Int J Multimed Info Retr (2014) 3:259–274
DOI 10.1007/s13735-014-0067-7

REGULAR PAPER

Improving the quality of K-NN graphs through vector sparsification: application to image databases

Michael E. Houle · Xiguo Ma · Vincent Oria · Jichao Sun

Received: 18 July 2014 / Revised: 26 August 2014 / Accepted: 29 August 2014 / Published online: 21 September 2014

Abstract K-nearest neighbor (K-NN) graphs are an essential component of many established methods for content-based image retrieval and automated image annotation. The performance of such methods relies heavily on the semantic quality of the graphs, which can be measured as the proportion of neighbors sharing the same class label as their query images. Due to the noise in image features, the K-NN graphs produced by existing methods may suffer from low semantic quality. In this article, we propose *NNF-Descent* for the efficient construction of K-NN graphs based on nearest-neighbor and feature descent, in which selective sparsification of feature vectors is interleaved with neighborhood refinement operations in an effort to improve the semantic quality of the result. A variant of the Laplacian Score is proposed for the identification of noisy features local to individual images, whose values are then set to 0 (the global mean value after standardization). We show through extensive experiments on several datasets that *NNF-Descent* is able to increase the proportion of semantically-related images over unrelated images within the neighbor sets, and that the proposed method generalizes well for other types of data which are represented by high-dimensional feature vectors.

M. E. Houle
National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
e-mail: meh@nii.ac.jp

X. Ma · V. Oria · J. Sun (✉)
New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA
e-mail: js87@njit.edu

X. Ma
e-mail: xm23@njit.edu

V. Oria
e-mail: oria@njit.edu

Keywords K-nearest neighbor graph · Semantic quality · Image database · Feature selection · Locally noisy feature · Vector sparsification · Iterative method

1 Introduction

The construction of K-nearest neighbor (K-NN) graphs has been widely adopted as an essential operation for many applications, such as object retrieval [21], data clustering [4], manifold learning [2,25], and other machine learning tasks [33].

In the research field of multimedia where images are represented by high-dimensional feature vectors, K-NN graphs built for fixed image sets serve as important data structures for a number of established methods. For example, Qin et al. [21] proposed a method for improving the accuracy of image retrieval wherein different ranking functions are applied to disjoint subsets of the database, the 'close set' and the 'far set', as defined relative to the query image. A K-NN graph is pre-computed to efficiently identify the reciprocal nearest neighbors of the query image, which constitute the initial close set. The close set is then expanded to include more images according to certain selection rules.

Manifold ranking, which has received much attention in the context of content-based image retrieval (*CBIR*), often uses K-NN graphs to represent the similarity relationships between images. Initially, a positive score is assigned to the query image, and a score of 0 is assigned to all other images. Each image iteratively computes its score as a weighted combination of its initial score and the scores of its neighbors. At termination, those images with larger scores are considered to be more related to the query. Examples following this protocol include [13] and [29].

Practical search engines for general images often require that the images be annotated beforehand. Due to the inherent

difficulty of preparing large volumes of images for search through manual annotation, automated image annotation (*AIA*) techniques have been extensively researched in recent years. One important approach to *AIA* is image label propagation, in which confidence scores are disseminated from initially labeled images to unlabeled images via a similarity graph, in which the nodes represent individual images, and the edges join pairs of images that meet certain similarity criteria. For each initially-unlabeled node in the graph, scores are computed individually for each label-node combination; at termination, the label with the highest score is assigned to the node. In [15], a keyword propagation method was developed using a modified K-NN graph in a graph-based semi-supervised learning framework. This work has been extended in [16] in a way that the graph edges are classified into 'strong' and 'weak' edges, with the strong edges having a higher weighting in the propagation process. In [28], the authors proposed a sparse graph reconstruction method to reduce links between semantically unrelated images in traditional K-NN graphs. The label inference step is formulated by minimizing a label reconstruction error function.

One major difficulty with the use of K-NN graphs for image databases is the large computational cost of construction. Due to the quadratic time complexity of brute-force methods, much effort has been devoted to the development of faster approximate K-NN graph construction techniques. One straightforward solution is to invoke approximate K-NN search for every graph node, using such indexing techniques as cover trees [3] or locality sensitive hashing [11]. Another approach involves the batch construction of K-NN graphs. Chen et al. proposed one such method based on recursive data partitioning in L_2 space [5]. In [7], *NN-Descent* was developed for iterative K-NN graph construction in generic metric space based on a simple transitivity principle: a neighbor of a neighbor is also likely to be a neighbor. A description of *NN-Descent* will be given in Sect. 4.1.

Another difficulty with the use of K-NN graphs for image databases lies in its semantic quality, which can be measured as the proportion of edges connecting two nodes with identical labels. The semantic quality of K-NN graphs depends crucially on the feature vectors describing the images. If many features are noisy or irrelevant for the class associated with the query image, the images in its neighborhood list may not be semantically related to the query, severely limiting the effectiveness of K-NN graph-based approaches. For example, for the case where the query image belongs to the database in question, a smaller number of correct neighbors in its K-NN list directly indicates a lower query result accuracy. In image label propagation, each graph edge connecting two unrelated image nodes is a source of error, in that it suggests that these two images should share the same label despite their belonging to different classes.

The negative impact of noisy or irrelevant features has motivated the use of feature selection techniques for improving the semantic quality of *CBIR* [8,12,18]. For image datasets, such feature selection techniques would also be relevant to our problem, since K-NN graph construction can be viewed as a batch of in-dataset content-based query operations. Traditional feature selection methods have been successfully applied in the reduction of noisy features in many contexts. However, as a rule, such reduction is performed over the entire dataset: any feature deemed to be noisy is discarded for each data point. This neglects the possibility that the importance of the feature may vary across different data points or classes of data points.

In this article, we present *NNF-Descent*, a new method for the efficient construction of K-NN graphs with improved semantic quality, for scenarios involving image databases where class label information is not available.

First, we propose the Local Laplacian Score (*LLS*), a variant of the Laplacian Score (*LS*) [14], to identify features that are 'locally noisy'—that is, noisy relative to the neighborhood of a given target image. We show that if a feature is indiscriminative for an image class, it is very likely that the feature will be identified as locally noisy for many images from this class.

Since we are interested in identifying features that are noisy only with respect to subsets of images (that is, neighborhoods of query images), and not with respect to the full image dataset, traditional feature selection techniques cannot be applied directly. To reduce the negative impact of locally noisy features, we modify the feature value so as to encourage the reduction of intra-class distances. Ideally, one suitable value for such replacement could be the mean for that feature, taken over all images from the class to which the image belongs. However, this is not feasible in practice, as the class labels of the images are not known in advance. As a heuristic solution, we instead change noisy feature values to the global mean for that feature. Assuming that the feature values have been standardized, as is common practice, this amounts to a replacement of noisy feature values by 0. This operation, referred to here as feature sparsification, is then embedded into the above-mentioned K-NN graph construction framework, *NN-Descent*. During the iterative feature sparsification process, as more and more images from a common class have had their locally noisy features identified and sparsified, the image vectors from this class gradually converge to a new class center in the image domain.

The initial version of this study previously appeared as [17]. We extend our work in the following significant aspects:

– The research literature of feature selection techniques for images, focusing on their applications to *CBIR*, is discussed.

– More details of the proposed method and its variants are provided.
– Experimental results on two additional (non-image) datasets are given, for the evaluation of our method on other types of data.
– In the experimental comparison of performance, an additional recent unsupervised feature selection method is included.

The remainder of this article is organized as follows. The related research literature is reviewed in Sect. 2. Section 3 formally introduces the Local Laplacian Score and explains the rationale for feature sparsification. Section 4 describes *NN-Descent* and our proposed K-NN graph construction method, *NNF-Descent*. In Sect. 5, we present and discuss the results of experiments in which our method is compared on six datasets against unsupervised feature extraction and selection methods, with respect to the semantic quality of the K-NN graphs produced. We conclude this paper in Sect. 6 with a discussion of future research directions.

2 Related work

In this section, we review the research literature of feature selection techniques, focusing on their applications in the field of *CBIR*, which is closely related to our research.

The existence of noise features has a negative impact on the discrimination of data from different classes. This has motivated the use of feature extraction and selection techniques on images represented by high-dimensional feature vectors. Feature extraction projects the original features into a new lower-dimensional vector space. Popular extraction techniques include principle component analysis *(PCA)* and linear discriminant analysis *(LDA)*, to name two. Feature selection, on the other hand, selects a subset of features from the original vector space, and is superior to feature extraction in terms of interpretability as the original feature values are maintained in a reduced feature space.

Feature selection methods can be broadly categorized as wrapper-based [19] or filter-based [24]. The wrapper model finds subsets of features using heuristic search strategies, and evaluates the quality of each reduced feature set using a target learning algorithm. The filter model instead evaluates feature relevance in terms of the intrinsic characteristics of the data. In the context of K-NN graph construction, filter-based approaches are more desirable, since no specific learning algorithm is required, and since the computational costs are typically much lower than with wrapper-based approaches.

