CS 245 Project 2: Distance Metric

Baochen Yang, Yunhao Zhang, Yiqun Diao

Abstract—In the field of machine learning, a typical kind of classification algorithms are based on distances between samples, for example, K-Nearest Neighbors (KNN) algorithm. Thus, how to calculate distances between samples becomes a hot research problem. In this paper, we implemented various distance metrics and then used KNN algorithm to test their performance on Animals with Attributes (AwA2) dataset. We evaluated a total of 14 distance metrics, including 4 simple metrics (Manhattan distance, Euclidean distance, Chebyshev distance and Cosine distance) and 10 metric learning algorithms (LMNN, LFDA, MLKR, NCA, RCA, ITML, LSML, MMC, SDML and Supervised RCA). Then we analyzed our results in detail and found that Euclidean distance performs best among simple metrics and even beats 9 out of 10 metric learning algorithms on AwA2 dataset. Only LMNN method slightly outperforms Euclidean distance (the baseline in each experiment) by less than 1%. Based on those analysis, We finally came to a conclusion that (1) Euclidean distance is the best choice considering both performance and complexity; (2) metric learning algorithms may fail to perform well due to the noise in the dataset.

Index Terms—simple metric, metric learning, evaluation

1 Introduction

In the past decade, Artificial Intelligence has greatly changed our life, based on dozens of effective machine learning algorithms. Among machine learning algorithms, one typical kind is designed to extract features from the distances between samples. The most classical and famous one is called K-Nearest Neighbors (KNN) algorithm, which refers to the neighbors of one sample to make predictions. The choice of distance metric can significantly affect the results of these algorithms, so it is rewarding to figure out how to reasonably measure the distances between samples or between distributions.

Generally, distance metrics can be roughly divided into two categories: simple distance metrics and metric learning methods. Simple distance metrics mean that we have a fixed formula to calculate the distance. For vector metric, the most well-known one is Minkowsky distance, or L_p distance. It just calculates the L_p norm of the difference of two vectors, containing a bunch of famous geometric distance, e.g. Manhattan distance $(p = 1)$, Euclidean distance $(p = 2)$, Chebyshev distance $(p = +\infty)$ and so on. Another simple vector metric is cosine distance, which calculates distances based on the angle between two vectors. These methods are easy and with clear geometric explanations. It is easy to verify that they all satisfy three rules of distances: non-negative, symmetric and triangle inequality. There are also simple metrics for distributions, such as Earth's Mover Distance (EMD), Maximum Mean Discrepancy (MMD), KL divergence, JD divergence, Bregman divergence and so on. But in this paper, we only focused on metrics for vectors.

Compared to simple distance metrics, metric learning methods do not have a fixed mathematical equation to calculate the distances. Instead, they set an objective function and learn a distance metric to optimize this function using gradient descendent or other optimization tools. Examples include Large Margin Nearest Neighbor (LMNN), Local Fisher Discriminant Analysis (LFDA), Metric Learning for Kernel Regression (MLKR), Neighborhood Components Analysis (NCA) and so on.

In this paper, we evaluated performances of both simple distance metrics and metric learning methods on Animals with Attributes (AwA2) data set by comparing the KNN classification accuracy based on different distance metrics. We made huge effort to finish a comprehensive experiment of a total of 14 different methods. We also made a detailed analysis on pros and cons of those distance metrics.

2 Method

2.1 K-nearest Neighbors (KNN)

The k-nearest neighbors algorithm (KNN) is a non-parametric, lazy learning method for regression and classification. We use KNN as a classifier in this project. As for KNN classification, an object is assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). Therefore, there are two main factors that will affect the performance of a KNN classifier: 1) **distance** used to determine "nearest neighbors"; 2) the parameter **k**.

As KNN is a lazy learning method, the training phase only stores the feature vectors and labels of training set. In testing phase, distance between every pair of samples is computed and label is assigned as the most frequent label among the k samples nearest to that query point. Therefore, KNN suffers great computational complexity. We have learned in Project 1 that principal component analysis (PCA) can reduce dimensionality of high dimensional data and still keeps most information used for classification. To reduce time complexity, we first use PCA to reduce data dimension to 50 and then use KNN for classification.

2.2 Simple Distance Metrics

KNN needs to compute the distance between pairs of data points. In this section, distance is computed by a deterministic function.

2.2.1 Minkowski Distance

The Minkowski distance of order p between two points $\mathbf{x} = \{x_1, x_2, \cdots x_m\}$ and $\mathbf{y} = \{y_1, y_2, \cdots y_m\}$ is defined as:

$$d_p(\mathbf{x}, \mathbf{y}) = (\sum_{i=1}^{m} |x_i - y_i|^p)^{\frac{1}{p}} \tag{1}$$

Assign different value to p, we can get different distance metrics. For example:

- Manhattan distance:
$$d_{manh}(\mathbf{x}, \mathbf{y}) = d_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} |x_i - y_i|$$
- Euclidean distance:
$$d_{eucl}(\mathbf{x}, \mathbf{y}) = d_2(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T(\mathbf{x} - \mathbf{y})}$$
- Chebyshev distance:
$$d_{cheb}(\mathbf{x}, \mathbf{y}) = \lim_{p \to +\infty} d_p(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i|$$

2.2.2 Cosine Distance

Cosine similarity measures similarity of two data points by the angle between them:

$$cos(\mathbf{x}, \mathbf{y}) = \frac{x^T y}{\|\mathbf{x}\| \|\mathbf{y}\|} \tag{2}$$

