Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Walking on Minimax Paths for k-NN Search

Kye-Hyeon Kim[1] and **Seungjin Choi**[1,2]
[1] Department of Computer Science and Engineering
[2] Division of IT Convergence Engineering
Pohang University of Science and Technology
77 Cheongam-ro, Nam-gu, Pohang 790-784, Korea
{fenrir,seungjin}@postech.ac.kr

Abstract

Link-based dissimilarity measures, such as shortest path or Euclidean commute time distance, base their distance on paths between nodes of a weighted graph. These measures are known to be better suited to data manifold with nonconvex-shaped clusters, compared to Euclidean distance, so that k-nearest neighbor (NN) search is improved in such metric spaces. In this paper we present a new link-based dissimilarity measure based on *minimax paths* between nodes. Two main benefits of minimax path-based dissimilarity measure are: (1) only a subset of paths is considered to make it scalable, while Euclidean commute time distance considers all possible paths; (2) it better captures nonconvex-shaped cluster structure, compared to shortest path distance. We define the total cost assigned to a path between nodes as L_p norm of intermediate costs of edges involving the path, showing that minimax path emerges from our L_p *norm over paths* framework. We also define *minimax distance* as the intermediate cost of the longest edge on the minimax path, then present a greedy algorithm to compute k smallest minimax distances between a query and N data points in $\mathcal{O}(\log N + k \log k)$ time. Numerical experiments demonstrate that our *minimax k-NN algorithm* reduce the search time by several orders of magnitude, compared to existing methods, while the quality of k-NN search is significantly improved over Euclidean distance.

Introduction

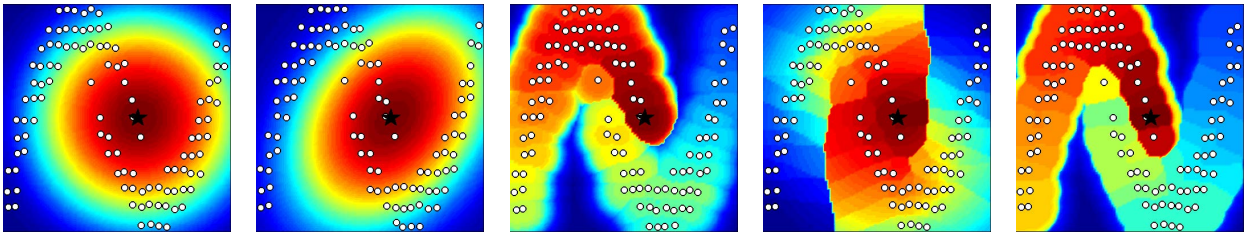
Given a set of N data points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, k-nearest neighbor (k-NN) search in metric spaces involves finding k closest points in the dataset $\mathcal{X}$ to a query $\boldsymbol{x}_q$. Dissimilarity measure defines distance d_{uv} between two data points (or nodes of a weighted graph) $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ in the corresponding metric space, and the performance of k-NN search depends on distance metric.

Euclidean distance $\|\boldsymbol{x}_u - \boldsymbol{x}_v\|_2$ is the most popular measure for k-NN search but it does not work well when data points $\mathcal{X}$ lie on a *curved manifold* with nonconvex-shaped clusters (see Fig. 1(a)). Metric learning (Xing et al. 2003; Goldberger et al. 2005; Weinberger and Saul 2009) optimizes parameters involving the Mahalanobis distance using labeled dataset, such that points in the same cluster become close together and points in the different cluster become far apart. Most of metric learning methods are limited to linear embedding, so that the nonconvex-shaped cluster structure is not well captured (see Fig. 1(b)).

Link-based (dis)similarity measures (Fouss et al. 2007; Yen, Mantrach, and Shimbo 2008; Yen et al. 2009; Mantrach et al. 2010; Chebotarev 2011) rely on paths between nodes of a weighted graph, on which nodes are associated with data points and intermediate costs (for instance Euclidean distance) are assigned to edge weights. Distance between nodes depends on the total cost that is computed by aggregating edge weights on a path connecting those nodes of interest. Total cost associated with a path is often assumed to be additive (Yen, Mantrach, and Shimbo 2008), so the aggregation reduces to summation. Dissimilarity between two nodes is calculated by integrating the total costs assigned to all possible paths between them. Such integration is often determined by computing the pseudo-inverse of the graph Laplacian, leading to Euclidean commute time distance (ECTD), regularized Laplacian kernel, and Markov diffusion kernel (see (Fouss et al. 2007; Yen, Mantrach, and Shimbo 2008; Fouss et al. 2012) and references therein), which are known to better capture the nonconvex-shaped cluster structure (see Fig. 1(c)). However, all possible paths between nodes need to be considered to compute the distances and the inversion of $N \times N$ matrix requires $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ space, so it does not scale well to the problems of interest.

Shortest path distance (Dijkstra 1959) is also a popular link-based dissimilarity measure (Tenenbaum, de Silva, and Langford 2000), where only shortest paths are considered to compute the distances between two nodes. Computational cost is reduced, but the cluster structure is not well captured when shortest path distance is used for k-NN search (see Fig. 1(d)). The distance between two nodes is computed along the shortest path only, Randomized shortest path (RSP) dissimilarity measures were proposed as a family of distance measures depending on a single parameter, which has interesting property of reducing, on one end, to the standard shortest path distance when the parameter is large, on the other hand, to the commute time distance when the parameter is near zero (Yen, Mantrach, and Shimbo 2008).

In this paper we present a new link-based k-NN search method with *minimax paths* (Pollack 1960; Gower and Ross



(a) Euclidean (b) LMNN (c) ECTD (d) Shortest path (e) Minimax distance

Figure 1: Real-world datasets usually contain nonconvex-shaped clusters. For example, two curved clusters (dots) and a query point (star) are given. Heat map shows the closeness to the query at each point, in terms of the distance measure used (best viewed in color; more reddish means closer to the query). **(a)** Euclidean distance cannot reflect the underlying cluster structure at all. **(b)** Metric learning (Weinberger and Saul 2009) cannot capture the nonconvex shapes. **(c)** Link-based distance (Fouss et al. 2007) captures the underlying cluster correctly, but requires a large amount of computation. **(d)** Shortest path distance is efficient to compute, but it is not reliable near the boundary between different clusters. **(e)** Our method is as efficient as computing the shortest path distance, while capturing the entire cluster structure correctly.