Feature selection techniques have been widely used in image retrieval for better semantic quality of the query result. Most of them are supervised. For example, in [30], a family of feature selection methods was designed based on the maximization of the mutual information between features and class labels. The selection of discriminative features and the reduction of redundant features are performed jointly for image retrieval and recognition. Guldogan and Gabbouj [12] integrated three feature selection criteria involving mutual information, intra-cluster relationships, and inter-cluster relationships. For the determination of the final ranking of features, majority voting is applied across the feature rankings computed according to each individual criteria.

Rashedi et al. [23] combined image feature adaptation and selection in a simultaneous process. The authors claimed that each image database should have its own parameters for the extraction of features, controlling such aspects of the process as (for example) the quantization levels in color histograms. The values of these parameters were encoded together with a binary vector corresponding to the selected features. A mixed gravitational search algorithm [22] was used for optimizing the parameter values.

Jiang et al. proposed a relevance feedback learning method for online image feature selection [18]. Given a query image, the returned results are labeled as 'relevant' or 'irrelevant' by the user. The most representative features for the query concept are then selected based on a form of similarity between the two labeled sets. A similar method was presented in [27], with feature selection being guided by a combination of a Bayesian classifier with a measure of inconsistency from relevance feedback. The mean feature vectors of the positive and negative labeled samples are constructed online in each feedback session, and the angle between the two vectors is computed as a measure of the inconsistency from relevance feedback.

The methods listed above require ground truth input for training images—either as a semantic labeling, or from relevance feedback. Dy et al. [8] proposed a wrapper-based unsupervised feature selection method for medical image retrieval. Sequential forward selection [9] is applied to produce candidate feature subsets, which are then used in expectation-maximization (*EM*) clustering. The quality of a feature set is then evaluated according to a measure of compactness and separability on the resulting clusters. However, the requirement of a target learning algorithm, as well as the huge computational costs involved, hinder the application of such wrapper-based methods to databases with high-dimensional feature vectors.

Laplacian Score (*LS*) [14] and spectral feature selection *(SPEC)* [32] were proposed as two powerful unsupervised filter-based methods for generic data. The basic idea of *LS* is to rank features according to their locality-preserving abilities. *LS* favors features whose values are similar within each local neighborhood, but which have a large variance over the entire dataset. *SPEC* presents a unified framework based on spectral graph theory for both supervised and unsupervised feature selection, in which features are evaluated according

to their consistency with the structure of a weighted similarity graph. Three ranking functions ($\phi 1$, $\phi 2$, and $\phi 3$) were proposed; it is claimed that *LS* is equivalent to *SPEC*-$\phi 2$ under certain conditions. Both methods achieve good generalization performance on learning tasks with reduced dimensionality. However, we cannot assume that the same also applies to the case of the semantic quality of image retrieval, especially when comparing them with the use of full feature vectors.

More recently, Yang et al. [31] proposed the unsupervised discriminative feature selection *(UDFS)* algorithm by incorporating discriminative analysis and $L_{2,1}$-norm minimization into a joint framework. Important features can be selected in batch mode, based on the optimization of an objective function. However, this algorithm requires the number of classes as an input, and suffers from large time complexity for data represented in high-dimensional feature spaces. Furthermore, the authors only evaluated *UDFS* in clustering scenarios—the performance of their method on semantic retrieval remains unknown.

Traditional feature selection computes a uniform set of features for all images in the database. However, it is possible that the discriminative power of a feature varies for different images. In our previous work [16], we proposed a feature selection method for linking labeled images to unlabeled images in a similarity graph for label propagation. Each feature of a labeled image is used in isolation to rank other labeled images; the features that assign high ranks to related neighboring images are treated as more important. By deleting the least important features, a different feature set is computed for each labeled image, for subsequent use in the ranking of unlabeled images. In this article, we also propose a scheme in which noisy features are identified locally for individual database images; however, unlike the method of [16], the entire process is fully unsupervised.

3 Locally noisy feature detection and sparsification

In this section we propose a local variant of the Laplacian Score (*LS*), the Local Laplacian Score (*LLS*), for the ranking of features with respect to individual data points. We then show the use of *LLS* in the identification of locally noisy features and the characterization of the features identified. The section concludes with a discussion of the effectiveness of sparsification of locally noisy features for the reduction of intra-class distances.

3.1 Local Laplacian Score

Given a dataset D consisting of n data points represented by m-dimensional feature vectors, we denote the r-th feature of the entire dataset by an n-dimensional vector $\mathbf{f}_r = (f_{r1}, \ldots, f_{rn})^T$, where $r = 1, \ldots, m$, and f_{ri} $(i = 1, \ldots, n)$ is the feature value of $\mathbf{f}_r$ taken from data point $x_i \in D$ (more generally, let f_r denote the r-th feature from an individual data point). For the sake of convenience, we will not distinguish between the r-th feature and its value(s), and simply refer to the feature in question as f_r (or $\mathbf{f}_r$).

Given a nearest neighbor graph G (for example, the K-NN graph) of dataset D, the Laplacian Score of the r-th feature over the entire dataset can be computed as follows [14]:

$$LS(r) = \frac{\sum_{ij}(f_{ri} - f_{rj})^2 S_{ij}}{\mathrm{var}(\mathbf{f}_r)}, \tag{1}$$

where $\mathrm{var}(\mathbf{f}_r)$ is the estimated variance of the values of feature $\mathbf{f}_r$, and S_{ij} is the (Gaussian) RBF kernel on feature vectors x_i and x_j representing the i-th and j-th data points, respectively:

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

where σ is a bandwidth parameter. *LS* favors those features that both preserve the nearest neighbor graph structure and have large variance values across all data points. Note that the similarity S_{ij} places a high weighting on node i's close neighbors, which are more likely to be from the same class as i.

LS evaluates the importance of a feature as regards its overall power in locality preservation, taken over all objects of a dataset D. Only one ranking score for each feature $\mathbf{f}_r$ is computed. When it is used as the criterion for traditional feature selection, $\mathbf{f}_r$ is either preserved for, or discarded from, the entirety of the dataset. This, however, neglects the possibility that a feature that is important for one data class (or one data point) may be irrelevant for another class (or point).

In this article, we propose the Local Laplacian Score (*LLS*) for the identification of noisy features relative to each data point. *LLS* represents the contribution to Eq. 1 that can be attributed to data point x_i:

$$LLS_i(r) = \frac{\sum_j (f_{ri} - f_{rj})^2 S_{ij}}{\mathrm{var}(\mathbf{f}_r)}. \tag{2}$$

As $\mathbf{f}_r = (f_{r1}, \ldots, f_{rn})^T$, it is easy to verify that

$$LS(r) = \sum_i LLS_i(r). \tag{3}$$

As with *LS*, a smaller *LLS* value indicates less variation in the feature value among the neighbors of the data point. Intuitively, by minimizing $LLS_i(r)$, *LLS* favors those features that have a high global variation and that have the greatest impact in establishing the neighborhood of data point i.

3.2 Locally noisy features and *LLS*

We adopt a straightforward method for the detection of noisy features local to node i using *LLS*, in which the m features

are sorted in descending order of $LLS_i(r)$, and returns the first z features (for some supplied value $z > 0$). We refer to these z features as the *locally noisy features* of x_i, and to the remaining $(m - z)$ features as the *subjective features* of x_i.

Let us assume that all feature values have been standardized in advance. If the original values of feature $\mathbf{f}_r$ are denoted by $\mathbf{f}'_r$, the standardized value of the r-th feature for data point x_i is:

$$f_{ri} = \begin{cases} (f'_{ri} - \mu_{\mathbf{f}'_r})/\sigma_{\mathbf{f}'_r} & \text{if } \sigma_{\mathbf{f}'_r} \neq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where

$$\mu_{\mathbf{f}'_r} = \frac{\sum_i f'_{ri}}{n}, \quad \text{and} \quad \sigma_{\mathbf{f}'_r} = \sqrt{\frac{\sum_i (f'_{ri} - \mu_{\mathbf{f}'_r})^2}{n}}$$

are the mean and standard deviation of the original feature values $\mathbf{f}'_r$, respectively. As a consequence, each standardized feature $\mathbf{f}_r$ has a mean of 0 and a variance of 1.

Standardization is possible provided that $\sigma_{\mathbf{f}'_r} \neq 0$. Note that if $\sigma_{\mathbf{f}'_r}$ were equal to 0, all the feature values for $\mathbf{f}'_r$ would be identical, and thus $\mathbf{f}'_r$ would have no impact in the computation of distances between data points, and could safely be eliminated altogether. We therefore consider only those cases in which $\sigma_{\mathbf{f}'_r} \neq 0$ for every original feature $\mathbf{f}'_r$.

Given that the values of feature $\mathbf{f}_r$ have been standardized in advance, *LLS* reduces to the following simpler form:

$$LLS_i(r) = \sum_j (f_{ri} - f_{rj})^2 S_{ij}. \tag{5}$$

Equation 5 can be viewed as a form of weighted local variance of the feature values for $\mathbf{f}_r$ in the neighborhood of node i. As with the computation of *LS* (Eq. 1), the close neighbors of node i are given higher weighting in the computation of $LLS_i(r)$, since they are more likely to belong to the same class as i.