$cos(\mathbf{x}, \mathbf{y})$ ranges between $[-1, 1]$. The more similar $\mathbf{x}$ and $\mathbf{y}$ are, the bigger $cos(\mathbf{x}, \mathbf{y})$ is. To scale distance to $[0, 1]$ and transform similarity to distance, cosine distance is defined as:

$$d_{cosine}(\mathbf{x}, \mathbf{y}) = \frac{1}{2}(1 - cos(\mathbf{x}, \mathbf{y})) \tag{3}$$

2.3 Large Margin Nearest Neighbor(LMNN)

Large Margin Nearest Neighbor(LMNN) [1] is a supervised method specially designed for KNN. The goal is to learn a Mahanalobis distance metric making k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin. Formally, the goal is to learn a linear transformation $L : R^d \to R^d$, which we will use to compute the squared distance as:

$$D(x_i, x_j) = \|L(x_i - x_j)\|^2 = (x_i - x_j)^T L^T L(x_i - x_j) \tag{4}$$

For each sample (x_i, y_i), we select k target neighbors (x_j, y_j) for it. Target neighbors are other samples with the same label y_i that we wish to have minimal distance to x_i. Without prior knowledge, target neighbors are often selected as the k nearest samples labels y_i in Euclidean space. We use $\eta_{ij} \in \{0, 1\}$ to denote whether x_j is a target neighbor of x_i. $\eta_{ij} = 1$ if x_j is a target neighbor.

The loss function is defined as:

$$\begin{aligned} L = &\sum_{ij} \eta_{ij} D(x_i, x_j) \\ &+ c \sum_{ijl} \eta_{ij}(1 - y_{jl})[1 + D(x_i, x_j) - D(x_j, x_l)]_+ \end{aligned} \tag{5}$$

Here $y_{jl} \in \{0, 1\}$, $y_{jl} = 0$ if $y_j = y_l$. $[z]_+ = \max(z, 0)$ denotes the standard hinge loss.

The loss function in 5 has two terms. The first term is to minimize the distance between x_i and its target neighbors. The second term is trying to separated samples with other labels and target neighbors by a margin 1. Ideally, we want $D(x_j, x_l) \geq D(x_i, x_j) + 1$. As hard margin is impossible to achieve when dataset is large, we have to use hinge loss to build soft margin.

2.4 Local Fisher Discriminant Analysis (LFDA)

Local Fisher Discriminant Analysis (LFDA) [2] is a linear supervised dimensionality reduction method. It is particularly useful when dealing with multi-modality, where one ore more classes consist of separate clusters in input space. The main idea of LFDA is same as Linear Discriminant Analysis(LDA): to minimum the "within-class scatter matrix" and maximum the "between-class scatter matrix". Formally, the "within-class scatter matrix" $\tilde{S}^{(w)}$ and "between-class scatter matrix" $\tilde{S}^{(b)}$ are defined as:

$$\begin{aligned} \tilde{S}^{(w)} &= \frac{1}{2} \sum_{i,j}^{n} \tilde{W}_{i,j}^{(w)} (x_i - x_j)(x_i - x_j)^T \\ \tilde{S}^{(b)} &= \frac{1}{2} \sum_{i,j}^{n} \tilde{W}_{i,j}^{(b)} (x_i - x_j)(x_i - x_j)^T \end{aligned} \tag{6}$$

where

$$\tilde{W}_{i,j}^{(w)} = \begin{cases} A_{i,j}/n_l, & \text{if } y_i = y_j = l \\ 0, & \text{if } y_i \neq y_j \end{cases} \tag{7}$$

$$\tilde{W}_{i,j}^{(b)} = \begin{cases} A_{i,j}(1/n - 1/n_l), & \text{if } y_i = y_j = l \\ 1/n, & \text{if } y_i \neq y_j \end{cases} \tag{8}$$

In Equation 7 and 8, $A_{i,j}$ is the (i, j)-th entry of the affinity matrix A. $A_{i,j}$ is large if x_i and x_j are close. We can give different weights to different pairs (x_i, x_j) using affinity matrix A. A imports local property into Fisher Discriminant Analysis. It can be proved that if $A_{i,j} = 1$, $\tilde{S}^{(w)}$ and $\tilde{S}^{(b)}$ equal to those we defined in LDA.

In this paper, we use local scaling to compute affinity matrix A:

$$\begin{aligned} A_{i,j} &= \exp - \frac{(\|x_i - x_j\|^2)}{\sigma_i \sigma_j} \\ \sigma_i &= \|x_i - x_i^{(K)}\| \end{aligned} \tag{9}$$

where $x_i^{(K)}$ is the K-th nearest neighbor of x_i.

Then the learning problem becomes derive the LFDA transformation matrix T_{LFDA}:

$$T_{LFDA} = \arg\max_T [\text{tr}((T^T S^{(w)} T)^{-1} T^T S^{(b)} T)] \tag{10}$$

2.5 Metric Learning for Kernel Regression (MLKR)

Kernel regression [3] is a well-established method for nonlinear regression in which the target value for a test point is estimated using a weighted average of the surrounding training samples. The weights are typically obtained by applying a distance-based kernel function to each of the samples, which presumes the existence of a well-defined distance metric. MLKR algorithm learn a task-specific metric over the input space in which small distances between two vectors imply similar target values. This metric gives rise to an appropriate kernel function with parameters set entirely from the data.

