Consensus of k-NNs for Robust Neighborhood Selection on Graph-Based Manifolds

Vittal Premachandran and Ramakrishna Kakarala
School of Computer Engineering, Nanyang Technological University, Singapore-639798
vittalp@pmail.ntu.edu.sg and ramakrishna@ntu.edu.sg

Abstract

Propagating similarity information along the data manifold requires careful selection of local neighborhood. Selecting a "good" neighborhood in an unsupervised setting, given an affinity graph, has been a difficult task. The most common way to select a local neighborhood has been to use the k-nearest neighborhood (k-NN) selection criterion. However, it has the tendency to include noisy edges. In this paper, we propose a way to select a robust neighborhood using the consensus of multiple rounds of k-NNs. We explain how using consensus information can give better control over neighborhood selection. We also explain in detail the problems with another recently proposed neighborhood selection criteria, i.e., Dominant Neighbors, and show that our method is immune to those problems. Finally, we show the results from experiments in which we compare our method to other neighborhood selection approaches. The results corroborate our claims that consensus of k-NNs does indeed help in selecting more robust and stable localities.

1. Introduction

Using the underlying manifold structure has proven to significantly improve performance in many vision-related tasks [14, 16]. The most notable of them is the task of shape/image retrieval. The task of shape retrieval is an especially difficult task because of the vast variability of shapes even within a particular class. Given a query object, the goal of retrieval tasks is to retrieve the most similar shapes in the database. Similar objects are usually retrieved using some similarity/dissimilarity measure (ex: [2, 6]), which is computed between pairs of shapes. Many of these similarity/dissimilarity measures violate the triangle inequality and, hence, are not metrics. If the underlying manifold structure of the shapes is curved, then the Euclidean distance between shapes cannot be a good metric for shape comparison. In such cases, the geodesic distance on the shape manifold is a better metric for comparing shapes than pairwise similarity/dissimilarity measures.

Many techniques have been proposed to capture this underlying manifold structure, and hence learn the correct geodesic distances between data points that lie on the manifold. Since the manifolds are usually of a much lower dimensionality than the space in which they lie, many manifold learning algorithms make use of dimensionality reduction methods (ex: [12, 11]) to reduce the dimensionality of the feature space. Such dimensionality reduction techniques map the features onto a lower dimensional subspace in hope that the Euclidean distance in this new lower dimensional subspace can capture geodesic distance from the original higher dimensional space.

In applications such as shape retrieval, we might not have access to the features of the data points, and might be forced to work in the data space. The shape manifold in the data space is represented as a graph with edge weights proportional to the similarity score. The true distance between two shapes can be learnt by considering the distance in context of other shapes in the neighborhood. The new distances are calculated by propagating the similarity information along the weighted edges of the graph.

Many recent papers make use of such contextual information to learn new affinity scores between pairs of data points [1, 4, 15, 16, 18]. The similarity information is usually propagated as a diffusion process on the graph. Yang et al. [16] perform diffusion on a locally constrained sparse graph, while in [18], the diffusion process is performed on a tensor product graph, thus allowing the capture of higher order information. The diffusion process is susceptible to noise [3], and hence the affinity propagation is not performed on a fully connected graph. Both [16] and [18] follow different styles of graph sparsification (k-nearest neighbors and dominant neighbors, respectively). The selection of a proper neighborhood is critical for diffusion to work.

This paper addresses the question of how to build a strong local neighborhood given data in the form of a graph. In this paper, we point out the drawbacks of using k-nearest neighbors (k-NN) and dominant neighbors (DN). Dominant neighbors are the nodes that form a maximal clique in a graph (Section 3.1 explains the process of finding the dominant neighbors in detail). We propose a new way for neighborhood selection by making use of the consensus information from various neighborhoods, and show that such a neighborhood is much more robust to parameter selections.

We also show that using our consensus neighborhood information, we are able to achieve better retrieval results on standard databases (Ex: MPEG7 shape database), as opposed to the use of k-NN or DN.

2. Sparse Affinity Matrix Generation

Given N shapes from a database, we can generate a NxN cost matrix by pairwise comparison of each pair of shapes s_i and s_j, $\forall i, j \in \{1, ...N\}$, using standard shape comparison methods such as [2, 6]. Ideally, we would want the shape comparison methods to generate costs such that similar shapes have less cost and dissimilar shapes have greater costs between them, and for them to obey the triangle inequality. Unfortunately, this is not true of any of the shape matching techniques. In order to learn the true geodesic distances on the shape manifold, contextual information has to be taken into consideration. Using the NxN matrix allows us to exploit much more information about the neighborhood structure of the data manifold than using just the pairwise information.