Denoting the class label of i by I, when standardized feature $\mathbf{f}_r$ is discriminative for class I, the variance of the values for $\mathbf{f}_r$ within I is likely to be relatively small. As a result, the *LLS* scores for the r-th feature are expected to be small for most data points from I. If feature f_{ri} nevertheless had a relatively high score $LLS_i(r)$ for node i, then f_{ri} is likely to be an outlier among all the feature values for $\mathbf{f}_r$ within class I.

On the other hand, when feature $\mathbf{f}_r$ is a noisy feature for class I, the variance of the standardized feature values for $\mathbf{f}_r$ is large in I. Thus, many data points from I are very likely to have large *LLS* scores for f_r, and to identify f_r as one of their own noisy features. In other words, if feature $\mathbf{f}_r$ is indeed noisy for a given class, many data points from this class would tend to agree on its identification as such. A consensus, however, does not in general occur among data points drawn from different classes.

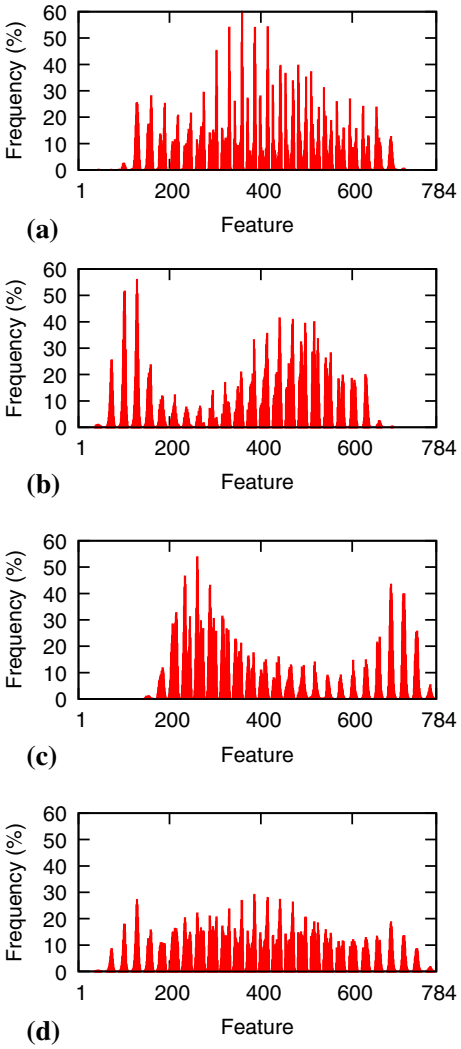

Fig. 1 Frequencies of features identified as noisy features in three image classes of MNIST. **a** Images of the digit 0, **b** images of the digit 6, **c** images of the digit 7, **d** images of digits 0, 6 and 7

This situation is illustrated in Fig. 1 for the MNIST handwritten digit image set [20] (see Sect. 5.1 for a description of this set). For three classes of handwritten digits, *LLS* is used to identify the top 50 noisy features from a total of 784 features. Figure 1a–c show the frequency by which each feature is identified as a noisy feature for the digit classes 0, 6 and 7, respectively. Figure 1d shows the frequency by which each feature is identified as a common noisy feature for all the three classes. It can be seen that even with less than 7 % of the features from each image deemed as noise, many features are selected as such for 40–60 % of the images within each class. However, the noisy feature sets receiving the most votes in the three image classes are very different. For all the three classes, the frequencies of noisy features are more balanced, with no feature receiving more than 30 % of the votes.

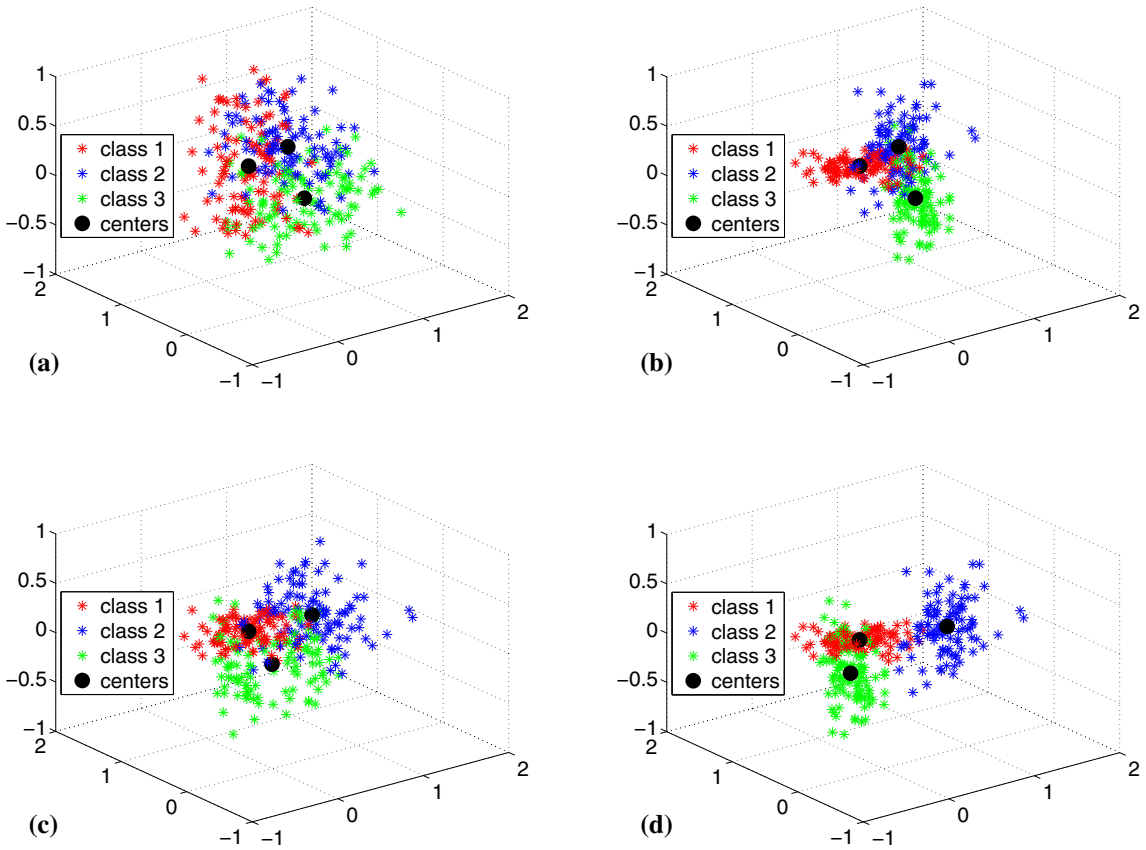

Fig. 2 Distribution of 3-D data points in a dataset of three classes. **a** Original data points, **b** noisy feature values changed to local mean, **c** 50 % noisy features sparsified, **d** 100 % noisy features sparsified

3.3 Feature sparsification

Traditional feature selection methods cannot be applied directly in the reduction of noisy features identified by *LLS*, as the feature importance is different across individual data points—each data point has its own subjective feature set. Instead of discarding a feature from the entire dataset, we modify the noisy feature values for individual data points in an effort to reduce intra-class distances.

Given a subset $D' \subset D$, we denote the mean value of the r-th feature for the data points in D' by:

$$\text{mean}(D', f_r) = \frac{\sum_{x_i \in D'} f_{ri}}{|D'|}. \tag{6}$$

More specifically, we denote the *global mean* of the r-th feature computed over the entire dataset by $\text{mean}(D, f_r)$, the *class mean* of the r-th feature computed in class P by $\text{mean}(P, f_r)$, and the *local mean* of the r-th feature with respect to node $p \in P$ by $\text{mean}(Q, f_r)$, where Q is the K-NN set of p.

Let us consider a simplified example wherein the data points of class P have a common noisy feature f_r. Ideally, if for all data point $p \in P$ we replace f_{rp} with $\text{mean}(P, f_r)$, the intra-class distances of P would tend to decrease (as the intra-class variance attributed to this feature dimension is eliminated), whereas the class mean value $\text{mean}(P, f_r)$ would not change. As a consequence, the data points of class P become closer, and the distances between P and other classes measured as the distances between the class centers remain the same.

Figure 2a, b illustrate a configuration of three classes of synthetic 3-D data points before and after such replacement. Figure 2a depicts the original distributions of the three classes of 3-D points and their class centers. The data points of each class share a common noisy feature, the noisy feature being different for each of the three classes. It can be seen from Fig. 2b that after replacing locally noisy feature values by their class mean values, the points of each class converge towards their class centers, while the class centers remain the same. Outlying feature values are effectively corrected, and discrimination of the classes is clearly improved.

Unfortunately, replacement of locally noisy feature values by class mean values is impractical, as the class labels are generally unavailable. As a heuristic solution, we instead perform a sparsification of the data vectors, by replacing

the value of each noisy feature f_r with the global mean mean(D, f_r) for standardized features, which is 0. Figure 2c, d show the configurations of the three classes after 50 and 100 % (respectively) of the data points in each class have been sparsified. During the sparsification process, the centers of the classes can change. In this example, the distances between the centers of classes 1 and 2, and classes 2 and 3, both increase; between the centers of classes 1 and 3, we observe a decrease. However, the data points in each class still converge towards their new centers as the sparsification rate increases. In fact, in Fig. 2d, the sparsified data points are reduced to 2-D points which converge towards their new class centers in three different 2-D planes.