In kernel regression, the value of $\hat{y}_i \approx f(\vec{x}_t)$ is approximated by a weighted average of the training samples:

$$\hat{y}_i = \frac{\sum_{j \neq i} y_j k_{ij}}{\sum_{j \neq i} k_{ij}} \tag{11}$$

where $k_{ij} = k(\vec{x}_i, \vec{x}_j) \geq 0$ is referred to as the kernel function.

MLKR applies to any distance-based kernel function $k(\vec{x}_i, \vec{x}_j) = k_D(D_\theta(\vec{x}_i, \vec{x}_j))$ with differentiable dependence on parameter θ specifying the distance function D_θ. Specifically, MLKR consists of setting initial values of θ, and then adjusting the values using a gradient descent procedure:

$$\Delta\theta = -\epsilon \frac{\partial \mathcal{L}}{\partial \theta} \tag{12}$$

where ϵ is an adaptive step-size, and the loss function $\mathcal{L}$ is the cumulative leave-one-out quadratic regression error of the training samples:

$$\mathcal{L} = \sum_i (y_i - \hat{y}_i)^2 \tag{13}$$

with $\hat{y}_i$ is defined as in (11).

2.6 Neighborhood Components Analysis (NCA)

NCA [4] is a supervised learning algorithm. Its purpose is to obtain a linear space transfer matrix by learning on the training set to maximize the average leave-one classification effect in the new conversion space. Therefore, the key of the algorithm is how to learn to obtain a positive definite matrix A related to the spatial transformation matrix. The matrix A can be obtained by defining a differentiable objective function of A and solving it using the iterative method.

We begin with a labeled data set consisting of n real-valued input vectors $x_1, ..., x_n$ in $\mathcal{R}^D$ and corresponding class labels $c_1, ..., c_n$. NCA essentially wants to learn a positive definite matrix $\mathbf{A}$ related to the conversion matrix $\mathbf{Q}$ such that $\mathbf{Q} = \mathbf{A}^T\mathbf{A}$ is established, so that the metric function can be written as:

$$d(x,y) = (x-y)^T Q (x-y) = (Ax - Ay)^T (Ax - Ay) \tag{14}$$

When calculating the error using the leave-one-out method, the error function is not continuous with respect to A, so we introduce a differentiable softmax function:

$$p_{ij} = \frac{exp(-||Ax_i - Ax_j||^2)}{\sum_{k \neq i} exp(-||Ax_i - Ax_k||^2)}, \qquad p_{ii} = 0 \tag{15}$$

where p_{ij} denotes the probability that x_i will eventually choose x_j as a neighbor and inherit its class label c_j in the process of randomly selecting neighbors. Then the probability that x is correctly classified is:

$$p_i = \sum_{j \in C_i} p_{ij} \tag{16}$$

where C_i denotes the set of points in the same class as i. The objective we maximize is the expected number of points correctly classified under this scheme:

$$f(A) = \sum_i \sum_{j \in C_i} p_{ij} = \sum_i p_i \tag{17}$$

Differentiating f with respect to the transformation matrix A yields a gradient rule which we can use for learning (denote $x_{ij} = x_i - x_j$):

$$\frac{\partial f}{\partial A} = -2A \sum_i \sum_{j \in C_i} p_{ij} (x_{ij} x_{ij}^T - \sum_k p_{ik} x_{ik} x_{ik}^T) \tag{18}$$

Through continuous iterative optimization, the optimal solution of A can be obtained.

2.7 Relevant Components Analysis (RCA)

Relevant Component Analysis (RCA) [5] is a method that seeks to identify and down-scale global unwanted variability within the data. The method changes the feature space used for data representation, by a global linear transformation which assigns large weights to ”relevant dimensions” and low weights to ”irrelevant dimensions”.

These”relevant dimensions” are estimated using chunklets. We define a chunklet as a subset of points that are known to belong to the same although unknown class; Chunklets are obtained from equivalence relations by applying a transitive closure. The RCA transformation is intended to reduce clutter, so that in the new feature space, the inherent structure of the data can be more easily unraveled. The method can be used as a preprocessing step for the unsupervised clustering of the data or nearest neighbor classification.

For a training set with n training points in k chunklets, the algorithm is efficient since it simply amounts to computing:

$$C = \frac{1}{n} \sum_{j=1}^{k} \sum_{i=1}^{n_j} (x_{ji} - \hat{m}_j)(x_{ji} - \hat{m}_j)^T \tag{19}$$

where chunklet j consists of $\{x_{ji}\}_{i=1}^{n_j}$ with a mean $\hat{m}_j$. The inverse of C^{-1} is used as the Mahalanobis matrix.

2.8 Information Theoretic Metric Learning (ITML)

ITML [6] is a semi-supervised learning approach. It minimizes the differential relative entropy between two multivariate Gaussians under constraints on the distance function, which can be formulated into a Bregman optimization problem by minimizing the LogDet divergence subject to linear constraints.