1969), which is well-known to capture the underlying cluster structure of data (e.g., Fig. 1(e)) (Kim and Choi 2007; Luo et al. 2008; Zadeh and Ben-David 2009). We develop a fast k-NN search algorithm that computes the minimax distance efficiently through the minimax paths between data points. Our method is as efficient as the shortest-path distance computation. More specifically, the overall time for k-NN search with N data points is only $\mathcal{O}(\log N + k \log k)$. In image retrieval experiments on some public image datasets, our method was several orders of magnitude faster, while achieving the comparable search quality (in terms of precision).

The main contributions of this paper are summarized:

1. We develop a novel framework for link-based (dis)similarity measures where we define the total cost (corresponding to dissimilarity) assigned to a path between nodes as L_p norm of intermediate costs of edges involving the path, showing that minimax path emerges from our L_p *norm over paths* framework. This framework has two main benefits: (1) it can reduce the number of paths considered for the link-based (dis)similarity computation, which improves the scalability greatly; (2) even using a small number of paths, it can capture any nonconvex-shaped cluster structure successfully.
2. We define *minimax distance* as the intermediate cost of the longest edge on the minimax path, then present a greedy algorithm to compute k smallest minimax distances between a query and N data points in $\mathcal{O}(\log N + k \log k)$ time, whereas the state of the arts, minimax message passing (Kim and Choi 2007) requires $\mathcal{O}(N)$ time for k-NN search.

Aggregation over Paths: L_p Norm

Link-based (dis)similarity is defined on a weighted graph, denoted by $\mathcal{G} = (\mathcal{X}, \mathcal{E})$:

- The set of nodes, $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, are associated with N data points. $\mathcal{E} = \{(i,j) \,|\, i \neq j \in \{1, \ldots, N\}\}$ is a set of edges between the nodes, excluding self-loop edges.
- The graph is usually assumed to be *sparse* such that the number of edges, $|\mathcal{E}|$, is limited to $\mathcal{O}(N)$. A well-known example is the *K-NN graph*, where an edge (i,j) exists only if $\boldsymbol{x}_i$ is one of the K nearest neighbors of $\boldsymbol{x}_j$ or $\boldsymbol{x}_j$ is one of the K nearest neighbors of $\boldsymbol{x}_i$ in the Euclidean space. The value of K is set to a small constant (usually 5-20) for ensuring sparsity.

Then a link-based (dis)similarity depends on the total cost associated with paths on the graph. In this section, we describe our L_p norm approach to the cost aggregation over paths to compute the total cost along paths.

Let $\boldsymbol{a} = (a_0, a_1, \ldots, a_m)$ be a *path* with m hops, connecting the nodes $\boldsymbol{x}_{a_0}, \boldsymbol{x}_{a_1}, \ldots, \boldsymbol{x}_{a_m} \in \mathcal{X}$ through the consecutive edges $\{(a_0, a_1), (a_1, a_2), \ldots, (a_{m-1}, a_m)\} \subseteq \mathcal{E}$. We denote a set of paths with m hops between an initial node $\boldsymbol{x}_u$ and a destination node $\boldsymbol{x}_v$ by

$$\mathcal{A}_{uv}^m = \Big\{ \boldsymbol{a} \,\Big|\, a_0 = u,\ a_m = v,\ (a_\ell, a_{\ell+1}) \in \mathcal{E},\ a_\ell \neq v,\quad \forall \ell = 0, \ldots, m-1 \Big\}, \tag{1}$$

and denote a set of all possible paths between $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ by

$$\mathcal{A}_{uv} = \cup_{m=1}^{\infty} \mathcal{A}_{uv}^m. \tag{2}$$

A weight $c(i,j)$ is assigned to each edge (i,j), representing the intermediate cost of following edge (i,j), usually defined as the Euclidean distance, i.e., $c(i,j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$. We define the total cost associated with a path $\boldsymbol{a}$ as

$$\|c(\boldsymbol{a})\|_p = \left[\sum_{\ell} c(a_\ell, a_{\ell+1})^p \right]^{1/p}, \tag{3}$$

where the L_p norm ($0 < p < \infty$) is applied. Then we define our link-based similarity s_{uv} between $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ as

$$s_{uv} = \sum_{\boldsymbol{a} \in \mathcal{A}_{uv}} \exp\left\{ -\frac{1}{T} \|c(\boldsymbol{a})\|_p \right\}, \tag{4}$$

where $T > 0$ is *temperature* which controls how rapidly the similarity falls off with distance.

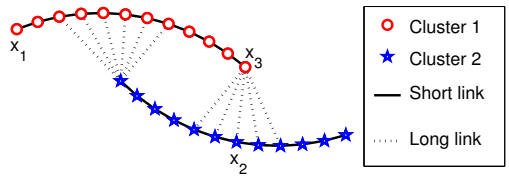


Figure 2: An example to show the importance of long edges. Two distinct clusters and three data points are given, where $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$ are in the same cluster and $\boldsymbol{x}_2$ belongs to the opposite cluster.

$\|c(\boldsymbol{a}_1)\|_1 = 11$
$\|c(\boldsymbol{a}_1)\|_2 = 31^{1/2} = 5.56$
$\|c(\boldsymbol{a}_1)\|_3 = 131^{1/3} = 5.08$

(a) A loose path, $\boldsymbol{a}_1$

$\|c(\boldsymbol{a}_2)\|_1 = 12$
$\|c(\boldsymbol{a}_2)\|_2 = 24^{1/2} = 4.89$
$\|c(\boldsymbol{a}_2)\|_3 = 48^{1/3} = 3.63$

(b) A compact path, $\boldsymbol{a}_2$

Figure 3: An illustrative example that increasing p makes the L_p norm of a "loose path" ($\boldsymbol{a}_1$) greater than the L_p norm of a "compact path" ($\boldsymbol{a}_2$). The L_p norm of a path converges to the longest edge, so that the L_∞ norm of a path containing a longer edge is always greater than that of any other path containing shorter edges only.

