Distance Learning Algorithm Comparing in Classification and Retrieval

Jiecheng Zhao
supervisor: Scott Sanner

October 31, 2012

Abstract

Learning a good distance function is crucial in a lot of applications. Different Learning distance with different will effect the performance of application in different ways. The most closely related application is classification and retrieval task.

In this report, three different distance learning algorithm, LMNN, Isomap and SimpleNPKL, is compared via their performance in three different kinds of application with various dataset to explore the characteristic of these algorithms and to compare them in a whole view. In the experiment a case of special over fitting was observed. It also report a new approach to improve the performance of Isomap via introduce side information to the algorithm.

1 Introduction

Distance function is a function which calculate a distance between elements of a set. A good learned distance functions can significantly improve the performance in classification, clustering and retrieval tasks.

For example, the kNN algorithm is one of the oldest and simplest methods for pattern classification. The kNN algorithm is heavily depends on distance metric. When skilfully select metric with prior knowledge, it often gives competitive results.[12] In clustering field, A good remapping of data will give a better result. Some Clustering algorithm like EM work poorly when applying to data with nonlinear structure. With the help of nonlinear-to-linear embedding, which is also a kind of distance learning algorithm, such algorithms will yield good results.[5] In the field of computer vision, given a picture, find another picture with the same thing. This is an information retrieval task. Usually we pick the neighbours of the target picture and this task is heavily rely on the distance function learning. With euclidean distance, it usually return a bad result.

Some of the distance learning algorithm can also assign coordinator to the points that only given their similarity relationships. For example in word processing, it is hard to simply assign a meaningful coordinator to a word. However with some distance metric learning algorithms which only need similarity information, it is possible to find a meaningful coordinator to a word.[6] This will make a lot of algorithm which based on coordinate can be implement in the area.

Depending on the availability of the training examples, algorithms of distance learning can be divided into three categories: Supervised distance metric learning, unsupervised distance metric learning and semi-supervised distance metric learning. Training examples are given with class labels in supervised distance metric learning and with pairwise constraints in semi-supervised learning.

This report describe experiments on three different algorithms, Large Margin Nearest Neighbour classification,Isometric Feature Mapping, SimpleNPKL . To compare the algorithm with different angles, the experiments involves comparing the algorithms with three different criterion where two of them are designed to test the performance and the other one is focus on retrieval tasks. And the algorithms are also tested with data with different structure. At the end of the experiment, a new approach is applied on one of the algorithm to try to improve the performance of the algorithm. The report will give the characteristic of each algorithm and compare the advantage and disadvantage between them in different ways.

2 Background

In machine learning community, distance learning is a hot area these years and there are lots of related work. Generally, there are three kinds of distance learning algorithm. And there are lots of work in each catagories.

2.1 Supervised Learning

Supervised distance function learning is learning from training data associated with class labels. The representative techniques include Linear Discriminant Analysis which try to find a linear combination of features, Neighbourhood Components Analysis which focus on the k-nearest neighbour result of the embedding, Maximally Collapsing Metric Learning and distance metric learning for Large Margin Nearest Neighbour classification(LMNN).[12][13] which also enforce the k-nearest neighbours belong to the same class and max the margin between different labelled examples. Invariant Large Margin Nearest Neighbour Classifier(ILMNN)[7], which add regularization and incorporating invariance to LMNN, DistBoost[10] which use the idea of Adaboost classifier to learn a good distance function from week distance functions.

2.2 Unsupervised Learning

The second category is Unsupervised Learning which attempt to find low-dimensional embeddings given high-dimensional input data. This is very useful when scientists work with large volumes of high-dimensional data, such as global climate patterns, stellar spectra. There are both linear approaches and nonlinear methods. Principal Component Analysis(PCA) might be most well known method; Multidimensional Scaling(MDS)[6] embed points via the distance between examples. PCA and MDS are both linear global method, and they are equivalent when MDS use Euclidean distance. The nonlinear reduction includes Isometric Feature Mapping (ISOMAP)[5] and Locally Linear Embedding(LLE)[8] which can give a better result when dealing with data with nonlinear structure.

2.3 Semi-supervised Learning

The third category is Semi-supervised Learning which learn the distance function with pairwise constraints. Each constraints indicate weather two data point are similar or dissimilar to each other in a particular learning task.

Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, which is much easier to get than label the whole example set, can produce considerable improvement in learning accuracy. A famous method was proposed by E.P.Xing[4], which minimize the distance between the similar pairs and keep the dissimilar pairs well separated. Relevant Component Analysis (RCA)[1] is another method with low computational load compared with Xing's method and empirical good performance but only use similar constraints. Simple Non-Parametric Kernel Learning(SimpleNPKL)[14] is an efficient method in this area, which has a closed-form solution with linear loss and quickly converge algorithm with

Square Hinge Loss. Constrained Metric Learning(CML)[11] is another fast and good performance method and take both global and local fact into consideration. Another method which deal with both local and global information is Globally-Consistent Local Distance Function[2].

2.4 Pairwise Constraints

Unlike supervised learning, semi-supervised distance metric learning use pairwise constraints as supervising information. There are two kinds of pairwise constraints. One is equivalence constraints, which state that the given pair are semantically-similar and should be close together in the learned metric; and the other kind is inequivalence constraints, which indicate that the given points are semantically-dissimilar and should not be near the learned metric.

Let $\mathcal{U} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ denote the collection of data points, where each data point $\mathbf{x}_i \in \mathcal{X}$. If $\mathbf{x}_i$ and $\mathbf{x}_j$ are known in the same class, then $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}$ and if $\mathbf{x}_i$ and $\mathbf{x}_j$ are known in the different class, then $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}$. The $\mathcal{S} and \mathcal{D}$ are often called as "side information"

Compared to the full labelled data, the side information are much easier to collect. For example, faces extracted from successive frames in a video in roughly the same location can be assumed to come from the same person. In search engine logs, two links a user clicked in a short time slide can be assumed related to each other. This make a lot of problems become practical via semi-supervised computing.

