Vector-Distance and Neighborhood Development for High Dimensional Data

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

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

Xiangyang You
Training Department, Xuzhou Air Force College of P. L. A, Xuzhou 221000, China
Email: xyyou@126.com

Ming Xu
Department of Logistic Command, Xuzhou Air Force College of P. L. A, Xuzhou 221000, China
Email: mingxu@sina.com

***Abstract*—This paper presents a novel distance concept, Vector-Distance (VD) for high dimensional data. VD extends traditional scalar-distance to a vector-like fashion by collecting multi partial distances from diverse angles. These partial distances are derived from random projections, and they preserve individual features of dimensions as much as possible. Based on VD definition, a method family for neighborhood development is proposed, where methods consist of some norm definitions and certain constrains specified for various purposes. Experiments on real datasets verify the quality of neighborhoods produced by the proposed method family better or competitive with the neighborhood produced by the state of the art.**

***Index Terms*—vector-distance (VD), high dimensional data, partial distances, neighborhood development**

I. Introduction

High dimensional data received renewed interest in recent years thanks to the increase of the available computer hardware. While the software or approaches handling to them didn’t achieve so sharp improvement as hardware due to the difficulty in learning high dimensional data structure. The structure of high dimensional space defies usual 3-dimensional geometric intuition, and it is extremely sparse with data points far away from each other. If using conventional metric to explore a data’s neighborhood, only a small number of neighbors may be inferred. Unless the neighborhood radius is set large, sufficient neighbors could be covered. But that leads to the loss of locality. This phenomenon is known in the statistical literatures as the curse of dimensionality, and its affect increases exponentially in the dimension [1, 2, 3].

Dimension reduction is the natural solution to address this issue. This approach family includes dimension selection that chooses important dimensions, dimension extraction that derives new dimensions from the original ones, and dimension weighting that equips dimensions with significance coefficients. Their idea relies on defining data-dependent metrics that can capture local distribution features from dimension analysis so as to generate data’s new representation. These metrics provide a scalar value, which reflects the distance information from a single angle. Yet in high dimensional space the single-source-based metric might not succeed in exploring exact distance information everywhere because the dimension significance might vary from region to region. For example, high dimensional data x, y and z, perhaps the dimensions that are critical to measure distance between x and y are not important to x and z. That inspires us to define a multi-source-based metric.

This paper defines a vector-fashion distance concept for high dimensional data, named as Vector-Distance (VD). VD equips a pair of data points with a vector as their distance; the components of that vector are partial distance values derived using random projections technique. The random projections are realized by iterations of data space partition. That idea is rooted from Local Sensitive Hashing (LSH) [4]. LSH is focused on nearest neighbor searching, and it develops neighborhood by collecting hash table buckets that query is projected to. The neighborhood produced by LSH is an unsorted set, so that an extra metric has to be consulted to find the nearest neighbor. Compared with classical LSH, VD is characterized with the ability to sort neighbors.

Based on VD, a method family of neighborhood formulation is proposed, where various metrics plus some constrains specify diverse manners of neighborhood formulation. Heuristics are given to facilitate VD computation, thus without suffering from huge cost in tuning parameters that many random-projection-based method need.

The rest of this paper is organized as below. Section 2 reviews some work, namely, state of the art of neighborhood development for high dimensional data. Then VD definition and method family are presented in Section 3. Consequently, Section 4 discusses self tuning of parameters of random projections. Experimental evidence and analysis are given in Section 5, followed by conclusion in the last section.

II. RELATED WORK

LSH originally aims to solve the ε–approximate nearest neighbor problem of high dimensional data. The relaxation from finding an exact answer to an approximate answer removes the curse.

LSH is asked to return a point whose distance from the query is at most $(1+\varepsilon)$ times the distance from the query to its nearest points. The appeal of this approximation fashion is that in many cases, an approximate nearest neighbor is almost as good as the exact one. The general LSH schema relies on existence of locality-sensitive hash functions.

