Improving k-NN Graph Accuracy Using Local Intrinsic Dimensionality

Michael E. Houle[1], Vincent Oria[2], and Arwa M. Wali[2,3(✉)]

[1] National Institute of Informatics, Tokyo 101-8430, Japan
meh@nii.ac.jp
[2] New Jersey Institute of Technology, Newark, NJ 07102, USA
{vincent.oria,amw7}@njit.edu
[3] King Abdulaziz University, Jeddah, Saudi Arabia

Abstract. The k-nearest neighbor (k-NN) graph is an important data structure for many data mining and machine learning applications. The accuracy of k-NN graphs depends on the object feature vectors, which are usually represented in high-dimensional spaces. Selecting the most important features is essential for providing compact object representations and for improving the graph accuracy. Having a compact feature vector can reduce the storage space and the computational complexity of search and learning tasks. In this paper, we propose NNWID-Descent, a similarity graph construction method that utilizes the NNF-Descent framework while integrating a new feature selection criterion, Support-Weighted Intrinsic Dimensionality, that estimates the contribution of each feature to the overall intrinsic dimensionality. Through extensive experiments on various datasets, we show that NNWID-Descent allows a significant amount of local feature vector sparsification while still preserving a reasonable level of graph accuracy.

Keywords: Intrinsic dimensionality · k-nearest neighbor graph · Feature selection · Vector sparsification

1 Introduction

The k-nearest neighbor (k-NN) graph is a key data structure used in many applications, including machine learning, data mining, and information retrieval. Some prominent examples for k-NN graph utilization include object retrieval [21], data clustering [3], outlier detection [8], manifold ranking [9], and content-based filtering methods for recommender systems [22]. In applications such as multimedia and recommender systems where data objects are represented by high-dimensional vectors, the so-called 'curse of dimensionality' poses a significant challenge to k-NN graph construction: as the dimensionality increases, the discriminative ability of similarity measures diminishes to the point where methods such as k-NN graph search that depend on them lose their effectiveness.

The construction of k-NN graphs using brute-force techniques requires quadratic time, and is practical only for small datasets [4]. One recent technique that efficiently constructs an approximate k-NN graph in a generic metric

C. Beecks et al. (Eds.): SISAP 2017, LNCS 10609, pp. 110–124, 2017.
DOI: 10.1007/978-3-319-68474-1_8

space is NN-Descent [4]. NN-Descent is an iterative algorithm that follows a simple transitivity principle: two neighbors of a given data object have a higher chance of being neighbors of each other. When ground truth class information is available, the accuracy of a k-NN graph can be measured in terms of the proportion of edges that connect nodes sharing the same class label. A common approach for maximizing k-NN graph accuracy is to incorporate dimensionality reduction techniques in the graph construction process. This can be done either independently as a preprocessing step using techniques such as Sparse Principal Component Analysis (Sparse PCA) [25], or integrated within the graph construction process itself, such as feature weighting [7] or other supervised feature selection approaches [23]. However, supervised feature selection would depend on ground truth information, which may not be always available.

In [15], an unsupervised method is presented, NNF-Descent, that iteratively and efficiently improves k-NN graph construction using the Local Laplacian Score (LLS) as a feature selection criterion. LLS favors those features that have high global variance among all objects, but less variance among the neighborhood of a given target object. The NNF-Descent method identifies locally noisy features relative to each object in the dataset — that is, those features having larger LLS scores. The noisy features are then gradually modified using a local sparsification process so as to decrease the distances between related objects, and thereby increase k-NN graph accuracy. NNF-Descent has already shown significant improvement in the semantic quality of the graphs produced, and superior performance over its competitors on several image databases [15]. However, NNF-Descent is a conservative method in that only a fixed small number of noisy features are sparsified in each iteration. With greater rates of feature sparsification, the k-NN graph accuracy tends to decrease. This also occurs when increasing the neighborhood size k beyond (roughly) 10. NNF-Descent is designed for datasets with dense feature vectors. In sparse datasets, vectors may contain very few non-zero features, in which case the sparsification process may incorrectly remove valuable features [15].

In this paper, we address the problem of improving the tradeoff between k-NN graph accuracy and the degree of data sparsification. We present the NNWID-Descent similarity graph construction method, which utilizes the NNF-Descent framework with a new feature selection criterion, Support-Weighted Intrinsic Dimensionality (support-weighted ID, or wID) [14]. Support-weighted ID is an extension of the Local Intrinsic Dimensionality (LID) measure introduced in [1,12], and is used within NNWID-Descent to identify and retain relevant features of each object. Unlike LLS, which is a variance-based measure, support-weighted ID penalizes those features that have lower locally discriminative power as well as higher density. In fact, support-weighted ID measures the ability of each feature to locally discriminate between objects in the dataset.