2.1. Affinity Matrix

The usual trend is to work with similarity scores rather than dissimilarity costs. The authors in [15] convert the cost matrix into a similarity matrix (also named as affinity matrix) and use this affinity matrix to learn the geodesics on the manifold. The affinity between a pair of shapes is calculated as follows:

$$A(i,j) = exp(-D(i,j)^2/\sigma_{ij}^2). \quad (1)$$

Here, A is the affinity matrix, D is the distance matrix, and σ specifies the kernel size. The choice of σ is critical in generating a "good" affinity matrix. A "good" σ would help in pulling intra-class objects together and in pushing inter-class objects far away from each other. Different methods have been proposed for choosing a proper σ. Perona et al. [10] use σ_{ij} as

$$\sigma_{ij} = \sigma_i \sigma_j, where \;\; \sigma_i = D(i, K(i)), \quad (2)$$

where $K(i)$ is the Kth nearest shape to shape s_i, and Yang et al. [15] use σ_{ij} as

$$\sigma_{ij} = mean(\text{K-NN}(s_i), \text{K-NN}(s_j)), \quad (3)$$

where $mean$(K-NN(s_i), K-NN(s_j)) is the mean of the K-nearest distances of shape s_i and shape s_j. Both these methods rely on the proper choice of the kernel neighborhood parameter, K, and choosing a "bad" K, would adversely affect the generation of a good affinity matrix.

2.2. Local Neighborhood Sparsification

While performing similarity propagation on a graph, it is extremely important to prune out noisy edges as the diffusion process is susceptible to noise [3]. Roweis et al. [11] assumed linearity of local neighbourhoods on a manifold, and graph sparsification follows a similar principle. It is assumed that the edge weights between data points that are close to each other on the data manifold approximate the geodesic information better than the edge weights between data points that are farther away from each other. Hence, graph sparsification tries to prune out edges between nodes that are not in a local neighborhood. For a graph $G(V, E)$, we can have different variants of the neighborhood graph G', such as,

- Symmetric k-NN graph neighborhood $G'(V, E)$, where there is an edge $E(v_i, v_j)$ if $v_j \in$ k-NN(v_i) or $v_i \in$ k-NN(v_j).
- Mutual k-NN graph neighborhood $G'(V, E)$, where there is an edge $E(v_i, v_j)$ if $v_j \in$ k-NN(v_i) and $v_i \in$ k-NN(v_j).
- ϵ-Neighborhood graph $G'(V, E)$, where there is an edge between v_i and v_j, if $D(i, j) \leq \epsilon$.
- Dominant Neighborhood graph $G'(V, E)$, where there is an edge $E(v_i, v_j)$ if $v_j \in DN(v_i)$.

Of these, the ϵ-neighborhood graph is susceptible to scaling. Different clusters can have different radii. So, selecting a single ϵ for all nodes in the graph might not properly capture the neighborhood structure of the nodes. The k-NN graph neighborhoods produce a fixed-size neighborhood. However, as pointed to in [17], the k-NN graph has a tendency to include noisy edges in the neighborhood of a node. Moreover, using a fixed-size neighborhood might not adequately capture the locality in the manifold.

The need for a variable-size neighborhood for manifold structure learning was pointed to in [17] and [19]. Zhang et al. [19] propose an adaptive neighborhood selection for manifold learning in the feature space, while Yang et al. [17] make use of dominant set computation method [9] for selecting the dominant subset of the k-nearest neighbors in the data space. The idea behind selecting a dominant neighborhood as opposed to just the k-nearest neighbors is that, the dominant neighborhood usually forms tight clusters and is therefore composed of nodes that are highly similar to each other. Therefore, dominant neighbors are less prone to noisy edges than k-nearest neighborhood.

While the dominant neighborhood graph can select variable sized neighborhoods, it is still dependent on the selection of the sparsification parameter, k[1]. Both k-NN and DN, can be adversely affected if the value of k is incorrectly chosen. In the next section, we explain in detail the problems related to the parameter selection. We will also explain how our consensus neighborhood selection strategy will help mitigate those problems.

[1]We use K to denote the kernel parameter and k to denote the sparsification parameter

3. Consensus k-NNs

Both k-NN and DN can select good local neighborhoods as long as the graph sparsification parameter, *k*, is properly selected. Looking back at symmetric k-NN neighborhood selection, we can see that an edge between v_i and v_j is selected if $v_j \in kNN(v_i)$ or $v_i \in kNN(v_j)$. This process is repeated for all nodes $i \in \{1, 2, ...N\}$ in order to obtain local neighborhoods of every node in the graph. Some of these edges might be noisy edges, while some of them are indeed accurate edges. Accurate edges are those edges that connect a particular node to other nodes, which are part of the true neighborhood, while noisy edges are those that connect a node to other nodes that are not part of the true neighborhood. As the neighborhood size, k, increases, so do the chances of adding in noisy edges. To make the neighborhood more stable even for large values of k, we propose to make use of consensus information from the multiple k-NN procedures that are applied to the graph. Consensus clustering has been previously used for other problems [5, 8] and has shown impressive results.