Definition. Locality-Sensitive Hashing.
A family $\mathcal{H}$ is called $(r, (1+\varepsilon)r, p_1, p_2)$-sensitive if for any $p, q \in R^d$:

1) If $\|p-q\| \leqq r$ then $\Pr[h(q) = h(p)] \geqq p_1$

2) If $\|p-q\| \geqq (1+\varepsilon)r$ then $\Pr[h(q) = h(p)] \leqq p_2$

Various families of hash functions can be defined to yield various LSH schemas. But the precondition is that the function must meet the locality-sensitiveness property, that is, the basic parameters r, p_1 and p_2 can be computed.

Many literatures have touched this issue. Reference [5] defines the hash function mapping from original space to Hamming space, and rectangular-shaped cell acts as the basic grid to form neighborhood. Reference [6] projects data to a R^1 space, where the projected line in R^1 is partitioned into equal-length intervals. The hash function returns the index of the interval containing the projection of query.

Literature [7] partitions data space with ball-shaped grid; the resulted ball boundaries actually correspond to hash functions definitions. In that paper, the number of balls is parameterized to ensure the union of balls can cover all space. In above methods, interval, rectangular and ball that contain query are collected to form neighborhood.

Recently an analysis in reference [8] uses one hash function to store all data. To search the neighborhood of query, not only query itself but also some neighbor-like points generated are hashed to find interesting hashing buckets.

III. VECTOR-DISTANCE AND PARAMETERIZATION

A. VD Definition

The novel of Vector-Distance definition is that it employs a vector as the distance representation of a pair of points. Elements of such a vector are partial distance values that are derived from some number of partitions of data space. And each partition is generated by random projections, a technique that used to handling high dimensional data.

Assume dataset $S = \{x_1 \ldots x_N\}$, $x_i \in R^n$, and q is the query. S is tessellated P times with random partitions. In each partition, a partial distance value is derived from C dimensions that are selected at random. In more details, each partition is defined 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. v_d acts as a benchmark coordinate to measure the local distance between q and x_i in d^{th} dimension. Denote q_d and x_{id} as the d^{th} coordinate, and then the local distance is defined as:

$$A_d(q, x_i) = \exp(-(q_d - v_d)(x_{id} - v_d)) \qquad (1)$$

Simultaneously, v_d is used to form the inequality '$x_{id} < v_d$'. q and x_i yield the true or false result under that inequality. After a partition q and x_i yield C-length Boolean vector with 0 meaning false and 1 meaning true. This Boolean vector is the projections of data under C random embeddings.

Denote PD_I as the set of selected dimensions of I^{th} partition, $b^I(x)$ as the Boolean vector of x, with $b^I(x)_d$ being its d^{th} component. We employ $b^I(q)_d$ and $b^I(x_i)_d$ to weight the local distance between q_d and x_{id}, to generate their partial distance of I^{th} partition:

$$A^I(q, x_i) = \Sigma_{d \in PD_I} \alpha_d \cdot A_d(q, x_i) \qquad (2)$$

With $\alpha_d = \exp(|b^I(q)_d - b^I(x_i)_d|)$.

The employment of α_d will strength the d^{th} local distance between q and x_i if they have different response to the inequality $x_{id} < v_d$. The VD between q and x_i is:

$$VD(q, x_i) = (A^1(q, x_i) \ldots A^P(q, x_i)) \qquad (3)$$

Points with the same Boolean vector are grouped into the same cell. View one partition as a hash function; then a cell is actually a hash table bucket. Compared with [4], which uses one-dimensional interval as bucket, our schema extends the bucket from one-dimension to multi-dimension, so brings richer hashing meaning and locality sensitiveness without expensive cost. That low computation cost also makes cell-bucket hashing competitive with ball-bucket hashing [9], which needs the nonlinear embeddings to fulfill random projections. Of course the nonlinear is at the gains of stronger hashing power and locality sensitiveness. Consider both performance and cost, cell-bucket hashing is a fine choice. VD definition integrates distance measurement into cell formulation, say random projections, so that neighbors are inherently sorted according to closeness with the query.

Basic parameters of VD, r, p_1 and p_2, are specified as in [10]. The time complexity is $O(N^{1+1/(1+\varepsilon)})$, and the query time is $O(dN^{1/(1+\varepsilon)})$.