The parameter $0 < p < \infty$ in Eq. (4) controls the *importance of longer edges* in a path. As p increases, the contributions of longer edges in a path $\boldsymbol{a}$ to the L_p norm $\|c(\boldsymbol{a})\|_p$ become more dominant than the contributions of shorter edges. Fig. 2 shows an example why we introduce this parameter. $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$ are in the same cluster, but they lie farther apart than $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ in the Euclidean space. We want to make the similarity $s_{1,3}$ larger than $s_{2,3}$, but it looks difficult because the path between $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$ is quite long compared to the direct edge between $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$. One can solve this problem by focusing only on the *longest edge* in each path. Connecting data points in the same cluster can be done through short edges only (e.g., the path between $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$ in Fig. 2) because they are surrounded with a dense region. By contrast, connecting data points in distinct clusters should involve one or more long edges in order to cross sparse regions between the clusters (e.g., any possible path between $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ in Fig. 2). That is, the existence of a path that consists solely of short edges can be a strong clue whether two data points are in the same cluster or not. In Eq. (4), increasing p makes the L_p norm of a path containing long edges (say "loose path") greater than the L_p norm of a path containing short edges only (say "compact path") (e.g., Fig. 3). That is, the exponential terms of loose paths become smaller than the terms of compact paths, making the similarity s_{uv} within the same cluster larger than the similarity between different clusters.

The parameter $0 < T < \infty$ in Eq. (4), so-called *temperature*, determines a favor toward a path having a small L_p norm. A larger value of T makes the exponential terms in Eq. (4) closer to 1, leading all paths to have more uniform weights for computing s_{uv}, regardless of their L_p norms.

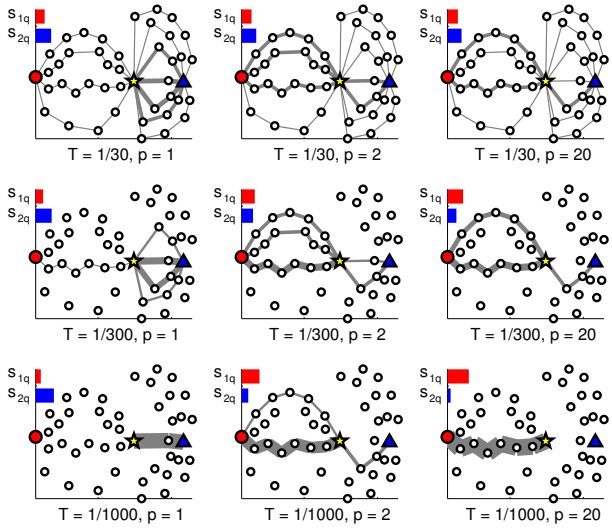


Figure 4: In the example, a query point $\boldsymbol{x}_q$ (star) is given near the boundary of the circle-shaped cluster. $\boldsymbol{x}_1$ (red circle) lies in the same cluster, but far from the query. $\boldsymbol{x}_2$ (blue triangle) lies are in a distinct, crescent-shaped cluster, but close to the query. There are four paths between $\boldsymbol{x}_1$ and $\boldsymbol{x}_q$ and seven paths between $\boldsymbol{x}_2$ and $\boldsymbol{x}_q$, where the line width of each path $\boldsymbol{a}$ represents its weight $\exp(-\frac{1}{T}\|c(\boldsymbol{a})\|_p)$ for computing the similarity as in Eq. (4) (thicker lines are paths having larger weights; no line means negligible). Then the link-based similarities s_{1q} (red horizontal bar) and s_{2q} (blue horizontal bar) are computed by summing weights over the paths between $\boldsymbol{x}_1$ and $\boldsymbol{x}_q$ and between $\boldsymbol{x}_2$ and $\boldsymbol{x}_q$, respectively.

By contrast, A smaller value of T increases the gap between the exponential terms of smaller L_p norms and the terms of larger L_p norms, so that makes paths having smaller L_p norms become more dominant for computing s_{uv} while paths having larger L_p norms become less significant. Thus, *a smaller value of T reduces the number of "effective paths" in $\mathcal{A}_{uv}$*, where the effective paths have sufficiently small L_p norms (so that have significant weights for computing s_{uv}) compared to the remaining paths.

Consequently, two parameters in our similarity measure, T and p, play a key role:

1. A larger value of p makes the weights assigned to paths lying within the same cluster larger than the weights assigned to paths lying between different clusters, so that the similarities between data points within the same cluster can be kept to be large regardless of their Euclidean distances (e.g., Fig. 2).
2. T reduces *the number of paths* considered for computing the link-based similarity. A small value of T allows only a small number of "effective paths" to have meaningful contributions to the similarity, and all the other paths become relatively negligible. We introduce this parameter in order to *improve the efficiency* of computing link-based similarities.

Fig. 4 illustrates the changes of the similarities between data points within the same cluster ($\boldsymbol{x}_q$ and $\boldsymbol{x}_1$) and between different clusters ($\boldsymbol{x}_q$ and $\boldsymbol{x}_2$) as T decreases and p increases. When p is small (the leftmost column in Fig. 4), some paths lying between different clusters have small L_p norms (due to their small total edge costs), so they have larger weights and more and more dominant effects on the similarities as T increases (from top to bottom rows in Fig 4). As p increases (from left to right columns in Fig 4), however, the L_p norms of those paths become larger (due to the long edges between two clusters) than the L_p norms of the paths lying in the same cluster, resulting that the paths lying in the same cluster become dominant.

Note that Eq. (1) allows the repetition of the nodes in a path (except for the destination node, $\boldsymbol{x}_v$, which is also called an *absorbing node*). The repetition makes $\mathcal{A}_{uv}^m$ and $\mathcal{A}_{uv}$ infinite, but in the remaining of this paper, we will show that the aggregation over paths, Eq. (4), is still finite in some cases of interest (specifically, Eq. (5) and (14)).

Minimax Distance

In the previous section, we show that setting large p and small T obtains a desirable link-based similarity. Now we consider the extreme case, i.e., $T \to 0$ and $p \to \infty$.

When $T \to 0$, all exponential terms in Eq. (4) become relatively negligible compared to the exponential term of the *smallest* L_p norm, i.e., Eq. (4) converges to

$$s_{uv} \;\to\; \max_{\boldsymbol{a}\in\mathcal{A}_{uv}} \; \exp\left\{-\frac{1}{T}\left\|c(\boldsymbol{a})\right\|_p\right\}. \tag{5}$$

Essentially, using Eq. (5) as a link-based similarity is equivalent to using the smallest L_p norm as the *dissimilarity*, i.e.,

$$d_{uv}^{(p)} = \min_{\boldsymbol{a}\in\mathcal{A}_{uv}} \left\|c(\boldsymbol{a})\right\|_p, \tag{6}$$

which is computed along *only one path* between $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ that has the smallest L_p norm.