3 Techniques Involved

In our work, we do experiment and survey on three different algorithms, Large Margin Nearest Neighbour classification(LMNN),Isometric Feature Mapping (Isomap) and Simple Non-Parametric Kernel Learning(SimpleNPKL), to try to find out the character of each algorithm. These works involved some background techniques and here is a simple overview of there background techniques.

3.1 Semi-definite Programming[9]

Semi-definite Programming is a kind of convex optimization. The objective function which we want to minimized is a linear function and constraints that we must satisfied is a cone of positive semi-definite matrices with an affine space.

A Semi-definite Programming Problem can be written as:

$$\begin{aligned} &maximize && \mathbf{tr}(CX) && (1)\\ &subject\ to && \mathbf{tr}(A_iX) = b_i, i = 1, ..., p && (2)\\ & && X \succeq 0 && (3) \end{aligned}$$

Semi-definite Programming is a relatively new field of optimization and arouses growing interest. Many piratical problems in combinatorial optimization and operations research can be formalized into Semi-definite programming. In our experiment LMNN is finally formulated as a Semi-definite Programming. SDPs are in fact a special case of cone programming and can be efficiently solved by interior point methods. A result with an additive error ϵ can be achieved in $O(\log(1/\epsilon))$. There are several standard solver available on the internet, such as cvxopt and cvx in matlab. However for efficient reasons we will not use standard solvers in LMNN.

3.2 Kernel Trick

For machine learning algorithms, the kernel trick is a way to map observations from a general set X into an inner product space V , without ever having to compute the mapping explicitly. Generic classifications in X will be equivalent to linear classifications in V which can bring much generalization to a bunch of linear classifiers. the trick is transform some linear classifier algorithms to use only products between vectors in V to avoid the explicit mapping, and choose the mappingby means of a kernel function such that these high-dimensional dot products can be computed within the original space. For example, Gaussian radial basis function kernel [3] is usually used as a default kernel for its close relationship with Gaussian distribution. The corresponding feature space of Gaussian radial basis function kernel is a Hilbert space of infinite dimensions which is impossible to calculate directly. With the help of kernel trick, do linear classification in such a feature space is practical now.

For x, y on S, certain functions $K(\mathbf{x}, \mathbf{y})$ can be expressed as an inner product. K is often referred to as a kernel or a kernel function. The word kernel is used in different ways throughout mathematics. If one is lucky or insightful regarding a particular machine learning problem, one may manually construct $\psi : S \to V$ such that $K(x, y) = \langle \varphi(x), \varphi(y) \rangle_V$ and verify that is indeed an inner

product.

SimpleNPKL is an algorithm to automatically learn a Kernel from geometry information and side information of the data. It also mapping points into a new space.[14] Although can not get the exact coordinate in a space X, it can also be regarded as a kind off distance learning. In our experiment, we will derive the distance from kernel.

Distance between points can be derived from kernel via the following equation.

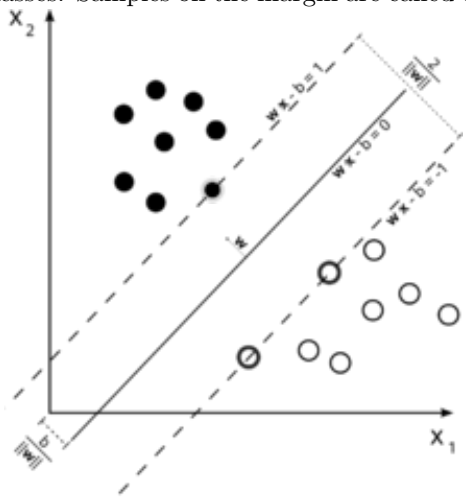
$$\|\mathbf{x}-\mathbf{y}\|_2^2 = \langle(\mathbf{x}-\mathbf{y}),(\mathbf{x}-\mathbf{y})\rangle = \langle\mathbf{x},\mathbf{x}\rangle - 2\langle\mathbf{x},\mathbf{y}\rangle + \langle\mathbf{y},\mathbf{y}\rangle \tag{4}$$

3.3 Support Vector Machine[3]

Support vector machines (SVM) are a kind of supervised learning models which used for classification and regression analysis. The basic SVM takes a set of examples from two classes and try to find a boundary with the maximized margin. After doing so an SVM model becomes a representation of the examples as points in space. That the examples of the separate categories are divided by a clear margin that is as wide as possible. New points are then mapped into that same space and predicted to belong to a class based on which side of the boundary they are on.

As well as modelling linear classification, SVMs can efficiently perform nonlinear classification using the kernel trick, implicitly mapping their inputs into high-dimensional, maybe infinite-dimensional feature spaces which might be very hard to calculate directly.

Figure 1: Maximum-margin boundary and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.



4 Distance Algorithms

In the work, we compare three distance algorithms to try to figure out their characteristic and the difference between them. The algorithm we compared is :

- One kind of supervised learning, Large Margin Nearest Neighbour classification (LMNN).
- One kind of unsupervised learning ,Isometric Feature Mapping (Isomap).
- Another kind of semi-supervised learning, and Simple Non-Parametric Kernel Learning (simpleNPKL).

Here is the introduction of each algorithm.

4.1 Large Margin Nearest Neighbour classification[12]

Large Margin Nearest Neighbour classification is a way to learn a Mahanalobis distance metric for kNN classification. With no prior knowledge, most kNN classifier simply use Euclidean distances metric to compare the similarity between examples represented as points in linear space. However Euclidean metric do not benefit from any statistical regularities in the data. Metric learned by LMNN is adapted to a particular problem being solved via learning from data labels. In fact, as shown by many researchers, kNN classification can be significantly improved by learning a distance metric from labelled examples.

4.1.1 Cost Functions

Before given the loss function of the problem, we will introduce an idea called target neighbours.