We define a consensus matrix, C, to keep track of the number of times a pair of nodes (v_p, v_q) appear together among all rounds of k-NN. A simple pseudocode to populate our consensus matrix is given below.

```
C = 0;
for i = 1 : N do
    S_i = k-NN(v_i);
    for p = 1 : N do
        for q = p + 1 : N do
            if p ∈ S_i and q ∈ S_i then
                C(p, q) = C(p, q) + 1;
                C(q, p) = C(q, p) + 1;
            end
        end
    end
end
```

Algorithm 1: Algorithm to collect the consensus information from multiple rounds of k-NNs.

The first advantage that we obtain from having such a consensus matrix is that it allows us to capture stronger relations between pairs of nodes. In the symmetric k-NN graph, a pair of nodes is either part of each other's neighborhood, or not. Whereas, if we use the consensus of k-NNs, we can be far more certain about the similarity, or dissimilarity, between pairs of vertices. The relation between a pair of nodes (v_p, v_q), which are a part of k-NN(v_i), were ignored in the case of symmetric k-NN (just the edges between v_i and v_p, and between v_i and v_q, were added to the graph). However, the fact that the two nodes v_p and v_q are a part of the same neighborhood, albeit some other node v_i, shows that v_p and v_q are similar to each other as well. If v_p and v_q keep appearing among the k-NNs of multiple nodes, then the chances of the two nodes being similar to each other further increases. This points to the second advantage of using consensus information.

Probabilistic Neighborhood Information: A row-normalized consensus matrix can be viewed as a probability matrix, where each value specifies the probability of that pair of nodes being similar to each other. We are not privy to such soft measures if we use the symmetric k-NN for neighborhood generation. In a symmetric k-nearest neighborhood, noisy edges have the same probability of being a part of a particular locality as accurate edges. Such noisy edges might have been included in the neighborhood by chance. The probability that such edges will be a part of multiple neighborhoods, however, is low. With the use of the consensus matrix, we can easily identify such noisy edges as those edges that have a low probability value, and can hence be ignored. Figure 2b shows a consensus matrix from one of our experiments.

With the use of such probabilistic information, we get more control over neighborhood tuning. A simple way to prune out noisy edges is to select only those pairs of edges that have a probability greater than some fixed threshold. More formally, the new probabilistic neighborhood graph $G'(V, E)$, has an edge $E(v_i, v_j)$, if $C(i, j) \geq \tau$. The threshold, τ, is a global threshold that does not need to be set independently for separate nodes. Unlike symmetric k-nearest neighborhoods, even for a fixed value of τ we can get variable-sized neighborhoods that can adaptively represent the local neighborhood structure.

Pruning out noisy edges leaves us with coherent clusters, where elements of the clusters are all highly similar to each other. Yang et al. [17] also tried to obtain such a neighborhood, but they used the dominant set method for identifying such clusters. Our consensus neighborhood identification method is better in many ways compared to the dominant set extraction, especially when used for manifold learning. In the following subsection, we explain the issues that the dominant sets cannot overcome, which are especially critical for manifold learning.

3.1. Advantages of Consensus Neighborhood over Dominant Sets

As mentioned in the previous sections, the goal of any graph sparsification method is to produce a neighborhood that best preserves the locally linear neighborhood property of manifolds. It is assumed that coherent clusters form good local neighborhoods and, hence, clustering algorithms are used to identify local neighborhoods. The dominant set extraction method, proposed by Pavan and Pellilo [9] has shown impressive results for identifying good clusters. The authors of [9] consider clusters as dominant sets and

propose an algorithm to calculate the maximal cliques in a graph. Given an affinity matrix A, and a probabilistic indicator vector $\mathbf{x}$, maximal cliques can be extracted by maximizing the following quadratic objective.

$$\begin{array}{ll} maximize & f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x} \\ subject\ to & \mathbf{x} \in \Delta \end{array} \tag{4}$$

where,

$$\Delta = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} \geq 0\ and\ \mathbf{1}^T\mathbf{x} = 1\}. \tag{5}$$

The above quadratic function can be maximized using the so-called replicator dynamics, an iterative procedure, which is guaranteed to converge at optimal locations:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t)\frac{(A\mathbf{x}(t))_i}{\mathbf{x}(t)^T A \mathbf{x}(t)}. \tag{6}$$

Local optima: The first problem that we face with the above process is that, it can easily get stuck at local optima. Finding all local optima, in order to find the global optima, is a computationally exhaustive task. To prevent the dynamics from converging at wrong optima, Yang et al. [17] restrict the search space to a local neighborhood. While finding the dominant neighborhood of v_i, they restrict the search to be among the k-nearest neighborhood of v_i, by setting $\mathbf{x}_j(1) = 1/k$, if $j \in kNN(i)$. A correct dominant neighborhood can be obtained only if prior information is available for selecting the proper value of k.

