Stochastic k-Neighborhood Selection for Supervised and Unsupervised Learning

Daniel Tarlow[†] DTARLOW@MICROSOFT.COM
Microsoft Research Cambridge

Kevin Swersky, Laurent Charlin {KSWERSKY, LCHARLIN}@CS.TORONTO.EDU
University of Toronto

Ilya Sutskever[†] ILYASU@GOOGLE.COM
Google Inc.

Richard S. Zemel ZEMEL@CS.TORONTO.EDU
University of Toronto

Abstract

Neighborhood Components Analysis (NCA) is a popular method for learning a distance metric to be used within a k-nearest neighbors (kNN) classifier. A key assumption built into the model is that each point stochastically selects a single neighbor, which makes the model well-justified only for kNN with $k = 1$. However, kNN classifiers with $k > 1$ are more robust and usually preferred in practice. Here we present kNCA, which generalizes NCA by learning distance metrics that are appropriate for kNN with arbitrary k. The main technical contribution is showing how to efficiently compute and optimize the expected accuracy of a kNN classifier. We apply similar ideas in an unsupervised setting to yield *kSNE* and *kt-SNE*, generalizations of Stochastic Neighbor Embedding (SNE, t-SNE) that operate on neighborhoods of size k, which provide an axis of control over embeddings that allow for more homogeneous and interpretable regions. Empirically, we show that kNCA often improves classification accuracy over state of the art methods, produces qualitative differences in the embeddings as k is varied, and is more robust with respect to label noise.

1. Introduction

Distance metrics are used extensively in machine learning, both as an essential part of an algorithm like in k-means or k-nearest neighbors (kNN) algorithms, and as a regularizer in, e.g., semi-supervised learning. An obvious problem is that the space in which data is collected is not always suitable for the target task (e.g., nearest neighbor classification); the choice of parameters like the scale of each dimension can significantly impact performance of algorithms. Thus, a long-standing goal is to learn the distance metric so as to maximize performance on the target task.

Neighborhood Component Analysis (NCA) is a method that aims at doing precisely this, adapting the distance metric to optimize a smooth approximation to the accuracy of the kNN classifier. A shortcoming of the NCA model is that it assumes $k = 1$, and indeed, all of the experiments in (Goldberger et al., 2004) and many in follow-up applications are performed with $k = 1$. This choice appears to be made for the sake of computational convenience: generalizing NCA for arbitrary k requires computing expected accuracies over all possible ways of choosing k neighbors from N points, which appears to be difficult when k is large. But since kNN with $k > 1$ tends to perform better in practice, it seems desirable to formulate an NCA method that directly optimizes the performance of kNN for $k > 1$.

The primary method we present, kNCA, is a strict generalization of NCA, which optimizes the distance metric for expected accuracy of kNN for any choice of k, and is equivalent to NCA when k is set to 1. The main algorithmic contribution of our work is a construction that allows the expected accuracy to be computed and differentiated exactly and efficiently. We also show that similar techniques can be applied to other problems related to the stochastic selection of k-neighborhoods, such as is required when extend-

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

[†] Work done primarily while authors were at the University of Toronto.

ing Stochastic Neighbor Embedding (SNE) methods to their k-neighbor analogs.

Thus, we make several contributions. The primary contribution is the extension of NCA so that it is appropriate when we wish to use a kNN classifier at test time with $k > 1$. Secondary contributions are the techniques for computing expected accuracy and the extension of SNE to the k-neighborhood setting. We explore the methods empirically: quantitatively, by comparing performance to several popular baselines on a range of illustrative problems, and qualitatively, by visualizing the learned embeddings by (supervised) kNCA and (unsupervised) kSNE.

2. Background

We begin with some basic notation. Input vectors are denoted by $x \in \mathbb{R}^D$. In supervised settings, the class label for a vector is denoted $y \in \{1, \dots, C\}$. A data set $\mathcal{D}$ is made up of N tuples $(x_i, y_i)_{i=1}^N$, and $X \in \mathbb{R}^{D \times N}$ is the concatenation of all x_i as column vectors. Similarly, Y is the vector of all the labels y_i. We use $z_i \in \mathbb{R}^P$ to represent a point in the transformed space that is associated with x_i, and Z to represent the concatenation of all z_i as column vectors. We use $[\cdot]$ as the indicator function.

2.1. Neighborhood Components Analysis

At a high level, the goal of NCA is to optimize a distance metric under the objective of performance of kNN algorithms. There are many ways to parameterize a distance metric, and this choice is not fundamental to the approach (for example, a non-linear extension like in Salakhutdinov & Hinton (2007) would be straightforward). For simplicity, we follow NCA and frame the presentation under the assumption that the distance between two vectors x and x' is defined as

$$d_A(x, x') = (x - x')^\top A^\top A(x - x'). \tag{1}$$

This choice has the interpretation that we are first linearly projecting points $x \in \mathbb{R}^D$ into P-dimensional space via the matrix $A \in \mathbb{R}^{P \times D}$, then computing Euclidean distances in the P-dimensional space. NCA also gives us the ability to visualize the learned metric by setting P to be small and plotting transformed points $Z = AX$.

