Project2 Report: Distance Metric

Hou Shengyuan 518021910604
Yan Binghao 518021910753
Wu Shiqu 518021910665

Abstract. Distance metrics are a key part of some machine learning algorithms, such as K-Nearest Neighbors (**KNN**) algorithm [1]. Moreover, an effective distance metric can improve the performance of machine learning models, whether that's for classification tasks or clustering. In this project, we conducted experiments using the deep learning features of the (AwA2) dataset. First, we use the KNN algorithm combined with 4 simple metrics (Manhattan distance, Euclidean distance, Chebyshev distance and cosine distance) to conduct experiments and evaluate their performance. We also use preprocessing to improve efficiency(Use LDA to reduce dimension). Secondly, we tried 7 metric learning algorithms (4 supervised metric learning methods : **LMNN** [2], **NCA** [3],**LFDA** [4], **MLKR** [5]; 3 weakly supervised metric learning methods : **ITML** [7], **SDML** [8], **MMC** [6]) to see different method's effect on KNN. Finally, we coordinated and analyzed all the experimental results, and found that.

Keywords: simple metric, metric learning, evaluation

1 Introduction

Among the machine learning algorithms, a typical algorithm is to extract features from the distance between samples, and the K-nearest neighbor (KNN) algorithm is one of the most famous ones. As a traditional learning method, KNN is performed without changing the spatial distribution of samples, so how to measure the distance between samples or distributions reasonably is very important [1].

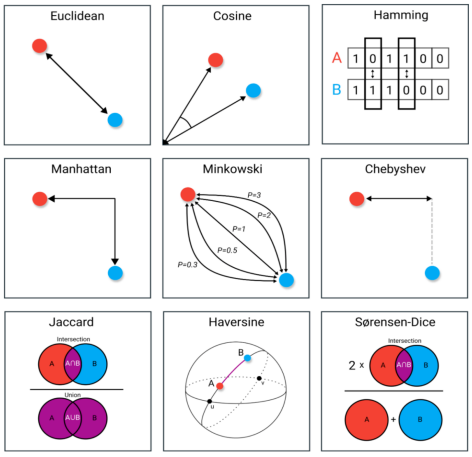

Fig. 1: 9 common standard distance metrics

Traditionally, practitioners will choose standard distance metrics (Chebyshev distance, Euclidean distance, Manhattan distance, and cosine distance, etc. as shown in Fig.1) and use prior knowledge of the domain to achieve their goals. These methods are simple and easy to implement, the geometric explanation is clear, and they all satisfy the three distance rules: non-negative, symmetric and triangular inequalities. Broadly speaking, it is meaningful to explore how to choose the best standard distance metric in different experimental configurations. But because we use a given data set, in the project we mainly strive to obtain the performance of different simple distance metrics through experiments, and try our best to give explanations for the different performance of various simple distance metrics.