B. Parameterization of P and C

P and C have a direct influence on the size of cell and the quality of neighborhood. But it is hard to integrate them into some cost function explicitly and find the optimal configuration through optimization. Reference [5] gave an approach that runs over all pairs of configuration and chooses the pair that can incur least time cost under a pre-specified error upper bound. Here we search P and C in an empirical way. That is, we do searching trails within an appreciate range, where a measurement of neighborhood quality is employed to find the best parameter pair.

When C increases, the number of cells increases and the average volume of one cell drops. When P becomes larger, more cells are produced and then the final neighborhood becomes large. Given the fixed C, only values of P below an upper bound are of interest. Because once P exceeds some bound, the neighbors it finds have been covered by smaller values of P, and the larger values of P only brings extra computation without any improvement in the neighborhood quality.

Therefore this paper fixes C by setting its value as an integer randomly generated from range [$\sqrt{n}$, n] in advance.

As to P, we run P over a specified range to find the one able to bring the best neighborhood quality. The neighborhood quality is measured in the way: $Qua(P)$ = ave{$|MEM_i| / |NEI_i|$}, where NEI_i is the neighborhood of x_i, and MEM_i is the set of x_i's same-class members in NEI_i. Then the optimal P is parameterized by the method of below steps:

```
1) Select $m$ data randomly;
2) For $P = P_{down} : P_{up}$
3)     Develop neighborhoods for $m$ data: $\{NEI_i\}$,
       $(i = 1\ldots m)$;
4)     $Qua(P)$ = ave$\{|MEM_i| / |NEI_i|\}$;
5) EndFor
6) $P_{optimal} = \max_P \{Qua(P)\}$.
```

The upper bound P_{up} can be specified by the memory available or problem at hand. The below bound is set as: $P_{down} = \max(n/C, C)$. Its underlying idea is following.

a) When $n/C > C$, that is, $C < \sqrt{n}$. There are a few conditions to specify a cell, which leads to the coarse boundaries of cells. So P should be large to produce more cells, so that the quality of the final neighborhood can be guaranteed. In this case, P is set as n/C.

b) When $n/C < C$, a cell can be constrained by the moderate number of conditions so it is refined sufficiently. Since every cell is of fine performance, a good neighborhood can be obtained without the need to yield many cells.

IV. Method Family of Neighborhood Development

VD spans a vector space where diverse norms can be defined for purposes. Assuming $\mu = VD(q, x_i)$, we give below five norm definitions:

$$\|\mu\|_1 = \min|\mu_j| \tag{4}$$

$$\|\mu\|_\infty = \max|\mu_j| \tag{5}$$

$$\|\mu\|_2 = \sqrt{\Sigma\ \mu_j^2} \tag{6}$$

$$\|\mu\|_3 = ave\{|\mu_j|\} \tag{7}$$

$$\|\mu\|_4 = \frac{\|\mu\|_\infty - \|\mu\|_1}{\|\mu\|_3} \tag{8}$$

The neighborhood is specified according to: $\|\mu\|_F < \delta$, where F = 1, 2, 3, 4, and ∞. Threshold δ is probed by following steps:

```
1) Sort VD values between $q$ and $x_i$ in the ascending
   order: $\{\|VD(q, x_1)\|_F \ldots \|VD(q, x_N)\|_F\}$;
2) Find the max gap between two adjacent values of
   that list, and let $\delta = \|VD(q, x_{gap})\|_F$, where $gap$ is
   defined as below (9):
   $gap = \max_j \{\frac{\|VD(q,x_{j+1})\|_F - \|VD(q,x_j)\|_F}{\|VD(q,x_{j+1})\|_F}\}$
   (9)
```

The above strategy thinks the max gap reveals the boundary of dense area around q, and this boundary provides a natural estimate of neighborhood. Points located before of the gap are taken as q's neighbors.

V. Experiments

A. Compared with Other Metric

As a kind of metric definition, VD is compared with some popular metrics: Euclidean metric Machete [11], Scythe [11], DANN [12], Adamenn [13].

These methods aim to reduce dimensionality and formulate data new representation by learning dimension relevance and then weighting dimensions. Machete is a recursive splitting procedure, in which the input variable used for splitting at each step is the one that maximizes the estimated local relevance. Scythe is a generalization of Machete method. DANN works as an adaptive nearest neighbor classifier. Adamenn is an adaptive nearest neighbor approach based on probability programming. Gaussian Kernel function, viewed as a kind of metric, is also compared, with its width parameter tuned by cross-validation.