Optimizing the entries of A requires the specification of a learning objective. A first attempt might be the accuracy of a kNN classifier. This approach is not feasible, because as a function of A, the performance of a kNN classifier is a piecewise constant function, which is not possible to optimize with gradient methods (note that any change in A that does not change the neighbor set of any point will not affect this objective). However, even if this approach were feasible, we still might prefer a smooth objective for the sake of robustness to noise in the data. For example, if a point has a neighbor of the proper class at a distance of b away, and many neighbors of other classes at a distance of $b+\epsilon$ away, the kNN accuracy where $k = 1$ will attain the maximum objective. This clearly is not a robust solution, though, since a slight perturbation of the data would likely lead to an error on this example.

These considerations lead to the central NCA idea of casting kNN in a probabilistic light. Specifically, (Goldberger et al., 2004) define a probability of selecting each point as its 1-nearest neighbor, which is a function of distance in transformed space. The learning objective is then the expected accuracy of a 1-nearest neighbor classifier under this probability distribution:

$$L(A) = \sum_i \sum_{j \neq i} p_i(j) \, [y_i = y_j] \,, \tag{2}$$

where $p_i(j) \propto \exp\left\{-\|Ax_i - Ax_j\|_2^2\right\}$ is the probability that i selects j as its (single) neighbor. This smooth objective has a nice interpretation as maximizing the expected accuracy of a 1-nearest neighbor classifier. Note that (Goldberger et al., 2004) also proposes an alternative objective, which can be interpreted as the (log) probability of obtaining an error free classification on the entire training set, $L(A) = \sum_i \log \sum_{j \neq i} p_i(j) \, [y_i = y_j]$, and note that performance is similar between the two objectives. Our generalization applies to either objective, and we similarly found performance to be similar between the two methods.

2.2. Stochastic Neighbor Embedding

Stochastic Neighbor Embedding (SNE) (Hinton & Roweis, 2002) is an unsupervised dimensionality reduction method that attempts to reproduce the local structure of high-dimensional data in a low-dimensional space. While this approach to dimensionality reduction is taken by popular methods such as Locally Linear Embedding (LLE) (Roweis & Saul, 2000) and Isomap (Balasubramanian et al., 2002), the SNE method differs from these methods because it uses a fundamentally smooth objective that is based on matching distances between distributions.

Given a set of high-dimensional points, $\{x_1, \dots, x_n\}$, SNE defines a distribution $p_i(j)$ for each point i that assigns a greater probability to its closer neighbors:

$$p_i(j) \propto \exp\left\{-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right\} \tag{3}$$

This distribution depends strongly on the length-scale σ_i^2, which is chosen in order to bring the entropy of p_i to a certain user-specified value (in experiments we set

σ_i^2 using the procedure described in Hinton & Roweis (2002)). This way, the distributions p_i smoothly describe the local neighborhood structure of the high-dimensional data.

Given a set of points $\{z_1, \ldots, z_n\}$ that are embedded in a low-dimensional space, we can define a similar distribution $q_i(j) \propto \exp\left\{-\|z_i - z_j\|^2\right\}$ which describes the local neighborhood structure of the embedded points.

SNE finds an embedding whose local neighborhood structure matches that of the original data by minimizing the following objective:

$$\sum_i KL(p_i \| q_i) = -\sum_i \sum_j p_i(j) \log q_i(j) + const. \quad (4)$$

In contrast to many other dimensionality reduction methods such as multidimensional scaling methods (for a good explanation see (van der Maaten et al., 2009)), SNE penalizes configurations that do not keep the neighbors close, while being more lenient to configurations that bring points together that are far apart.

SNE has been extended in several ways. Uni-SNE (Cook et al., 2007) alters the definition of $q_i(j)$ by the addition of a small constant: $q_i(j) \propto \exp\left\{-\|z_i - z_j\|^2\right\} + \kappa$. This provides SNE with more "effective space" in the low-dimensional space, since it can now place far points arbitrarily far. t-SNE (van der Maaten, 2009) alters the definition of $q_i(j)$ even more drastically, to $q_i(j) \propto (1 + \|z_i - z_j\|^2)^{-1}$, which tends to work even better in practice due to the heavier tails of the Cauchy distribution.

3. Related Work

Following the work of NCA, several researchers have proposed approaches to the problem of metric learning for kNN classification using the idea of stochastic neighbors. We note the method of Globerson & Roweis (2006), Maximally Collapsing Metric Learning algorithm (MCML), which presents a convex optimization function that approximates the desiderata that all points belonging to a class should be mapped to a single location in the embedding space infinitely far away from the points in other classes. A non-linear version of NCA (NLNCA) has also been introduced where a neural net is used to learn a non-linear mapping from original to embedding space (Salakhutdinov & Hinton, 2007). We are not aware of other NCA-based work that tailors the objective to the case of $k > 1$.