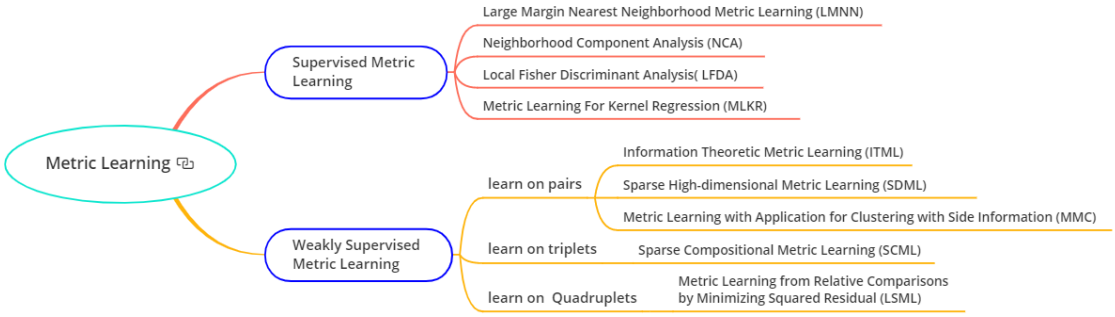

Fig. 2: Outline of Metric Learning Methods

However, although the standard distance metric is widely used, it also has its limitations-it is often difficult to design a metric suitable for specific data and tasks of interest. This is why scientists have introduced metric learning, which aims to automatically construct task-specific distance metrics from (weakly) supervised data in a machine learning manner as shown in Fig. 2. Then, the learned distance metric can be used to perform various tasks (eg, k-NN classification, clustering, information retrieval...). Therefore, we also tried some metric learning methods (**LMNN** [2], **NCA** [3], **LFDA** [4], **MLKR** [5], **ITML** [7], **SDML** [8], **MMC** [6]) to see their effect.

2 Method

2.1 K-nearest Neighbors (KNN)

KNN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. In simple terms, in the training phase, training examples are vectors in a multi-dimensional feature space, and each vector has a category label. The training phase of the algorithm only includes storing the feature vectors and category labels of the training samples. In the classification phase, k is a user-defined constant, and the unassigned vector (query or test point) is classified by assigning the most frequent label among the k training samples closest to the query point, as shown in Fig. 3.

It is not difficult to find that there are two main factors affecting the performance of the KNN classifier: 1) the distance used to determine the "nearest neighbor"; 2) the parameter k. In addition, in the experiment, we found that the computational complexity of KNN is very high. In order to reduce the time complexity, we first use LDA (LDA can reduce the dimensionality of high-dimensional data while retaining most of the information used for classification) to reduce the

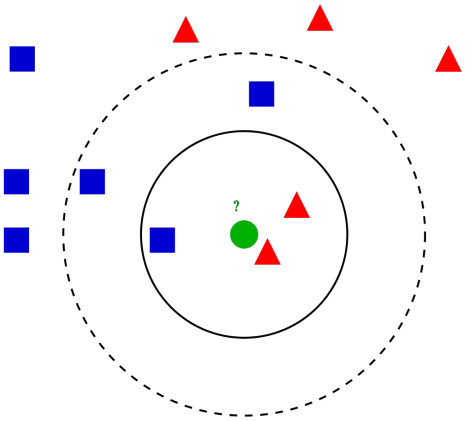

Fig. 3: KNN algorithm example

dimensionality of the data to 49, and then use KNN for classification. The reason for setting the dimension to 50 is that it can better balance training speed and accuracy.

2.2 Simple Distance Metrics

Simple Distance denotes data-independent distance functions, e.g. Minkowski Distance and Cosine Distance.

Minkowski Distance The Minkowski distance of order p (where p is an integer) between two points $X = \{x_1, x_2, \cdots, x_n\}$ and $Y = \{y_1, y_2, \cdots, y_n\}$ is defined as:

$$D_p(X,Y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$$

. **Manhattan distance:**

$$D_{Manhattan} = D_1(X,Y) = (\sum_{i=1}^{n} |x_i - y_i|)$$

. **Euclidean distance:**

$$D_{Euclidean} = D_2(X,Y) = \sqrt{(\sum_{i=1}^{n} |x_i - y_i|^2)}$$

. **Chebyshev distance:**

$$D_{Chebyshev} = \lim_{p \to \infty} D_p(X,Y) = \max_i |x_i - y_i|$$

Cosine Distance We can use cosine formula to calculate the similarity of two vectors:

$$cos(X,Y) = \frac{X^T Y}{||X|||Y||}$$

$cos(X,Y)$ ranges between $[-1,1]$. The more similar X and Y are, the bigger $cos(X,Y)$ is. To scale distance to $[0,1]$ and transform similarity to distance, cosine distance is defined as:

$$D_{cos(X,Y)} = \frac{1}{2}(1 - cos(X,Y))$$

2.3 Supervised Metric Leaning

The goal in Supervised Learning is to learn a distance metric that puts points with the same label close together while pushing away points with different labels. That is to say, transform points in a new space, in which the distance between two points from the same class will be small, and the distance between two points from different classes will be large. The data we needed to prepare for supervised learning algorithm are a set of data points, each of them belonging to a class (label). As we can see that the original data form satisfies the requirement, we just use the KNN training data as the input of supervised learning algorithms.