The remainder of this paper is organized as follows. Section 2 provides background on the relevant feature selection research literature, and on unsupervised approaches in particular. An overview of the NNF-Descent framework is presented in Sect. 3. We outline the proposed NNWID-Descent method in Sect. 4.

In Sect. 5, the performance of our method — with experimental results and analysis on several real datasets — is compared to NNF-Descent and other competing methods from the literature. Finally, we conclude in Sect. 6 with a discussion of future research directions.

2 Related Work

A brief review of feature selection techniques is provided in this section, with a particular emphasis on unsupervised methods.

2.1 Supervised Feature Selection and *k*-NN Graph Construction

Feature selections methods are commonly used in supervised learning methods to maximize their predictive accuracy. For example, Han et al. [7] proposed a Weight Adjusted k-Nearest Neighbor (WAKNN) classification scheme where the weights of the features are learned using an iterative algorithm. In [23], a supervised feature selection method is presented that uses an improved k-NN graph-based text representation model to reduce the number of features and predict the category of the text in the test set.

2.2 Unsupervised Feature Selection

In unsupervised feature selection methods, class information is not available, and thus it is difficult to decide the importance of a feature — especially when many of the features may be redundant or irrelevant [5]. Most existing unsupervised feature selection approaches are customized to a particular search or clustering algorithm.

Unsupervised feature selection methods can be further classified into global feature selection methods and local feature selection methods. In global feature selection methods, the features are selected based on their relevancy that has been computed globally using the entire dataset. The Laplacian Score (LS) [10] is one of the most popular unsupervised filter-based methods for generic data. LS selects the features to be used for all objects in the dataset based on their ability to discriminate among object classes. LS favors features that have high variance on the entire datasets and low variance within local neighborhoods. Local feature selection methods are based on the idea that the discriminative power and the importance of a feature may vary from one neighborhood to another; they aim to select features based on their relevancy to a given neighborhood. For example, Li et al. [18] introduced a localized feature selection algorithm for clustering that is able to reduce noisy features within individual clusters. Their algorithm computes, adjusts, and normalizes the scatter separability for individual clusters before applying a backward search technique to find the optimal (local) feature subsets for each cluster. Mitra et al. [20] introduced an algorithm that partitions the original feature set into clusters based on a k-NN graph principle. To detect and remove redundant features, their algorithm uses a pairwise

feature similarity measure, the Maximum Information Compression index, that find the linear correlation between features in the clusters. This algorithm has a low computational complexity, since it does not involve any search for feature subsets [20]. However, their model may be too restrictive for real datasets, since correlations among features within clusters may not exist, or may be non-linear when they do exist [24].

3 Overview of NNF-Descent

As the basis for the work presented in this paper, in this section we provide an overview of the NNF-Descent algorithm [15]. We also describe its feature selection criterion, the Local Laplacian score LLS, and discuss its utilization in feature ranking and sparsification processes.

3.1 Local Laplacian Score, Feature Ranking, and Sparsification

Local Laplacian Score LLS is used for feature ranking with respect to individual data objects. Assume we have a dataset X with n data objects, each represented by a D-dimensional feature vector $\mathbf{f} = (f_1, f_2, \ldots, f_D)$. We further assume that the vectors are normalized. Then, for an object $x_i \in X$, the LLS score for each of its feature f_i can be computed using the following formula:

$$LLS(f_i) = \sum_j \frac{(f_i - f_j)^2 S_{ij}}{var(\mathbf{f})} \tag{1}$$

where $var(\mathbf{f})$ is the variance of feature $\mathbf{f}$, and S_{ij} is the (Gaussian) RBF kernel similarity between two object vectors x_i and x_j defined as:

$$S_{ij} = \begin{cases} exp(\frac{-\|x_i - x_j\|^2}{2\sigma^2}), & \text{if } i \text{ and } j \text{ are connected;} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Here, σ is a bandwidth parameter. S_{ij} favors neighboring objects x_i and x_j that are likely to share the same class label. A smaller value for LLS(f_i) indicates that the feature is stable among the neighbors of object x_i. The features are ranked for each object in decreasing order of their LLS values, and the top-ranked proportion Z of the ranked list is deemed to be noise. In the sparsification process, the impact of noisy features is minimized by changing their values in the feature vectors to the global mean, which is zero due to normalization.

3.2 NNF-Descent

The NNF-Descent framework interleaves k-NN graph construction using NN-Descent [4] with a feature ranking and sparsification process. Algorithm 1 gives the complete algorithm for NNF-Descent. After normalizing the original vectors

of the dataset X, the algorithm starts by computing the initial approximate k-NN graph using NN-Descent [4] (lines 1–2). The NN-Descent procedure depends on the so-called *local join* operation. Given a target point p, the local join operation checks whether any neighbor of p's neighbors is closer to p than any points currently in its neighbor list, and also whether pairs of neighbors of p can likewise improve each other's tentative neighbor list. Noisy features are gradually identified using LLS, ranked, and then sparsified.