The objective function of ITML is shown as follows:

$$\begin{aligned} \min \quad & KL(p(x; M_0) || p(x; M)) \\ s.t. \quad & d_M(x_i, x_j) \leq u, \quad (x_i, x_j) \in S \\ & d_M(x_i, x_j) \geq l, \quad (x_i, x_j) \in D \end{aligned} \tag{20}$$

In the objective function above, M represents the metric to be learned, M_0 represents the prior metric. Two points are considered similar if the Mahalanobis distance between them is smaller than a given upper bound, i.e., a relatively small value u; two points are considered dissimilar if the Mahalanobis distance between them is larger than a given lower bound, i.e., a sufficiently large value l. S represents all pairs of similar points and D represents all pairs of dissimilar points. The objective function is to minimize the KL divergence between M and M_0 to avoid overfitting, and make the learned metric M to satisfy the threshold. Since the "closeness" between M and M_0 is measured via KL divergence, this method measures the difference between two distributions in an entropy perspective. Therefore, this metric learning method is called an information-theoretic approach.

2.9 Least Squared-residual Metric Learning (LSML)

LSML algorithm wants to keep the partial ordering of distances between each two samples after the projection. It also learns a Mahalanobis matrix M to project samples to a new space which facilitates the classification, so we need to tell the label information to supervise this learning process.

Let us denote $X = \{x_1, x_2, ..., x_n\}$ as the samples. First, we calculate a set $S \subset \{1, 2, ..., n\}^4$ such that

$$S = \{(a, b, c, d) \mid d(x_a, x_b) < d(x_c, x_d)\} \tag{21}$$

where $d(x_a, x_b) = \|x_a - x_b\|_2$ means the Euclidean distance between sample x_a and x_b.

After we apply the Mahalanobis distance matrix, we would like to keep the partial ordering of distances. For example, if $(a, b, c, d) \in S$, which means $d(x_a, x_b) < d(x_c, x_d)$, we want $d_M(x_a, x_b) < d_M(x_c, x_d)$, where $d_M(x_a, x_b)$ equals the Mahalanobis distance between x_a and x_b. If $d_M(x_a, x_b) > d_M(x_c, x_d)$ in the projection space, we shall punish this behavior using the loss function. So the loss function is defined as

$$\begin{gathered} L = D_{ld}(M, M_0) + \sum_{(a,b,c,d)\in S} H(d_M(x_a, x_b) - d_M(x_c, x_d)) \\ H(x) = \begin{cases} 0 & x \le 0 \\ x^2 & x > 0 \end{cases} \\ D_{ld}(M, M_0) = tr(MM_0) - \log\ \det(M) \end{gathered} \tag{22}$$

where M_0 is a prior distance matrix and is generally set as identity matrix if there is no prior knowledge.

Let us examine (22). For each $(a, b, c, d) \in S$, if $d_M(x_a, x_b) < d_M(x_c, x_d)$ remains, the loss won't increase. However, when $d_M(x_a, x_b) > d_M(x_c, x_d)$ happens, $[d_M(x_a, x_b) - d_M(x_c, x_d)]^2$ will be added to the loss to punish this case. Hence we call this algorithm Least Square-residual, because it has a squared loss for residual part. $D_{ld}(M, M_0)$ makes sure that M is neither too small nor just the trivial case. If values in M are too small, each distance becomes close to 0, and the squared loss $H(x)$ will become meaningless. If $M = I$, there is no error in the distance ordering, but nothing is learned! This is why $D_{ld}(M, M_0)$ should be added to the loss based on some prior knowledge M_0.

2.10 Mahalanobis Metric for Clustering (MMC)

MMC [7] is a classical method which aims to minimize the sum of Mahalanobis distances between samples of the same clusters, while making the sum of Mahalanobis distances between samples of different clusters larger than a threshold. Thus, it becomes a convex optimization problem, which can be solved efficiently. MMC has the similar idea as Linear Discriminative Analysis (LDA) in dimensionality reduction, so it also needs labels to supervise its training. To formulate its idea rigorously in mathematics, let us denote $X = \{x_1, x_2, ..., x_n\}$ as the samples. With labels of each sample, we can calculate two sets as follows:

$S = \{(x_i, x_j) \mid x_i$ and x_j have the same label$\}$
$D = \{(x_i, x_j) \mid x_i$ and x_j have different labels$\}$

Thus, we can write the objective function, i.e.

$$\begin{aligned} \min_M \sum_{(x_i,x_j)\in S} d_M(x_i, x_j) \\ s.t. \sum_{(x_i,x_j)\in D} d_M(x_i, x_j) \ge 1 \end{aligned} \tag{23}$$

where M is a positive semi-definite matrix, and $d_M(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)}$ is the Mahalanobis distance between x_i and x_j.

The constraints in (23) makes sure that we can split clusters, rather than get a trivial solution and mix up every clusters, for example $M = O$.

2.11 Sparse Determinant Metric Learning (SDML)

SDML [8] furthers the idea of MMC. It also wants to maximize the Mahalanobis distances between different clusters, while minimizing the Mahalanobis distances inside each cluster, so it is a supervised metric learning algorithm. Besides that, it also prefers M to be sparser. Moreover, we can set a prior M_0 with our prior knowledge on a specific problem, and the algorithm will prefer M closer to M_0.