Large Margin Nearest Neighbor Metric Learning (LMNN)

LMNN learns a **Mahalanobis distance metric** in the KNN classification setting. The learned metric attempts to keep close k-nearest neighbors from the same class, while keeping examples from different classes separated by a large margin. This algorithm makes no assumptions about the distribution of the data.

Let the training data consist of a data set $D = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \cdots, (\boldsymbol{x_n}, y_n)\} \subset R^d \times C$, where the set of possible class categories is $C = \{1, 2, \cdots, c\}$.And the Mahalanobis distance metric is:

$$D(\boldsymbol{x_i}, \boldsymbol{x_j}) = \sqrt{(\boldsymbol{x_i} - \boldsymbol{x_j})^\top M (\boldsymbol{x_i} - \boldsymbol{x_j})}$$

What's more, for $D(\cdot,\cdot)$ to be well defined,the matric M needs to be positive semi-definite.The Euclidean metric is a special case, where M is the identity matrix. Fig. 4 illustrates the effect of the metric under varying M. The two circles show the set of points with equal distance to the center $\boldsymbol{x_i}$. In the Euclidean case this set is a circle, whereas under the modified (Mahalanobis) metric it becomes an ellipsoid.

The distance is learned by solving the following optimization problem:

$$\min_{\mathbf{L}} \sum_{i,j} \eta_{ij} D(\boldsymbol{x_i}, \boldsymbol{x_j}) + c \sum_{i,j,l} \eta_{ij}(1 - y_{il})[1 + D(\boldsymbol{x_i}, \boldsymbol{x_j}) - D(\boldsymbol{x_i}, \boldsymbol{x_l})]_+)$$

where x_i is a data point, x_j is one of its k-nearest neighbors sharing the same label, and x_l are all the other instances within that region with different labels, η_{ij},$y_{il} \in \{0,1\}$ are both the indicators, η_{ij} represents x_j is the k-nearest neighbors (with same labels) of x_i, $y_{il} = 0$ indicates x_i,x_l belong to different classes, $[\cdot]+ = max(0,\cdot)$ is the Hinge loss.

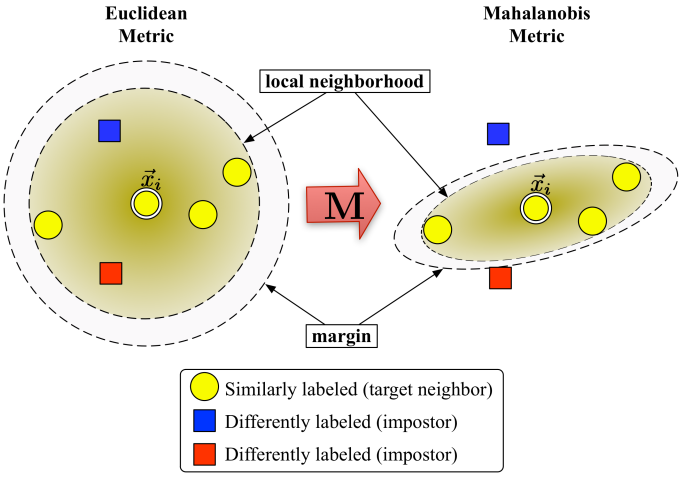

Fig. 4: Schematic illustration of LMNN

Neighborhood Components Analysis (NCA)

Neighbourhood components analysis is a supervised learning method for classifying multivariate data into distinct classes according to a given distance metric over the data.The key points can be summarized as follows: the task is KNN Classification, the sample similarity calculation method is based on **Mahalanobis Distance**, and the parameter selection method is **Leave One Out**. The optimization problem is to find matrix $L(M = L^T L)$ that maximizes the sum of probability of being correctly classified:

$$\mathbf{L} = \text{argmax} \sum_i p_i$$

where p_i represents probability that x_i will be correctly classified by the stochastic nearest neighbors rule. The p_i is calculated by:

$$p_i = \sum_{j:j \neq i, y_j = y_i} p_{ij}$$

p_{ij} represents the probability that x_i is the neighbor of x_j,it's defined as:

$$p_{ij} = \begin{cases} \dfrac{\exp(-||\mathbf{L}x_i - \mathbf{L}x_j||_2^2)}{\sum_{l \neq i} \exp(-||\mathbf{L}x_i - \mathbf{L}x_l||_2^2)}, & i \neq j \\ 0, & i = j \end{cases}$$