When $p \to \infty$, the L_p norm of a path converges to the intermediate cost of the *longest edge*, regardless of any other edges in the path:

$$\left\|c(\boldsymbol{a})\right\|_\infty = \max_\ell \; c(a_\ell, a_{\ell+1}). \tag{7}$$

Plugging the infinity norm into Eq. (6), we derive the *minimax distance*, denoted by d_{uv}:

$$d_{uv} = \min_{\boldsymbol{a}\in\mathcal{A}_{uv}} \left\|c(\boldsymbol{a})\right\|_\infty = \min_{\boldsymbol{a}\in\mathcal{A}_{uv}} \left(\max_\ell \; c(a_\ell, a_{\ell+1})\right), \tag{8}$$

and the path between $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ having the smallest infinity norm is called the *minimax path*. In other words, minimax distance is the longest edge cost on the minimax path:

$$d_{uv} = \max_\ell c(a_\ell^*, a_{\ell+1}^*), \tag{9}$$

where $\boldsymbol{a}^*$ denotes the minimax path between $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$. Needless to say, if $u = v$ then $d_{uu} = 0$.

Dynamic Programming

Minimax distance is computed along only one path, the minimax path between nodes. Using this property, minimax distance can be computed more efficiently than the existing link-based (dis)similarities. Now we derive an efficient computation method for the minimax distance.

First, we reformulate Eq. (8) as a dynamic programming model. Given $\mathcal{A}_{uv} = \{(a_0 = u, \ldots, a_m = v) \text{ for } m \geq 1\}$, suppose that we already know the minimax path and its longest edge (i.e., minimax distance) between $\boldsymbol{x}_{a_1}$ and $\boldsymbol{x}_v$, for all $a_1 = r$ such that $(u, r) \in \mathcal{E}$. Then, for computing d_{uv}, we only need to (1) compute the longest edge cost, $\max(c(u, r), d_{rv})$, for each path adding the edge (u, r) to the minimax path $(r, \ldots, v)$, and then (2) choose the smallest one among them. Hence, Eq. (8) can be rewritten as the following recursive form:

$$d_{uv} = \min_{\boldsymbol{x}_r\in\mathcal{N}_u} \Big(\max(c(u, r), d_{rv})\Big), \tag{10}$$

where $\mathcal{N}_u = \{\boldsymbol{x}_r \mid (u, r) \in \mathcal{E}\}$.

Now we derive some useful propositions for computing Eq. (10) efficiently:

Proposition 1 *Let $\boldsymbol{a}^*$ be the minimax path from $\boldsymbol{x}_u$ to $\boldsymbol{x}_v$, i.e., $\boldsymbol{a}^* = (a_0^* = u, \ldots, a_m^* = v)$ for some m. Then, $d_{rv} \leq d_{uv}$ for all subsequent nodes $r \in \{a_1^*, \ldots, a_m^*\}$.*

Proof Suppose that $d_{rv} > d_{uv}$ for some $a_\ell^* = r$. From Eq. (8), we have $\max_\ell c(a_\ell, a_{\ell+1}) \geq d_{rv} > d_{uv}$ for all paths $\boldsymbol{a} \in \mathcal{A}_{rv}$. Since the subsequence $(a_\ell^* = r, \ldots, a_m^* = v)$ of $\boldsymbol{a}^*$ is also in $\mathcal{A}_{rv}$, the longest edge in the subsequence should be greater than d_{uv}. However, this is contradictory to the definition of the minimax distance (Eq. (9)).

Proposition 2 *For $\boldsymbol{x}_r \in \mathcal{N}_u$, the term $\max(c(u, r), d_{rv})$ in Eq. (10) is negligible for computing d_{uv} if $d_{rv} > d_{uv}$.*

Proof From Proposition 1, $\boldsymbol{x}_r$ is not involved in the minimax path from $\boldsymbol{x}_u$ to $\boldsymbol{x}_v$, so the term is negligible. Also, if $d_{rv} > d_{uv}$, then $\max(c(u, r), d_{rv}) > d_{uv}$, meaning that the term has definitely no effect on d_{uv}.

Proposition 3 *For computing d_{uv} using Eq. (10), we need to know only d_{rv} such that $d_{rv} \leq d_{uv}$ and $(u, r) \in \mathcal{E}$.*

Proof From Proposition 2, d_{uv} is independent of any d_{rv} such that $d_{rv} > d_{uv}$, i.e., only the case $d_{rv} \leq d_{uv}$ among $\boldsymbol{x}_r \in \mathcal{N}_u$ is possible to affect d_{uv}.

Greedy Algorithm

For k-NN search, we compute the minimax distance d_{uq} between each data point $\boldsymbol{x}_u \in \mathcal{X}$ and a given query point $\boldsymbol{x}_q$. Using Eq. (10) and the propositions, now we develop an efficient algorithm for computing the minimax distances to $\boldsymbol{x}_q$ in k-NN search.

Before deriving the algorithm, we need to consider the case when the query is a new, unseen data point (i.e., $\boldsymbol{x}_q \notin \mathcal{X}$, called *out-of-sample*), which is usual in k-NN search. Since $\boldsymbol{x}_q \notin \mathcal{X}$, there is no predefined edge from/to $\boldsymbol{x}_q$ in the graph $\mathcal{G} = (\mathcal{X}, \mathcal{E})$, nor path between $\boldsymbol{x}_q$ and any $\boldsymbol{x}_u \in \mathcal{X}$. Thus, we should first connect $\boldsymbol{x}_q$ to the graph by adding new edges (q, j) for some $\boldsymbol{x}_j \in \mathcal{X}$. For a *$K$-NN graph* (described

in the beginning of the previous section), $\boldsymbol{x}_q$ is connected with its K-NNs among $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ in terms of the Euclidean distance, which can be done in $\mathcal{O}(\log N)$ time by space partitioning (Beis and Lowe 1997; Samet 2005).