These metrics are introduced into $\|x - q\|_F < \delta$ to develop neighborhoods. The neighborhood size takes two fashions: the pre-specified size NS1 and the adaptive size NS2. NS1 is expressed as a selectivity percentage to tellhow many data are selected from dataset as neighbors. NS2 is computed using our strategy mentioned in Section 4. Neighborhood quality is evaluated by:

$$\eta = ave\{|MEM_q| / |NEI_q|\} \qquad (10)$$

News Group [14] is used as experimental dataset. This dataset contains about 20,000 articles (email messages). These articles are evenly divided into the 20 newsgroups. In this paper, each newsgroup is labeled as following:

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.

We apply the usual *tf.idf* weighting schema to express documents. We delete words that appear too few times and normalize each document vector to have unit Euclidean length. We do experiments in the whole dataset with the ever-increasing NS1.

Figure 1 shows the average dependence of η on NS1, where 0.5% data are sampled randomly as queries.

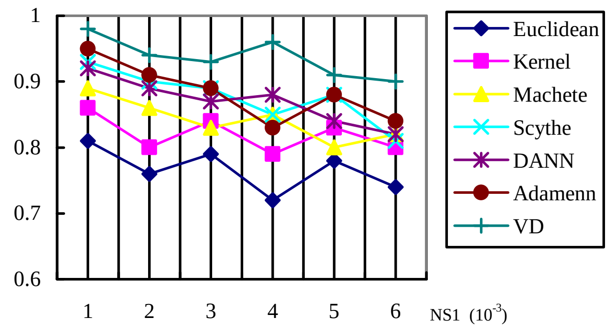

Figure 1. Neighborhood quality comparison

According to results reflected from Figure 1, it is easy to find that with NS1 increasing, η values of these methods drop at different speed. On average, the dropping speed of VD is the least sharp. Adamenn is competitive with VD. Their ratio curves are relatively gentle. Other methods yield somewhat sharp tendency curves; that suggests their performance is unsteady and they are readily to be influenced by the changes of neighborhood size. VD sees a local peak at about NS1 = 0.004; this NS1 value could be considered as the desired neighborhood size searched by cross-validation under VD metric.

Then take a look of other methods. Firstly, it is easy to find that Adamenn also has a local peak, but its peak is located at about 0.005, a larger size than that of VD. This is because Adamenn extracts complete and profound dimension relevance from data distribution, and consequently shows more effectiveness when more data are absorbed into the neighborhood.