Local Fisher Discriminant Analysis (LFDA)

LFDA is a linear supervised dimensionality reduction method which effectively combines the ideas of Linear Discriminant Analysis and Locality-Preserving Projection. Its main idea is to find the transformation matrix L so that adjacent data pairs in the same class are close to each other, while data pairs in different classes are separated from each other.

$$\mathbf{L}_{LFDA} = \arg\max_{\mathbf{L}} [\text{tr}((\mathbf{L}^T \mathbf{S}^{(w)} \mathbf{L})^{-1} \mathbf{L}^T \mathbf{S}^{(b)} \mathbf{L})]$$

Solved as a generalized eigenvalue problem.

The algorithm define the Fisher local within-/between-class scatter matrix $S^{(w)}/S^{(b)}$ in a pairwise fashion:

. **Fisher local within class scatter matrix:**

$$\mathbf{S}^{(w)} = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}^{(w)}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T$$

. **Fisher local between class scatter matrix:**

$$\mathbf{S}^{(b)} = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}^{(b)}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T$$

where:

$$W_{ij}^{(w)} = \begin{cases} 0 & y_i \neq y_j \\ \mathbf{A}_{i,j}/n_l & y_i = y_j \end{cases} \qquad W_{ij}^{(b)} = \begin{cases} 1/n & y_i \neq y_j \\ \mathbf{A}_{i,j}(1/n - 1/n_l) & y_i = y_j \end{cases}$$

here $A_{i,j}$ is the $(i,j)-th$ entry of the affinity matrix A, which can be calculated with local scaling methods, $n,\ n_l$ are the total number of points and the number of points per cluster l respectively.

Metric Learning for Kernel Regression (MLKR)

MLKR is an algorithm for supervised metric learning, which learns a distance function by directly minimizing the leave-one-out regression error. This algorithm can also be viewed as a supervised variation of PCA and can be used for dimensionality reduction and high dimensional data visualization.

Theoretically, MLKR can be applied with many types of kernel functions and distance metrics, we hereafter focus the exposition on a particular instance of the Gaussian kernel and Mahalanobis metric, as these are used in our empirical development. The Gaussian kernel is denoted as:

$$k_{ij} = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)}{\sigma^2}),$$

where the $d(\cdot,\cdot)$ is Mahalanobis, $d(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j)||$, the transition matrix L is derived from the decomposition of Mahalanobis matrix $M = L^T L$.

Then we use the cumulative leave-one-out quadratic regression error of the training samples as the loss function:

$$\mathcal{L} = \sum_i (y_i - \hat{y}_i)^2, \qquad \hat{y}_i = \frac{\sum_{j\neq i} y_j k_{ij}}{\sum_{j\neq i} k_{ij}}$$

2.4 Weakly Supervised Metric Leaning

The difference between weakly supervised learning and supervised learning is that weakly supervised learning algorithm has access to a set of data points with supervision only at the tuple level (typically pairs, triplets, or quadruplets of data points). The tuples of points can also be called

"constraints". A classic example of weaker supervision is a set of positive and negative pairs. In this case, the goal is to learn a distance metric that puts positive pairs close together and negative pairs far away. The weakly supervised learning algorithms can be divided into the following three types:

- **Learning on Pairs:** In this case, one should provide the algorithm with n pairs of points, with a corresponding target containing n samples values being either +1 or -1. These values indicate whether the given pairs are similar points or dissimilar points. MMC, ITML, SDML, RCA are typical pair metric learning algorithms. And in this project, we only try the first three algorithms.
- **Learning on Triplets:** In this case, one should provide the algorithm with n triplets of points. The semantic of each triplet is that the first point should be closer to the second point than to the third one. SCML is one of the most famous metric learning algorithms for this type.
- **Learning on Quadruplets:** In this case, one should provide the algorithm with n quadruplets of points. The semantic of each quadruplet is that the first two points should be closer together than the last two points. LSML is a powerful quadruplet metric learning algorithm and it minimizes a convex objective function corresponding to the sum of squared residuals of constraints. More details about LSML will be displayed later.

