[24] http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html
[25] G. B. Wang, X. J. Li, K. F. He, "Kernel Local Fuzzy Clustering Margin Fisher Discriminant Method Faced on Fault Diagnosis," Journal of Software, vol 6(10), 2011, pp. 1993-2000.
[26] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," Pattern Recognition, vol 44(8), 2011, pp. 1761–1776.
[27] J. H. Friedman,. "Flexible Metric Nearest Neighbor Classification," Technique Report, Department of Statistics, Stanford University, 1994.
[28] T. Hastie, R. Tibshirani, "Discriminant Adaptive Nearest Neighbor Classification," IEEE Trans. on Pattern Analysis and Machine Intelligence. vol. 18(6), 1996, pp. 607-615.
[29] Y. P. Li, Y. M. Ye, X. L. Du, "A New Vertex Similarity Metric for Community Discovery: a Local Flow Model," Journal of Software, vol 6(8), 2011, pp. 1545-1553.

Ping Ling was born in Xuzhou, Jiangsu Province, China, Feb. 1979. She received her Bachelor's degree in 2000, from College of Computer Science and Technology, Xuzhou Normal University. And then she received her Master's degree and PHD from College of Computer Science and Technology, Jilin University in 2006 and 2010 respectively. She research field focuses on data mining, intelligence computing, support vector machine and support vector clustering, etc.

Xiangsheng Rong was born in Yanggu, Shandong Province, China, 1975. He received his Bachelor's degree in 1997, from Department of Logistic Command, Xuzhou Air Force College of P. L. A. And then he received his Master's degree in 2003 from Xuzhou Air Force College of P. L. A. His major research directions include the application of information technology and dynamic programming technique in military logistic command, intelligence command in combined operations of a sham battle, etc.

Yongquan Dong was born in Xuzhou, Jiangsu Province, China, 1979. He received his Bachelor's degree in 1994, from College of Computer Science and Technology, Harbin Institute of Technology. Now his research directions are operational research in military logistic command, intelligent computing application in logistic command, etc.