There has also been a plethora of research on metric learning for kNN classification using deterministic neighborhoods. Standard learning techniques such as random forests (Xiong et al., 2012), boosting (Shen et al., 2009), and large margin classification approaches (Weinberger & Saul, 2009) have been applied to this problem. A good review of this work is provided in Yang (2007), which also surveys some of the classic methods first introduced in the field such as RCA and LDA. We highlight the work of Weinberger & Saul (2009), which introduced a Large Margin Nearest Neighbor method (LMNN). They frame the problem as the optimization of a cost function which penalizes large within-class distance and small out-of-class distances in the embedding space. The within-class distances is only taken with respect to a number of *target neighbors*. Setting the size of this target neighborhoods acts in a similar fashion as setting k (and we will compare experimentally to this). Finally, we briefly note the method Information Theoretical Metric Learning (ITML) (Davis et al., 2007), where the problem is framed as a Bregman optimization problem and does not require the solution of an expensive semidefinite program. Empirically ITML often rivals LMNN in performance but its run-time is generally significantly shorter.

For the unsupervised case, a brief review of existing approaches is provided in Section 2.2; a good survey is also available (van der Maaten et al., 2009).

4. k-Neighborhood Components Analysis

Our starting point is the NCA objective from Eq. 2. While the objective has a desirable simplicity, it is heavily tailored to the case of a 1-nearest neighbor classifier: the distribution over neighbors assumes so, and the accuracy measure within the expected accuracy objective is the accuracy relative to selecting 1-nearest neighbor. Our goal in this section is to tailor NCA to the case of a kNN classifier for arbitrary k.

There are two components. First, we define a probability distribution over the selection of sets of k neighbors. Second, we modify the accuracy measure to reflect that the kNN procedure selects a label by majority vote of the k neighbors. Putting these two components together yields the kNCA expected accuracy objective:

$$L^{(k)}(A) = \sum_i \sum_{s \in S_i} p_i(s \mid k; A) \left[\mathtt{Maj}(s) = y_i\right], \quad (5)$$

where S_i is the set of all subsets of neighbors of i, $\mathtt{Maj}(s)$ denotes the majority function—equal to the kNN classifier output that would result from choosing s as the set of neighbors—and $p_i(s \mid k; A)$ represents the probability that i chooses s as its set of neighbors given that it chooses k neighbors. We define $p_i(s \mid k; A)$ to be a function of the distances between i

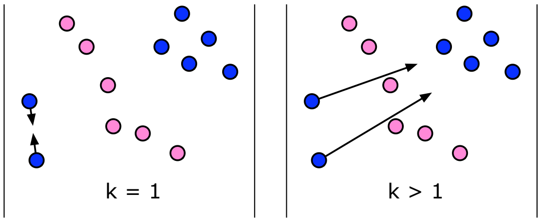


Figure 1. An illustration of the effect of k on the kNCA objective. (left) When $k = 1$, two points can be isolated from others in their class and have no pressure to seek out larger clusters. (right) When k is greater than 1, small clusters of points have greater pressure to join larger clusters, leading the objective to favor larger consistent groups of points.

and the points $j \in s$ as follows:

$$p_i(s \mid k; A) = \begin{cases} \frac{1}{Z(A)} \exp\{-\sum_{j \in s} d_{ij}(A)\} & \text{if } |s| = k \\ 0 & \text{otherwise} \end{cases},$$

where we use the shorthand $d_{ij} = d_{ij}(A) = ||Ax_i - Ax_j||_2^2$. While it is not yet obvious that Eq. 5 can be optimized efficiently, as the normalization $Z = Z(A)$ entails considering all subsets of size k, it should be clear that it is the expected accuracy of a kNN classifier in the same way that the NCA objective is the expected accuracy of a 1NN classifier. Indeed, with $k = 1$ we recover NCA.

5. Efficient Computation

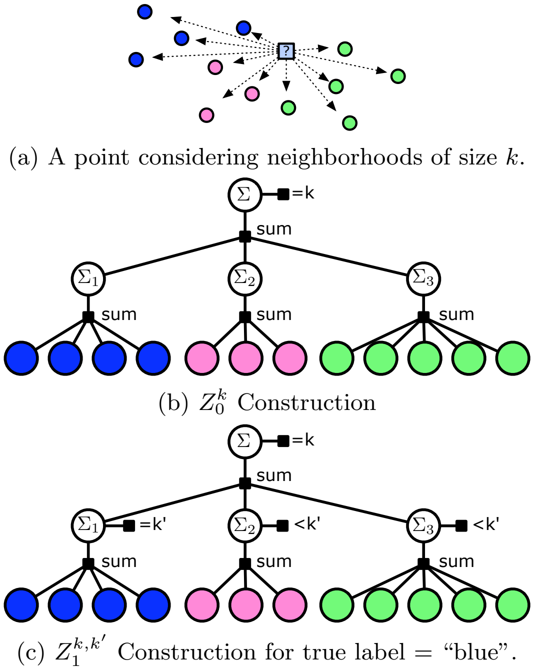
The main computational observation in this work is that Eq. 5 can be computed and optimized efficiently. To describe how, we begin by rewriting the inner summation from Eq. 5 (throughout this section, we will focus on the objective for a single point i, and will drop dependencies on i in the notation):

$$\frac{\sum_{s:|s|=k} \exp\{-\sum_{j \in s} d_{ij}\} \left[\mathtt{Maj}(s) = y_i\right]}{\sum_{s \in S:|s|=k} \exp\{-\sum_{j \in s} d_{ij}\}}, \qquad (6)$$

Our strategy is to formulate a set of factor graphs to represent the components of this problem such that the objective can be expressed as a sum of ratios of partition functions. Afterwards, we will show how efficient exact inference can be done on these factor graphs.