In order to provide the specific data form (pairs, triplets, quadruplets) for weakly supervised learning algorithms, we design a pre-processor to get the target input representations (pair,triplet or quadruplets) from the original input representation(array-like). Due to the fact that the implementation details of this pre-processor are not important for this project, we just skip this part and directly analyse the output results.

Mahalanobbis Metric for Clustering(MMC)

MMC exploits the idea similar to LDA, where data points in different clusters should be separated far away and points within the same class should be considered closer together. Taking this into consideration, MMC aims to minimize the sum of Mahalanobbis distances among all data pairs within the same class while to exert a lower bound constraint on data pairs between different classes.

Define S as the set of all data point pairs within the same class, and D as the set of all point pairs between different classes.Then the objective function is as follows.

$$\begin{aligned} \min \quad & \sum_{(i,j)\in S} ||\boldsymbol{x_i} - \boldsymbol{x_j}||_A^2 \\ s.t. \quad & \sum_{(i,j)\in D} ||\boldsymbol{x_i} - \boldsymbol{x_j}||_A^2 \geq 1 \\ & \boldsymbol{A} \succeq 0 \end{aligned}$$

This is a convex and thus local-minimum free optimization problem which can be solved efficiently. However, it has an important assumption that all data points within one class should be cluster-shaped rather than any other non-regular shapes since it;s originally used for center-based clustering. Class distribution should satisfy unimodal distribution.

Information Theoretic Metric Learning (ITML)

ITML aims to learn a class of Mahalanobbis distance functions. It incorporates information theoretic setting by transforming the object of learning best distance metrics to learn the best Gaussian distribution w.r.t KL entropy objective function.

ITML bears a close resemblance to MMC in the idea of partitioning data pairs as similar and dissimilar one. When problem setting is semi-supervised, we could manually set a threshold to identify similar and dissimilar point pair, which means (1)if $d_{\boldsymbol{A}}(\boldsymbol{x_i}, \boldsymbol{x_j}) \leq u$, they are considered as similar. (2)if $d_{\boldsymbol{A}}(\boldsymbol{x_i}, \boldsymbol{x_j}) \geq l$, they are considered as dissimilar. When all data points are labeled, pair within the same class would be regarded as in similar set and otherwise in dissimilar set.

Given a Mahalanobbis distance parameterized by $\boldsymbol{A}$, we express its corresponding multivariate Gaussian distribution as $p(\boldsymbol{x}, \boldsymbol{A}) = \frac{1}{Z}\exp(-\frac{1}{2}d_{\boldsymbol{A}}(\boldsymbol{x}, \boldsymbol{u}))$(Z is normalizing constant). Then we could measure the distance between two distance metric functions defined by calculating KL divergence of multivariate Gaussian distribution.

$$\boldsymbol{KL}(p(\boldsymbol{x}; \boldsymbol{A_0})||q(\boldsymbol{x}; \boldsymbol{A})) = \int p(\boldsymbol{x}; \boldsymbol{A_0}) \log \frac{p(\boldsymbol{x}; \boldsymbol{A_0})}{q(\boldsymbol{x}; \boldsymbol{A})} d\boldsymbol{x}$$

Then given similar pair set S and dissimilar pair set D, the objective is as follows.

$$\begin{aligned} \min_{A} \ & KL(p(\boldsymbol{x}; \boldsymbol{A_0})||p(\boldsymbol{x}; \boldsymbol{A})) \\ s.t. \ & d_{\boldsymbol{A}}(\boldsymbol{x_i}, \boldsymbol{x_j}) \leq u \quad (i,j) \in S \\ & d_{\boldsymbol{A}}(\boldsymbol{x_i}, \boldsymbol{x_j}) \geq l \quad (i,j) \in D \end{aligned}$$