False Neighborhoods: Secondly, even when the dynamics converges at global maxima, there is no guarantee that the node whose neighborhood we are actually searching is even part of the maximal clique! Suppose we are searching for maximal cliques in a given $kNN(i)$, and the maximal clique is composed of a set of nodes, which is a subset of $kNN(i)$, but does not include the node v_i. Then, the replicator dynamics will converge with the value for $\mathbf{x}_i(t_{end}) = 0$. If we now construct a sparse neighborhood graph $G'(V, E)$, we would end up adding edges between v_i and other nodes v_j, for which $\mathbf{x}_j(t_{end}) > 0$. Clearly, these edges do not accurately represent the true neighborhood of v_i.

Liu et al. [7] devised a means to ensure that a particular node is always a part of the final solution. They did so by forcing the node under consideration to fall within a restricted neighborhood, $\mathbf{x}_i \in [\epsilon, 1]$, where $\epsilon \in (0, 1)$. While such constraints can force the involvement of a particular node in the final solution, it still cannot guarantee that the outcome of the optimization procedure is the true local neighborhood on the manifold.

Consider the example situation shown in Figure 1, where the graph has two non-intersecting cliques, of which one is the true neighborhood of the node v_1, and other is "far away" from v_1. Let S_1 be the set of nodes in the true clique

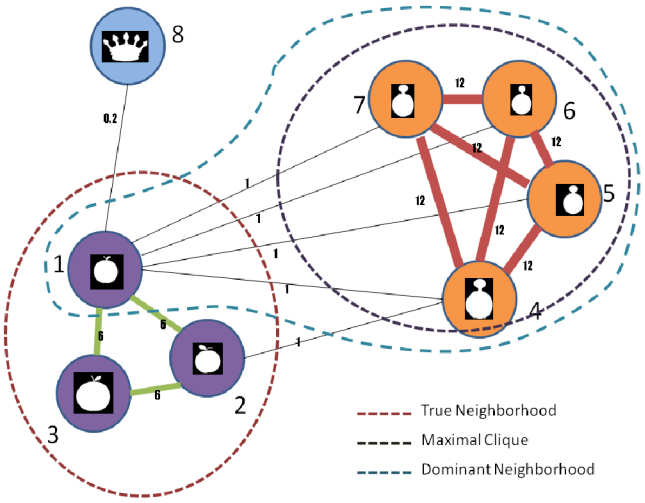


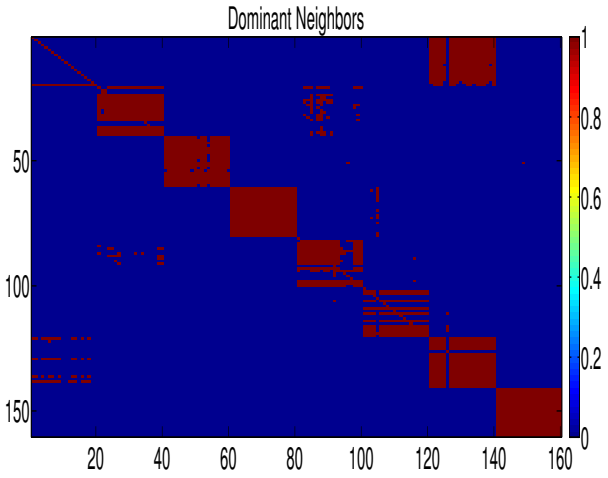
Figure 1: An example where the Dominant Neighbors of a particular node (node 1), does not include its true neighbors. Since there is less variability among pocket-watches, the affinity among them is very high. Due to the variations among the class Apples, the affinity among them is not as high as among the class of Pocket-Watches. Also, since the pocket watches do look similar to apples, they fall in the k-NN of apples. The dominant neighborhood, in this example, is a false neighborhood because, $W(S_{Pocket-Watches} \cup \{1\} > W(S_{Apples})$.

(nodes belonging to the class *Apples*), and S_2 be the set of nodes in the other clique (nodes belonging to the class *Pocket-Watches*). Also, let $W(S)$ denote the weight of the set S and be defined as

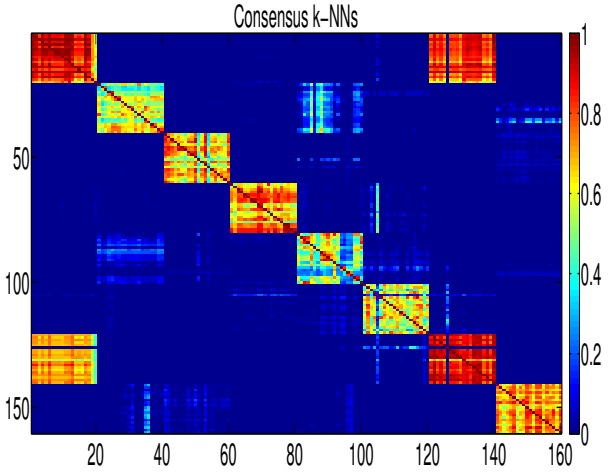
$$W(S) = \mathbf{x}_\mathbf{S}^\top \mathbf{A} \mathbf{x}_\mathbf{S} \tag{7}$$