Factor graph construction. We start by constructing a factor graph for which the associated partition function is equal to the denominator in Eq. 6; see Fig. 2 (b). Note this is all with respect to a single point i considering its set of k neighbors. First (at the bottom of the factor graph), there is a binary indicator variable h_j for each possible neighbor j, indicating whether it is chosen to be a neighbor of i. Unary potentials (not drawn) are added, with potential set to $\theta_j(h_j) = h_j d_{ij}$. We group these variables according to class and introduce an auxiliary variable for each class c (next level up in the factor graph, denoted by Σ_c) that deterministically computes the number of neighbors that are chosen from class c. Finally (top level in the factor graph), there is an auxiliary variable Σ that counts the total number of neighbors that were chosen across all classes. At this point, we can add a constraint that the total number of neighbors selected across all classes is k, and this is simply a unary potential on Σ that disallows all values other than k. Once this constraint is added, the partition function of the resulting model is equal to the denominator of Eq. 6.

To construct the numerator, we first observe that $\mathtt{Maj}(s) = y_i$ is true if and only if there is some k' such that $\sum_{j \in s}[y_j = y_i] = k'$ and $\sum_{j \in s}[y_j = c] < k'$ for all $c \neq y_i$. That is, the true label gets k' votes, and all other labels get fewer than k' votes. For a given k', these constraints can be expressed as unary potentials on the intermediate sum variables Σ_c. Counts for the true class y_i are constrained to exactly equal k', and counts for the other classes are constrained to be less than k'. This is illustrated in Fig. 2 (c). By then summing over the partition function of these models for $k' = 1, \ldots, k$, we cover all possible ways for



(a) A point considering neighborhoods of size k.

(b) Z_0^k Construction

(c) $Z_1^{k,k'}$ Construction for true label = "blue".

Figure 2. Factor graph constructions used to efficiently compute expected kNN accuracy. The bottom row of each figure shows h variables grouped according to class label. The Σ_c variables represent the number of neighbors from class c that are selected, and Σ represents the total number of neighbors that are selected. The text next to factors notes the constraint that is represented by the factor. Unary potentials are omitted to reduce clutter.

$\texttt{Maj}(s) = y_i$ to be true, and thus recover the numerator of Eq. 6. Specifically, Eq. 6 can be rewritten as follows:

$$= \frac{\sum_{k'=1}^{k} \sum_{s:|s|=k} \exp\{-\sum_{j\in s} d_{ij}\}[\phi_{k'}(s)]}{\sum_{s\in S:|s|=k} \exp\{-\sum_{j\in s} d_{ij}\}} \tag{7}$$

$$= \frac{\sum_{k'=1}^{k} Z_1^{k,k'}}{Z_0^{k}}, \text{ where} \tag{8}$$

$$\phi_{k'}(s) = (\sum_{j\in s}[y_j = y_i] = k') \wedge (\forall c \neq y_i, \sum_{j\in s}[y_j = c] < k').$$

Efficient inference. At this point, we have reduced the difficulty of the kNCA objective to computation of partition functions in the models constructed in the previous section. If these partition functions can be efficiently computed and differentiated, then the kNCA objective can be optimized.

The key observation is that the models in Fig. 2 (b) and (c) are special cases of Recursive Cardinality (RC) models (Tarlow et al., 2012). An RC model defines a probability distribution over binary vectors $h = (h_1, \ldots, h_N)$ based on an energy function of the form $E(h) = \sum_i \theta_i(h_i) + \sum_{s\in\mathcal{S}} f_s(\sum_{j\in s} h_j)$, where $\mathcal{S}$ is a set of subsets of $\{1, \ldots, N\}$ that must obey a nestedness constraint: for all $s, s' \in \mathcal{S}$, either $s \cap s' = \emptyset$ or $s \subset s'$ or $s' \subset s$. The $f_s(\cdot)$ functions are arbitrary functions of the number of variables within the associated subset that take on value 1 and can be different for each s. Given this energy function, an RC model defines the probability of a binary vector h as a standard Gibbs distribution: $p(h) = \frac{1}{Z}\exp\{-E(h)\}$, where Z is the partition function that ensures the distribution sums to 1. The key utility of RC models is that the partition function (and marginal distributions over all h_i variables) can be computed efficiently. Briefly, inference works by constructing a binary tree that has h_i variables at the leaves, and variables representing counts of progressively larger subsets at internal nodes, then doing fast sum-product updates up and down the tree. See Tarlow et al. (2012) for more details.