Although the global mean of each feature is 0 due to standardization, individual feature values could be positive or negative, and thus in general, if two data points have different features sparsified, the distance between them could increase or decrease. Ideally, data objects from a common class should identify the same sets of noisy features. Two data points from different classes could conceivably share many sparsified features, resulting in an undesirable reduction of the distance between them. However, one would expect this to be more than offset by the sparsification of common noisy features across many members of the same class, since the *LLS* ranking favors such features.

For data points with outlying feature values in a class (the features in question are otherwise discriminative for the class), the sparsification of the outlying features does not guarantee a reduction of the distances between the data points and the other members of their class. The reason is that the features in question are less likely to be identified as noisy features by the other data members and thus remain unchanged. However, even if the distances did increase, one would expect the number of such outliers (data points with outlying feature values) to be relatively small, and thus the overall negative impact would likely be outweighed by the positive impact on class cohesion by the sparsification of common noisy features for the class.

With more locally noisy features detected, data points from different classes are more likely to share common noisy features. Thus, unlike traditional feature selection methods, the feature sparsification scheme should be employed conservatively, by modifying only a relatively small proportion of features.

Another heuristic solution is to replace each noisy feature value f_{rp} by an approximation of the class mean mean(P, f_r). Here, we use the local mean mean(Q, f) as the approximation, where Q is the K-NN set of p taken with respect to the full feature set. The K-NN set of each data point could be precomputed in the process of the initial graph construction for *LLS* feature ranking. However, this strategy suffers from several drawbacks:

- Averaging featue values incurs cost overheads that can significantly reduce the efficiency when the dataset is large.
- It is difficult to determine whether the original or the updated feature values should be used in subsequent averaging processes.
- The local mean of a feature is not fixed for a data class, so that data points from the same class may have their common noisy features changed to different values.

In our experimentation, we compare this variant with the feature sparsification scheme, and discuss the result in Sect. 5.3.

4 K-NN graph construction with feature sparsification

In this section, we give the details of our proposed adaptation of *NN-Descent* for the construction of a K-NN graph for images described as high-dimensional vectors. As will be seen, this method generalizes well for non-image data having similar representations. A brief description of *NN-Descent*, and the complete algorithm of our method *NNF-Descent*, are given in Sects. 4.1 and 4.2, respectively.

4.1 NN-Descent

NN-Descent is an iterative algorithm for the construction of approximate K-NN graphs with arbitrary similarity measures [7]. Let p, q and r denote three data points. *NN-Descent* seeks to take advantage of a tendency toward transitivity in the neighbor relationship: if q is a neighbor of p, and r is a neighbor of q, then r is likely to be a neighbor of p (Fig. 3). Starting from a random tentative K-NN graph, the *NN-Descent* strategy is to repeatedly check for each point p as to whether any neighbors of its neighbors (such as r) could serve as a closer neighbor of p than any of the nodes currently in the neighbor list of p.

If the neighborhood relationship is undirected, checking data pairs of the form (p, r) is equivalent to checking all pairs of neighbors of a common point q. This operation is referred to as a *local join*. The *NN-Descent* strategy can thus be described as that of checking whether two neighbors of a

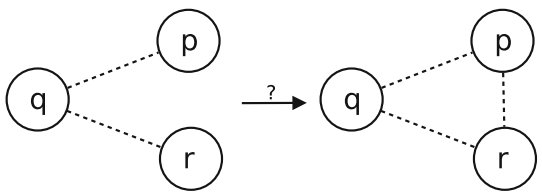

Fig. 3 The principle of *NN-Descent*

Algorithm 1: *NN-Descent*

input : dataset D, distance function $dist$, neighborhood size K
output: K-NN graph G
1 **foreach** *data point* $p \in D$ **do**
2 Initialize G by randomly generating a tentative K-NN list for p with an assigned distance of $+\infty$;
3 **end**
4 **repeat**
5 **foreach** *data point* $p \in D$ **do**
6 Check different pairs of p's neighbors (x, y) in p's K-NN and R-NN (reverse nearest neighbor) lists, and compute $dist(x, y)$;
7 Use $\langle x, dist(x, y)\rangle$ to update y's K-NN list, and use $\langle y, dist(x, y)\rangle$ to update x's K-NN list;
8 **end**
9 **until** *G converges;*
10 Return G.

common data point could improve over any of the tentative neighbors in each other's neighbor list.

The basic algorithm of *NN-Descent* is summarized in Algorithm 1. The algorithm starts with an initial random K-NN graph that will be iteratively refined in an effort to produce the true K-NN graph (lines 1–3). Lines 6–7 correspond to the local join operation. In the implementation, a K-NN list of a query point consists of K entries, each of which is an ordered pair $\langle x, d\rangle$ with x being a data point ID, and d being the distance between x and the query point. In line 7, the K-NN entry $\langle x, \text{dist}(x, y)\rangle$ is used to update y's K-NN list if and only if $\text{dist}(x, y) < \text{dist}(q_K, y)$, where q_K is the K-th neighbor of y. The same rules apply to the case for $\langle y, \text{dist}(x, y)\rangle$. The algorithm stops when the graph G is not changed in consecutive iterations, or the proportion of updated K-NN entries is smaller than a user-specified threshold.

4.2 NN-Descent with sparsification

We now show how *LLS* feature ranking and sparsification can be integrated into the *NN-Descent* framework. Starting from a near-exact K-NN graph, noisy features are gradually sparsified as the nearest-neighbor descent progresses. After each sparsification, the feature vector is updated for use in subsequent refinements of neighborhoods. This allows the effects of sparsification and graph refinement to influence each other promptly: an updated K-NN graph improves the feature ranking accuracy, and the sparsification of noisy features improves the semantic quality of the K-NN graph in return.

The details of the *NNF-Descent* method can be found in Algorithm 2. For simplicity, we sparsify a fixed number of features from each feature vector per iteration, which is controlled by the parameter z. As mentioned before, the value of z should be relatively small in comparison with the total number of features. The other two parameters, K and N, determine the neighborhood size of the target graph, and the desired number of iterations, respectively.

Lines 1–2 of Algorithm 2 are preprocessing steps, the latter of which uses the original *NN-Descent* to compute a K-NN graph for the original (standardized) feature vectors. We expect this graph to be of reasonable semantic quality; otherwise, the initial feature ranking may be too unreliable for the sparsification strategy to further improve the graph. Although chosen for reasons of efficiency, *NN-Descent* can be replaced with other exact or approximate K-NN graph construction methods if desired.

Lines 3–13 correspond to one iteration of our method, in which three main phases are involved: feature ranking, sparsification, and K-NN updates.

In line 6, the updated K-NN graph is used to rank the features for data point p. If desired, the feature ranking step may use a subset of the K-NN lists. For example, a 10-NN graph can be used for *LLS* feature ranking in the construction of a 100-NN graph.

Line 7 sparsifies a small number of highly-ranked (noisy) features for p. The value of parameter z is chosen empirically as described in Sect. 5. Since the values of the noisy features will eventually be changed to 0, we considered only those features having non-zero values. In particular, if the original data points have identical values for a given feature, the standardized values of this feature will be 0 for every point, as indicated by Eq. 4. In traditional feature selection, such features will be removed, as they provide no discriminative information. However, *LLS* sparsification simply ignores zero-valued features as they do not affect the semantic quality of the K-NN graph. Ignoring zero-valued features also ensures that a sparsified feature will not be sparsified again in further iterations.

Lines 8–11 correspond to the K-NN update phase. Lines 8 and 9 update the current K-NN graph to be consistent with the newly-sparsified feature vector. The distances between p and its current K-NN and R-NN neighbors are recomputed, and the lists of neighbors are re-sorted. Note that as a heuristic method, for the sake of efficiency, *NNF-Descent* does not recompute the K-NN lists of p or of p's R-NNs. However, the implementation of the local join operation requires that the order of the K-NN entries be correct. In the local join operations performed in lines 10–11, new candidate K-NN members are created and compared with the existing neighbors, after which the neighbor lists are updated. A data pair (x, y) that has been checked is subsequently flagged in order to prevent it from being checked again.