Now we derive the following theorem that implies "*for computing the $(t+1)$-th smallest minimax distance to $\boldsymbol{x}_q$, we need to know only t smallest minimax distances to $\boldsymbol{x}_q$*":

Theorem 1 *Let $\boldsymbol{x}_u$ be the $(t+1)$-th NN of $\boldsymbol{x}_q$ in terms of the minimax distance. Then computing d_{uq} using Eq. (10) needs only the terms $\max(c(u,r), d_{rq})$ such that $(u,r) \in \mathcal{E}$ and $\boldsymbol{x}_r \in \{\boldsymbol{x}_q, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}$, where $\boldsymbol{x}^{(t)}$ denotes the t-th NN of $\boldsymbol{x}_q$ in terms of the minimax distance.*

Proof From Proposition 3, computing d_{uq} requires d_{rq} such that $d_{rq} \leq d_{uq}$ and $(u,r) \in \mathcal{E}$. Since d_{uq} is the $(t+1)$-th smallest minimax distance to $\boldsymbol{x}_q$, such $\boldsymbol{x}_r$ should be the query $\boldsymbol{x}_q$ ($d_{qq} = 0$) or one of the 1st, ..., t-th NNs of $\boldsymbol{x}_q$.

Theorem 1 leads us to develop an *incremental algorithm* for k-NN search, as shown in Algorithm 1. The source code, programmed in C and MATLAB, is publicly available at http://home.postech.ac.kr/∼fenrir/mm_knn.html. Below we list some notes for easier understanding of the algorithm:

- When a given query is out-of-sample, the algorithm first connects $\boldsymbol{x}_q$ with some nodes $\boldsymbol{x}_r \in \mathcal{X}$ (line 2, assuming that $\mathcal{G}$ is a K-NN graph).
- We introduce a set of *candidates*, $\boldsymbol{x}_u \in \mathcal{Q}$, which are connected with one or more of $\{\boldsymbol{x}_q, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}$ so that can only be expected to be the $(t+1)$-th NN. The algorithm searches $\mathcal{Q}$ instead of the whole set $\mathcal{X}$, thereby reducing the computational cost of k-NN search greatly.
- For each candidate $\boldsymbol{x}_u \in \mathcal{Q}$, the algorithm maintains and updates the *estimate* of d_{uq}, denoted by $\widehat{d}_{uq}$. At each iteration t, the term $\max(c(u,r), d_{rq})$ of the t-th NN $\boldsymbol{x}_r$ is applied to every $\widehat{d}_{uq}$ such that $(u,r) \in \mathcal{E}$ (line 5-9).
- The update procedure (line 5-9) *monotonically reduces* $\widehat{d}_{uq}$. At iteration t, each $\widehat{d}_{uq}$ becomes the minimum among $\max(c(u,r), d_{rq})$ such that $(u,r) \in \mathcal{E}$ and $\boldsymbol{x}_r \in \{\boldsymbol{x}_q, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}$. From Theorem 1, $\widehat{d}_{uq}$ becomes the exact $(t+1)$-th smallest minimax distance if $\boldsymbol{x}_u$ is the $(t+1)$-th NN. Otherwise, $\widehat{d}_{uq} \geq d_{uq}$ and will monotonically decrease until $\widehat{d}_{uq} = d_{uq}$.
- After the update procedure, we have (1) the exact estimate of the $(t+1)$-th smallest minimax distance and (2) the estimates of the $(t+2)$-th, $(t+3)$-th, ... smallest minimax distances that are clearly never smaller than the $(t+1)$-th smallest minimax distance. Thus, the algorithm can obtain the $(t+1)$-th NN by greedily choosing the smallest estimate $\widehat{d}_{uq}$ among $\boldsymbol{x}_u \in \mathcal{Q}$ (line 10-11).

Time Complexity

Now we show that the time complexity of our algorithm for k-NN search is $\mathcal{O}(\log N + k \log k)$ when the graph $\mathcal{G} = (\mathcal{X}, \mathcal{E})$ is *sparse* such that $|\mathcal{E}| = \mathcal{O}(N)$ or $\frac{1}{N}\sum_{i=1}^{N} |\mathcal{N}_i| = \mathcal{O}(1)$ (e.g., a K-NN graph with a small constant K).

First, we summarize the computation time for each step:

Algorithm 1 Minimax k-NN search (MM-kNN)

Input: a query point $\boldsymbol{x}_q$, the number of NNs k, and a graph $\mathcal{G} = (\mathcal{X}, \mathcal{E})$ where $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$

Output: k-NNs of $\boldsymbol{x}_q$ in terms of the minimax distance, denoted by $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(k)}\} \subset \mathcal{X}$

1: **if** $\boldsymbol{x}_q \notin \mathcal{X}$ ("out-of-sample") **then** compute
2: | $\mathcal{N}_q = \{K\text{-NNs of } \boldsymbol{x}_q \text{ w.r.t. Euclidean distance}\}$
3: Initialize $\mathcal{Q} \leftarrow \{\}$, $d_{qq} \leftarrow 0$, and $r \leftarrow q$
4: **for** $t = 0$ to k **do** /* $\boldsymbol{x}_r$ denotes $\boldsymbol{x}^{(t)}$ */
5: | **for each** $\boldsymbol{x}_u \in \mathcal{N}_r$ where $\boldsymbol{x}_u \neq \boldsymbol{x}_q, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}$ **do**
6: | | **if** $\boldsymbol{x}_u \notin \mathcal{Q}$ **then**
7: | | | $\widehat{d}_{uq} \leftarrow \max(c(u,r), d_{rq})$; INSERT $\boldsymbol{x}_u$ into $\mathcal{Q}$
8: | | **else**
9: | | | UPDATE: $\widehat{d}_{uq} \leftarrow \min\left(\widehat{d}_{uq},\ \max(c(u,r), d_{rq})\right)$
10: | EXTRACT-MIN $\boldsymbol{x}_r$ from $\mathcal{Q}$: $r = \arg\min_{u:\boldsymbol{x}_u \in \mathcal{Q}} \widehat{d}_{uq}$
11: | Set $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}_r$ and $d_{rq} \leftarrow \widehat{d}_{rq}$

- Line 2: $\mathcal{O}(\log N)$ time in order to find the K-NNs by space partitioning (Beis and Lowe 1997; Samet 2005).
- Line 3: a constant time.
- Line 4-11 repeats $k+1$ times, where
  - Line 5-9 repeats at most $|\mathcal{N}_r|$ time, where UPDATE and INSERT take a constant amortized time in a *Fibonacci heap* (Fredman and Tarjan 1987) of $\{\widehat{d}_{uq} \mid \boldsymbol{x}_u \in \mathcal{Q}\}$;
  - Line 10: $\mathcal{O}(\log |\mathcal{Q}|)$ time using the Fibonacci heap.