Algorithm 1. NNF-Descent

Input : Dataset X, distance function dist, neighborhood size K, sparsification rate Z, number of iterations T
Output: k-NN graph G

1 Normalize the original feature vectors of X;
2 Run NN-Descent(X, dist, K) to convergence to obtain an initial k-NN graph G;
3 **repeat**
4 Generate a list L of all data points of X in random order;
5 **foreach** *data point* $p \in L$ **do**
6 Rank the features of p in descending order of their LLS scores, as computed over the current k-NN list of p;
7 Change the value of the top-ranked Z-proportion of features to 0;
8 Recompute the distances from p to its k-NN and RNN points;
9 Re-sort the k-NN lists of p and its RNNs;
10 For each pair (q, r) of points from the k-NN list and RNN list of p, compute $\text{dist}(q, r)$;
11 Use $(q, \text{dist}(q, r))$ to update the k-NN list of r, and use $(r, \text{dist}(q, r))$ to update the k-NN list of q;
12 **end**
13 **until** *maximum number of iterations* T *is reached*;
14 Return G

4 Improving NN-Descent Graph with Weighted ID

The NNF-Descent framework, which integrates feature ranking and sparsification with k-NN graph construction, serves as the basis for the method presented in this paper, NNWID-Descent. In NNWID-Descent, instead of feature variance, a measure of the discriminability of features is used for feature ranking. In this section, we first provide a brief overview of this measure of discriminability, the Support-Weighted Local Intrinsic Dimensionality or support-weighted ID (Sect. 4.1). The utilization of support-weighted ID as a feature selection criterion is then presented in Sect. 4.2. Finally, the details of the proposed NNWID-Descent algorithm is given in Sect. 4.3.

4.1 Support-Weighted Local Intrinsic Dimensionality

As an alternative to the Local Laplacian Score, we propose in this paper a new feature evaluation strategy based on the Local Intrinsic Dimension ('Local ID', or 'LID') model originally appearing [12]. Given a distribution of distances with a univariate cumulative distribution function F that is positive and continuously differentiable in the vicinity of distance value x, the indiscriminability of F at x is given by

$$\mathrm{ID}_F(x) \triangleq \frac{x \cdot F'(x)}{F(x)}\,. \tag{3}$$

The indiscriminability reflects the growth rate of the cumulative distance function at x; it can be regarded as a probability density associated with the neighborhood of radius x (that is, $F'(x)$), normalized by the cumulative density of the neighborhood (that is, $F(x)/x$). The local intrinsic dimension has been shown to be equivalent to a notion of local intrinsic dimensionality, which can be defined as the limit $\mathrm{ID}_F^* = \lim_{x \to 0^+} \mathrm{ID}_F(x)$. However, the notion of local ID as proposed in [13,14] is considerably more general, in that the original model of [12] has been extended to handle multivariate real-valued functions that are not necessarily the cumulative distribution functions of distance distributions.

When considering a distance distribution on a space of many features, it is natural to ask which variables or features are contributing most to the overall discriminability of the function or cumulative distribution function (as the case may be). Two variables or features with the same local ID value may not necessarily have the same impact on the overall ID value. To illustrate this, let Φ and Ψ be the respective cumulative distribution functions of two univariate distance distributions on distance variable x.

The indiscriminability $\mathrm{ID}_\Phi(x)$ can be thought of as having a 'support' equal to the probability measure associated with distance x — namely, $\Phi(x)$; similarly, the support for $\mathrm{ID}_\Psi(x)$ would be $\Psi(x)$. Even when the indiscriminabilities $\mathrm{ID}_\Phi(x)$ and $\mathrm{ID}_\Psi(x)$ are equal, if (say) the support $\Phi(x)$ greatly exceeded $\Psi(x)$, one would be forced to conclude that the features associated with ID_Φ are more significant than those of ID_Ψ, at least within the neighborhood of radius x.

For the comparison of the discriminabilities of different features in our proposed adaptation of NNF-Descent, we will adopt the following Support-Weighted ID complexity measure. This measure has the highly desirable theoretical advantage of being additive across features (for more details we refer the reader to [14]).

Definition 1 (Support-Weighted ID [14]). *Let F be a real-valued multivariate function over a normed vector space $(\mathbb{R}^m, \|\cdot\|)$, and let $\mathbf{x} \neq \mathbf{0} \in \mathbb{R}^m$ be a vector of positive norm. The* support-weighted indiscriminability of F *at* $\mathbf{x}$ *is defined as*

$$\mathrm{wID}_F(\mathbf{x}) \triangleq F(\mathbf{x})\,\mathrm{ID}_F(\mathbf{x}) = \mathbf{x} \cdot \nabla F(\mathbf{x})\,. \tag{4}$$

Estimating support-weighted ID for the purpose of assessing indiscriminability can be complicated by the need to standardize the distance within which the indiscriminabilities are measured — in a k-NN graph, each neighborhood is

associated with its own potentially-unique k-NN distance. If each feature were to assessed at widely-varying distances, there would be no basis for the fair comparison of feature performance.