- Target neighbours
  In addition to the class label y_i, for each input x_i we also specify k "target" neighbours. That is k other inputs with the same label y_i that we wish to have minimal distance to x_i. In the absence of prior knowledge, the target neighbour can simply be indentified as the k nearest neighbours, determined by Euclidean distance. We use $\eta_{ij} \in \{0, 1\}$ to indicate whether input x_j is a target neighbour of input x_i. Like the binary matrix y_{ij}, the matrix η_{ij} is fixed and does not change during Learning.

Let $\{(x_i, y_i)\}_{i=1}^n$ denote a training set of n labelled examples with inputs $x_i \in R^d$ and discrete class labels y_i. We use the binary matrix $y_{ij} \in \{0, 1\}$ to indicate whether or not the labels y_i and y_j match. Our goal is to learn a linear transformation $L : R^d \to R^d$, which we will use to compute squared distances as: $D(x_i, x_j) = \|L(x_i - x_j)\|^2$.

Our cost function over the distance metrics parameter by has two competing terms. The first term penalizes large distances between each input and its target neighbours, while the second term penalizes small distances between each input and all other inputs that do not share the same label. Specifically, the

cost function is given by:

$$\mathcal{L} = \sum_{ij} \eta_{ij} \|L(x_i - x_j)\|^2 + c \sum_{ijl} \eta_{ij}(1 - y_{il})[1 + \|L(x_i - x_j)\|^2 - \|L(x_i - x_l)\|^2]_+ \quad (5)$$

Where $[x]_+ = max(0, x)$.

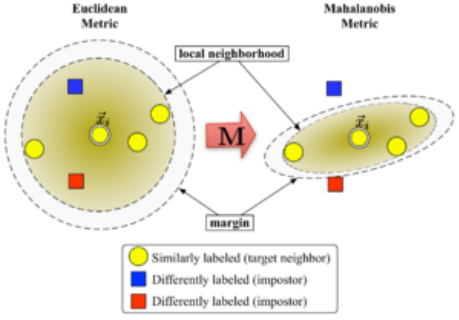
4.1.2 convex optimization reformulation

To solve the problem in practical we will reformulate the problem in to a Semi-Definite Problem. We can rewrite $D(x_i, x_j)$ as $(x_i - x_j)^T M(x_i - x_j)$. Where the matrix $M = L^T L$, parameterize the Mahalanobis distance metric introduced by the linear transformation L. Rewrite (5) as an SDP. The hinge loss can be mimicked by introducing slack variables ξ_{ijl} for all pairs of differently labelled inputs.

then the problem can be formulated as a SDP:

$$\begin{aligned}
&\textit{minimize} && \sum_{ij} \eta_{ij}(x_i - x_j)^T M(x_i - x_j) + c \sum_{ij} \eta_{ij}(1 - y_{ij})\xi_{ijl} && (6)\\
&\textit{subject to} && (x_i - x_l)^T M(x_i - x_l) - (x_i - x_j)^T M(x_i - x_j) \geq 1 - \eta_{ijl} && (7)\\
& && \eta_{ijl} \geq 0 && (8)\\
& && M \succeq 0 && (9)\\
& && && (10)
\end{aligned}$$

Figure 2: Schematic illustration of LMNN.



While this SDP can be solved by standard on-line solvers. General-purpose solvers tend to scale poorly in the number of constraints. The inventor of LMNN designed a special-purpose solver. That exploit the fact that most of the slack variables never attain positive values to avoid constraints. It was based on a combination of sub-gradient descent in both the matrices L and M. M is used mainly to verify whether the global minimum was reached. Alternating projection algorithms provably converge, and in this case our implementation worked much faster than generic solvers.

4.2 Isometric Feature Mapping[5]

Isomap is an algorithm to solve the problem of dimensionality reduction with non-linear data : pending meaningful low-dimensional structures hidden in their high-dimensional observations. Because with the effect of dimension curses, some algorithm will not work with the high dimension data and some will lose it efficiency when dealing with high dimension data. High dimension will also bring burden to the memory system. Finding a good low-dimensional enbedding can solve these problem. Although PCA and MDS work well with data with

Figure 3: Image set obtained by rotating the object about a single axis. These images are scale and brightness normalized.

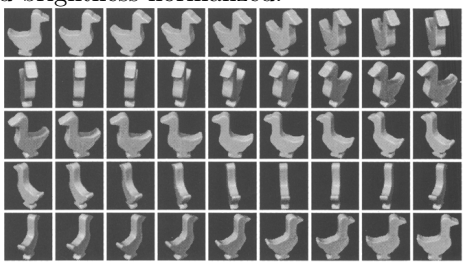
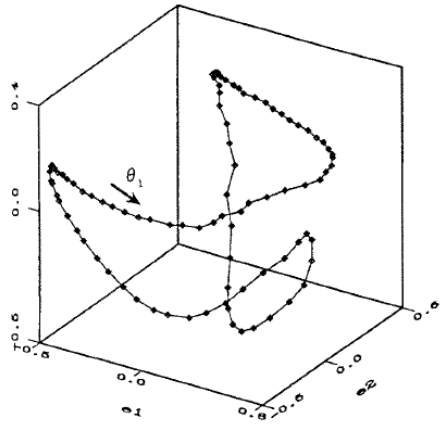


Figure 4: Parametric eigenspace representation computed using the image set shown above. Only the three most prominent dimensions of the eigenspace are displayed here. The dots correspond to projections of learning samples. Since illumination is constant in this case, appearance is given by a curve with a single parameter (rotation) rather than a surface.



linear structure. They do not work well with data which contain non-linear structure.[] However, this kind of data is widely exists in practical problems. Here are figures show the non-linear structure of the image set of a rotating object.

The Isomap is an algorithm that builds on classical MDS but seeks to preserve the intrinsic geodesic, as captured in the geodesic manifold distances between all pairs of data points. The crux is estimating the geodesic distance between faraway points, given only input-space distance. For neighbouring points, input space distance provides a good approximation to geodesic distance. For faraway points, geodesic distance can be approximated by adding up a sequence of short hops between neighbouring points.