It is worth mentioning that we do not re-standardize the dataset after sparsification, for the reason that sparsification would introduce large computational overheads, and change the representation of the feature vectors dramatically. During the iterative process of feature sparsification, with respect

Algorithm 2: *NNF-Descent*

input : dataset D, distance function $dist$, neighborhood size K, number of sparsifications per iteration z, and number of iterations N
output: K-NN graph G

1 Standardize the original feature vectors of D;
2 Run *NN-Descent*$(D, dist, K)$ until convergence to obtain an initial K-NN graph G;
3 **repeat**
4 Generate a list L of all data points in random order;
5 **foreach** *data point* $p \in L$; **do**
6 Rank p's features in descending order of their *LLS* computed from p's current K-NN;
7 Change the values of the top z ranked features to 0;
8 Recompute the distances from p to its K-NN and R-NN;
9 Re-sort p's K-NN list and p's R-NN's K-NN lists;
10 Check different pairs of p's neighbors (x, y) in its K-NN and R-NN, and compute $dist(x, y)$;
11 Use $\langle x, dist(x, y)\rangle$ to update y's K-NN list, and use $\langle y, dist(x, y)\rangle$ to update x's K-NN list;
12 **end**
13 **until** *Max number of iterations N is reached;*
14 Return G.

to a given class, the class mean of an affected feature $\mathbf{f}_r$ eventually tends to 0 if most or all data points of this class have this feature sparsified; the variance of $\mathbf{f}_r$ tends to 0 as well. For simplicity, when computing the *LLS* for a feature $f_r \in \mathbf{f}_r$, we treat the global mean and variance of $\mathbf{f}_r$ as if they maintained their original (standardized) values throughout the sparsification process: with the mean fixed at 0, and the variance fixed at 1.

In the implementation, the length of an R-NN list is limited to K for efficiency. As a result, in terms of the number of distance computations, the time complexity of each *NNF-Descent* iteration is in $O(K^2 n)$, determined by the maximum cost of local join operations. If the dist function is L_2, the cost in terms of the number of operations on feature values is in $O(K^2 mn)$. The feature ranking and selection performed by *LLS* entails a small run-time overhead of $O(Kmn)$ for each iteration of *NNF-Descent*. This indicates that the algorithm scales well in terms of n, for reasonable values of K. Several optimizations of *NN-Descent* can be applied directly to *NNF-Descent* (Algorithm 2); we refer the reader to [7] for the full details.

4.3 Variants of NNF-Descent

In this section, we present several variants of *NNF-Descent* that will also be evaluated in the experimentation.

First, as an alternative to feature sparsification, another heuristic solution for adjusting the values of a locally noisy feature is to replace it with the approximate class mean (that is, the local mean) for that feature. More formally, we create a variant (*Var1*) from Algorithm 2 by modifying line 7 to:

For each feature f_{rp} appearing among the z top-ranked noisy features of p, set f_{rp} to $mean(Q, f_r)$, where Q is the current K-NN set of p.

Note that:

1. Unlike *NNF-Descent*, *Var1* does not skip the zero-valued features. However, a modified feature will not be modified again in subsequent iterations.
2. The computation of $mean(Q, f_r)$ uses the original standardized feature values of f_r instead of newly computed values.

In order to illustrate the effect of iterative feature ranking, we also contrast *NNF-Descent* against two variants (*Var2* and *Var3*) of Algorithm 2 with iterative feature ranking disabled. *Var2* maintains only the nearest-neighbor descent procedure of *NNF-Descent*, while *Var3* performs neither nearest-neighbor descent nor feature descent. Both *Var2* and *Var3* compute the *LLS* for features of each data point only once before the iteration begins, based on the initial K-NN graph. In each iteration, both variants sparsify z noisy features from each feature vector, with the features occupying ranks $iz - z + 1$ to iz being sparsified in the i-th iteration. The two variants differ in the K-NN update phase:

- *Var2* maintains the iterative K-NN updating step as in Algorithm 2 (lines 8–11), so that the K-NN graph is gradually changed.
- At the end of each iteration, after all data points have had z noisy features sparsified, *Var3* recomputes in its entirety an exact K-NN graph from the new feature vectors.

5 Experiments

We conducted experiments using six datasets, of which four were image sets. First, the influence of the rate of feature sparsification was investigated. *NNF-Descent* was then compared with the proposed variants so as to demonstrate the effectiveness of feature sparsification and iterative feature ranking. Finally, we compared our method with several unsupervised feature extraction and selection methods with respect to the semantic quality of the K-NN graphs produced, and for a labeling task.

5.1 Datasets

Table 1 summarizes the datasets used in our experimentation.

ALOI-100 and Google-23 are described in [15]. The former is a subset of the ALOI image dataset [10]. It contains 10,800 images of 100 simple objects, each image being described by a 641-D color and texture histogram. The latter

Table 1 Datasets used in the experiments

Datasets	Features	Instances	Subjects	Instances per subject
ALOI-100	641	10,800	100	108
MNIST	784	10,000	10	1,000
Google-23	1,937	6,686	23	97–406
ORL faces	10,304	400	40	10
Movement	90	360	15	24
Secom	590	1,567	2	1,463 and 104

dataset consists of 6,686 faces extracted from web images of 23 celebrities. The dimension of the face descriptors is 1,937.

The original MNIST dataset [20] contains 70,000 images of handwritten digits, each image being represented by 784 texture values. As in [16], we constructed a reduced set for our experiments by randomly selecting 1,000 images for each digit.

The ORL face dataset [26] (collected by AT&T Laboratories Cambridge) contains 400 images of 40 distinct subjects, each image consisting of 92 × 112 pixels. Each pixel is an 8-bit (0–255) gray scale integer, and is treated as one image feature.

The competing methods were also evaluated on two non-image datasets, Libras Movement and Secom, whose data objects are represented by high-dimensional feature vectors. Libras Movement (referred to as Movement for the remainder of this article) [6] contains 15 classes of 24 instances each, with each class referring to a hand movement type in Brazilian sign language. The 90-D feature vector for each instance is composed of normalized coordinates captured in 45 frames of a video clip of the hand gesture. Secom [1] consists of surveillance data from a semi-conductor manufacturing process. Each instance represents a single production entity with 590 measured features. This dataset has 1,463 positive instances and 104 negative instances.

We used the four image datasets for the testing of parameter z, and the comparison between *NNF-Descent* and its variants. All six datasets were used in the comparison between *NNF-Descent* and other methods for feature selection or extraction. For each experiment, image descriptors were standardized within each dataset, and the Euclidean (L_2) distance was employed. The class labels of data objects were used solely for evaluating the quality of the resulting K-NN graph.

5.2 Number of features sparsified per iteration

Testing was performed for different choices of the number of features to be sparsified from each data object per iteration, using the four image sets.

On ALOI-100, MNIST and Google-23, the choices of z were in {3, 5, 10, 15, 20}, whereas on ORL faces the choices were in {30, 50, 100, 150, 200}. K was set at 10, and the updated K-NN graph in each iteration was used for *LLS* feature ranking. The parameter σ in the RBF kernel was set to the average distance value stored in the exact 10-NN graph.

For the evaluation of the semantic quality of the resulting K-NN graphs, we define the *graph correctness* as follows:

$$\text{graph correctness} = \frac{\#\text{correct neighbors}}{\#\text{data} \times K},$$

where a correct neighbor is one whose class label coincides with that of the query object. An alternative measure for the semantic quality of K-NN graphs could be the *edge precision*, which is the proportion of graph edges connecting two data points from the same class. Of the two evaluation criteria, we present experimental results only in terms of graph correctness, since all methods tested exhibited very similar performance trends for both criteria.

In Fig. 4 we plot the performances of our method with different values of z, reporting the graph correctness at every second iteration. As a baseline, the correctness of the exact K-NN graph computed from the original feature vectors is also presented in the figure (indicated as 'exact K-NN'). At iteration 0, instead of performing *NN-Descent* without feature selection, we simply used the correctness values produced by the exact K-NN for all configurations of *NNF-Descent*, so that all curves converged to a single point at the left-hand side of the figures.

On the four image datasets, our proposed method achieves significant improvements in terms of the graph correctness, indicating that the the average number of correct neighbors per individual images is increased. With a larger number of features z sparsified per iteration, our method achieves its performance peak after fewer iterations, but thereafter degrades faster, as the number of sparsified features shared by images of different classes increases. Smaller choices of z lead to more gradual changing in performance, and occasionally a better peak performance (for example, on ALOI-100 and MNIST). However, it may require substantially more iterations to reach the performance peak. In practice, as a reasonable starting point for parameter tuning, we recommend that z be set to approximately 1 % of the number of features.

Although the number of iterations at which peak performance is reached varies from dataset to dataset, it also is influenced by the semantic quality of the initial K-NN graph, and the number of features sparsified in each iteration. It is difficult to determine an ideal value for parameter N; however, we still observe a notable improvement of *NNF-Descent* over the exact K-NN in the first 30–50 iterations.