where, $\mathbf{x}_\mathbf{S}$ is a probabilistic vector where $\mathbf{x}_i > 0$ if $i \in S$, and $\mathbf{x}_j = 0$, if $j \notin S$ (see [9] for details). If $W(S_2) >> W(S_1)$, then, the dynamics (Eq. 6) will always converge with $\mathbf{x}_j(t_{end}) > 0\ \forall\ j \in S_2$, and, $\mathbf{x}_i(t_{end}) = 0\ \forall\ i \in S_1$. Even when the node v_1 is forced to be present in the final solution, false neighborhoods can be obtained when $W(S_2 \cup \{1\}) > W(S_1)$. In our experiments, we noticed that the sparse graph generated by connecting a node to its dense neighbors, suffered from the above problem (see Figure 2).

This second problem that we have just described is actually quite a serious problem, especially when the objective is to learn a local neighborhood on the manifold. The task of neighborhood selection is one in which the algorithm selects an optimal local neighborhood, which need not always be the solution that globally maximizes a function. This is a case of an ill-formed objective since the objective function does not give importance to the similarity between the node v_i and other nodes in the solution. k-NNs are not suscep-



(a)



(b)

Figure 2: (Best viewed in color) Neighborhood matrix for a subset of the MPEG7 shape database. We use $k = 50$ for both cases. (a) Figure shows the dominant neighbors, where all nodes have a binary status. Also, note that class 1 (rows 1 to 20) has chosen a neighborhood that belongs to a completely different class, class 7 (columns 121 to 140). This is because of the false neighborhood problem that was described in Section 3.1. (b) Figure shows a probabilistic version of the neighborhood matrix obtained from consensus of k-NNs. Note that the neighborhood of class 1 still includes others nodes from the same class, with high probability, unlike DN.

tible to this problem as the first goal of k-NN is to sort the rest of the nodes in decreasing order of similarity. So, we are always guaranteed to select the most similar nodes to a particular node v_i. The problem with k-NN (i.e., its tendency include noisy edges), which forced the adoption of DN, is overcome by our method of using consensus information from multiple k-NNs. Consensus information not only retains the most similar nodes, but also gives a means to prune out noisy edges. Consensus of k-NNs has the advantages that motivated the use of DN, and is not affected by the issues that affect DN.

Disconnected Graphs: The final problem that we will discuss is the case of disconnected graphs. In order to propagate similarity information between every pair of nodes, there should be at least one path connecting a node v_i to every other node v_j. This means to say that, the sparse graph should be a connected graph (need not be a fully connected graph). The graph sparsification step should not output a graph with two or more subgraphs that have no connections between them. This is critical if we wish to learn the true geodesic distance between every pair of nodes.

Dominant set extraction procedure has a tendency to output tight-knit clusters. The different optima of the optimization objective in Eq. (4) are nodes that are subsets of V. These subsets are coherent subsets with a high degree of intra-set similarity. This causes inter-cluster edges to be pruned off, resulting in a fragmented graph with multiple connected subgraphs. If diffusion was performed on such a graph, the true geodesic distances between pairs of nodes would be learnt only among the nodes within a connected subgraph. All distances between nodes belonging to different subgraphs would end up being very large, and therefore, meaningless. In our experiments, we noticed such fragmentation of the graphs, even when k was set to moderate values ($k = 10$). This meant that, while using the dominant set for neighborhood extraction, we were never able to learn the geodesic distance between nodes belonging to mutually disconnected subgraphs for particular values of k.

The probability of ending up with fragmented subgraphs is much less while using k-nearest neighborhoods. This is because the edges are not forced to remain only among a selected subset of nodes. We noticed fragmentation of the graph, while using k-NN, only when k was chosen to be very small ($k = \{1, 2, 3\}$). Such small neighborhoods do not provide any locality information. Hence, choosing such small k's is hardly ever the case.

Summary: To summarize, consensus of k-NNs has lots of advantages over dominant neighbors. Firstly, they are not prone to problems such as local optimality. While using dominant sets, one can end up with neighborhoods that arise out of locally optimal solutions. Secondly, the "neighbors" generated by the dominant sets are not guaranteed to contain the true neighbors of the node under consideration. There are no such problems while using consensus of k-NNs as k-NN guarantee that the most similar nodes to a particular node are always part of the local neighborhood. Finally, the chances of graph fragmentation is much less (in fact, hardly ever the case) in the case of consensus k-NNs when compared to the dominant neighbors. This allows similarity information to propagate between all pairs of nodes.

3.2. Diffusion Using Consensus Information

The consensus information just gathered can be used in two ways before performing diffusion. As mentioned in Section 2, there are two pre-processing steps that are performed before propagating the similarity information. The cost matrix to affinity matrix conversion stage relies on a good choice of the kernel, σ. Eq. (3) selects σ_{ij} as the mean of the K-NN distances of the two nodes v_i and v_j. Using the consensus information, the parameter can be chosen as the mean of the distances between pairs of nodes that have a probability greater than, say, τ. This would better ensure similar nodes to be grouped together much more tightly, and dissimilar nodes to be pushed much further away.