First, similar to MMC, we also calculate the similarity set S and dissimilarity set D:

$S = \{(x_i, x_j) \mid x_i$ and x_j have the same label$\}$
$D = \{(x_i, x_j) \mid x_i$ and x_j have different labels$\}$

Then the loss function intended to maximize inter-cluster distances while minimizing intra-cluster distances is

$$\begin{aligned} L_{split} &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} K_{i,j} d_M^2(x_i, x_j) \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} K_{i,j}(x_i^T M x_i + 2x_i^T M x_j + x_j^T M x_j) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} K_{i,j}(x_i^T M x_i + x_i^T M x_j) \\ K_{i,j} &= \begin{cases} 1 & (x_i, x_j) \in S \\ -1 & (x_i, x_j) \in D \end{cases} \end{aligned} \tag{24}$$

Second, if we want M to be sparser, we should use L1 loss on M. In this case, we do not want the elements in the diagonal of M to be 0, so we use the off-diagonal L1 loss where

$$L_{sparse} = \sum_{i=1}^{n}\sum_{j\ne i} |M_{i,j}| \tag{25}$$

Third, if we want M to be closer to our prior knowledge M_0, we should use the log-determinant divergence function:

$$D_{ld}(M, M_0) = tr(MM_0) - \log\ \det(M) \tag{26}$$

Combining (24), (25) and (26) together, we can derive the overall loss function of SDML:

$$\begin{aligned} \min_{M} \quad & L_{split} + \lambda L_{sparse} + \mu D_{ld}(M, M_0) \\ s.t. \quad & M \succeq 0 \end{aligned} \tag{27}$$

where $\lambda > 0$ and $\mu > 0$ are hyper parameters indicating the trade-off between those three loss terms.

2.12 Supervised Relative Components Analysis (RCA)

RCA can be used as a supervised metric learning algorithm. The Mahalanobis matrix is the weighted sum of covariance matrix for each cluster. It tries to assign larger weights to relevant dimensions and smaller weights to irrelevant ones. Since it is a supervised version, clusters are divided by label information.

Suppose the input X is projected to Y in a new space, RCA intends to maximize the mutual information $I(X, Y)$ while the weighted sum of covariance matrix stays below a given threshold. To formulate the problem in mathematical forms, we denote the K original clusters $x_1, ..., x_k$, each cluster x_i has samples $x_{i,1}, ..., x_{i,n_i}$. The projected samples are denoted as $y_{i,j}$ correspondingly. Then we have

$$\begin{aligned} & \max_{Y} \quad I(X, Y) \\ s.t. \quad & \frac{\sum_{i=1}^{k} \sum_{j=1}^{n_i} \|y_{i,j} - \mu_i\|^2}{\sum_{i=1}^{k} n_i} \leq C \\ & \mu_i = \frac{1}{n_i} \sum_{j=1}^{n_i} y_{i,j} \end{aligned} \tag{28}$$

where C is a constant to limit the weighted sum of covariance matrix.

3 Experiment

3.1 Data Set and Preprocessing

We evaluated the different distance metrics on Animals with Attributes (AwA2) data set. This data set consists pre-extracted 2048-dimension deep learning features for 37322 images of 50 animal classes. We split the images in each category into 60% for training and 40% for testing.

As the original features have dimension of 2048 and KNN is a lazy learning method whose computational complexity is highly related to the dimension, we have to reduce the dimensionality. In this experiment, we use PCA to reduce dimensionality from 2048 to 50. In Project 1, the experiment shows that this 50-dimensional feature still keeps most of the information for classification. The following experiments takes the preprocessed feature as input.

3.2 Hyperparameter Selection

The number of nearest neighbors k is a very important hyperparameter for KNN algorithm. In this section, we perform 5-fold cross-validation on training set to determine the optimal range for k. We only use Euclidean distance as distance metric. Because we only need to select a range for k, such simplification is reasonable. The validation result is shown in Table 1.

TABLE 1: 5-fold cross-validation accuracy for different K

K	Accuracy
2	85.51%
5	89.02%
10	89.31%
15	89.23%
20	88.94%
30	88.49%
50	87.51%
80	86.39%
100	85.88%

We can get the optimal range of k for Euclidean distance is $[5, 15]$. Smaller k is not robust to noise, and larger k makes the boundary for different classes unclear. In order to apply to other distance metrics, we expand the optional range to $[2, 30]$.

3.3 Simple Distance Metrics

We tried 4 different simple metrics mentioned in Section 2.2 and the results are shown in Table 2 and Figure 1.

TABLE 2: Classification Accuracy of Simple Distance Metrics

Acc / K \ Dist	Manh	Eucl	Cheb	Cosine
2	86.15%	86.66%	83.72%	86.22%
5	89.48%	89.90%	87.45%	89.17%
6	89.35%	89.78%	87.51%	88.86%
7	**89.66**%	**90.13**%	87.61%	89.17%
8	89.49%	90.07%	87.58%	89.01%
9	89.52%	90.09%	87.72%	89.22%
10	89.51%	89.87%	87.52%	89.15%
11	89.40%	89.96%	87.61%	**89.39**%
12	89.32%	89.74%	87.53%	89.12%
13	89.37%	89.75%	**87.80**%	89.21%
14	89.24%	89.66%	87.78%	89.07%
15	89.25%	89.77%	87.67%	89.00%
20	89.09%	89.56%	87.28%	88.72%
30	88.55%	89.14%	86.79%	88.30%

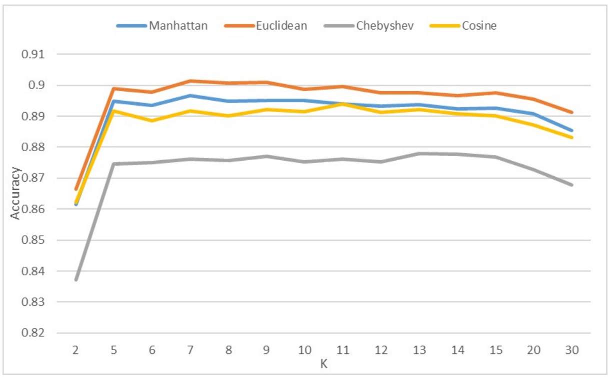

Fig. 1: Accuracy Curve of Simple Distance Metrics

Getting the result, we make the following observation and analysis:

- All these curves increase first and decrease after reaching the optimal value of K. This is because when K is too small, noise will affect classification greatly. However, when K is too large, including too many neighbors into the voting process makes the boundary between different clusters unclear.