For all applications of RC models considered here, this computation of each partition function and associated marginals would take $\mathcal{O}(N \log^2 N)$ time. Below, we alternatively show how to implement the same computations in $\mathcal{O}(Nk + Ck^2)$ time (where recall N is the total number of points considered as neighbors, k is the number of neighbors to select, and C is the number of classes). For our purposes where k is typically small, and due to the smaller constant factors, an efficient C++ implementation of this algorithm outperformed a generic implementation from (Tarlow et al., 2012), so we used the special-purpose algorithm throughout.

Algorithm 1 kNCA inference for point i

```
for $c = 1, ..., C$ do
    $f_c^1(1) \leftarrow [1, \exp\{-\textsc{Dist}(i, c, 1)\}, 0, \ldots, 0]$
    for $j = 2, ..., J_c$ do
        $f_c^1(j) \leftarrow \textsc{Forward1}(f_c^1(j-1), \textsc{Dist}(i, c, j))$
    end for
end for
$f^2(1) \leftarrow f_c^1(J_c)$
for $c = 2, ..., C$ do
    $f^2(c) \leftarrow \textsc{Forward2}(f^2(c-1), f_c^1(J_c), \theta_c(\Sigma_c))$
end for
$b^2(C) \leftarrow \theta(\Sigma)$
for $c = C-1, ..., 1$ do
    $b^2(c-1) \leftarrow \textsc{Backward2}(b^2(c), f_c^1(J_c), \theta_c(\Sigma_c))$
end for
for $c = C, ..., 1$ do
    $b_c^1(J_c) \leftarrow b^2(c)$
    for $j = J_c - 1, ..., 2$ do
        $b_c^1(j-1) \leftarrow \textsc{Backward1}(b_c^1(j), \textsc{Dist}(i, c, j))$
    end for
end for
```

Alternative $\mathcal{O}(Nk + Ck^2)$ algorithm. Here we present the alternative algorithm for computing marginal probabilities (used for gradients) and partition functions (used to evaluate the expected kNCA objective) for the models illustrated in Fig. 2. The structure of the algorithm is given in Alg. 1. The overall idea is to do dynamic programming over two levels of chain-structures. The first level of the forward pass computes probabilities over the number of neighbors chosen from each class separately, then the second level combines the results across classes to compute probabilities over the total neighbors selected. The backward pass propagates information from the other classes backward to the individual classes.

The computations use dynamic programming, incrementally computing a vector $f_c(j) \in \mathbb{R}^{k+1}$ that stores the probability for each $\hat{k} \in 0, \ldots, k$ that $\hat{k}$ variables from class c were chosen as neighbors of i from amongst the first j points of class c, assuming that the point j is chosen independently with probability $p_{ij} = \frac{\exp(-d_{ij})}{1+\exp(-d_{ij})}$. The Forward1 function computes $f_c^1(j+1)$ from $f_c^1(j)$ in $O(k)$ time, using the update that $f_c^1(j+1)[\hat{k}] = f_c^1(j)[\hat{k}-1]p_{ij} + f_c^1(j)[\hat{k}](1-p_{ij})$. Intuitively, there are only two ways for $\hat{k}$ variables to be chosen from amongst the first j variables: either $\hat{k}$ were chosen from the first $j-1$ and the j^{th} was not used, or $\hat{k}-1$ were chosen from the first $j-1$, and the j^{th} was used.

In the second level chain, we use the result of the first level forward pass to compute vectors $f^2(c)$, which store the upward probabilities that each possible number of neighbors were chosen from amongst the first c classes. The Forward2 functions compute these updates on $O(k^2)$ time using the update $f^2(c)[\hat{k}] = \sum_{k_c, k_{c-1}: k_c + k_{c-1} = \hat{k}} f^2(c-1)[k_{c-1}] \cdot f_c^1(J_c)[k_c] \cdot \theta_c(k_c)$, where $\theta_c(k_c)$ expresses the constraints that are given as unary factors on the Σ_c variables in Fig. 2. Similar reasoning can be used to derive the backward updates.

It can be shown that the above updates correspond to performing sum-product belief propagation on the factor graphs in Fig. 2, which are tree-structured, so the partition function and marginal probabilities of each neighbor being selected can be read off from the results. See Tarlow et al. (2012) for more details on this interpretation. Algorithmically, relative to Tarlow et al. (2012), the main difference is that we take advantage of the fact that any configuration with more than k neighbors chosen is disallowed.

Objective Function Variations. The construction from the previous section allows for other choices of the accuracy measure than the `Maj` one. For example, by requiring that *all* selected neighbors are of the target class (i.e., $[\sum_{j \in s}[y_j = y_i] = |s|]$), we get an accuracy measure that only rewards neighborhoods where *all* neighbors are of the target class. While this alternative measure which maximizes the number of points with "perfect" k-neighborhoods does not correspond to optimizing the expected accuracy of a kNN classifier, we will show below that it can boost the classifier's performance in practice. The preceding derivations can also easily be applied to a "probability of error free classification" variant of kNCA, analogous to the NCA variants discussed in Sec. 2.1, which amounts to taking a sum of logs of Eq. 6 instead of just a sum.