Since $|\mathcal{N}_r|$ is bounded by a constant in average, line 5-9 requires a constant time. Since the size of $\mathcal{Q}$ increases at most $|\mathcal{N}_r| - 1$ (i.e., bounded by a constant; -1 is due to EXTRACT-MIN) in each iteration, the size $|\mathcal{Q}|$ can be at most $\mathcal{O}(k)$ until the k-th iteration. Hence, line 10 requires at most $\mathcal{O}(\log k)$ time. Finally, line 11 takes a constant time.

Consequently, the computation time for line 4-11 is $\mathcal{O}(k \log k)$. Adding $\mathcal{O}(\log N)$ time taken at line 2, finally we have the overall time complexity of $\mathcal{O}(\log N + k \log k)$.

Related Work

In our L_p norm framework, we can also derive two existing link-based (dis)similarities, (1) a link-based similarity (Yen, Mantrach, and Shimbo 2008) that is computed along all possible paths and (2) the shortest path distance.

From the theory of absorbing Markov chains, the aggregation of all possible paths between nodes can be formulated as the (pseudo-)inverse of the graph Laplacian matrix or similar variants, and many existing link-based (dis)similarities come from the theory (e.g., the (Euclidean) commute time distance, regularized Laplacian kernel, Markov diffusion kernel, and so on; see (Fouss et al. 2007; Yen, Mantrach, and Shimbo 2008; Fouss et al. 2012) and references therein). In Appendix, we derive one of the (dis)similarities, proposed in (Yen, Mantrach, and Shimbo 2008), from our L_p norm framework with $p = 1$. The derivation clearly shows the reason why the existing link-based (dis)similarities, which are computed along all possible paths, require high computational cost.

Our L_p norm framework also derives the shortest path distance. When $p = 1$ and $T \to 0$ in Eq. (4), we have the smallest L_1 norm as the dissimilarity (i.e., $p = 1$ in Eq. (6)):

$$d_{uv}^{(1)} = \min_{\boldsymbol{a} \in \mathcal{A}_{uv}} \sum_{\ell} c(a_\ell, a_{\ell+1}), \tag{11}$$

which is the shortest path distance, and the path having the smallest L_1 norm is the shortest path between $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$. Similar to our algorithm for the minimax distance, Eq. (11) can be computed efficiently along the shortest path only. However, the shortest path distance cannot capture the underlying cluster structure of data well. When a query point is given near a boundary of a cluster, k-NN search using the shortest path distance prefers data points in different clusters. Some examples are shown in Fig. 1(d) and the bottom-left plot in Fig. 4 (when $p = 1$ and $T \approx 0$).

Our L_p norm framework provides a solution to overcome the limitations of the existing link-based (dis)similarities. First, we introduce the temperature parameter, T, where a smaller value of T reduces the number of effective paths for the link-based (dis)similarity computation, improving the scalability. Second, we define the total cost of each path as the L_p norm, where a larger value of p makes paths lying within the same cluster more effective. Finally, we obtain the minimax distance, which is computed very efficiently along only one path (by setting $T \to 0$), while capturing the underlying cluster structure better than the shortest path distance (by setting $p \to \infty$).

Experiments

We compared our minimax k-NN search method (MM-kNN) to the following (dis)similarity measures:

- Euclidean distance (Euclidean).
- The shortest path distance (Shortest) with Dijkstra's algorithm (Dijkstra 1959).
- The pseudo-inverse of the Laplacian matrix ($\boldsymbol{L}^+$), which is a fundamental link-based similarity for many other existing link-based (dis)similarities (Fouss et al. 2007).

The comparison to the first two measures aims to show that MM-kNN obtains significantly better k-NN search quality (in terms of precision). The comparison to $\boldsymbol{L}^+$ aims to show that our MM-kNN is much faster ($\mathcal{O}(N)$ versus $\mathcal{O}(\log N + k \log k)$), while obtaining comparable k-NN search quality. All methods were implemented in MATLAB, and run on an Intel Core2 Duo 2.4GHz CPU with 4GB memory. For $\boldsymbol{L}^+$, we used a recent method of partial computation (Yen, Mantrach, and Shimbo 2008). For all link-based (dis)similarities, we constructed the K-NN graph $\mathcal{G} = (\mathcal{X}, \mathcal{E})$ with $K = 20$ for each dataset.

First, we compared the methods on the Two-moon toy datasets (Fig. 5). Under the varying size of the dataset as $N = 10^3, \ldots, 10^6$, Fig. 6 clearly shows that the k-NN search time of MM-kNN and Shortest grows very slowly, compared to the time of $\boldsymbol{L}^+$. For varying k, the time of MM-kNN and Shortest grows slightly faster than $\mathcal{O}(k)$, which is consistent with the time complexity w.r.t. k, $\mathcal{O}(k \log k)$.

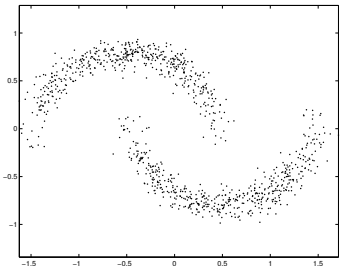


Figure 5: Two-moon dataset.

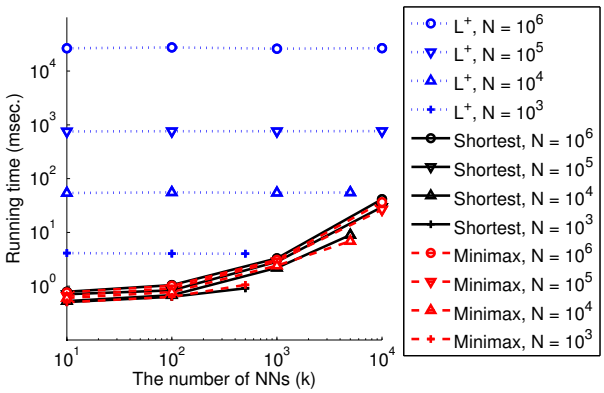


Figure 6: The k-NN search time of each method on Two-moon datasets with varying N and k.