In practice, however, estimation of ID requires samples that are the result of a k-nearest neighbor query on the underlying dataset. Across such samples, standardization can be achieved using the local ID representation theorem:

Theorem 1 (Local ID Representation Theorem [13]). *Let $\Phi : \mathbb{R} \to \mathbb{R}$ be a real-valued function, and let $v \in \mathbb{R}$ be a value for which $\mathrm{ID}_\Phi(v)$ exists. Let x and w be values for which x/w and $\Phi(x)/\Phi(w)$ are both positive. If Φ is non-zero and continuously differentiable everywhere in the interval $[\min\{x,w\}, \max\{x,w\}]$, then*

$$\frac{\Phi(x)}{\Phi(w)} = \left(\frac{x}{w}\right)^{\mathrm{ID}_\Phi(v)} \cdot G_{\Phi,v,w}(x), \text{ where} \tag{5}$$

$$G_{\Phi,v,w}(x) \triangleq \exp\left(\int_x^w \frac{\mathrm{ID}_\Phi(v) - \mathrm{ID}_\Phi(t)}{t}\,\mathrm{d}t\right), \tag{6}$$

whenever the integral exists.

For a univariate cumulative distribution function Φ at distance x, we can use Theorem 1 with $v = 0$ to relate the support $\Phi(x)$ with the support at another desired distance w. If n is the size of the dataset that we are given, we choose the distance at which over n selection trials one would expect k samples to fall within the neighborhood — that is, w would satisfy $\Phi(w) = k/n$. The support-weighted ID would thus be:

$$\mathrm{wID}_\Phi(x) \;=\; \Phi(x)\,\mathrm{ID}_\Phi(x) \;=\; \frac{k\,\mathrm{ID}_\Phi(x)}{n} \cdot \left(\frac{x}{w}\right)^{\mathrm{ID}^*_\Phi} \cdot G_{\Phi,0,w}(x)\,. \tag{7}$$

In [13] it is shown that (under certain mild assumptions) the function $G_{\Phi,0,w}(x)$ tends to 1 as $x, w \to 0$ (or equivalently, as $n \to \infty$); also, $\mathrm{ID}_\Phi(x)$ would tend to ID^*_Φ, for which reliable estimators are known [1,11]. Thus, for reasonably large dataset sizes, we could use the following approximation:

$$\mathrm{wID}_\Phi(x) \;\approx\; \frac{k\,\mathrm{ID}^*_\Phi}{n} \cdot \left(\frac{x}{w}\right)^{\mathrm{ID}^*_\Phi}. \tag{8}$$

4.2 Defining Support-weighted ID (wID) for each Feature

Let $X = \{x_1, x_2, x_3, \ldots, x_n\}$ be a dataset consisting of n objects such that each object x_i is represented as a feature vector in $\mathbb{R}^D$. The set of features is denoted as $F = \{1, 2, \ldots, D\}$ such that $j \in F$ is the j-th feature in the vector representation. Since the factor k/n in Eq. 8 can be regarded as constant, the support-weighted ID criterion for feature f_j of object x_i can be simplified:

$$\mathrm{wID}_i(f_j) = \mathrm{ID}_{f_j} \cdot \left(\frac{a_f}{w_{f_j}}\right)^{\mathrm{ID}_{f_j}} \tag{9}$$

where ID_{f_j} is the local intrinsic dimensional estimate for the neighborhood, and w_{f_j} is the distance to the k-th nearest neighbor with respect to feature f_j, respectively. a_f is any positive constant representing the distance value x. For simplicity, a_f can be set as an average of a sample of k-NN distances across many objects for feature f_j. Equation (9) helps to find the most discriminative features by considering both the density of neighborhood around each object and the complexity of local ID with respect to a particular feature f_j.

For feature ranking, a straightforward method is used for selecting the most local discriminative features for each object using wID_i, in which the D features are ranked in descending order of $\mathrm{wID}_i(f_j)$, and a proportion Z of the top-ranked features are determined as candidates for sparsification. Assuming that the feature vectors have been normalized, the sparsification process (described in Sect. 3.1) will set the values of the least important features to 0.

4.3 NNWID-Descent

Algorithm 2 shows how NNWID-Descent proceeds. The input parameters are K, Z, and T, where $K \geq k$ is the working neighborhood size during the construction of the output k-NN graph, Z is a fixed proportion of features that are sparsified in each iteration, and T is the total number of desired iterations. The feature sparsification rate Z should be relatively small.

The algorithm has two phases: an initialization phase, and a sparsification and refinement phase. In the initialization phase, the algorithm computes a K-NN graph using NN-Descent after normalizing the original vectors of the dataset X (lines 2–4). This step is crucial, since a neighborhood of reasonably high quality is needed for the subsequent refinement phase to be effective.