Secondly, consensus information has significant uses during the graph sparsification stage i.e., neighborhood generation stage. We have explained above, how a good neighborhood graph, G', can be obtained from the consensus information. Given a neighborhood graph G' generated using consensus of k-NNs, one can obtain a probabilistic transition matrix P as

$$P(i,j) = \frac{E'(i,j)}{\sum_j E'(i,j)}, \tag{8}$$

where, E' is the edge set obtained from the sparse graph G'. Once P is calculated, we can now perform diffusion using any of the graph diffusion procedures (Ex: LCDP [16], or TPG [18]). In our experiments, we primarily use TPG diffusion as it takes into account higher-order similarity relations for the same space and time complexity as classical diffusion on the original graph.

4. Experiments

To demonstrate the stability of using consensus neighborhood, we compare the diffusion process when using k-NN, DN, and consensus of k-NNs for many different values of k. We perform our experiments on a spiral data and on the standard MPEG-7 shape retrieval database.

4.1. Spiral Data

The spiral data is obtained by generating samples from the Archimedes spiral as a function of arc length. We sample 1000 points from the spiral. Each generated point is perturbed by random noise. The spiral, which lies in the 2-D space, has an intrinsic 1-D structure. Here, we compare the performance of consensus of k-NNs with the simple k-NN, by purposely setting the value of k to be "large".

Figure 3 shows the output of the two approaches. The first row is obtained by using the naive k-NN, and the second row from consensus of k-NNs. The graphs on the left show the plots of the second most-important eigen vector against the arc length. The figures on the right show a color-coded spiral. Similar colors mean that the points

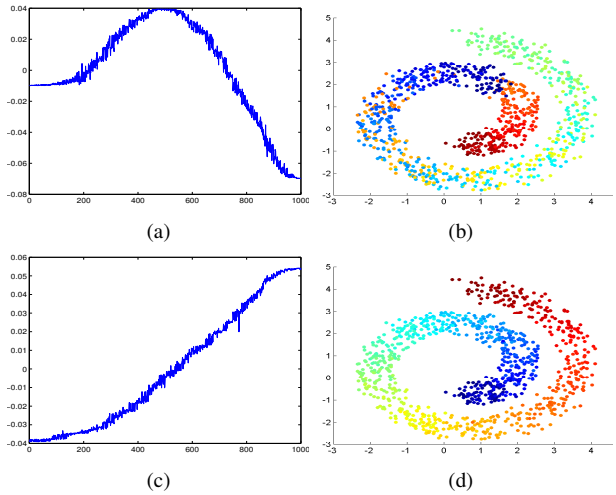


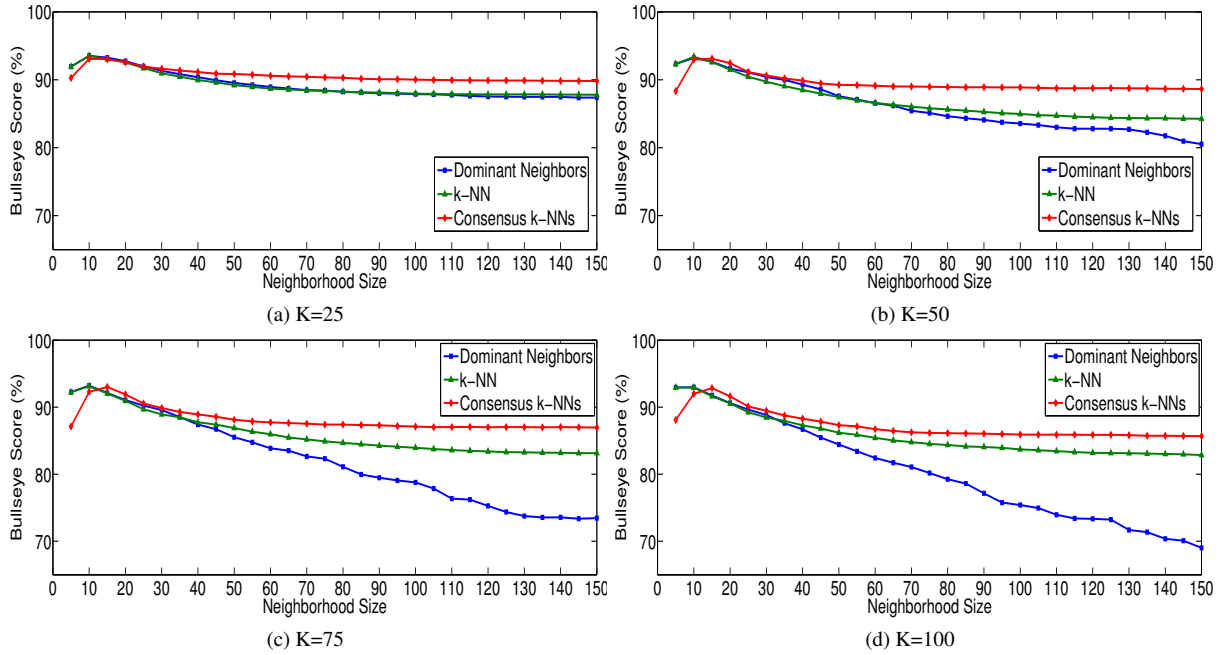
Figure 3: Top row shows the diffusion results from using simple k-NN sparse graph, and the bottom row, using the sparse graph generated using consensus of k-NNs. (a) and (c) show a plot of the mapped 1-D coordinates of the points versus the arc length. (b) and (d) show a color-coded spiral. Points that are close to each other after mapping to the lower dimension are similarly colored.