6. k-Stochastic Neighbors Embedding

So far, we have focused on the supervised case, where class labels are available for all points. In this section, we consider the unsupervised analog of kNCA. The starting point is Stochastic Neighbor Embedding (SNE), which like NCA has an interpretation that involves the stochastic selection of one neighbor.

Rather than selecting one neighbor, we proceed again by defining distributions over sets of k neighbors. For the unsupervised version, we have a target distribution p and an approximating distribution q. The goal is to minimize the sum of KL divergences for each point i: $-\sum_i \sum_{s \in S: |c| = k} p_i(s \mid k) \log q_i(s \mid k) + const$, where $p_i(\cdot)$ is defined in terms of target distances d^*_{ij} in the original space, while $q_i(\cdot)$ is defined in terms of distances in the lower dimensional space:

$$p_i(s \mid k) = \begin{cases} \frac{1}{Z} \exp\{-\sum_{j \in s} d^*_{ij}\} & \text{if } |s| = k \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

$$q_i(s \mid k) = \begin{cases} \frac{1}{Z} \exp\{-\sum_{j \in s} d_{ij}\} & \text{if } |s| = k \\ 0 & \text{otherwise} \end{cases} . \quad (10)$$

We can leverage the previous efficient computations of this objective after some re-arranging of the objective. Focusing on a single i:

$$\sum_{s \in S: |s| = k} p_i(s \mid k) \log q_i(s \mid k) \quad (11)$$

$$= -\sum_{s \in S: |s| = k} p^k_{ij} \sum_{j \in s} d_{ij} - \log Z^k_0 = -\sum_j d_{ij} p^k_{ij} - \log Z^k_0,$$

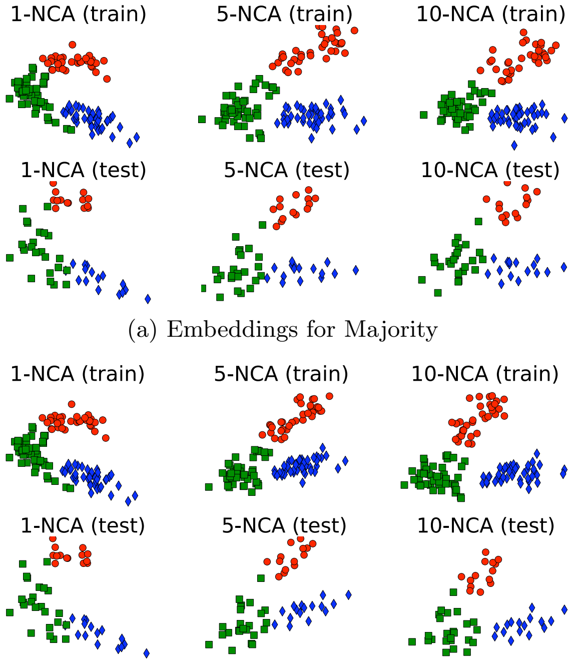
where p^k_{ij} denotes the probability that i chooses j, given that i chooses sets of k neighbors according to $p_i(c \mid k)$. The computations involved here are the same as are involved in computing Z_0 for kNCA. Note that p^k_{ij} can be precomputed once, then used throughout learning, and also that this formulation is agnostic to the distance measure used and therefore can be easily adapted to suit the measures used in SNE, t-SNE and other variants. We focus on the t-SNE variant.

7. Experiments

In a similar fashion to Goldberger et al. (2004), we experimented with various loss functions on several UCI datasets as well as the USPS handwritten digits dataset. For our experiments we divide the datasets into 10 different partitions of training and testing sets. Each partition uses 70% of the data for training and the remainder for testing. We inititialize the embedding matrix using PCA. We experimented with various values for k between 1 and 10, where $k = 1$ implies normal NCA. For each value of k, we trained separate models using the two loss functions discussed in the text: one that favors all k neighbors belonging to the same class (All) and one that favors that the majority belong to the same class (Majority) (when $k = 1$ these are equivalent). We experimented with both NCA objective variants and found performance to be similar, but we found the sum of logs variant (probability of error free classification) to be slightly easier to work with numerically, so we report results using it. Once each model is trained, we test using k-nearest neighbors on the learned models for $k \in \{1, 2, ..., 15\}$. We experimented with learning parameters that either project the data down to 2 dimensions, or retain the original data dimensions. To optimize, we used stochastic gradient descent with momentum, subsampling a set of points to compute the gradients (but still always considering all points as possible neighbors).

7.1. kNN Classification and kNCA Embedding

Our first set of experiments mimics those found in Goldberger et al. (2004). The extended results are given in the supplementary materials. For purposes



(a) Embeddings for Majority

(b) Embeddings for All

Figure 3. Illustrative learned embeddings on wine dataset for various choices of k. Setting $k = 1$ is standard NCA. The Majority measure uses the standard objective, and the All objective is described in Sec. 5.

of illustration, we show the learned embeddings from the different kNCA methods for one dataset (wine) in Fig. 3. For $k = 1$ (in (a) and (b), top left) the green class can be seen to wrap around the blue in the training embedding. While this satisfies the 1NCA objective well, it leads to worse generalization, as the boundary between blue and green becomes confused in the test data (bottom left). With larger k, generalization is improved. The All-trained models promote a larger margin while the Majority-trained models allow for a smaller margin and more dispersion within the clusters. The key point to note is that the All models can be seen as trying to build a larger margin between classes with k=1 being the weakest example of this. Meanwhile the Majority models are given more freedom to manipulate the projection. We found that this translates to the Majority objective converging more quickly in terms of optimization.