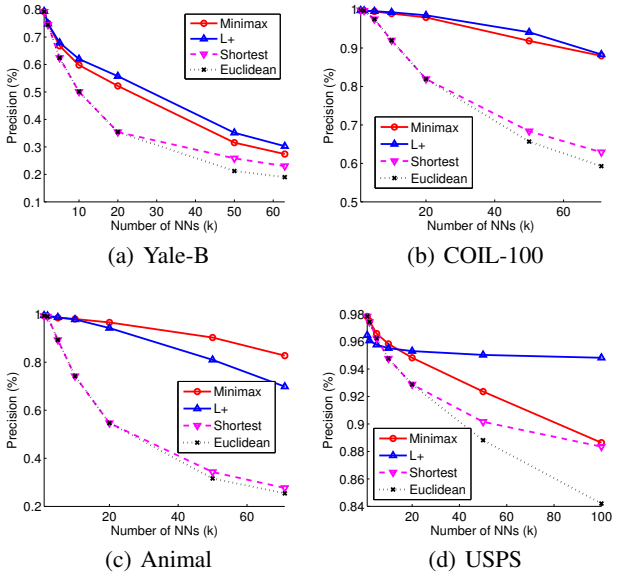
We also considered the content-based image retrieval task, performing k-NN search (using MM-kNN, $\boldsymbol{L}^+$, Shortest, and Euclidean) on the following image databases:

- Yale-B database (Yale-B, http://vision.ucsd.edu/∼leekc/ExtYaleDatabase/Yale Face Database.htm) (Georghiades, Belhumeur, and Kriegman 2001; Lee, Ho, and Kriegman 2005) contains images of 10 people with various poses and illumination conditions. We chose the images of all 64 different illumination conditions of the frontal faces (i.e. $N = 640$).
- Columbia University Image Library database (COIL-100, http://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php) (Nene, Nayar, and Murase 1996) consists of color images of 100 objects, given 72 different views for each object (i.e. $N = 7200$).
- Animal data (Animal, http://ml.nec-labs.com/download/data/videoembed/) (Mobahi, Collobert, and Weston 2009) is a database of $N = 4371$ color images of 60 toy animals (such as horse and several kinds of birds). There are about 72 images of different views per animal.
- USPS database (USPS) contains $N = 7291$ handwritten digit images. It is popularly used for pattern recognition and content-based image retrieval.

We performed N experiments of k-NN search ($k = 63$ on Yale-B; $k = 71$ on COIL-100 and Animal; $k = 100$ on USPS) where the i-th image in a database was used as a

Table 1: Average computation time per query (msec.), with 1 standard deviation

	MM-kNN	Shortest	$\boldsymbol{L}^+$
Yale-B	0.23±0.06	**0.20±0.11**	5.10±2.37
COIL-100	**0.17±0.12**	0.20±0.17	131.1±8.91
Animal	**0.56±0.35**	0.85±0.48	151.6±25.2
USPS	**0.73±0.47**	0.80±0.59	376.5±39.1



(a) Yale-B (b) COIL-100

(c) Animal (d) USPS

Figure 7: Average precision of k-NN search using some (dis)similarity measures.

query for the i-th experiment. Then we averaged the precisions (the proportion of retrieved images in the same class with the query image) and the computation times for k-NN search over all N experiments.

Table 1 summarizes the average running time of k-NN search using each method on each dataset. MM-kNN and Shortest clearly outperformed $\boldsymbol{L}^+$ up to several orders of magnitude, since they consider only one path (minimax path and shortest path, respectively) from each data point to a query point. Fig. 7 shows the average precision of k-NN search with each (dis)similarity measure. MM-kNN achieved significantly better precision than the shortest path distance and Euclidean distance. It means that the minimax path can capture the underlying cluster structure of data better than the shortest path. While MM-kNN was somewhat poorer than $\boldsymbol{L}^+$ in precision, MM-kNN can be a scalable alternative to $\boldsymbol{L}^+$ and other similar existing link-based (dis)similarities.

Conclusions

We have addressed k-NN search with the minimax distance. We showed that the minimax distance is more efficient to compute than the existing link-based measures, since it ignores any other paths between nodes except for the minimax path. We also showed that the minimax path better captures the latent cluster structure of data than the shortest path. We proposed a novel efficient algorithm for k-NN search that greedily computes the 1st, 2nd, ..., and the k-th smallest minimax distances to a given query point. We proved that the time complexity of our method is $\mathcal{O}(\log N + k \log k)$, which is far faster than computing the existing link-based (dis)similarities. We applied our method to image retrieval tasks, and showed that our method can be a scalable alternative to the existing link-based (dis)similarity measures.

Acknowledgments

This work was supported by National Research Foundation (NRF) of Korea (2012-0005032), NIPA ITRC Support Program (NIPA-2013-H0301-13-3002), the Converging Research Center Program funded by MEST (2012K001343), NIPA and Microsoft Research Asia Joint Support Program, POSTECH Rising Star Program, and NRF World Class University Program (R31-10100).

Appendix: L_p Norm Aggregation with $p = 1$

When $p = 1$ in Eq. (4), each exponential term of the L_p norm can be decomposed into the product of the exponential factors:

$$s_{uv} = \sum_{\boldsymbol{a}\in\mathcal{A}_{uv}} \exp\left\{-\frac{1}{T}\sum_{\ell} c(a_\ell, a_{\ell+1})\right\} = \sum_{\boldsymbol{a}\in\mathcal{A}_{uv}} \prod_{\ell} \exp\left\{-\frac{1}{T} c(a_\ell, a_{\ell+1})\right\}. \quad (12)$$