The algorithm of isomap to find a d-dimensional embedding:

1 Construct neighborhood graph
Define a undirected weighted graph G over all the points by connecting points i and j with weight $\|\mathbf{x}_i - \mathbf{x}_j\|_2$, if i is one of the k nearest neighbour of j.

2 Compute shortest paths
Given graph G calculate the shortest path between each pair of points. Define a matrix D that D_ij is the length of shortest path from i to j.

3 Construct d-dimensional embedding
Let λ_p be the p-th largest eigenvalue of the matrix $\tau(D)$, and v_p^i be the i-th component of the p-th eigenvector. Then the embedding coordinate of i, $\mathbf{x}_i = [\sqrt{\lambda_1}v_1^i, \sqrt{\lambda_2}v_2^i, .., \sqrt{\lambda_d}v_d^i]^T$

The first two steps are graph algorithm, the third step is a kind of MDS algorithm. Although mathematically the Isomap has a close form solution, finding k-Nearest neighbour would be slow when the dimension of the data is very high and the short path search and the eigen-decomposition will also introduce much calculation when the size of sample set is very large.

4.3 Simple Non-Parametric Kernel Learning[14]

Non-Parametric Kernel Learning is a family of algorithms that try to learn a kernel automatically with side information. Usually the Non-Parametric Kernel Learning will be formulated as a Semi-definite Problem, which may be low efficient. Simple Non-Parametric Kernel Learning with linear loss has a closed-form solution that can be simply computed. The Simple is formulated as the following.
Let $\mathcal{U} = \{x_1, x_2, ..., x_N\}$ denote the entire data collection, where each data point $x_i \in \mathcal{X}$. With pairwise constraints that $\mathcal{S}, \mathcal{D}$. Given $\mathcal{S}$ and $\mathcal{D}$, we construct a similarity matrix $T \in \mathbf{R}^{N \times N}$ to represent the pairwise constraints, i.e.,

$$T_{ij} = \begin{cases} 1 & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, \\ -1 & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}, \\ 0 & \text{otherwist.} \end{cases} \tag{11}$$

An intuitive principle for kernel earning is that the kernel entry K_{ij} should be aligned with the side information T_{ij} as much as possible.
In addition to constraints information, geometric information of the data can also be explored to improve the performance of kernel learning. Typically, most existing kernel learning approaches adopt the data manifold to preserve the locality. Below is an approach for exploring manifold in kernel learning. Let us denote $f(x, x')$ as a similarity function that measures the similarity in geometric

view between any two data points x and x' and $S \in \mathbf{R}^{N\times N}$ as the similarity matrix where each element S_{ij}. Generally, a Non-Parametric Kernel (NPK) matrix K with respect to N patterns can be expressed as $K = V^T V \succeq 0$,where $V = [v_1, ..., v_N]^T$ is the matrix of the embedding of data points in feature space. The cost function of the kernel matrix K, which captures the local dependency between the embedding of v_i and v_j , can be defined as:

$$\begin{aligned} \Omega(V, S) &= \sum_{i,j=1}^{N} S_{i,j} \| \frac{v_i}{\sqrt{d_i}} - \frac{v_j}{\sqrt{d_j}} \|_2^2 && (12) \\ &= tr(VLV^T) = tr(LK) && (13) \end{aligned}$$

where L is the graph Laplacian matrix defined as:

$$L = I - D^{-1/2} S D^{-1/2} \tag{14}$$

Here $D = diag(d_1, d_2, ..., d_N)$ is a diagonal matrix with the diagonal element defined as $d_i = \sum_{j=1}^{N} S_{ij}$.

The Simple NPKL pick loss function $l(f) = -f$, we can write the problem of Simple NPKL as the following:

$$\begin{aligned} &minimize && tr((L - CT)K) && (15) \\ &subject\ to && K \succeq 0 && (16) \\ & && tr(KK) \leq B && (17) \end{aligned}$$

here $C > 0$ is a trade off scaler parameter to control the loss.

Take $A = CT - L$, The SimpleNPKL has the following closed-form solution.

$$K^* = A_+ \sqrt{\frac{B}{tr(A_+ A_+)}} \tag{18}$$

where $A = U\Sigma U^T$, $A_+ = U\Sigma_+ U^T$ and Σ is a diagonal matrix and $\Sigma_{+i} = (\Sigma_i)_+$.

5 Approach

Our approach is comparing different performance between three different distance metric with different angles to figure out some character in and difference between these algorithms.

5.1 Criterion Selection

As the algorithm with different approach can hardly compared with a simple single criterion, we set 3 different criterion to compare the algorithm from different angle. As mentioned above the distance metric which learned via algorithm plays an important role when people apply some other machine learning algorithms to the example with learned distance metric. Considering three main area distance learning applied, classification, clustering and retrieval tasks, clustering is not easy to compare within these three kind of algorithms for the information given to the algorithm is on different level. We will pick three ten-fold cross validation accuracy ratio given by some practical classification and retrieval algorithms as the criterion. The three algorithm we pick is:

- Support Vector Machine classifier.
- k-Nearest Neighbour classifier.
- k-Nearest Neighbour retrieval.

Nowadays SVM and K-nearest neighbour classifier are very popular classifier algorithm. As they are popular and widely used, the classification accuracy given by these two algorithm should be a good criterion that reflect the actual state of the distance learning algorithm in practical situation.

For SVM classifier, we will pick the Gaussian radial basis kernel, as the kernel to be used when classifying data given by LMNN and Isomap. The reason to use this kernel is that the kernel is widely used and is closely related to the distance between data. The formula of Gaussian radial basis kernel is $k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2)$. And the learned distance metric of LMNN and Isomap can be represented as Euclidean distance in a new space of mapped data. The SVM classifier with this kernel should be sensitive with the distance metric learned. To simpleNPKL, we will train the SVM with the kernel learned by the algorithm.