In line 4, the value of a_f for each feature is precomputed for use in calculating wID values, during the sparsification and refinement phase. As will be described in Sect. 5.4, the value a_f can be computed as the average of the K-NN distances using the feature f alone, over a sample of the data objects. The K-NN graph entries are then improved using the sparsification and refinement phase (Lines 6–16). This phase includes three steps: feature ranking, sparsfication, and graph updating. In lines 9–10, the features are ranked in decreasing order according to the wID values obtained from the set of K-NN distances determined by each feature alone. For each object p, the top Z-proportion of features are then sparsified (line 11). As will be described in Sect. 5.4, the value Z is chosen depending on the density of the dataset X. As in [15], only non-zero features are candidates for sparsification, since features with value 0 do not provide discriminative information in the vicinity of p, and thus do not affect the quality of the K-NN graph. Ignoring zero features will ensure that once sparsified, a feature will not be evaluated again in subsequent iterations. Sparsifying a feature vector for p in one iteration will more likely change the nearest neighbors for each feature of p; for this reason, to determine the correct wID value in subsequent iterations, recomputation of the K-NN distances is required for each feature.

Lines 12–14 correspond to Lines 8–11 in NNF-Descent (Algorithm 1) which identify the local join operation and graph update step to improve the graph

Algorithm 2. NNWID-Descent

Input : Dataset X, distance function dist, neighborhood size K, sparsification rate Z, number of iterations T
Output: k-NN graph G

{Initialization Phase}
Normalize the original feature vectors of X;
Run NN-Descent(X, dist, K) to convergence to obtain an initial K-NN graph G;
For each feature f, set the value of a_f to the average of K-NN distances computed for the feature over a sample of objects.
{Sparsification and Refinement Phase}
repeat
    Generate a list L of all data points of X in random order;
    foreach *data point* $p \in L$ **do**
        For each feature, compute the K-NN distances from p with respect to X;
        Rank the features of p in descending order of their wID scores (Eq. 9), as computed over the current K-NN list of p;
        Change the value of the top-ranked Z-proportion of features to 0;
        Recompute the distances from p to its K-NN and RNN points;
        Re-sort the K-NN lists of p and its RNNs;
        For each pair (q, r) of points from the K-NN list and RNN list of p, compute $\text{dist}(q, r)$;
        Use $(q, \text{dist}(q, r))$ to update the K-NN list of r, and use $(r, \text{dist}(q, r))$ to update the K-NN list of q;
    end
until *maximum number of iterations* T *is reached*;
Return G

accuracy. In the implementation, we set $K \geq k$ to be the length for both RNN and NN lists used in computing wID.

The time complexity of NNWID-Descent can be divided according to its phases as follows: For the initialization phase, data normalization and NN-Descent —in terms of distance computation until convergence— take $O(Dn)$ and $O(K^2Dn)$ time, respectively. Computing the values of a_f for all features using average k-NN distances takes $O(Dn^2)$. For each iteration of the sparsification and refinement phase, feature ranking and selection using wID takes $O(KDn + D \log D)$ time per object, with total time in $O(KDn^2 + Dn \log D)$ over all objects. As with NN-Descent, assuming that the lengths of the RNN lists are in $O(K)$, each iteration of NNWID-Descent takes $O(K^2Dn)$ time for the neighbor list update step. However, the optimizations that have been defined for NN-Descent in [4] can also applied for NNWID-Descent to speed up the local join operation and update steps.

5 Experiments

For the comparison of NNWID-Descent with competing methods, we conducted experiments to study the influence on performance of varying the feature sparsification rate Z and the working neighbor list size K.

5.1 Datasets

Six real datasets of varying sizes and densities were considered, of which five are image sets:

- The Amsterdam Library of Object Images (ALOI) [6] contains 110,250 images of 1000 small objects. Each image is described by a 641-dimensional feature vector based on color and texture histograms.
- The MNIST dataset [17] contains 70,000 images of handwritten digits. Each image is represented by 784 gray-scale texture values.
- Google-23 [16] contains 6,686 faces extracted from images of 23 celebrities. The dimension of the face descriptors is 1,937.
- The Isolated Letter Speech Recognition dataset (ISOLET) [19] contains 7797 objects generated by having 150 subjects speak the name of each letter of the alphabet twice. Each object is described by 617 features, and were scaled so that all values lie in the interval $[-1.0, 1.0]$.
- The Human Activity Recognition Using Smartphones dataset (HAR) [2] contains 10,299 instances of accelerometer and gyroscope data from 30 subjects performing 6 different activities. Each instance is represented by a feature vector of 561 time and frequency domain variables.
- The Relative Location of CT dataset (RLCT) [19] contains 53500 axial CT slice images from 74 different patients. Each CT slice is described by two histograms in polar space. The feature vectors of the images are of 385 dimensions.