This objective function is equivalent to LogDet Bregman Divergence, and thus could be solved by adapting Bregman's method. LogDet Divergence is generated by convex function $\phi(\boldsymbol{X}) = -\log\det \boldsymbol{X}$, and it has been proved that KL divergence w.r.t multivariate Gaussian distribution could be expressed as LogDet Divergence.

$$\begin{aligned} D_{ld}(\boldsymbol{A}, \boldsymbol{A_0}) &= tr(\boldsymbol{A}\boldsymbol{A_0}^{-1}) - \log\det(\boldsymbol{A}\boldsymbol{A_0}^{-1}) - n \\ KL(p(\boldsymbol{x}; \boldsymbol{A_0})||p(\boldsymbol{x}; \boldsymbol{A})) &= \frac{1}{2} D_{ld}(\boldsymbol{A_0}^{-1}, \boldsymbol{A}^{-1}) \\ &= \frac{1}{2} D_{ld}(\boldsymbol{A}, \boldsymbol{A_0}) \end{aligned}$$

Therefore the original objective function could be written as follows

$$\begin{aligned} \min_{A \succeq 0} \ & D_{ld}(\boldsymbol{A}, \boldsymbol{A_0}) \\ s.t. \ & Tr(A(\boldsymbol{x_i} - \boldsymbol{x_j})(\boldsymbol{x_i} - \boldsymbol{x_j})^T) \leq u \quad (i,j) \in S \\ & Tr(A(\boldsymbol{x_i} - \boldsymbol{x_j})(\boldsymbol{x_i} - \boldsymbol{x_j})^T) \geq l \quad (i,j) \in D \end{aligned}$$

Sparse High-Dimensional Metric Learning (SDML)

SDML inherits the core idea of MMC, which aims to find Mahalanobbis distance metrics that minimize distances within the same class while maximize distances between diffeerent classes. Rather

than just assign a lower bound constraint to maximization problem in MMC, SDML tries to incorporate both two goals into one objective function. This is realized by adding a signal parameter K_{ij} to the sum of distances. S and D are data pair set already defined in MMC.

$$K_{ij} = \begin{cases} 1 & (i,j) \in S \\ -1 & (i,j) \in D \end{cases}$$

Then integrated sum of distance could be formulated as follows. $\boldsymbol{A}$ and $\boldsymbol{A_0}$ is respectively parameterized matrix for new and old distance metrics.

$$\begin{aligned} \frac{1}{2}\sum_{i,j} K_{ij}||\boldsymbol{x_i} - \boldsymbol{x_j}||^2_{\boldsymbol{A}} &= \frac{1}{2}\sum_{i,j} K_{ij}(\boldsymbol{x_i} - \boldsymbol{x_j})^T\boldsymbol{A}(\boldsymbol{x_i} - \boldsymbol{x_j}) \\ &= \sum_{i,j} K_{ij}(\boldsymbol{x_i}^T\boldsymbol{A}\boldsymbol{x_i} - \boldsymbol{x_i}^T\boldsymbol{A}\boldsymbol{x_j}) \\ &= Tr(\boldsymbol{X^T A X D}) - Tr(\boldsymbol{X^T A X K}) \\ &= Tr(\boldsymbol{X(D-K)X^T A}) \\ &= Tr(\boldsymbol{X L X^T A}) \end{aligned}$$

Additionally, SDML aims to make distance metric to be compact based on the prior knowledge that non-diagonal elements of Mahalanobbis matrix in high dimensional space tends to be sparse for the follwing three reasons:(1)due to weak relationship of different features in high dimensional space, off-diagonal elements of concentration matrix are often remarkably small which can be safely ignored.(2)sparse principle could yield the most compact model. (3)Sparse model enjoys low training cost with high efficiency. SDML attains this by adding L1 regularization on non-diagonal elements.

$$L1 = \sum_i \sum_{j \neq i} |\boldsymbol{A}_{ij}|$$

Also, similar to ITML, SDML learned metrics should not deviate too much from our prior metrics knowledge $\boldsymbol{A_0}$. Therefore, we should alse take LogDet Divergence into consideration like in ITML.