Eq. (12) can be represented as a matrix form. Consider an $N \times N$ matrix $\boldsymbol{W}$ (or $(N+1)\times(N+1)$ when a given query $\boldsymbol{x}_q$ is out-of-sample and is included as $\boldsymbol{x}_q = \boldsymbol{x}_{N+1}$) whose (i,j)-th entry is $[\boldsymbol{W}]_{ij} = \exp\left\{-\frac{1}{T}c(i,j)\right\}$, except that the v-th row $[\boldsymbol{W}]_{v1}, \ldots, [\boldsymbol{W}]_{vN} = 0$ in order to make $\boldsymbol{x}_v$ an absorbing node. Then, it is well known that the (u,v)-th entry of $\boldsymbol{W}^m$ ($\boldsymbol{W}$ to the power m) is the sum of the products in Eq. (12) over all m-hop paths between $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$, i.e.,

$$[\boldsymbol{W}^m]_{uv} = \sum_{\boldsymbol{a}\in\mathcal{A}^m_{uv}} \prod_{\ell} \exp\left\{\frac{1}{T} c(a_\ell, a_{\ell+1})\right\}, \quad (13)$$

e.g., $[\boldsymbol{W}^2]_{uv} = \sum_r \exp(-\frac{1}{T}c(u,r)) \exp(-\frac{1}{T}c(r,v))$ summing over all 2-hop paths $\boldsymbol{a} = (u,r,v) \in \mathcal{A}^2_{uv}$.

Since $\mathcal{A}_{uv} = \cup_{m=1}^{\infty} \mathcal{A}^m_{uv}$, Eq. (12) can be rewritten as $s_{uv} = \sum_{m=1}^{\infty} [\boldsymbol{W}^m]_{uv}$. When the largest absolute eigenvalue of $\boldsymbol{W}$ is less than 1, the series of powers converges as $\sum_{m=1}^{\infty} \boldsymbol{W}^m = (\boldsymbol{I} - \boldsymbol{W})^{-1} - \boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix. Thus, we have

$$s_{uv} = [(\boldsymbol{I} - \boldsymbol{W})^{-1} - \boldsymbol{I}]_{uv}, \quad (14)$$

which is exactly the same form as a partition function proposed in (Yen, Mantrach, and Shimbo 2008). Similar to the pseudo-inverse of the graph Laplacian matrix (Fouss et al. 2007), aggregating all possible paths between nodes requires the inversion of the $N \times N$ matrix, $(\boldsymbol{I} - \boldsymbol{W})^{-1}$, which is

infeasible when N is large because of the high computational cost of $\mathcal{O}(N^3)$ in time and $\mathcal{O}(N^2)$ in space. For k-NN search, however, we need only the q-th column in Eq. (14), denoted by $\boldsymbol{l}_q = [s_{1q}, \ldots, s_{Nq}]^\top$, which can be computed by solving the following linear system:

$$(\boldsymbol{I} - \boldsymbol{W})^\top \boldsymbol{l}_q = \boldsymbol{e}_q, \tag{15}$$

where $[\boldsymbol{e}_q]_q = 1$ and $[\boldsymbol{e}_q]_i = 0$ for all $i \neq q$. The computation time can be reduced up to $\mathcal{O}(N)$, but it is still not sufficiently fast for real-time search in large-scale applications.

References

Beis, J. S., and Lowe, D. G. 1997. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 1000–1006. San Juan, Puerto Rico: IEEE Computer Society.

Chebotarev, P. 2011. A class of graph-geodetic distances generalizing the shortest-path and the resistance distances. *Discrete Applied Mathematics* 159(5):295–302.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271.

Fouss, F.; Pirotte, A.; Renders, J. M.; and Saerens, M. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. Knowledge and Data Engineering* 19(3):355–369.

Fouss, F.; Francoisse, K.; Yen, L.; Pirotte, A.; and Saerens, M. 2012. An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification. *Neural Networks* 31:53–72.

Fredman, M. L., and Tarjan, R. E. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM* 34(3):596–615.

Georghiades, A. S.; Belhumeur, P. N.; and Kriegman, D. J. 2001. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Analysis and Machine Intelligence* 23(6):643–660.

Goldberger, J.; Roweis, S.; Hinton, G.; and Salakhutdinov, R. 2005. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, 513–520. Vancouver, Canada: MIT Press.

Gower, J. C., and Ross, G. J. S. 1969. Minimum spanning tree and single-linkage cluster analysis. *Applied Statistics* 18:54–64.

Kim, K.-H., and Choi, S. 2007. Neighbor search with global geometry: A minimax message passing algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, 401–408. Corvallis, Oregon, USA: ACM.

Lee, K.-C.; Ho, J.; and Kriegman, D. 2005. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Trans. Pattern Analysis and Machine Intelligence* 27(5):1–15.

Luo, H.; Niu, C.; Shen, R.; and Ullrich, C. 2008. A collaborative filtering framework based on both local user similarity and global user similarity. *Machine Learning* 72(3):231–245.

Mantrach, A.; Yen, L.; Callut, J.; Francoisse, K.; Shimbo, M.; and Saerens, M. 2010. The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph. *IEEE Trans. Pattern Analysis and Machine Intelligence* 32(6):338–361.

Mobahi, H.; Collobert, R.; and Weston, J. 2009. Deep learning from temporal coherence in video. In *Proceedings of the International Conference on Machine Learning (ICML)*, 737–744. Montreal, Canada: ACM.

Nene, S. A.; Nayar, S. K.; and Murase, H. 1996. Columbia object image library (COIL-100). Technical Report CUCS-006-96, Department of Computer Science, Columbia University.

Pollack, M. 1960. The maximum capacity through a network. *Operation Research* 8(5):733–736.

Samet, H. 2005. *Foundations of Multidimensional and Metric Data Structures*. San Francisco, California, USA: Morgan Kaufmann Publishers Inc.

Tenenbaum, J. B.; de Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290:2319–2323.

Weinberger, K. Q., and Saul, L. K. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research* 10:207–244.

Xing, E. P.; Ng, A. Y.; Jordan, M. I.; and Russel, S. 2003. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, 505–512. Vancouver, Canada: MIT Press.

Yen, L.; Fouss, F.; Decaestecker, C.; Francq, P.; and Saerens, M. 2009. Graph nodes clustering with the sigmoid commute-time kernel: A comparative study. *Data & Knowledge Engineering* 68(3):338–361.

Yen, L.; Mantrach, A.; and Shimbo, M. 2008. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 785–793. Las Vegas, Nevada, USA: ACM.

Zadeh, R. B., and Ben-David, S. 2009. A uniqueness theorem for clustering. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI)*, 639–646. Montreal, Canada: AUAI Press.