5.2 Competing Methods

The performance of NNWID-Descent is contrasted with that of 3 competitors:

- NNF-Descent: uses LLS criterion for feature ranking and sparsification (as described in Sect. 3).
- Random: as per NNF-Descent, except that for each object the features to be sparsified are selected randomly. The rationale for the comparison with this method is to establish a baseline for the performance of the feature ranking and sparsification criterion.
- Sparse PCA: is similar to wID in such that it takes into account the dataset sparsity. In this method, the feature extraction and graph construction are conducted as two separate processes. To allow a fair comparison with other methods, after choosing the highest principal components, an exact k-NN graph is computed (at a computation cost of $O(Dn^2)$).

5.3 Performance Measure

We use the graph accuracy as a performance measure. The class labels of data objects were used to measure the quality of the resulting k-NN graph at every iteration. The accuracy of the resulting k-NN graph is evaluated, as in [15], using the following formula:

$$graph\ accuracy = \frac{\#correct\ neighbors}{\#data \times K}, \tag{10}$$

where the 'correct' neighbors share the same label as the query object.

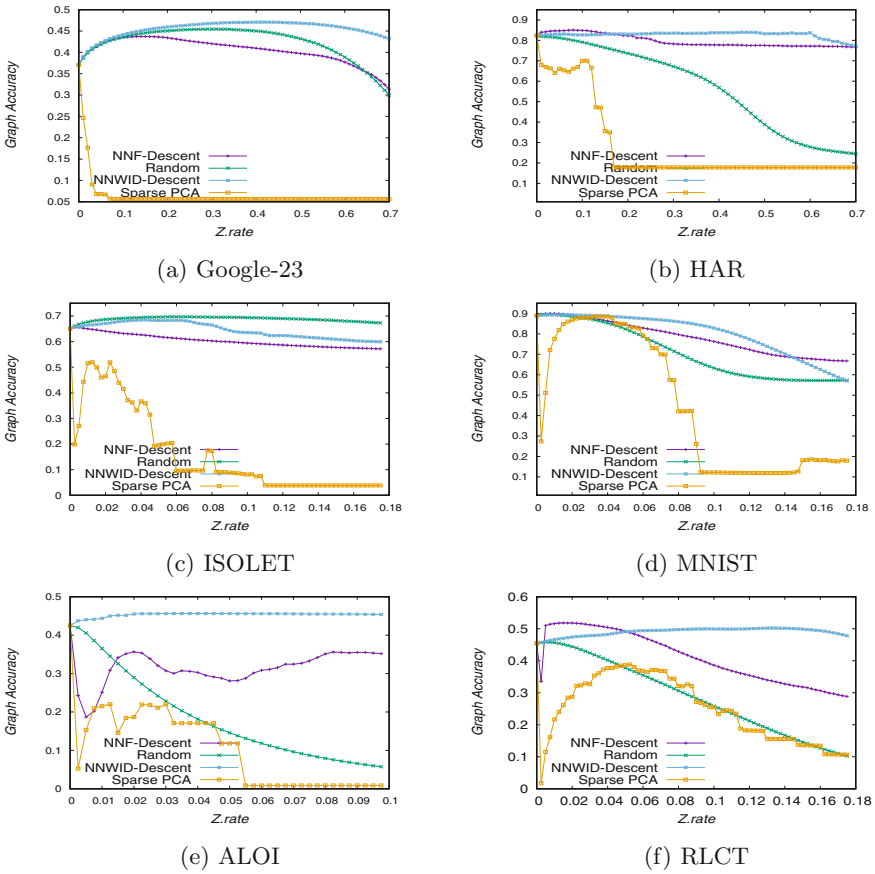
5.4 Default Parameters

Except for the case of Sparse PCA, the feature vectors were normalized within each dataset in each experiment performed, and the Euclidean (L_2) distance was employed. In NNWID-Descent, for the datasets Google-23, HAR, and ISOLET, the value of a_f in the weight parameter of Equation (9) is set to be the average of distances (using feature f) computed from the neighbors of all objects in the dataset; for ALOI, MNIST, and RLCT, the average was computed over a random sample of 100 objects. Furthermore, for all features, the value a_f is precomputed in advance using the original feature vectors without sparsification. For simplicity, the number of neighbors used for computing wID and LLS is set to be equal to the input parameter K.

5.5 Effects of Varying the Sparsification Rate Z

Parameter Setting. In this experiment, we tested the effect on performance of varying Z while keeping K fixed. The choices of Z is varied with different datasets as it depends heavily on the density of the feature vectors. For example, in each iteration, smaller choice of Z (= 0.0025%) for the sparse datasets (MNIST, ALOI, ISOLET, and RLCT) was required to produce gradual changes in graph accuracy with acceptable performance. On the other hand, the dense datasets (Google-23 and HAR) require a larger starting point for Z (= 0.1%)to produce perceptible changes in performance from iteration to iteration. For Sparse PCA, the parameter controlling sparsity was set to Z, and the number of principle components selected were set to ZD. The total number of iterations T is set to 70 for all datasets except ALOI, for which T is set to 40. For all methods in the comparison, the value of K is fixed at 100. Figure 1 shows plots of the graph accuracy in each iteration for all the methods, across a range of Z values.