$$KL(p(\boldsymbol{x};\boldsymbol{A_0})||p(\boldsymbol{x};\boldsymbol{A})) = \frac{1}{2}D_{ld}(\boldsymbol{A},\boldsymbol{A_0}) = \frac{1}{2}tr(\boldsymbol{A}\boldsymbol{A_0}^{-1}) - \frac{1}{2}\log\det(\boldsymbol{A}) + \frac{1}{2}\log\det(\boldsymbol{A_0}) - \frac{1}{2}n$$

n and $\det \boldsymbol{A_0}$ is constant here taking Euclidean distance as prior knowledge. So in short, our objective function is comprised of three components as follows.

$$\begin{aligned} &\min_{\boldsymbol{A}} Tr(\boldsymbol{X L X^T A}) + \lambda\sum_i\sum_{j\neq i}|\boldsymbol{A}_{ij}| + \mu(Tr(\boldsymbol{A}\boldsymbol{A_0}^{-1}) - \log\det\boldsymbol{A}) \\ &s.t.\ \boldsymbol{A} \succeq 0 \end{aligned}$$

Sparse Compositional Metric Learning (SCML)

SCML learns a squared Mahalanobis distance from triplet constraints by optimizing sparse positive weights assigned to a set of K rank-one PSD bases. This can be formulated as an optimization problem with only K parameters, that can be solved with an efficient stochastic composite scheme.

The Mahalanobis matrix M is built from a basis set $B = \{b_i\}_{i=\{1,\dots,K\}}$ weighted by a KK dimensional vector $w = \{w_i\}_{i=\{1,\dots,K\}}$as:

$$M = \sum_{i=1}^{K} w_i b_i b_i^T = B \cdot diag(w) \cdot B^T \quad w_i \geq 0$$

Learning M in this form makes it PSD by design, as it is a nonnegative sum of PSD matrices. The basis set B is fixed in advance and it is possible to construct it from the data. The optimization problem over w is formulated as a classic margin-based hinge loss function involving the set C is added to yield a sparse combination. The formulation is the following:

$$\min_{w \geq 0} \sum_{(x_i, x_j, x_k) \in C} [1 + d_w(x_i, x_j) - d_w(x_i, x_k)]_+ + \beta ||w||_1$$

where $[\cdot]_+$ is the hinge loss.

Metric Learning from Relative Comparisons by Minimizing Squared Residual (LSML)

LSML proposes a simple, yet effective, algorithm that minimizes a convex objective function corresponding to the sum of squared residuals of constraints. This algorithm uses the constraints in the form of the relative distance comparisons, such method is especially useful where pairwise constraints are not natural to obtain, thus pairwise constraints based algorithms become infeasible to be deployed. Furthermore, its sparsity extension leads to more stable estimation when the dimension is high and only a small amount of constraints is given.

The loss function of each constraint $d(\mathbf{x}_i, \mathbf{x}_j) < d(\mathbf{x}_k, \mathbf{x}_l)$ is denoted as:

$$H(d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - d_{\mathbf{M}}(\mathbf{x}_k, \mathbf{x}_l))$$

where $H(\cdot)$ is the squared Hinge loss function defined as:

$$H(x) = \begin{cases} 0 & x \leq 0 \\ x^2 & x > 0 \end{cases}$$

The summed loss function $L(C)$ is the simple sum over all constraints $C = \{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l) : d(\mathbf{x}_i, \mathbf{x}_j) < d(\mathbf{x}_k, \mathbf{x}_l)\}$

The distance metric learning problem becomes minimizing the summed loss function of all constraints plus a regularization term w.r.t. the prior knowledge:

$$\min_{\mathbf{M}} (D_{ld}(\mathbf{M}, \mathbf{M_0}) + \sum_{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l) \in C} H(d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - d_{\mathbf{M}}(\mathbf{x}_k, \mathbf{x}_l))$$

where $\mathbf{M}_0$ is the prior metric matrix, set as identity by default, $D_{ld}(\cdot, \cdot)$ is the LogDet divergence:

$$D_{ld}(\mathbf{M}, \mathbf{M_0}) = \mathrm{tr}(\mathbf{M}\mathbf{M_0}) - \mathrm{logdet}(\mathbf{M})$$

3 Experiment

3.1 Experimental Setup

In the experiment, We select the **Animals with attributes 2 dataset (AwA2)**. The data is composed of 37322 images of 50 animals and $2048-dimensional$ learned representation features are pre-extracted using $Resnet101$ from every image. We split the images in each category into 60% for training and 40% for testing.