are mapped close to each other after learning the manifold structure [11, 13]. Clearly, consensus of k-NNs has learnt a better neighborhood than the simple k-NN. From Figure 3a, we can see there there is no one-to-one mapping of the coordinates to the arc length. This indicates that the geodesic distances were incorrectly learnt due to the presence of noisy edges in the simple k-nearest neighborhood. On the other hand, Figure 3c shows a clear one-to-one mapping of the coordinates to the arc length, which shows that the neighborhood generated by consensus of k-NNs was more robust to noise. For this experiment, k is set to 15, and we use TPG to perform diffusion. We have tried for many different k's and found that consensus of k-NNs performs better than the simple k-NN.

4.2. MPEG-7 Shape Retrieval Database

In this sub-section, we show the results from our experiments on a more challenging and real-world application i.e., image retrieval by learning the manifold of shapes. The well-known, and widely used, MPEG7 CE-Shape-1 Part B database consists of silhouettes of 1400 images with a wide variety among them. The database is split into 70 classes, with each class containing 20 example images. The shape-retrieval performance is measured by the so-called Bullseye score. The Bullseye score is basically the percentage of objects belonging to the same class as the query object among its top-40 best matching objects.

We learn the true shape manifold by starting off with the 1400×1400 pairwise dissimilarity matrix. We make use



(a) K=25 (b) K=50

(c) K=75 (d) K=100

Figure 4: The figure shows plots of Bullseye score v. neighborhood size, for four different affinity matrices, while using TPG diffusion process. The affinity matrices were generated using different kernels, which are calculated using different neighborhood sizes of (a) 25, (b) 50, (c) 75, and (d) 100. For each affinity matrix, we show plots of the performance of DN, k-NN and Consensus of k-NNs. We can see that DN is better than k-NN for small values of k, but deteriorates quickly for larger k's. Consensus of k-NNs performs better than DN and k-NN for almost all neighborhood sizes. Also, the performance deterioration rate is significantly slower than both k-NN and DN for larger neighborhood sizes, thus pointing towards stable localities.

of the IDSC shape dissimilarity matrix [6], which is extensively used in the literature, for learning the shape manifold structure. The matrix has a Bullseye score of 85.40% before diffusion. While the previous trend in the literature is to carefully hand-tune a good neighborhood size, and to properly select a good number of diffusion iterations, for getting better Bullseye scores, in our experiments, we purposely select neighborhood sizes that are bound to include noisy edges and show that consensus k-NNs produce a much more stable neighborhood than k-NN or DN.

Objects belonging to the same class, usually, belong to the same neighborhood. Since the MPEG-7 shape database has 20 objects per class, it comes as no surprise that the previous state-of-the-art Bullseye scores were reported for neighborhood sizes that were purposely selected to be smaller than or equal to 20 ($k = 20$ in [16] and $k = 10$ in [18]). Such parameter selections are an example of supervised neighborhood selection. However, in a completely unsupervised setting, selecting k to be less than or equal to the number of items in a particular class is highly unlikely. Therefore, we would like our neighborhoods to be stable enough even when k is chosen to be greater than the number of examples in a particular class, and thus including more objects into the local neighborhood than there would be in the "true neighborhood".

We have experimented with multiple values of k, and in Figure 4, we show the plots of Bullseye scores v. neighborhood sizes, when using k-NN, DN and consensus k-NN. Remember from Section 2 that the generation of a good sparse affinity matrix requires the choice of two neighborhood parameters: one while generating the affinity matrix and one while sparsifying the matrix. The four plots correspond to four different choices of $K(= 25, 50, 75$ and $100)$ that were used to compute the kernel σ_{ij}, while generating the affinity matrix. For each of these affinity matrices, we experiment over different neighborhood sizes while sparsifying the graph. From the plots, we can see that the neighborhood generated by using consensus of k-NNs is quite stable to the neighborhood size parameter. Once the neighborhood size goes above the true neighborhood size (i.e., $k > 20$), the performance obtained from both k-NN and DN starts deteriorating at a quicker rate than consensus k-NNs. We can also see that, up to a neighborhood size of 35-40, DN outperforms k-NN, but after that, k-NN performs better. This is because, DN starts converging onto false neighborhoods because of the problems that were explained in Section 3.1. Moreover, even while selecting small neighborhood sizes of up to 35-40, we can see that consensus of k-NNs has learnt a