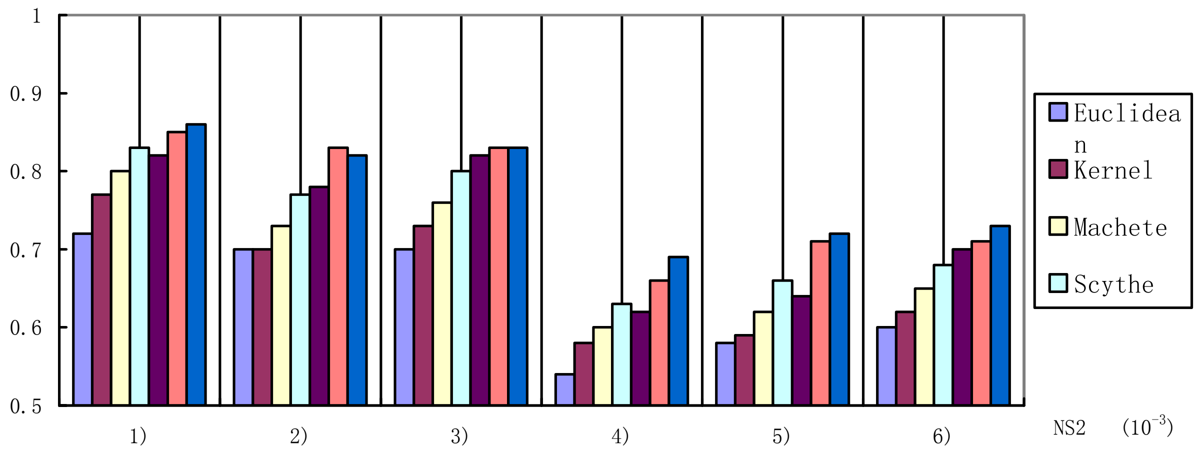

Figure 2. Neighborhood quality comparison

Secondly, it comes to DANN and Scythe. Obviously, the performance of DANN and Scythe follows first two methods and they produce similar results. But the curve of Scythe is a little sharper than DANN because DANN is equipped with adaptation to data distribution. But DANN approximates the weighted *Chi-squared* distance, which will cause its failure in datasets of non-Gaussian distribution. The local peaks of DANN and Scythe are between Adamenn and VD. Thirdly, Machete shares the same spirit of Scythe, while it employs a greedy idea, so its job is not good as Scythe.

Finally, Euclidean metric works poorly, and the reason lies in the mismatch between its measurement meaning and the high-dimensional data space. And its ratio curve is somewhat devious with more local peaks than other methods, which shows the unsteady behavior of this metric. Although Kernel's results are better than Euclidean, its curve also experiences more local peaks. Compared with the first 5 methods, Euclidean metric and Kernel metric present unsteady performance and their multi local peaks prevent finding optimal NS1 value. For Adamenn and VD, VD is a fine choice because it has computation ease brought by parameterization strategies while Adamenn has six parameters to be tuned carefully.

Then we do experiments in subsets consisting of several news groups. These experimental subsets are below six ones:

1) {NG1, NG2, NG7} (400);
2) {NG2, NG8, NG12, NG17} (300);
3) {NG11, NG12, NG16， NG19} (400);
4) {NG2 (200), NG3 (350), NG4 (400)};
5) {NG4 (200), NG5 (300), NG6 (300), NG7 (200)};
6) {NG17 (300) NG18 (500), NG19 (300)}.

Therein, the number in bracket is the size of random samples selected from the original set. Now suppose that all methods use their own NS2 value as neighborhood size. Then the corresponding ratios are described in Figure 2.

In Figure 2, all ratios are lower than corresponding peaks of Fig. 1. This is because those peak ratios are found in the searching way, while ratios of Fig. 2 are computed under fixed NS2 values. If ratios of Fig. 2 are close to peak ratios of Fig. 1, it suggests that NS2 is a qualified neighborhood size and the specification heuristic is fine. As expected, the difference between them is not far. If the cost is taken into consideration, NS2-based procedure is more welcome than NS1-based methods.

From Figure 2, the analysis of Euclidean, Kernel methods and four dimension derivation methods follow the above patterns. For VD and Adamenn, the former is competitive or even outperforms the later. In those subsets containing similar news groups, say 4), 5) and 6), VD takes more advantage than Adamenn. In that cases, class boundaries are not distinct, data original representations are unfriendly to reveal class features, and consequently the probability derivation based on these representations is less confident. That leads to the metric produced by Adamenn being not so informative. VD relies on repeating random projections without much direct dependence on data representation, therefore it is less influenced.

TABLE I.
CLASSIFICATION ACCURACY COMPARISON (%)

Data	kNN	AkNN	VkNN	SVM$_{1r}$	ASVM$_{1r}$	VSVM$_{1r}$	DAGSVM	ADAGSVM	VDAGSVM
Vote	92.2	96.8	96.8	96.1	96.7	96.1	96.1	96.5	96.7
BC1	88.3	92.5	92	89	93.9	93.7	91.1	94.2	94.1
BC2	86.2	90.7	90.5	88.4	92.6	92.6	89.8	93.4	93.2
Musk1	89.6	93.8	94.1	94.8	95.9	96	94.8	95.9	96
Musk2	59.5	62.4	63.1	62.8	67.3	68	62.8	63.7	63.8
Iris	94	97	97	95.9	97	97	96.2	96.4	96.8
Wine	92	93.9	94.7	93.1	93	90.9	93.6	94	90.6
1)	83	86.9	88.1	84.2	86.3	86.5	85.3	86.8	86.9
2)	87.1	89.2	90.5	90.1	92.1	91.6	91.2	92.8	93
3)	79.6	82	82.2	82.6	84.7	84.7	83	83.6	84.1
4)	67.7	70.3	70.1	70.2	72.9	73.4	71.1	73.5	73
5)	69.4	73.2	73.5	73.5	75.1	75.9	73.4	75.1	75.3
6)	70.4	72	72.1	71.8	73.2	73.5	72	73.8	74.1

B. Test VD Performance through Classifiers

In this experimental section, VD definition and Adamenn are introduced into some metric-based classifiers to fulfill classification task. Those metric-based classifiers are: pure kNN [15], SVM$_{1r}$ [16, 17] and DAGSVM [18].