K-Nearest Neighbour classifier is also a distance sensitive algorithm. Unlike SVM try to figure out a global linear boundary in the feature space. The way k-Nearest Neighbour classifier work is quite local. This will help us to see the effect of learned distance metric in a local view.

K-Nearest Neighbour retrieval is an important step when doing k-Nearest Neighbour classifier. It is to find k-nearest neighbour of the example given. Unlike the classifier, we will focus on how many data we want (with the same label) are found in k nearest neighbours, instead of how many right predictions we made via kNN algorithm. This is a criterion with the view of retrieval task.

5.2 Semi-supervised Isomap

We also try to improve the performance of the algorithm. For isomap is a good unsupervised embedding algorithm, Our approach is to improve the performance by introducing some side information semi-supervise the Isomap algorithm. Inspired by NPKL, we mixture the information of constraints and the geometry information of the data. In the semi-supervised learning, we want to push the points that with different label away. The new algorithm is like the following, Semi-supervised isomap:

1 Construct neighborhood graph
Define a undirected weighted graph G over all the points by connecting points i and j with weight $\|\mathbf{x}_i - \mathbf{x}_j\|_2$, if i is one of the k nearest neighbour of j.

2 Introduce dissimilar penalty
In graph G, if an edge connect two points in a dissimilar constraint $\mathcal{D}$, update the weight of the edge by multiply it with a penalty scaler C which is great then 1.

3 Compute shortest paths
Given graph G calculate the shortest path between each pair of points. Define a matrix D that D_ij is the length of shortest path from i to j.

4 Construct d-dimensional embedding
Let λ_p be the p-th largest eigenvalue of the matrix $\tau(D)$, and v_p^i be the i-th component of the p-th eigenvector. Then the embedding coordinate of i, $\mathbf{x}_i = [\sqrt{\lambda_1}v_1^i, \sqrt{\lambda_2}v_2^i, .., \sqrt{\lambda_d}v_d^i]^T$

After introduced information, the gap between two classes will be extend, it might be easier for classifiers work in this new space.

6 Experiment

6.1 Implementation of Algorithms

In our experiment, Isomap and SimpleNPKL is implemented in python with the help of Scipy and Numpy, and the LMNN is called from a matlab package provided by Kilian Weinberger, the inventor of LMNN, in which a specified SDP solver is implemented to solve the problem fast. We also implement LMNN via cvxopt, but for the efficiency reason, it was not used in experiment. KDTree provide by Scipy was used to find the k-nearest neighbour in isomap, simpleNPKL and kNN classifier (retrieval). Scipy also provided an interface to Arpack, which is a popular toolkit to do eigen-decomposition on sparse matrix. With the help of these algorithms, the program is quite efficient. All the experiment costs only a few minutes to run.

The SVM classifier we use is LibSVM[**?**]. we pick Gaussian radial basis kernel as kernel in learned linear space by LMNN and Isomap, and learned kernel from SimpleNPKL.

When doing k-Nearest Neighbour classification (retrieval). $k = 3$ which would represent a more local result was picked.

6.2 LMNN

As mentioned above, LMNN is an algorithm try to grab target neighbour close to it and push the neighbour that labelled as different away. We will test LMNN with some artificial and natural data. To see the character of LMNN.

6.2.1 LMNN with linear data structure

Here is the linear structure we use to indicate the ability LMNN to learn. It is generate as following. The points in the first class is $\{(x_i, y_i)|x_i = 90i/N + U(0, 0.01), y_i = U(0, 0.01), i = 1, ..., N\}$, and the second class is $\{(x_i, y_i)|x_i = 90i/N + U(0, 0.01), y_i = U(0.015, 0.025), i = 1, ..., N\}$. where U is the uniform distribution and $N = 150$. Although this set is linearly well separated, it is a very hard for kNN classifier and SVM with radial basis kernel to do classification. The original figure is showed as following:

After learned a Linear projection via LMNN. we project the data into a New space. As the graph show, the x coordinate is from 0 to about 16 now and y coordinate is from 0 to 0.04. This show that the projection shrink the distance in x direction for about 5 times, and extend the distance in y direction for about 1.5 times.

	svm classifier	knn classifier	knn retrival
lmnn	35.0	62.66	60.44
baseline	4.33	15.33	35.79

Table 1: The Result of LMNN on the linear data

The result shows that the problem used to be very hard for the 3 criterion algorithms we picked, The algorithms are more likely to make the opposite predictions instead of the right ones. After implementing LMNN, although the

Figure 5: A very narrow two class data set

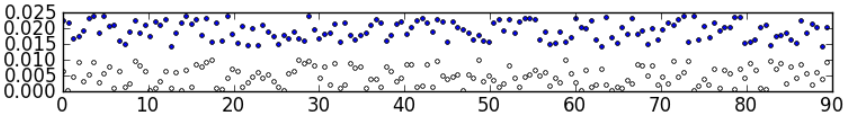
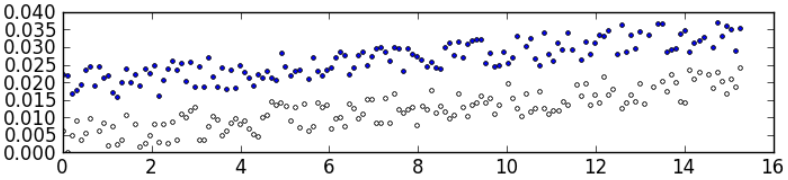


Figure 6: After LMNN learning



result is still not very good, there are significant improvements in each of the criterion and in kNN criterion, now the algorithms do the right prediction more than a half.

6.2.2 LMNN with non-linear data structure

Here is the non-linear data structure we use to indicate that LMNN can not work with non-linear structure. It is generated similarly with the linear data. For the first class $\{(x_i, y_i)|x_i = 10\cos(2\pi i/N)+U(0, 0.01)\ , y_i = 10\sin(2\pi i/N)+U(0, 0.01), i = 1, ..., N\}$, and the second class is $\{(x_i, y_i)|x_i = 10.015\sin(2\pi i/N)+U(0, 0.01)\ , y_i = 10.015\sin(2\pi i/N) + U(0, 0.01), i = 1, ..., N\}$. where U is the uniform distribution and $N = 150$.
The figure of this data is showed blow.