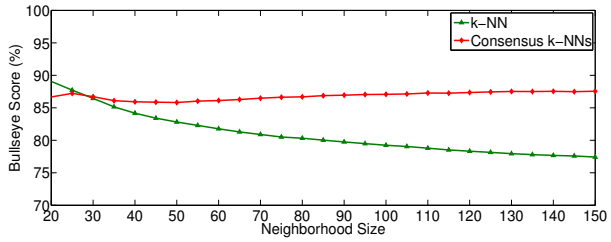


Figure 5: Bullseye score v. neighborhood size plot while using LCDP. The affinity matrix was generated using $K = 75$. We see a similar performance as in Figure 4. Consensus of k-NNs produces a more stable neighborhood than k-NN for larger values of k.

better neighborhood than DN, thus producing better Bullseye scores after diffusion. For extremely small values of k, there is a relative drop in performance of our method because there is hardly any consensus information that can be extracted while using such small neighborhood sizes.

We also compared the effect of the using k-NN versus consensus of k-NNs using the Locally Constrained Diffusion Process (LCDP). We found that the neighborhood selection had a similar effect as while using TPG. Figure 5 shows the plot for one such trial. Thus we see that the selection of neighborhood is independent of the diffusion process and behaves similarly across different diffusion processes.

As a final comment, we would like to point out that our method can be applied to other forms of graph sparsification techniques as well. Ex: We can generate a consensus of ϵ-neighborhood graphs while using ϵ-neighborhood sparsification. In our experiments we found that consensus of ϵ-neighborhoods outperformed the simple ϵ-neighborhood by a Bullseye score of 1-3%, when we experimented with multiple threshold values (ϵ). We do not discuss ϵ-neighborhood graphs in detail due to the lack of space and also because it is well-known that k-NN graphs are more stable than ϵ-neighborhood graphs (Section 2.2).

5. Conclusion

In this paper, we have identified some of the problems with the currently used neighborhood selection methods. We also propose a new way for neighborhood selection, which makes use of the consensus information from different k-NNs. We have shown that making use of such information increases the robustness of the neighbors, and thus, helps the similarity information to propagate better on the data manifolds. In the future, we would like to explore how such consensus information can be used to obtain adaptive neighborhoods on graph-based manifolds.

References

[1] X. Bai, X. Yang, L. Latecki, W. Liu, and Z. Tu. Learning context-sensitive shape similarity by graph transduction. *IEEE Trans. Pattern Anal. Machine Intell.*, 32(5):861–874, 2010.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Machine Intell.*, pages 509–522, 2002.

[3] M. Jaakkola. Partially labeled classification with markov random walks. In *NIPS*, volume 2, page 945. MIT Press, 2002.

[4] P. Kontschieder, M. Donoser, and H. Bischof. Beyond pairwise shape similarity analysis. In *ACCV*, pages 655–666. Springer, 2010.

[5] A. Lancichinetti and S. Fortunato. Consensus clustering in complex networks. *Scientific Reports*, 2, 2012.

[6] H. Ling and D. Jacobs. Shape classification using the inner-distance. *IEEE Trans. Pattern Anal. Machine Intell.*, pages 286–299, 2007.

[7] H. Liu, X. Yang, L. Latecki, and S. Yan. Dense neighborhoods on affinity graph. *Int. J. of Computer Vision*, pages 1–18, 2011.

[8] S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine learning*, 52(1):91–118, 2003.

[9] M. Pavan and M. Pelillo. Dominant sets and pairwise clustering. *IEEE Trans. Pattern Anal. Machine Intell.*, 29(1):167–172, 2007.

[10] P. Perona and L. Zelnik-Manor. Self-tuning spectral clustering. *NIPS*, 17:1601–1608, 2004.

[11] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[12] B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[13] J. Tenenbaum, V. De Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[14] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on riemannian manifolds. In *CVPR*, pages 1–8. IEEE, 2007.

[15] X. Yang, X. Bai, L. Latecki, and Z. Tu. Improving shape retrieval by learning graph transduction. *ECCV*, pages 788–801, 2008.

[16] X. Yang, S. Koknar-Tezel, and L. Latecki. Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval. In *CVPR*, pages 357–364. IEEE, 2009.

[17] X. Yang and L. Latecki. Affinity learning on a tensor product graph with applications to shape and image retrieval. In *CVPR*, pages 2369–2376. IEEE, 2011.

[18] X. Yang, L. Prasad, and L. Latecki. Affinity learning with diffusion on tensor product graph. *IEEE Trans. Pattern Anal. Machine Intell.*, 2012.

[19] Z. Zhang, J. Wang, and H. Zha. Adaptive manifold learning. *IEEE Trans. Pattern Anal. Machine Intell.*, 34(2):253–265, 2012.