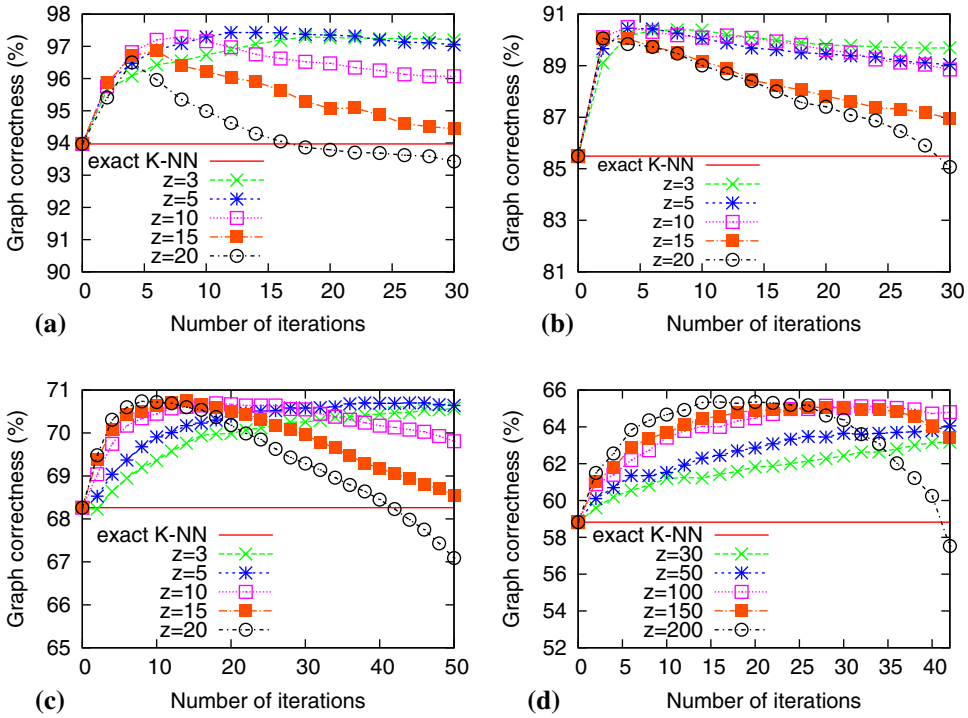

Fig. 4 Performances of *NNF-Descent* for different numbers of features sparsified per iteration. **a** ALOI-100, **b** MNIST, **c** Google-23, **d** ORL faces

For the remainder of the experiments, the value of N was not fixed—instead, we show the results over a large range of iterations.

5.3 Replacing noisy feature values by the local mean

We compared *NNF-Descent* with *Var1* on the four image sets using $K = 10$. As in Sect. 5.2, the K-NN graphs produced were subsequently used by *LLS* for feature ranking. The value of z was set at 5 for ALOI-100, MNIST and Google-23, and at 100 for ORL faces.

The results can be found in Fig. 5, from which we see that both methods can improve the correctness of produced K-NN graphs, indicating the effectiveness of the feature modification scheme. On Google-23, *Var1* achieves better results, whereas the performance gap is small—the largest difference between *Var1* and *NNF-Descent* is roughly 0.6 %. On the other three datasets, *NNF-Descent* outperforms *Var1* within several iterations, and has higher peak values for graph correctness.

In practice, the local mean of a feature is computed from different neighborhoods, and is not fixed for a data class—this can be observed by considering a semantic image class that contain several visually distinct subclasses. As a result, data objects from the same class may be assigned different values for a common noisy feature, and thus, the intra-class distances of the objects may not be reduced by assignment of the local mean. One possible explanation of the better performance of *Var1* on Google-23 is that the cases in which neighboring images have many noisy features in common may occur less often than with the other three datasets.

5.4 Effectiveness of iterative feature ranking

To demonstrate the effectiveness of iterative feature ranking, we compared *NNF-Descent* with the two remaining variants, *Var2* and *Var3*. The framework for the experiments of Sect. 5.3 was employed here as well.

The results can be found in Fig. 6. They show that the performance of *NNF-Descent* is consistently better than those of the two variants. This implies that iterative feature ranking and K-NN updating are mutually beneficial: an updated K-NN graph improves the accuracy of feature ranking, and the sparsification of noisy features improves the semantic quality of the K-NN graph in return.

It is also interesting to note that *Var2* outperforms *Var3* on Google-23 and ORL faces. On ALOI-100, *Var2* has better performance after 12 iterations. On MNIST, *Var3* is better, but the difference is small. A possible reason for the relatively poor performance of *Var3* is that in each iteration, the K-NN graph is computed from scratch using new feature vectors. If the feature ranking is unreliable, the semantic quality of the graph is severely affected. In contrast, *Var2* adopts a conservative neighborhood updating scheme in which a K-NN graph is updated from its previous status.

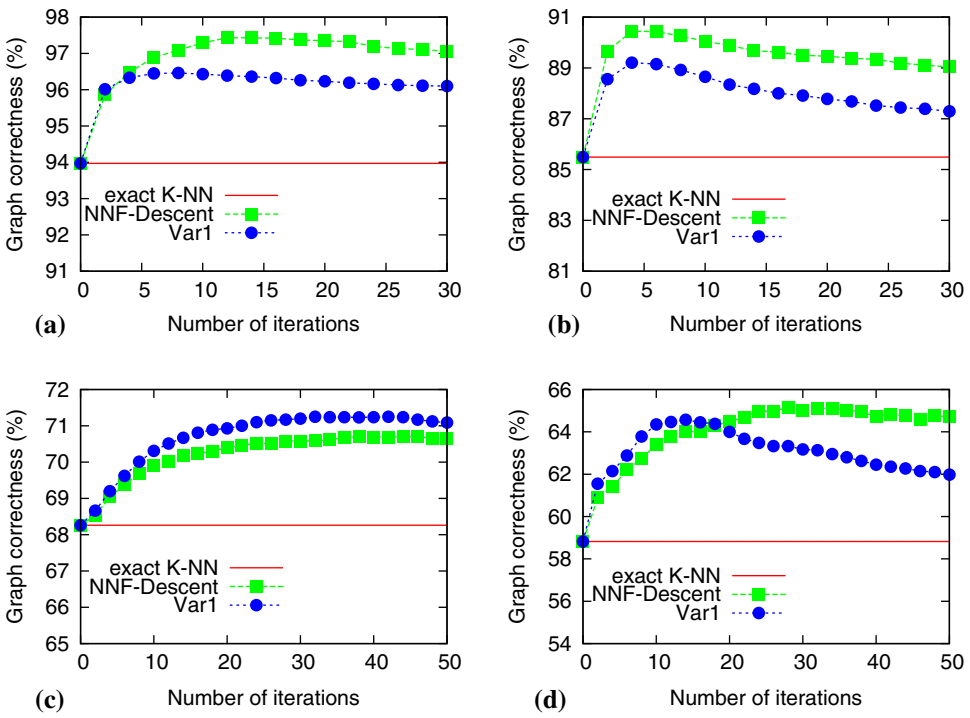

Fig. 5 Comparing *NNF-Descent* with *Var1*. **a** ALOI-100, **b** MNIST, **c** Google-23, **d** ORL faces

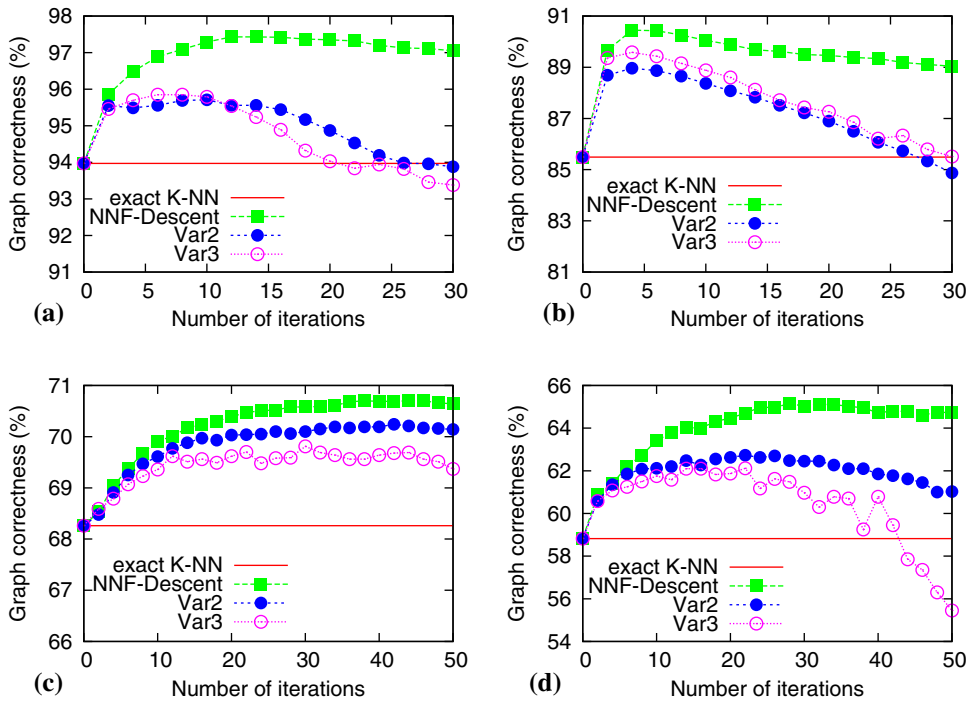

Fig. 6 Comparing *NNF-Descent* with *Var2* and *Var3*. **a** ALOI-100, **b** MNIST, **c** Google-23, **d** ORL faces

5.5 Comparison against existing methods with respect to graph correctness

On all six datasets, we compared *NNF-Descent* with *PCA*, *LS*, *SPEC-ϕ1*, *SPEC-ϕ3* and *UDFS* with respect to the correctness of produced K-NN graphs. Among the competing methods, *NNF-Descent*, *PCA*, *LS* and *SPEC-ϕ1* are fully unsupervised, while *SPEC-ϕ3* and *UDFS* require the number of classes as an input.