Due to the introduction of VD definition and Adamenn, resulted methods form two groups of variants of original algorithm, and these two groups of variants are denoted by adding two prefixes to the original names, namely, adding 'V-' and 'A-' before these method names. That is, for kNN, there are two versions: AkNN and VkNN.For kNN, two metrics can work directly; for SVM$_{1r}$ and DAGSVM, two metrics appear in their Gaussian Kernel in the way that:

$$K(x,y) = \exp(-[VD(x,y)]^2 / \sigma^2) \tag{11}$$

$$K(x,y) = \exp(-\| x-y \|^2_{Adamenn} / \sigma^2) \tag{12}$$

The original classifiers and variants are compared on some real datasets that are taken from UCI Machine Learning Repository [11]. Among these experimental datasets, 1% data are sampled at random as training data and metric quality is measured by the classification accuracy listed in Table I.

Note that for bi-classification cases, SVM$_{1r}$ and DAGSVM yield same result because both of them train one basic SVM. It is easy to find that 'A-' and 'V-' variants improve their original models, which indicates two metrics do improve effectiveness. But two metrics take their own advantage in different scenarios. In low-dimensional datasets, 'A-' methods behave better, while in high-dimensional space, 'V-' family is relatively preferred.

According to experimental evidence of this section, it concludes that VD is more suitable to high-dimensional space than Adamenn because the later derives new metric from statistics of data distribution. Those statistics, as measurements, might be caught by the curse of dimensionality, and consequently the resulted metric becomes less informed. However, VD focuses on learning valid information from repeating projections, thus does not suffer from this problem. For SVM$_{1r}$ and DAGSVM, the later accounts to a weighted framework of basic SVM, while the former is the non-weighted framework of SVM, so naturally the later gives higher accuracies.

C. Test VD Performance for High-dimensional Data

After investigating the performance of VD through comparing it with other metrics and other classifiers, This section experiment aims to compare VD with some NN searching techniques that are developed specially for high-dimensional data. These NN searching techniques are those famous ones: VA-file [19, 20, 21, 22, 23], iLSH [6], cLSH [10] and bLSH [9]. To check their property, another evaluation is used, neighborhood cohesion. For q's neighborhood, *NEI* (q), its cohesion is defined as formula (13):

$$\Sigma_{x,y \in NEI(q)} \frac{\exp(-\| x-y \|^2)}{| NEI(q) |} \tag{13}$$

Suppose that each method finds its own NS2 value. Here, 10% data are sampled randomly as queries. The average cohesion values of each method are collected and shown in Table II.

TABLE II.

COMPARISON OF LSH-BASED APPROACHES THROUGH AVERAGE COHESION

Data	1)	2)	3)	4)	5)	6)
VA-file	0.73	0.78	0.86	0.7	0.67	0.65
iLSH	0.71	0.75	0.83	0.68	0.68	0.63
cLSH	0.74	0.74	0.87	0.71	0.67	0.66
bLSH	0.79	0.79	0.86	0.73	0.71	0.68
VD	0.78	0.79	0.85	0.75	0.71	0.7

From Table II, it is clear that bLSH and VD behave similarly and take their own advantage in various cases.

bLSH does better in subsets with clear class boundaries, while VD exhibits its merit in handling subsets with blurred classes. cLSH and VA-file yield close results for they actually construct the same-shaped bucket using different strategies. iLSH is relatively poor due to its weak hashing power carried by the one-dimensional hashing embeddings. bLSH can be seen the best one among four indexing approaches because the shape of its bucket is the best to approximate the inherent shape of neighborhood, and thus it has more ability to formulate the good neighborhood which contains true neighbors.