- Chebyshev distance performs poorer than the other three. This is because Chebyshev distance only takes dimension with max value into consideration. Therefore, it does not fully use the information hidden in other dimensions.
- Cosine distance does not satisfy the triangle inequality. Therefore, $sklearn$ does not imply implement this method and we have to implement it by ourselves. Experiments shows that Cosine distance runs much slower than the other three and performance is poorer than Euclidean and Manhattan.

3.4 Large Margin Nearest Neighbor(LMNN)

As we have discussed in 2.3, LMNN is specially designed for KNN. The number of target neighbors k is an important parameter. We first set $k = 3$ as most users of LMNN do. After analysis, we infer that $k = K$(K is the number of nearest neighbors in KNN) may achieve better result. Results for this experiment is shown in Table 3.

TABLE 3: Test Accuracy of LMNN

K	Baseline	k = 3	k = K
2	86.66%	86.93%	86.94%
5	89.90%	90.15%	90.17%
10	89.87%	**90.15%**	**90.34%**
15	89.76%	90.08%	90.26%
20	89.56%	89.90%	–
30	89.12%	89.54%	–

From the result we can see that LMNN improves the performance of KNN. This is intuitive because the goal of LMNN is to pull target neighbors close and push away samples from different classes. By setting $k = K$ we further improve the performance of LMNN. In this way, we assume the closest neighbors in KNN are the target neighbors selected in LMNN. However, for LMNN, larger k means more memory consumption. Setting $k = K = 20$ requires 31.1 GB memory, which we are unable to allocate. For this reason, we didn't perform experiments for $k = 20$ and $k = 30$. Therefore, while using LMNN for KNN classifier, it is important to balance the performance improvement and memory consumption.

3.5 Local Fisher Discriminant Analysis (LFDA)

LFDA is derived from LDA by adding affinity matrix $A_{i,j}$ into scatter matrix. In this experiment, we use local scaling to compute affinity matrix. A turning hyperparameter for local scaling is k, and [9] demonstrated that $k = 7$ works well on the whole. Therefore, we use this setting through this experiment.

As we have analyzed in Section 2.4, the main idea for LFDA is to import local property into LDA. To find out whether this idea increase performance, we also tried LDA($n_components = 49$). The result for LFDA and LDA is shown in Table 4.

The result shows that performances of LFDA and Euclidean distance is similar. And LDA slightly outperforms LFDA, which means importing local property does not improve the performance. This is because LFDA is specially designed for multi-modality tasks where samples in a class form several separate clusters. But mutli-clusters classes may be rare in our dataset. Although LDA is often used as a dimensionality reduction method, the goal of it is same as metric learning: to find a projection that satisfies some properties. The result shows that LDA performs well and improves performance.

TABLE 4: Test Accuracy of LFDA and LDA

K	Baseline	LFDA	LDA
2	86.66%	86.96%	87.26%
5	89.90%	89.44%	**90.25%**
10	89.87%	89.50%	90.05%
15	89.76%	**89.54%**	89.89%
20	89.56%	89.46%	89.54%
30	89.12%	88.76%	89.28%

3.6 Metric Learning for Kernel Regression (MLKR)

MLKR is an algorithm of supervised metric learning, which learns a distance function by directly minimizing the leaveone-out regression error. Results for this experiment are shown in Table 5.

TABLE 5: Test Accuracy of MLKR

K	Baseline	MLKR
2	86.66%	82.65%
5	89.90%	86.40%
10	89.87%	86.18%
15	89.76%	86.18%
20	89.56%	86.23%
30	89.12%	85.60%

From the results we can see that MLKR does not perform well on AwA2 data set, since its accuracy is consistently below the baseline. We know that the MLKR algorithm is based on kernel regression, and we hope to construct a new feature space based on the original data to calculate the similarity. Therefore, the choice of feature space is crucial to the performance of the model. Without knowing the form after feature mapping, we do not know which kernel function is suitable, and the kernel function only implicitly defines this feature space. And actually the process of MLKR algorithm does not include the adjustment of kernel function. Therefore, the reason for the poor algorithm performance may be because we failed to select the appropriate kernel function.

3.7 Neighborhood Components Analysis (NCA)

NCA is a distance metric learning algorithm which aims to improve the accuracy of nearest neighbors classification compared to the standard Euclidean distance. It aims to maximize function $f(A)$, which is the the sum of probability of being correctly classified. Results for this experiment are shown in Table 6

From the results we can see that NCA does not perform well on AwA2 data set, since its accuracy is consistently below the baseline. The method of NCA is not to consider K nearest neighbors in the new feature space, but to consider

TABLE 6: Test Accuracy of NCA

K	Baseline	NCA
2	86.66%	83.83%
5	89.90%	86.93%
10	89.87%	86.88%
15	89.76%	87.05%
20	89.56%	86.68%
30	89.12%	86.08%

the entire data set as random nearest neighbors in the new space. Therefore, when we consider k-nearest neighbors in the new feature space, the accuracy may be affected. In addition, NCA measures the similarity in the new feature space by applying Euclidean distance. If combined with other distance metric methods, it may improve the performance of NCA on the AwA2 dataset.