Results and Analysis. On five of the six datasets, compared with its competitors, NNWID-Descent achieves consistent improvements for graph accuracy and resistance to performance degradation as sparsification increases — for ISOLET, it is outperformed only by Random. For the MNIST dataset, Sparse PCA has



(a) Google-23 (b) HAR

(c) ISOLET (d) MNIST

(e) ALOI (f) RLCT

Fig. 1. Performance of NNWID-Descent with varying values of Z, and $K = 100$.

a performance comparable to that of NNWID-Descent for small sparsification rates.

It is important to realize that obtaining accurate estimates of wID requires that the neighborhood be of generally good quality. In NNWID-Descent, the recomputation of neighborhoods after sparsification at each iteration is essential to the quality of wID estimation. However, using distance values computed from the current k-NN graph may lead to less accurate ID estimation, if the initial graph is of low quality.

Execution Time. The cost of sparsification and refinement dominates the overall computational performance of the three methods that employ this strategy. For these methods, the execution time for the sparsification and refinement phase is displayed in Table 1. The displayed times account for the operations of feature ranking, sparsification, and updating of neighbor lists. The table shows

Table 1. Average time in seconds per iteration.

	NNF-Descent	Random	NNWID-Descent
Google-23	320.96	70.77	1431.56
ISOLET	204.92	73.34	1152.43
HAR	248.75	141.46	1275.44
MNIST	5274.55	4429.77	8281.03
ALOI	13053.55	11183.65	55363.56
RLCT	3125.64	2853.78	9549.33

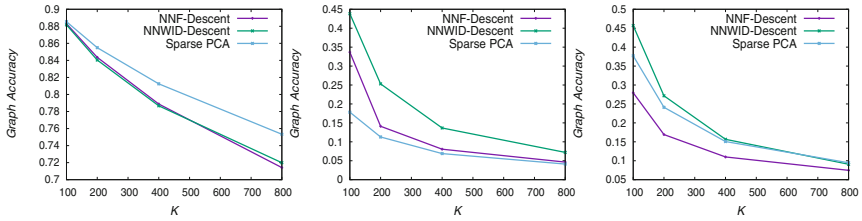
the average running time in seconds per iteration for all datasets under consideration.

Since the time for sparsification and neighbor list updating is expected to be the same for all three methods, the observed differences in execution time related to differences in the costs of the feature ranking step. As can be observed from Table 1, NNWID-Descent has the highest execution cost. This is due to the necessity of computing neighborhood distances for each object per feature in each iteration. Despite its larger running time relative to its competitors, NNWID-Descent shows a better potential for the improvement of graph accuracy, and better resistance to performance degradation as sparsification increases.

5.6 Effects of Varying the Neighbor List Size K

Parameter Setting. In this experiment, we compare the performance of NNWID-Descent against NNF-Descent and Sparse PCA as the neighbor list size increases beyond $K = 100$. We show the results only for the largest datasets (ALOI, MNIST and RLCT), as the values of K are too large relative to the size of the other datasets. Concretely, K is set to 100, 200, 400, and 800, and Z is fixed at 4% for MNIST and RLCT, and at 2% for ALOI. These Z values represent approximately the peak graph accuracy achieved in Fig. 1. The performances across these choices of K are plotted in Fig. 2.

Results and Analysis. We note that for ALOI and RLCT, NNWID-Descent still provides better accuracy than other methods as the neighborhood list size K is increased. With MNIST, Sparse PCA outperforms other methods as K increases, which indicates that this method can lead to a reasonable graph accuracy for a sparse dataset when Z is small. For all methods, the performance degrades as K increases. In addition, we observe that the relative performances of all methods shown when varying K (Fig. 2) is still consistent with the performances observed when varying Z (Fig. 1).



(a) MNIST (b) ALOI (c) RLCT

Fig. 2. Performance of NNWID-Descent with different values of K and fixed Z (Z = 4% for RLCT, MNIST, and 2% for ALOI).

6 Conclusion and Future Work

In this paper, we presented the NNWID-Descent similarity graph construction method, which utilizes the NNF-Descent framework with a new unsupervised feature selection criterion. This method aimed to improve or maintain k-NN graph accuracy while achieving a significant amount of sparsification of object feature vectors. We proposed the use of support-weighted ID (wID) to identify relevant features with higher discriminative power local to each object. NNWID-Descent ranks the features according to their wID values, then sparsifies those features achieving the smallest values.