The mismatch between neighborhood region shape and rectangular, cell and interval leads to some irrelevant data absorbed into neighborhood, so affects the quality of neighborhood spanned by other three methods. Although VD's bucket is of cell-like shape, it exploits the informed weighted metric to sort neighbors. That removes the influence brought by the mismatch.

D. Test VD Performance for Real Dataset

Finally, VD is conducted on the real datasets: Musk [24] and Mutagenesis [25]. Musk dataset has two versions Musk1 and Musk2. They record 476 and 6598 conformations for musk molecules and non-musk molecules. We fix the data with some normalization and develop its relation frame shown in Figure 3.

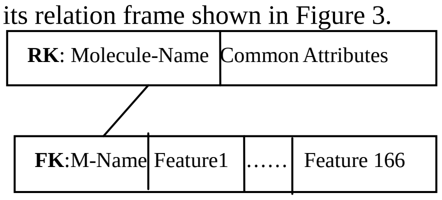

Figure 3. Dataset Relationship of Musk

Mutagenesis dataset records 230 aromatic and heteroaromatic nitro compounds. They are divided into two groups based on the mutagenicity: the active and the inactive. Its relation schema is formulated in Figure 4.

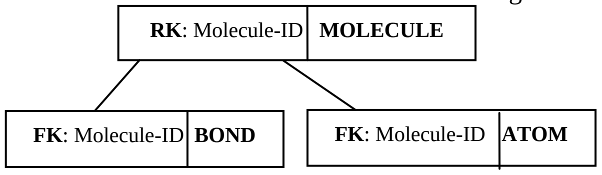

Figure 4. Dataset Relationship of Mutagenesis

Here, for two datasets, their tables are integrated into one big table, which includes all records. Note that since there are two types of Mutagenesis molecules: regression-friendly and regression-unfriendly, some data are sampled from two subsets and the whole dataset respectively. Table III shows classification accuracy of above methods.

TABLE III.
COMPARISON OF LSH-BASED APPROACHES THROUGH CLASSIFICATION ACCURACY (%)

Data	Musk	Mutagenesis (*friendly*)	Mutagenesis (*unfriendly*)	Mutagenesis (*full*)
VA-file	86.3	79.5	63.2	73.1
iLSH	87.6	82.9	68.7	75.2
cLSH	85.2	92.7	70.2	78.9
bLSH	88.4	90.5	70.5	77.2
VD	90.5	93.5	70.6	79.5

According to the experimental results, it is easy to know that VD exhibits outstanding behaviors by presenting highest classification accuracy among five methods. That is the similar conclusion with above section. Other LSH-based methods follow VD.

Through above experiments, the fine performance of VD definition can be verified.

VI. CONCLUSION

This paper proposes a novel distance concept customized to high-dimensional data, Vector-Distance (VD). VD is a vector with its entries reflecting multi-aspect information summarized from random projections. Through applying some metric to VD values, a family of neighborhood formulation methods is yielded. Empirical evidence on real datasets demonstrates the fine performance of VD and the proposed method family.

On the other hand, VD provides another thinking angle by extending the common distance information to a compound fashion. Consequently the new distance collects rich information, and this information reveals more to reflect the relationship among data. That is expected to make more importance in high-structured data environment.

Furthermore, high-dimensional data always attracts a lot of interest in diverse applications, more work is needed to promote the improvement of algorithms of high-dimensional data, even high-structured data.

ACKNOWLEDGMENT

This work is supported by the Youth National Natural Science Foundation of China under Grant No. 61105129. And this work thanks for the contribution of Dr. Xiangsheng Rong.