The value of z was set at 5 for ALOI-100, MNIST, Google-23 and Secom, at 100 for ORL faces and at 1 for Movement. The neighborhood size K for the target graph was set at 10, 20, 30, 50 and 100 on ALOI-100, MNIST, Google-23 and Secom. On ORL faces and Movement, only $K = 10$

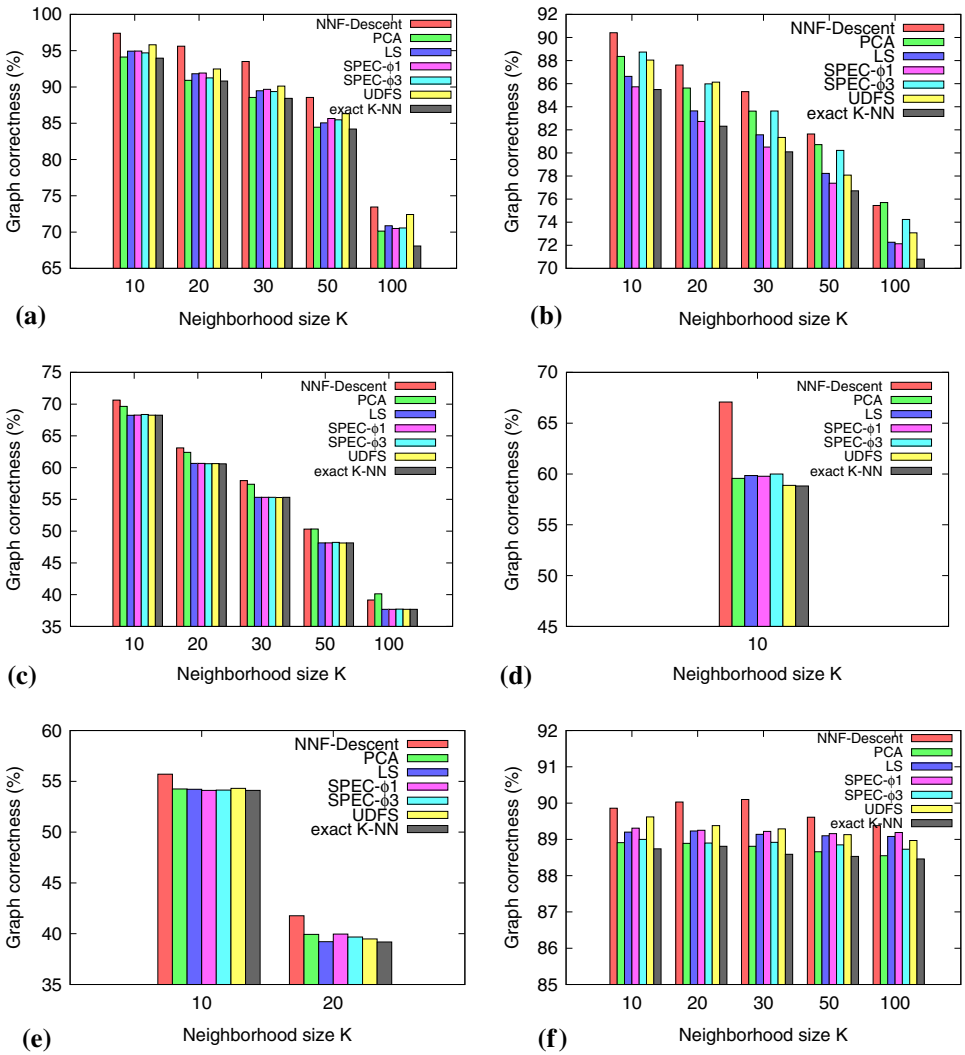

Fig. 7 Comparing the graph correctness of *NNF-Descent* with that of competing methods. **a** ALOI-100, **b** MNIST, **c** Google-23, **d** ORL faces, **e** Movement, **f** Secom

and $K \in \{10, 20\}$ were tested, respectively, since the number of objects in each subject of the two datasets is small. The full K-NN graph was used for feature ranking in *LS*, *SPEC*, *UDFS* and our method *NNF-Descent*. The RBF kernel spread parameter σ for *LS*, *SPEC* and *NNF-Descent*, and the regularization parameter for *UDFS*, were tuned using $K = 10$ for all datasets. For each method, the best values obtained by tuning were chosen for use in the remainder of the experiments.

For each experimental run of *NNF-Descent*, we computed the best graph correctness score over 50 iterations. The average computed from 5 runs was reported for each dataset.

For the other methods, for each data point, the z least important features were sparsified per iteration, and a K-NN graph was computed from the resulting set of reduced feature vectors. Over all K-NN graphs produced (one per iteration)—from the original full-sized vectors to those having fewer than z features—the best correctness value achieved over the feature reduction process was reported.

Note that we did not compare the running time used for the methods evaluated in the experiments. Unlike *NNF-Descent*, which integrates *LLS* in a graph construction framework, for the other methods feature selection and graph computation are two separate processes. After every z features are discarded by *LS*, *SPEC*, *UDFS* and *PCA*, an exact K-NN graph is computed—at a computational cost of $O(mn^2)$—so as to allow a fair comparison among the methods.

The results can be found in Fig. 7. The performance of the exact K-NN graph computed from the original feature vectors is plotted as a baseline. On all six datasets tested, for all choices considered for the value of K, *NNF-Descent* is able to achieve graph correctness scores better than those of the exact K-NN graphs. In almost all cases, our method clearly outperforms its competitors.

On ALOI-100, our method has consistently better results than its competitors. *PCA* fails to construct a K-NN graph better than exact K-NN except when $K = 100$. *LS*, *SPEC* and

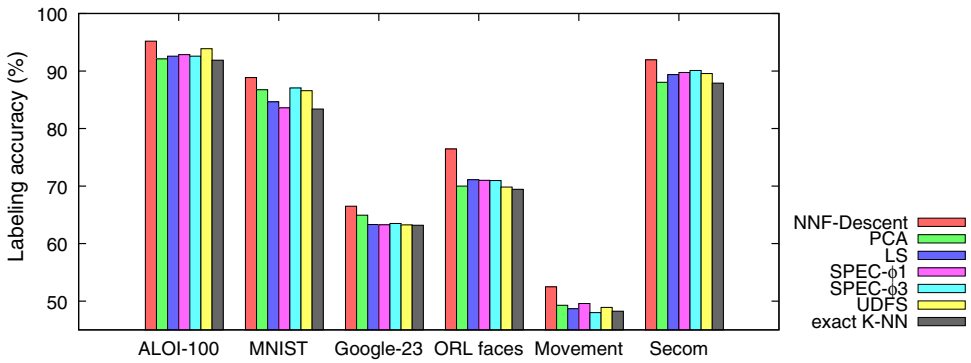

Fig. 8 Comparing *NNF-Descent* with its competitors on a labeling task

UDFS feature selection methods outperform *PCA* and exact K-NN by taking advantage of the high semantic quality of the initial K-NN graphs for this simple dataset.

On ORL faces, our method outperforms its competitors by a large margin. When $K = 10$, the best correctness value achieved by our method is 67.1 %, while the nearest competitors *SPEC-*ϕ3 and exact K-NN achieve 60.0 and 58.8 %, respectively.

On MNIST, although *LS* and *SPEC-*ϕ1 are both better than exact K-NN, the best-performing methods are *PCA*, *SPEC-*ϕ3, *UDFS* and *NNF-Descent*. When $K \leq 30$, our method outperforms *SPEC-*ϕ3, which in turn outperforms *PCA*. When $K = 50$, *PCA* overtakes *SPEC-*ϕ3, and when $K = 100$, it also outperforms *NNF-Descent* slightly, by 0.3 %. Similar outcomes were observed for *PCA* and our method on Google-23, where *LS*, *SPEC* and *UDFS* failed to make improvements over exact K-NN. *NNF-Descent* maintains its advantage over *PCA* until $K = 50$. When $K = 100$, *PCA* outperforms our method by a margin of 0.9 %. This outcome can be explained by the semantic quality of the K-NN graphs upon which *NNF-Descent* rank features. As can be seen from the degradation of the performance of exact K-NN in Fig. 7a–c, e, when the neighborhood size increases, the proportion of correct neighbors in the K-NN graph becomes smaller. All of the evaluated methods except for *PCA* utilize K-NN graphs for feature ranking: if the semantic quality of the K-NN graph degrades, the detection of noisy features becomes less reliable.

On Movement and Secom, *NNF-Descent* outperforms its competitors, which indicates that it can be easily adapted to other data types as long as the instances are represented as high-dimensional feature vectors. It is worth mentioning that on Movement, *NNF-Descent* is able to improve the semantic quality of the K-NN graph, even when the initial K-NN graph has a correctness value less than 40 %. On Secom, there is no obvious degradation of the quality of produced K-NN graphs when K increases, the reason being that there are many more positive examples than negative examples in this dataset.