In the supplementary materials, we present quantitative results and an extensive comparison to other distance metric learning methods discussed in Sec. 3, including LMNN, MCML, 1NCA, and ITML (when A is full rank since ITML cannot be used to reduce the dimensionality of the data). In general, the UCI results are more variable, but kNCA compares favorably (either being the best, or near the best) in all cases.

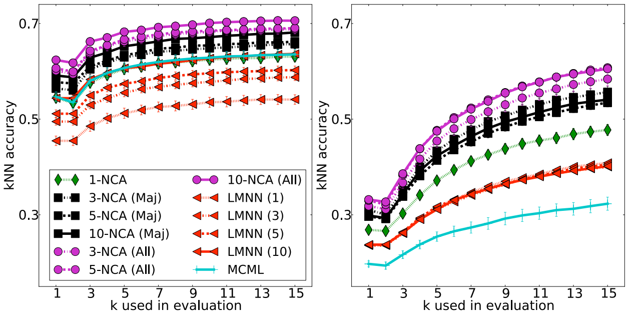
In the experiments on the USPS digits, we evaluate performance of the various algorithms when the data is more difficult and noisy. To study this in a controlled manner, we created three variations of the data with increasing levels of corruption in the labels. The first variant is the uncorrupted, original dataset while the others have 25% and 50% of the labels resampled uniformly. To evaluate performance, the votes of the neighbors come from the corrupted data, but we report correctness based on the uncorrupted labels. As can be seen by the increasing trend of all the curves in Fig. 4, using larger k at test time results in better performance. The improvements are steepest in the high noise cases. We also see that the kNCA methods substantially outperform 1-NCA, LMNN, and MCML. Although not reported here for lack of space, the above conclusions hold when comparing kNCA to ITML (in the full dimensional setting). Training accuracies are similar to test accuracies.

For a final experiment in the supervised setting, we tried to better understand why (a) kNCA with larger k outperforms 1NCA, and (b) why the All-trained models outperformed the Majority-trained models on the USPS data. One hypothesis is that the performance can be explained in terms of the severity of non-convexity in the objectives: since 1NCA is so narrowly focused on its immediate neighborhood, there are many local optima to fall into; and since Majority is forgiving of impure neighborhoods, there are more configurations that it is satisfied with, and thus more local optima. To test this, we took the parameters learned by kNCA, with both the Majority and All objective, and evaluated the 1NCA objective (Eq. 2). We repeated this several times across 10 different folds of the data (with different random initialization of the parameters for each fold) to measure the variance, which we attribute to reaching different local optima. Results are shown in Fig. 5. As hypothesized, the kNCA methods with larger k do actually achieve better 1NCA objectives, and the All training achieves better 1NCA objectives than the Majority training.

7.2. kt-SNE Embeddings

We also experimented with kt-SNE. The details for construction of the target distribution p followed the details presented in (Hinton & Roweis, 2002). In Fig. 6, we show the embeddings that have been learned by both t-SNE and kt-SNE with $k = 5$ at two points of learning: first, at iteration 25, where clusters are beginning to take form; second, at iteration 250, which had reached convergence. Note the global rearrangement that occurs even after iteration 25 when $k = 5$.

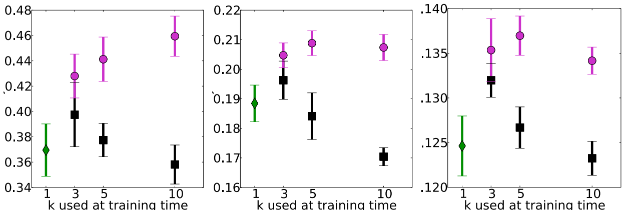
Quantitatively, in Fig. 6 (e), we use the true labels to



Test, 0 noise

Test, 0.5 noise

Figure 4. Test accuracies on USPS digits data for projections to 2 dimensions with varying levels of noise. 0.25 noise is omitted for space but interpolated 0 and 0.5.



0% noise

25% noise

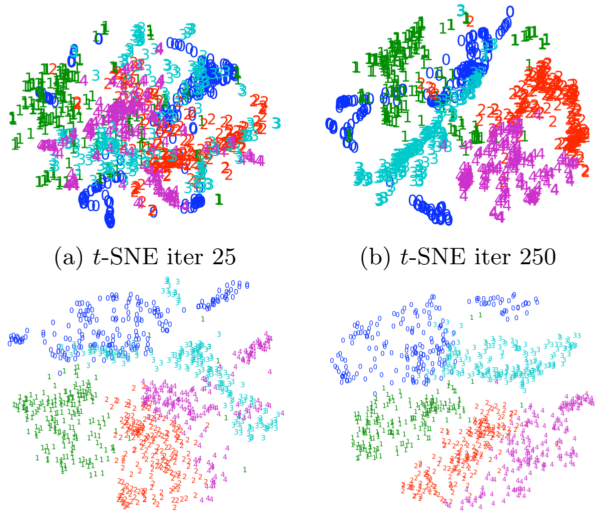
50% noise

Figure 5. Mean and standard deviations of training 1-NCA objectives achieved by optimizing kNCA objectives for varying k (higher is better). Legend follows Fig. 4.