As being showed, this time after projection, the data was projected to a new circle with radius about 15. The algorithm did nothing but scale the coordinate in each direction with scaler about 1.5. It failed to learn.

metric	svm classifier	knn classifier	knn retrival
lmnn	17.16	19.16	37.72
euclidean	17.0	19.164	37.72
projection L	22.0	17.64	38.5

Table 2: The result of LMNN learning and an arbitrary projection on the non-linear data

The result also shows that the LMNN failed to learn a better, and L is an

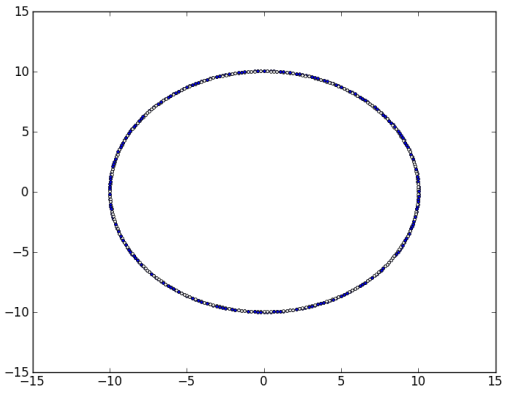


Figure 7: the graph of the nonlinear data, in the graph two classes are nearly over lapped

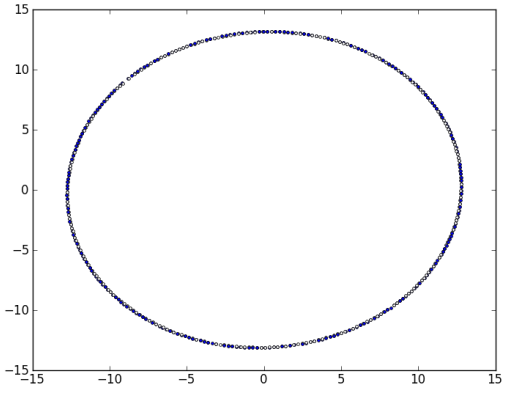


Figure 8: After LMNN learning

arbitrary projecting(, and here it is $\begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$) which gives a better result in both SVM classifier and knn retrieval.

6.2.3 LMNN with over fitting

In our experiment, an interesting over fitting case was observed that LMNN will get a distance metric even worse than the Euclidean distance metric with a particular kind of data set. Here is the data set. The points in the first class is :
$\{(x_i, y_i)|x_i = 5i/N + U(0, 0.01), y_i = U(0, 0.01), i = 1, ..., N\} \cup \{(x_i, y_i)|x_i = 5i/N + U(0, 0.01), y_i = U(0.022, 0.32), i = 1, ..., N\}$,
and the second class is :
$\{(x_i, y_i)|x_i = 5i/N + U(0, 0.01), y_i = U(0.011, 0.021), i = 1, ..., N\} \cup \{(x_i, y_i)|x_i = 5i/N + U(0, 0.01), y_i = U(0.033, 0.43), i = 1, ..., N\}$.
The algorithm gives a Mahanalobis metric which will introduce error instead of eliminate it. This is a kind of overfitting.

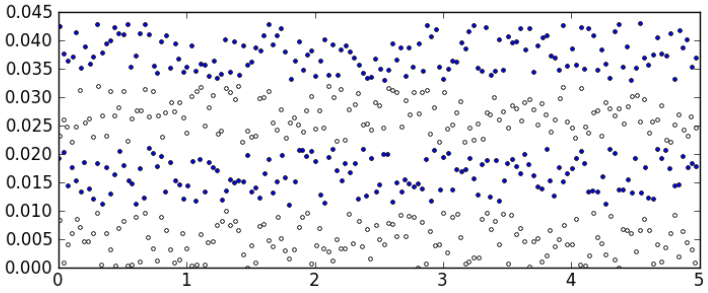


Figure 9: The original structure of data set

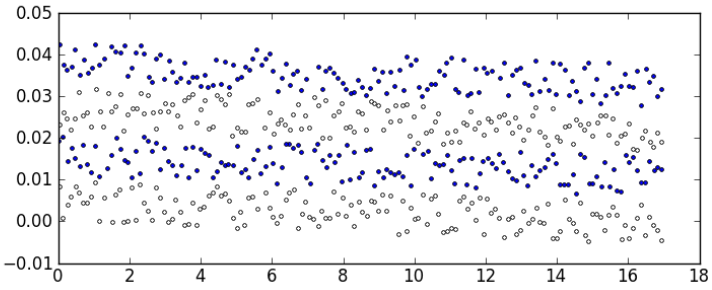


Figure 10: After Learned the result become even worse

The reason can be explained like this. The algorithm pick the target neighbour in another lane of the same class. this will make the learned L to shrink the y coordinate. Which is really bad. The figure 6.2.3 shows the reason how the over fitting occurred. The algorithm should grab B close to A. However because C is close to A with Euclidean distance, it pick C as the target neighbour.

Then the algorithm try to shrink the distance from A to C via linear projection. which will make the two class even closer and introduce error.

Figure 11: Schematic illustration of Over fitting in LMNN.

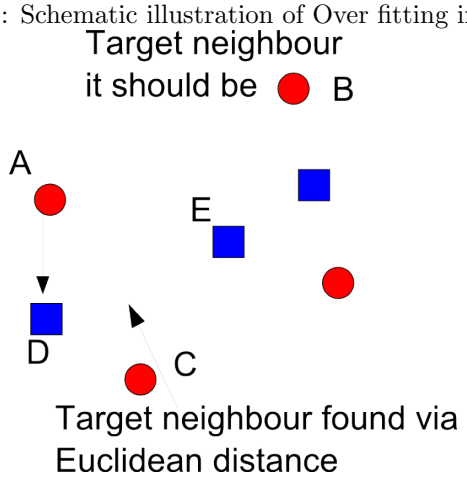


Table 3: This table shows some data

metric	svm classifier	knn classifier	knn retrieval
lmnn	50.5	8.33	34.72
euclidean	51.16	26.83	41.17

Here is the criterions on that data. It is cleared showed that the result is worse than do nothing.