3.8 Relative Components Analysis (RCA)

RCA learns a full rank Mahalanobis distance metric based on a weighted sum of in-chunklets covariance matrices. It applies a global linear transformation to assign large weights to relevant dimensions and low weights to irrelevant dimensions. Those relevant dimensions are estimated using “chunklets”, subsets of points that are known to belong to the same class. Results for this experiment are shown in Tabel 7

TABLE 7: Test Accuracy of RCA

K	Baseline	RCA
2	86.66%	84.55%
5	89.90%	87.38%
10	89.87%	87.08%
15	89.76%	87.40%
20	89.56%	86.63%
30	89.12%	86.18%

From the results, we can see that RCA does not perform well on AwA2 data set, since its accuracy is consistently below the baseline. The possible reason is that RCA reduces the global unnecessary variability in the data. Therefore, there is bound to be some information lost in the process of our fitting, but this part of information may still contribute to our subsequent classification problems. In other words, we made a compromise between the integrity of the information and the dimensions of the data, similar to the projection of data in the direction of the largest variance in PCA. This is why our performance has declined.

3.9 Information Theoretic Metric Learning (ITML)

ITML minimizes the (differential) relative entropy, aka Kullback–Leibler divergence, between two multivariate Gaussians subject to constraints on the associated Mahalanobis distance, which can be formulated into a Bregman optimization problem by minimizing the LogDet divergence subject to linear constraints. This algorithm can handle a wide variety of constraints and can optionally incorporate a prior on the distance function. Unlike some other methods, ITML does not rely on an eigenvalue computation or semi-definite programming. Results for this experiment are shown in Table 8.

TABLE 8: Test Accuracy of ITML

K	Baseline	ITML
2	86.66%	84.33%
5	89.90%	87.73%
10	89.87%	87.53%
15	89.76%	87.28%
20	89.56%	87.25%
30	89.12%	86.88%

From the results, we can see that ITML does not perform well on AwA2 data set, since its accuracy is consistently below the baseline. The possible reason is that there are two parameters u and l in our formula that are not confirmed by training, but are generated by our initialization. But these two parameters are our most important two constraints. Therefore, if these two parameters can be adjusted adaptively, the ITML method may get better results. In addition, what the data does not show is that ITML’s training speed is very fast. Compared with NCA and MLKR, ITML takes less time to complete the training.

3.10 Least Squared-residual Metric Learning (LSML)

LSML aims to maintain the partial ordering from the original space to projected space. Thus, it would not take too much time. Results for this experiment are shown in Table 9.

TABLE 9: Test Accuracy of LSML

K	**Baseline**	**LSML**
2	**85.51%**	**75.04%**
5	**89.02%**	**76.15%**
10	**89.31%**	**84.76%**
15	**89.23%**	**85.71%**
20	**88.94%**	**84.94%**
30	**88.49%**	**84.39%**

From the results, we can see that LSML does not perform well on AwA2 data set, since its accuracy is consistently below the baseline. Maybe it is because we supervise this learning process without any regularization, and it projects into a very good space for training data. If there is a direction that can improve training accuracy, the Mahalanobis matix will have a tendency to move there. However, for testing data, this way may spoil its partial ordering and lead to lower testing accuracy. Another possible explanation is that the Euclidean distance $d(x_a, x_b)$ is not a good measurement, and it cannot effectively represent the actual relationship between x_a and x_b. But LSML is based on the Euclidean distance and tries to maintain its partial ordering. If the latent information does not lie directly in the Euclidean distance, it will be difficult for LSML to perform well.

3.11 Mahalanobis Metric for Clustering (MMC)

MMC intends to minimize the sum of Mahalanobis distances between samples of the same clusters, while making the sum of Mahalanobis distances between samples of different clusters larger than a threshold. In our experiment, we put features into MMC function after doing PCA to reduce them to 50-d space.

TABLE 10: Test Accuracy of MMC

K	Baseline	MMC
2	85.51%	73.94%
5	89.02%	75.70%
10	89.31%	87.53%
15	89.23%	88.15%
20	88.94%	88.42%
30	88.49%	87.99%

As is shown in Table 10, MMC has relatively lower accuracy than the baseline method using Euclidean distances. This result makes us somewhat puzzled. Now it seems that using a complicated learning method does not necessarily guarantee a better classification result, especially in the cases that K is small. As we analyze, since the data set has a total of 50 categories, if we learn a projection matrix M, it is very likely that a few samples become lost in other clusters.