With respect to the correctness of k-NN graph produced using six real datasets, NNWID-Descent has been shown to generally outperform its closest competitors, NNF-Descent and Sparse PCA. NNWID-Descent can be applied to obtain more compact representations for high-dimensional features vectors, which is important to reduce the storage and computational complexity for many applications. However, the ID estimator used in NNWID-Descent generally requires relatively large dataset sizes to provide a reasonable accuracy. Of the six datasets used in our experiments, three are considered too small for the extreme-value-theoretic LID model to be applicable. Further improvement of NNWID-Descent could be achieved through the development of ID estimators that can more accurately handle smaller dataset sizes and smaller neighborhood sample sizes.

Acknowledgments. M.E. Houle acknowledges the financial support of JSPS Kakenhi Kiban (A) Research Grant 25240036 and JSPS Kakenhi Kiban (B) Research Grant 15H02753. V. Oria acknowledges the financial support of NSF Research Grant DGE 1565478.

References

1. Amsaleg, L., Chelly, O., Furon, T., Girard, S., Houle, M.E., Kawarabayashi, K., Nett, M.: Estimating local intrinsic dimensionality. In: KDD, pp. 29–38 (2015)
2. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: A public domain dataset for human activity recognition using smartphones. In: ESANN (2013)

3. Brito, M., Chávez, E., Quiroz, A., Yukich, J.: Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. Stat. Probab. Lett. **35**(1), 33–42 (1997)
4. Dong, W., Moses, C., Li, K.: Efficient K-nearest neighbor graph construction for generic similarity measures. In: WWW, pp. 577–586 (2011)
5. Dy, J.G., Brodley, C.E.: Feature selection for unsupervised learning. J. Mach. Learn. Res. **5**, 845–889 (2004)
6. Geusebroek, J.M., Burghouts, G.J., Smeulders, A.W.M.: The Amsterdam library of object images. Int. J. Comput. Vis. **61**(1), 103–112 (2005)
7. Han, E.-H.S., Karypis, G., Kumar, V.: Text categorization using weight adjusted k-nearest neighbor classification. In: Cheung, D., Williams, G.J., Li, Q. (eds.) PAKDD 2001. LNCS (LNAI), vol. 2035, pp. 53–65. Springer, Heidelberg (2001). doi:10.1007/3-540-45357-1_9
8. Hautamaki, V., Karkkainen, I., Franti, P.: Outlier detection using k-nearest neighbour graph. In: ICPR, vol. 3, pp. 430–433, August 2004
9. He, J., Li, M., Zhang, H.J., Tong, H., Zhang, C.: Manifold-ranking based image retrieval. In: ACM MM, pp. 9–16 (2004)
10. He, X., Cai, D., Niyogi, P.: Laplacian score for feature selection. In: NIPS, vol. 186, p. 189 (2005)
11. Hill, B.M.: A simple general approach to inference about the tail of a distribution. Ann. Stat. **3**(5), 1163–1174 (1975)
12. Houle, M.E.: Dimensionality, discriminability, density & distance distributions. In: ICDMW, pp. 468–473 (2013)
13. Houle, M.E.: Local intrinsic dimensionality I: an extreme-value-theoretic foundation for similarity applications. In: SISAP, pp. 1–16 (2017)
14. Houle, M.E.: Local intrinsic dimensionality II: multivariate analysis and distributional support. In: SISAP, pp. 1–16 (2017)
15. Houle, M.E., Ma, X., Oria, V., Sun, J.: Improving the quality of K-NN graphs through vector sparsification: application to image databases. Int. J. Multimedia Inf. Retrieval **3**(4), 259–274 (2014)
16. Houle, M.E., Oria, V., Satoh, S., Sun, J.: Knowledge propagation in large image databases using neighborhood information. In: ACM MM, pp. 1033–1036 (2011)
17. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
18. Li, Y., Dong, M., Hua, J.: Localized feature selection for clustering. Pattern Recogn. Lett. **29**(1), 10–18 (2008)
19. Lichman, M.: UCI Machine Learning Repository (2013). http://archive.ics.uci.edu/ml
20. Mitra, P., Murthy, C., Pal, S.K.: Unsupervised feature selection using feature similarity. IEEE TPAMI **24**(3), 301–312 (2002)
21. Qin, D., Gammeter, S., Bossard, L., Quack, T., van Gool, L.: Hello neighbor: accurate object retrieval with k-reciprocal nearest neighbors. In: CVPR 2011, pp. 777–784, June 2011
22. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Application of dimensionality reduction in recommender systems – a case study. Technical report, DTIC Document (2000)
23. Wang, Z., Liu, Z.: Graph-based KNN text classification. In: FSKD, vol. 5, pp. 2363–2366, August 2010
24. Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. J. Mach. Learn. Res. **5**, 1205–1224 (2004)
25. Zou, H., Hastie, T., Tibshirani, R.: Sparse principal component analysis. J. Comput. Graph. Stat. **15**(2), 265–286 (2006)