6.2.4 LMNN with real problem

Here is the 3 criterion on data Iris and breast cancer which is from uci repository[]. It shows that with Iris, the algorithm doesn't improve the result a lot. However in breast cancer. It improves the result of SVM classifier a lot. This may be the reason that the data structure of breast cancer is more local and the LMNN which aim to improve the nearest neighbour retrieval will bring benefit to that kind of data set.

Table 4: The learning result of breast cancer compared to Euclidean distance

metric	svm classifier	knn classifier	knn retrieval
lmnn	96.24	95.35	94.28
euclidean	62.84	92.85	92.02

Table 5: The learning result of Iris compared to Euclidean distance

metric	svm classifier	knn classifier	knn retrieval
lmnn:	93.0	94.0	92.67
euclidean	93.0	94.0	92.67

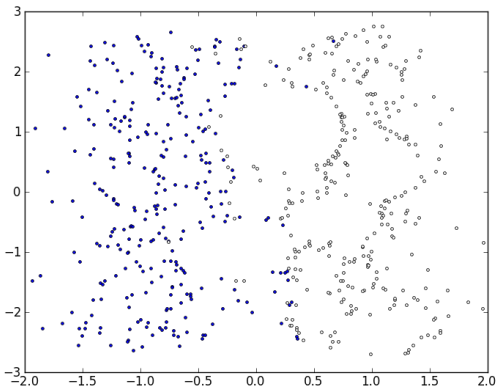
6.3 Isomap

6.3.1 Isomap with Nonlinear data

As mentioned above, isomap is designed to unfold the nonlinear data. To justify this, we prepared two artificial data set.

- 3/4 cylinder
  The first class is: $\{(x_i, y_i, z_i)|x_i = \cos(\frac{3\pi i}{4N}) + N(0, 0.01), y_i = \sin(\frac{3\pi i}{4N}) + N(0, 0.01), z_i = N(0, 0.3), i = 1, ..., N\ \}$,
  The second class is: $\{(x_i, y_i, z_i)|x_i = \cos(\frac{3\pi i}{4N}) + N(0, 0.01), y_i = \sin(\frac{3\pi i}{4N}) + N(0, 0.01), z_i = N(1, 0.3), i = 1, ..., N\ \}$. The isomap can perfectly unfolding and embedding this data into 2-dimensional space.

Figure 12: Embedding 3/4 cylinder into 2D space

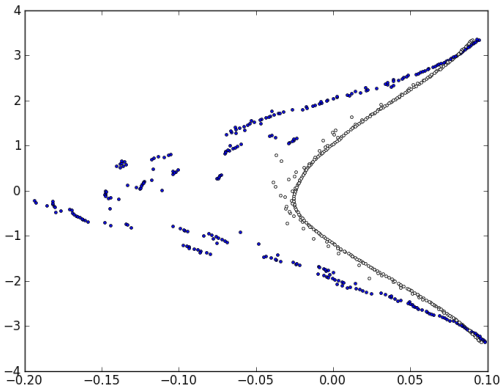


- 3/4 circle
  The first class is: $\{(x_i, y_i)|x_i = \cos(\frac{3\pi i}{4N}) + N(0, 0.01)\ ,\ y_i = \sin(\frac{3\pi i}{4N}) + N(0, 0.01), i = 1, ..., N\}$,
  The second class is: $\{(x_i, y_i)|x_i = 1.1\cos(\frac{3\pi i}{4N}) + N(0, 0.01)\ ,\ y_i = 1.1\sin(\frac{3\pi i}{4N}) + N(0, 0.01), i = 1, ..., N\}$.
  The figure 13 is a 2D dimension data set. We apply Isomap to try to find a good unfold in 2D space. The Isomap give us a fair result. The nonlinear structure still exists but much expended than 3/4 circle.

6.3.2 Isomap with nature data

Generally the result of the isomap is quite good. Although embedded into a lower dimension. The loss in the sense of the three criterion compared to the baseline is not significant. Here breast cancer is a 31 dimension data set and Iris is a 4 dimension data set. Isomap map these data into 2 dimension linear space which is a very strong data embedding. The result show in table 6.3.2 clearly show that the Isomap really hold the structure of the data when embedding.

Figure 13: Unfolding 3/4 circle into 2D space



data	SVM		kNN classifier		kNN retrieval	
	isomap	Euclidean	isomap	Euclidean	isomap	Euclidean
breast cancer	62.85	62.85	89.64	92.85	88.45	92.02
iris	94.0	93.0	96	94.0	93.66	92.66

Table 6: Comparing Result between isomap and Euclidean

6.4 SimpleNPKL

SimpleNPKL is a kernel learning algorithm with semi-supervising information. The performance of Simple NPKL is closely related to the information given to it. here is the experiment result. It is clearly show that the simpleNPKL is sensitive to the information. With 70% information, the algorithm perform better than only give half of the points with constraints. However compare to the baseline, the euclidean distance the learned is not good even bad. This might because that the kernel learned does not fit the algorithm we use. And the criterion is not fair to that algorithm.

Table 7: SimpleNPKL result with data where 50% of examples are assigned side information.

data	svm classifier	knn classifier	knn retrieval
breast cancer	70.35	58.57	55.29
Iris	61.0	51.0	51.66

Table 8: SimpleNPKL result with data where 70% of examples are assigned side information.

data	svm classifier	knn classifier	knn retrieval
breast cancer	81.78	76.25	73.92
Iris	79.0	74.0	71.66

6.5 New Approach on isomap

After introduce more information to the isomap, the performance is really improved. In Iris data with SVM classifier, it even reduced about 20% errorate. Normally, it can reduce a error rate at about 5%.