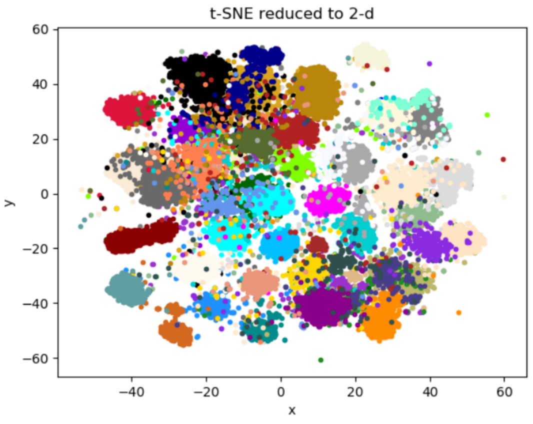

Fig. 2: Data distribution after reducing to 2-d via t-SNE

We use t-SNE to do dimensionality reduction to 2-d space and visualize the distribution of AwA2 data set. As is illustrated in Figure 2, there are a small portion of points mixed with the other cluster. These outliers can confuse the KNN algorithm, especially when K is small. Even if we would like to minimize the distances between samples of the same clusters, our loss function focuses on the whole picture and cannot totally solve these small errors. In fact, yet no one can tell what really happens for those outliers, since their initial position are too far away from their fellows. Thus, the metric learning methods still fail to solve the problem of outliers, but they can solve the cases when we need to resize or rotate the co-ordinate to split different clusters well.

3.12 Sparse Determinant Metric Learning (SDML)

SDML aims to make a trade-off between the following four tasks:
1) maximize the Mahalanobis distances between different clusters;
2) minimize the Mahalanobis distances inside each cluster;
3) have a sparser Mahalanobis matrix M (except diagonal elements);
4) M becomes closer to a prior knowledge M_0.

In our experiments, we do not have any prior knowledge, so we use the default value $M_0 = I$. Thus, we can regard it as MMC plus the preference for sparsity. We also put the reduced features (50-d, same as previous sections) into SDML function. Results are in Table 11.

TABLE 11: Test Accuracy of SDML

K	Baseline	SDML
2	85.51%	77.42%
5	89.02%	84.33%
10	89.31%	81.85%
15	89.23%	82.30%
20	88.94%	82.02%
30	88.49%	81.47%

SDML also fails to outperform the classical Euclidean metric on AwA2 data set, as expected. Since part of its idea is similar to MMC, it shares the problem of MMC. The influence of outliers cannot be eliminated. And the projection can make it even worse. For further studies, we should try to smooth the data first and then do metric learning. After de-noising, the projection could perform better.

3.13 Supervised Relative Components Analysis (RCA)

Supervised RCA maximizes the mutual information $I(X, Y)$ between original samples X and projected ones Y. It uses the label information to divide clusters. In this experiment, we just put the PCA features (50-d) into RCA_Supervised function.

TABLE 12: Test Accuracy of Supervised RCA

K	Baseline	Supervised RCA
2	85.51%	83.65%
5	89.02%	86.52%
10	89.31%	87.80%
15	89.23%	88.16%
20	88.94%	87.66%
30	88.49%	87.41%

As is shown in Table 12, Supervised RCA does not perform better than the classic Euclidean metric. But its performance is better than previously mentioned supervised methods (LSML, MMC, SDML). Thus, we again verify our statement that metric learning method does not perform well in AwA2 data set. We regard the noise in the deep learning features as the crux of this phenomenon. For future studies, one can try de-noising and then test the processed features on these distance metrics. If provided with more computational resources, we would like to try the de-noising experiments and see if our analysis is reasonable or not.

4 Conclusion

In this paper, we implemented various distance metrics for vectors and then used KNN algorithm to test their performance on Animals with Attributes (AwA2) dataset. We evaluated a total of 14 distance metrics, including 4 simple metrics and 10 metric learning algorithms. We also visualized the data distribution to help us analyze.

From all the results above, we can conclude that (1) Euclidean distance is the best choice considering both performance and complexity; (2) most metric learning algorithms have worse performance than Euclidean distance; (3) only LMNN outperforms Euclidean distance by less than 1%, but consider the computational complexity, we do not recommend it; (4) metric learning algorithms may fail to perform well due to the noise in the dataset.

For further studies, provided with more computational resources, we would like to know how noise affects the performance of metric learning algorithms. One can try denoising algorithms before metric learning algorithms, and see how much their performances can enhance.

References

[1] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[2] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis," *Journal of machine learning research*, vol. 8, no. May, pp. 1027–1061, 2007.

[3] K. Q. Weinberger and G. Tesauro, "Metric learning for kernel regression," in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, M. Meila and X. Shen, Eds., vol. 2. San Juan, Puerto Rico: PMLR, 21–24 Mar 2007, pp. 612–619. [Online]. Available: http://proceedings.mlr.press/v2/weinberger07a.html

[4] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. MIT Press, 2005, pp. 513–520. [Online]. Available: http://papers.nips.cc/paper/2566-neighbourhood-components-analysis.pdf

[5] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning distance functions using equivalence relations," in *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, T. Fawcett and N. Mishra, Eds. AAAI Press, 2003, pp. 11–18. [Online]. Available: http://www.aaai.org/Library/ICML/2003/icml03-005.php

[6] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 209–216. [Online]. Available: https://doi.org/10.1145/1273496.1273523

[7] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell, "Distance metric learning with application to clustering with side-information." in *International Conference on Neural Information Processing Systems*, 2002.

[8] G.-J. Qi, J. Tang, Z.-J. Zha, and T.-S. Chua, "An efficient sparse metric learning in high-dimensional space via l 1-penalized log-determinant regularization," 01 2009.

[9] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Advances in neural information processing systems*, 2005, pp. 1601–1608.