REFERENCES

[1] E Novak, K Ritter, "The Curse of Dimension and a Universal Method for Numerical Integration," *Multivariate Approximation and Splines*, pp. 177-187, 1998.
[2] Michael E. Houle, Hans-Peter Kriegel, Peer Kröger, Erich Schubert and Arthur Zimek, "Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?" *Lecture Notes in Computer Science*, Volume 6187, pp. 482-500, 2010.
[3] Jiajun Liu, Zi Huang, Heng Tao Shen and Xiaofang Zhou, "Efficient Histogram-Based Similarity Search in Ultra-High Dimensional Space," *Lecture Notes in Computer Science*, Volume 6588, pp. 1-15, 2011.
[4] 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.
[5] D. Comaniciu, P. Meer: Mean shift, "A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, Vol. 24, No. 5, pp. 603-619, 2002.
[6] M. Datar, N Immorlica, P Indyk, "Locality-sensitive hashing scheme based on p-stable distributions," *Proceeding of 20th annual symposium on Computational geometry*, p-. 253–262, 2004.
[7] A. Chakrabarti, S.Khot, S. Xiaodong, "Near-Optimal Lower Bounds on the Multi-Party Communication Complexity of Set Disjointness," *Proceeding of 18th Annual IEEE Conference on Computational Complexity*, pp. 107-117, 2003.
[8] X.G. Cheng, C. Yang, J. Yu, "A New Approach to Group Signature Schemes," *Journal of Computers*, Vol. 6, No 4. pp. 812-817, 2011.
[9] 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.
[10] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: towards Removing the Curse of Dimensionality," *Proceeding of Symposium on Theory of Computing*, pp. 604-613, 1998.
[11] http://www.uncc.edu/knowledgediscovery
[12] T. Hastie, R. Tibshirani, "Discriminant Adaptive Nearest Neighbor Classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*. Vol. 18(6), pp. 607-615, 1996.
[13] C. Domeniconi, J. Peng, D. Gunopulos, "An Adaptive Metric Machine for Pattern Classification," *Advances in Neural Information Processing Systems*, Vol. 13. pp. 458-464, 2001.
[14] http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html
[15] K.N.N. Unni, R. de Bettignies, S.-D. Seignon and J.-M. Nunzi, "Application Physics Letter," Vol. 85, pp. 1823. 2004.

[16] V.Murino, M. Bicego, I.A. Rossi, "Statistical classification of raw textile defects," *Proceedings of the 17th International Conference on Pattern Recognition*, Vol. 4, pp. 311-314, 2004.

[17] C.C. Chang, C.J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, Volume 2 Issue 3, pp. 35-63, 2011.

[18] J. C. Platt, N. Cristianini, J. Shawe-Taylor, "Large margin DAG's for Multiclass classification," *Advances in neural information processing systems*. MIT press, Cambridge, MA, 12: 547-553, 2000.

[19] R. Weber, H. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," *Proceeding of 24th International Conference on Very Large Data Bases*, pp. 194–205. 1998.

[20] Peisen Yuan, Chaofeng Sha, Xiaoling Wang, Bin Yang, Aoying Zhou, "Efficient Approximate Similarity Search Using Random Projection Learning", *Lecture Notes in Computer Science*, Volume 6897, pp. 517-529, 2011.

[21] C.H. Zhou, M.L. Cao, M. Ye, Z.H. Qian, "SAT-based Algorithmic Verification of Noninterference," *Journal of Computers*, Vol. 6, No 11, 2310-2320, 2011.

[22] Z.Q. Wang, X. Sun, "An Efficient Discriminant Analysis Algorithm for Document Classification," *Journal of Computer*, Vol 6, No 7, pp. 1265-1272, 2011.

[23] Chunyang Ma, Yongluan Zhou, Lidan Shou, Dan Dai, Gang Chen, "Matching Query Processing in High-dimensional space," Proceeding of the 20th ACM International Conference on Information and Knowledge Management, pp. 32-40, 2011.

[24] J. Renders, "Kernel Methods in Natural Language Processing," *Learning Methods for Text Understanding and Mining Conference Tutorial*, 2004.

[25] L. Lopriore, "Page Protection in Multithreaded Systems," *Journal of Computers*, Vol. 5, No 9, pp. 1297-1304, 2010.

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.

Xiangyang You was born in Xuzhou, Jiangsu Province, China, 1972. He received his Bachelor's degree in College of Computer Science and Technology, Harbin Institute of Technology. His research directions are operational research in military logistic command, etc.

Ming Xu was in Suqian, Jiangsu Province, China, 1968. He received his Bachelor's degree in 1994, from Department of Logistic Command, Xuzhou Air Force College of P. L. A. And then he received his Master's degree in 2005 from Xuzhou Air Force College of P. L. A. His research fields are centered in data integration, data mining, semantic network, etc.