Table 9: Semi-isomap perform better than isomap

data	SVM		kNN classifier		kNN retrieval	
	isomap	semi-isomap	isomap	semi-isomap	isomap	semi-isomap
breast cancer	62.85	62.85	89.64	90.00	88.45	88.45
iris	94.0	95.0	96	96.0	93.66	93.0
circle	91.00	91.66	90.66	92.33	86.0	87.88

6.6 Summery

As being showed in table 10 11 12, the LMNN and Isomap perform good that LMNN can improve some results significantly and do nearly no worse than the Euclidean in other case. Isomap can find a lower embedding with minor loss. The performance of SimpleNPKL is not good, only be better than baseline on the case that ding kNN retrieval on breast cancer data set.

Table 10: Overall Result with SVM Classifier

data	Euclidean	LMNN	isomap	semi-isomap	SimpleNPKL
breast cancer	62.85	96.24	62.85	62.85	81.78
iris	93.0	93.0	94.0	95.0	79.0
circle	93.33	92.0	91.00	91.66	69.66

Table 11: Overall Result with kNN Classifier

data	Euclidean	LMNN	isomap	semi-isomap	SimpleNPKL
breast cancer	92.85	95.35	89.64	90.00	76.25
iris	94.0	94.0	96.0	96.0	74.0
circle	94.00	94.00	90.66	92.33	69.66

Table 12: Overall Result with kNN Retrieval

data	Euclidean	LMNN	isomap	semi-isomap	SimpleNPKL
breast cancer	92.02	94.28	88.45	88.45	73.92
iris	92.66	92.66	93.66	93.0	71.66
circle	91.22	91.11	86.0	87.88	69.44

7 Future Work

As the result of experiment showed, each algorithm has its particular characteristic. However, these algorithm are just tested in toy data set which the size of is small and the structure is simple. It is necessary to do some more experiments with adult data set to simulate the true practical situation.
The future work can also do something to combine the strong points of there algorithms to find a stronger distance learning. For example, The LMNN is quite good when dealing with data with linear structure with supervised information. and Isomap is good at dealing with data with nonlinear structure. If we can combine them together to obtain a strong distance learning algorithm to dealing with data with nonlinear structure and supervised information. Our approach on semi-supervised isomap had already try to do so and achieve a miner progress. There should be more work to do in this area. Another future might be avoiding the over fitting in LMNN although the situation to trigger that phenomenon is rare. But if we can to do so, then we are able to dealing with data with multi layers, this will also benefit clustering works.
Although NPKL can automatically learn a kernel without parameter. The result is not friendly enough to other algorithms and meaningless to people. It is worth to develop some way to study the structure to learn the data structure of the kernel and then some algorithm with automatically parameter would be able to developed.

8 Conclusion

The distance learning algorithm with different approaches can improve the performance of classification and retrieval algorithms. However it is not always the case. Each algorithm has its own characteristic. The learning algorithm was used in a situation that unfit to its feature. To be fortunate, it is just waste time and calculation. To be worse, it will make the whole performance even worse. It is necessary to understand the algorithm clearly and doing some experiment before using it. In our experiment, LMNN works really good when dealing with data with linear structure, and learn little when data with nonlinear structure was given. Sometimes it will occur over fitting. Isomap can do really good embedding with nonlinear structure, but it not guarantee that the map data is linear in the new space. The calculation cost should be very high when the size of data set is huge. It is not a good idea to use it in all the embedding cases. The performance of SimpleNPKL is very poor. It might because that the criterion or the data set we pick is not suitable to the algorithm. When side information was introduced, The performance of Isomap can be improved but not significant. Overall, with different situations and different data, it is tricky to find a good distance learning algorithm. To find an distance learning algorithm which work well in most situations is still an important task to fulfil in the machine learning area.

References

[1] N. Shental D. Weinshall A. Bar-Hillel, T. Hertz. Learning distance functions using equivalence relations. *International Conference on Machine Learning (ICML)*, 2003.

[2] Fei Sha Jitendra Malik Andrea Frome, Yoram Singer. Learning globally-consistent local distance functions for shape-based image retrieval and classification. *ICCV*, 2007.

[3] Christopher M. Bishop. Pattern recognition and machine learning. 2007.

[4] Michael I. Jordan Stuart Russell Eric P. Xing, Andrew Y. Ng. Distance metric learning, with application to clustering with side-information. *NIPS*, 2002.

[5] John C. Langford Joshua B. Tenenbaum, Vin de Silva. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000.

[6] J. M. Bibby Kanti V. Mardia, J. T. Kent. Multivariate analysis. 1979.

[7] A. Zisserman M. Pawan Kumar, P.H.S. Torr. An invariant large margin nearest neighbour classifier. 2007.

[8] Lawrence K. Saul Sam T. Roweis. Nonlinear dimensionality reduction by locally linear embedding. *science*, 2000.

[9] Lieven Vandenberghe Stephen Boyd. Convex optimization. 2004.

[10] Daphna Weinshall Tomer Hertz, Aharon Bar-Hillel. Learning distance functions for image retrieval. *CVPR*, 2004.

[11] Dacheng Tao Jianzhuang Liu Wei Liu, Xinmei Tian. Constrained metric learning via distance gap maximization. *Proc. AAAI*, 2010.

[12] Blitzer J. C. Saul L. K. Weinberger, K. Q. Distance metric learning for large margin nearest neighbor classification. *NIPS*, 18, 2006.

[13] Saul L. K. Weinberger, K. Q. Distance metric learning for large margin classification. *Journal of Machine Learning Research*, 10, 2009.

[14] Jinfeng Zhuang, Ivor W. Tsang, and Steven C. H. Hoi. Simplenpkl: simple non-parametric kernel learning. 382, 2009.