5.6 Comparison against existing methods in data labeling

We performed in-dataset labeling using the K-NN graphs produced during the procedure of feature sparsification (for *NNF-Descent*) and reduction (for the other methods).

We chose the same values for z as in the experiments of Sect. 5.5. With all six datasets, 10 % of the data objects from each category were randomly selected for initial labeling in each run. A simple labeling strategy was adopted: the class label of each initially unlabeled data object is determined by its nearest labeled object in its K-NN list. A large K was used to guarantee that each object would be labeled eventually (K was 100 in this experiment). The neighborhood size for feature ranking was set at 10 for all methods evaluated (except for *PCA*).

The semantic quality of the K-NN graphs was assessed using the labeling accuracy:

$$\text{labeling accuracy} = \frac{\#\text{correctly labeled data}}{\#\text{initially unlabeled data}}.$$

As in Sect. 5.5, for our method, we computed the best labeling accuracy over 50 iterations; for its competitors, features were reduced iteratively (z per iteration), until all features were exhausted. All results reported in Fig. 8 were obtained by averaging the best accuracies from 5 trials of experiments for each method evaluated.

NNF-Descent has the best performance over all the competing methods on the six datasets. With respect to the labeling accuracy, the differences between our method and its closest competitor are 1.3, 1.8, 1.6, 5.3, 2.9 and 1.9 % for ALOI-100, MNIST, Google-23, ORL faces, Movement and Secom, respectively. For all methods tested, the results shown in Fig. 8 present a trend for labeling accuracy that is generally consistent with that of graph correctness when $K = 10$ (Fig. 7).

The experiment provides evidence that our method can improve the semantic quality of K-NN graphs, in that semantically related data objects are ranked higher within the neighborhoods in which they appear.

6 Conclusion and future work

In this paper, we presented a K-NN graph construction method, *NNF-Descent*, that uses sparsification of feature values within a nearest-neighbor descent framework to improve the semantic quality of K-NN graphs for image databases, when the class labels are unavailable.

We proposed the use of a local variant of the Laplacian Score for assessing whether a feature helps or hinders the association between an image and the other members of the class to which it belongs. To reduce intra-class image distances, we adopted a heuristic solution in which locally noisy features (as identified using the Local Laplacian Score) are sparsified from initially standardized feature vectors. Feature ranking and sparsification steps were then incorporated into the *NN-Descent* iterative K-NN graph construction framework so as to improve the semantic quality of the graph.

An experimental comparison was provided of the performance of our method against those of several unsupervised feature extraction and selection methods, with respect to the correctness of the K-NN graphs produced, and in an in-dataset labeling task, on four image datasets and two non-image datasets whose objects are also represented by high-dimensional feature vectors. Our method significantly outperformed its competitors in most cases.

It is worth mentioning that unlike traditional feature selection schemes, *NNF-Descent* is not a supervised learning method—it does not make use of separate training and test datasets as would most classifiers. The goal of this paper is to build a K-NN graph with better semantic quality for a fixed dataset. This technique can be applied in such applications as in-dataset image querying, indexing, labeling, image clustering and graph-based semi-supervised learning.

NNF-Descent is designed mainly for dense vectors and may not work well for sparse features, such as the bag-of-visual-words representation. When the feature vectors are too sparse, the standardized features may still contain many zero entries. In such situations, the sparsification process may undesirably remove valuable information and greatly change the K-NN graph structure. The application of locally noisy feature selection for sparse feature vectors would be a worthwhile topic for future research.

Other research directions in this area include:

- Empirical evaluation of the K-NN graphs produced by *NNF-Descent* in applications such as image clustering.
- A wider study as to the best strategies for the identification and modification of locally noisy features.
- Investigation of the potential for adapting *LLS* and feature sparsification in the development of new feature selection methods.

Acknowledgments Michael Houle acknowledges the financial support of JSPS Kakenhi Kiban (C) Research Grant 24500135 and the JST ERATO Kawarabayashi Large Graph Project. Vincent Oria acknowledges the financial support of NSF under Grant 1241976.

References

1. Bache K, Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml
2. Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput 15(6):1373–1396
3. Beygelzimer A, Kakade S, Langford J (2006) Cover trees for nearest neighbor. In: Proceedings of the international conference on machine learning, pp 97–104
4. Brito M, Chávez E, Quiroz A, Yukich J (1997) Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. Stat Probab Lett 35(1):33–42
5. Chen J, Fang H, Saad Y (2009) Fast approximate kNN graph construction for high dimensional data via recursive Lanczos bisection. J Mach Learn Res 10:1989–2012
6. Dias DB, Madeo RCB, Rocha T, Bíscaro HH, Peres SM (2009) Hand movement recognition for Brazilian sign language: a study using distance-based neural networks. In: Proceedings of the international joint conference on neural networks, pp 697–704
7. Dong W, Charikar M, Li K (2011) Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proceedings of the 20th international conference on world wide web, pp 577–586
8. Dy JG, Brodley CE, Kak AC, Broderick LS, Aisen AM (2003) Unsupervised feature selection applied to content-based retrieval of lung images. IEEE Trans Pattern Anal Mach Intell 25(3):373–378
9. Fukunaga K (1990) Introduction to statistical pattern recognition, 2nd edn. Academic Press, San Diego
10. Geusebroek JM, Burghouts GJ, Smeulders AWM (2005) The Amsterdam library of object images. Int J Comput Vision 61(1):103–112
11. Gionis A, Indyk P, Motwani R (1999) Similarity search in high dimensions via hashing. In: Proceedings of the 25th international conference on very large data bases, pp 518–529
12. Guldogan E, Gabbouj M (2008) Feature selection for content-based image retrieval. Signal Image Video Process 2(3):241–250
13. He R, Zhu Y, Zhan W (2009) Fast manifold-ranking for content-based image retrieval. ISECS Int Colloq Comput Commun Control Manag 2:299–302
14. He X, Cai D, Niyogi P (2006) Laplacian score for feature selection. In: Advances in neural information processing systems, vol 18. MIT Press, Cambridge, MA, pp 507–514
15. Houle ME, Oria V, Satoh S, Sun J (2011) Knowledge propagation in large image databases using neighborhood information. In: Proceedings of the ACM multimedia, pp 1033–1036
16. Houle ME, Oria V, Satoh S, Sun J (2013) Annotation propagation in image databases using similarity graphs. TOMCCAP 10(1):7
17. Houle ME, Ma X, Oria V, Sun J (2014) Improving the quality of K-NN graphs for image databases through vector sparsification. In: Proceedings of the international conference on multimedia retrieval, pp 89–96
18. Jiang W, Er G, Dai Q, Gu J (2006) Similarity-based online feature selection in content-based image retrieval. IEEE T Image Process 15(3):702–712
19. Kohavi R, John GH (1997) Wrappers for feature subset selection. Artif Intell 97(1–2):273–324

20. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
21. Qin D, Gammeter S, Bossard L, Quack T, Gool LJV (2011) Hello neighbor: accurate object retrieval with *k*-reciprocal nearest neighbors. In: Proceedings of the 24th IEEE conference on computer vision and pattern recognition, pp 777–784
22. Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
23. Rashedi E, Nezamabadi-pour H, Saryazdi S (2013) A simultaneous feature adaptation and feature selection method for content-based image retrieval systems. Knowl Based Syst 39:85–94
24. Robnik-Sikonja M, Kononenko I (2003) Theoretical and empirical analysis of ReliefF and RReliefF. Mach Learn 53(1–2):23–69
25. Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290:2323–2326
26. Samaria FS, Harter AC (1994) Parameterisation of a stochastic model for human face identification. In: Proceedings of the second IEEE workshop on applications of computer vision, pp 138–142
27. Sun Y, Bhanu B (2010) Image retrieval with feature selection and relevance feedback. In: Proceedings of the 17th IEEE international conference on image processing, pp 3209–3212
28. Tang J, Hong R, Yan S, Chua TS, Qi GJ, Jain R (2011) Image annotation by kNN-sparse graph-based label propagation over noisily tagged web images. ACM Trans Intell Syst Technol 2(2):14
29. Tong H, He J, Li M, Ma WY, Zhang HJ, Zhang C (2006) Manifold-ranking-based keyword propagation for image retrieval. In: Proceedings of EURASIP journal advances in signal processing
30. Vasconcelos N, Vasconcelos M (2004) Scalable discriminant feature selection for image retrieval and recognition. Proc IEEE Conf Comput Vis Pattern Recognit 2:770–775
31. Yang Y, Shen HT, Ma Z, Huang Z, Zhou X (2011) $l_{2,1}$-norm regularized discriminative feature selection for unsupervised learning. In: Proceedings of the international joint conferences on artificial intelligence, pp 1589–1594
32. Zhao Z, Liu H (2007) Spectral feature selection for supervised and unsupervised learning. In: Proceedings of the 24th international conference on machine learning, pp 1151–1157
33. Zhu X, Ghahramani Z, Lafferty JD (2003) Semi-supervised learning using gaussian fields and harmonic functions. In: Proceedings of the 20th international conference on machine learning, pp 912–919