measure the leave-one-out accuracy for a kNN classifier applied to the points output by t-SNE and kt-SNE. t-SNE performs better on 1-nearest neighbor accuracy, but when k is increased, 5t-SNE overtakes t-SNE. This further illustrates the myopic nature of using $k = 1$ in t-SNE. In the supplementary material, we provide animations illustrating the evolution of kSNE embeddings on USPS digits for various choices of k. In these animations, qualitative differences are visible, where t-SNE exhibits the myopic behavior illustrated in Fig. 1.

8. Discussion

There are several desirable properties of kNCA. First, it provides a proper methodology for doing NCA-like learning when the desire is to use kNN with $k > 1$ at test time. Our work here derives the NCA-like objective that is properly matched to using kNN at test time. kNN classifiers are ubiquitous, and a choice of $k > 1$ is nearly always used, so the method has wide applicability. Second, it provides robustness in two ways: first, the majority objective is relatively unconcerned with outliers, so long as the majority of neighbors in a region have the correct label; second, the objective optimizes an expectation over the selection of all sets of k neighbors, so we do not expect small perturbations in the data to have a significant effect on the learning objective. Robustness is not achieved by the methods we compare to, which we attribute to either their global or non-probabilistic nature.



(a) t-SNE iter 25

(b) t-SNE iter 250

(c) 5t-SNE iter 25

(d) 5t-SNE iter 250

	$k = 1$	$k = 5$	$k = 9$	$k = 13$
t-SNE	**0.934**	0.925	0.916	0.914
5t-SNE	0.928	**0.946**	**0.953**	**0.949**

(e) kNN accuracy after unsupervised learning.

Figure 6. Unsupervised USPS digits results.

Curiously, in most cases, the All objective outperformed the Majority objective. The argument can be made that All is like a margin-enhanced version of Majority, which inherits robustness due to the probabilistic formulation, but generalizes well due to its margin-enforcing tendencies. We believe this to be an interesting result for those people wishing to design better learning objectives; it challenges the common intuition that the best learning objective is to minimize expected loss. However, our final supervised experiments suggest that the story may be more complicated, and that we might need to find better ways of initializing and optimizing the two methods before having a clear answer.

One disadvantage of NCA (and thus also kNCA) is the inherently quadratic nature of the algorithm that comes from basing it on pairwise distances. We believe the method to still have desirable properties when the set of potential neighbors is restricted (either randomly or deterministically) but a fuller exploration of the tradeoffs involved require further investigation.

Finally, we believe the general technique used to compute the expected majority function to be of interest beyond just for kNN classifiers and for kNCA learning. It would be interesting to find further applications.

References

Balasubramanian, M., Shwartz, E. L., Tenenbaum, J. B., de Silva, V., and Langford, J. C. The isomap algorithm and topological stability. *Science*, 2002.

Cook, JA, Sutskever, I., Mnih, A., and Hinton, GE. Visualizing similarity data with a mixture of maps. *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

Davis, Jason V., Kulis, Brian, Jain, Prateek, Sra, Suvrit, and Dhillon, Inderjit S. Information-theoretic metric learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007.

Globerson, Amir and Roweis, Sam T. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

Goldberger, Jacob, Roweis, Sam T., Hinton, Geoffrey E., and Salakhutdinov, Ruslan. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

Hinton, Geoffrey E. and Roweis, Sam T. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

Roweis, Sam T. and Saul, Lawrence K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, 2000.

Salakhutdinov, Ruslan and Hinton, Geoffrey. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 11, 2007.

Shen, Chunhua, Kim, Junae, Wang, Lei, and van den Hengel, Anton. Positive semidefinite metric learning with boosting. In *Advances in Neural Information Processing Systems (NIPS)*. 2009.

Tarlow, Daniel, Swersky, Kevin, Zemel, Richard S., Adams, Ryan P., and Frey, Brendan J. Fast exact inference for recursive cardinality models. In *Uncertainty in Artificial Intelligence (UAI)*, 2012.

van der Maaten, Laurens. Learning a parametric embedding by preserving local structure. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

van der Maaten, L.J.P., Postma, E.O., and van den Herik, H.J. Dimensionality reduction: A comparative review. Technical Report TiCC-TR 2009-005, Tilburg University, 2009.

Weinberger, K.Q. and Saul, L.K. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research (JMLR)*, 2009.

Xiong, Caiming, Johnson, David, Xu, Ran, and Corso, Jason J. Random forests for metric learning with implicit pairwise position dependence. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, New York, NY, USA, 2012. ACM.

Yang, Liu. Distance metric learning: A comprehensive survey. Technical report, Carnegie Mellon University, 2007.