Since the dimensionality of the original feature is as high as 2048, and KNN is a lazy learning method, its computational complexity is highly related to the dimensionality, so we must reduce the dimensionality. We use LDA for dimensionality reduction. The following experiments takes the preprocessed feature as input.

In addition, we use **metric-learn** package to implement all metric learning algorithms. The basic training framework is **sklearn**.

3.2 Simple distance

The experimental results of the four standard simple distance metrics are shown in Table 1. The parameter k refers to the number of neighbors. According to the experimental results, we make the

K	Euclidean	Manhattan	Chebyshev	Cosine	Euclidean_PCA
2	0.910	0.908	0.905	0.917	0.86
5	0.928	0.926	0.924	0.929	0.894
8	0.928	0.928	0.924	0.929	0.896
10	0.928	**0.928**	0.924	0.929	0.897
15	0.929	0.928	**0.926**	0.930	**0.898**
20	0.929	0.927	0.925	0.930	0.895
25	**0.930**	0.926	0.925	**0.930**	0.895

Table 1: Accuracy of KNN with different N neighbors and distance functions.

following observations and analyses:

1. The performance of all distance metrics increases first, and decreases after reaching the optimal value of k. This is because when k is too small, there are too few neighbors to deal with the noise. However, when k is too large, too many neighbors are included in the voting process, which will make the boundary between different clusters unclear.
2. When k is small, the performance of cosine distance is the best; when k is large, Euclidean distance also performs very well, and the performance is close to cosine distance. Such results indicate that our input data may be locally distributed in a circular manner, so the cosine distance can best evaluate the data relationship.
3. Chebyshev distance metric performed the worst. This is because it only takes dimension with max value into consideration. Therefore, it cannot fully use the hidden information in other dimensions.
4. From the positive and negative Euclidean distance experiments of whether to add LDA, we found that LDA pretreatment can improve the performance by about 3%, as shown in Figure 5. This is because LDA can reduce the distance within a group and increase the distance between groups, which is consistent with the motivation of KNN. In contrast, PCA only does a simple dimensionality reduction, which only improves the speed of model training.

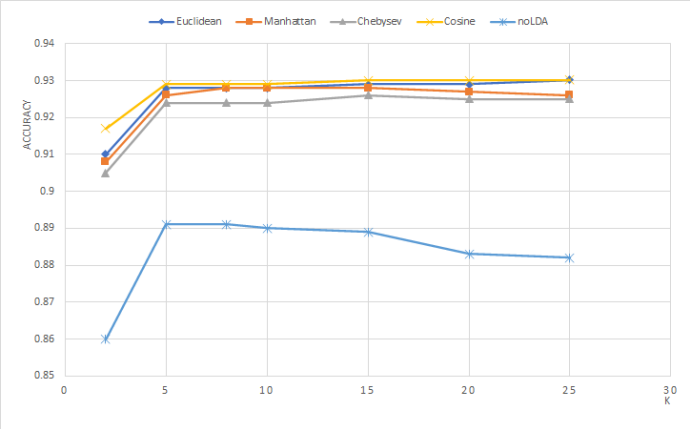

Fig. 5: Accuracy of different distance metrics

3.3 LMNN

Considering that LMNN is a special form of KNN, the number of target neighbors k is still the most important parameter. The experimental results are shown in Table 2.

K	LMNN_LDA	LMNN_PCA
2	0.910	0.864
5	0.925	0.899
8	0.927	0.900
10	0.928	**0.902**
15	0.929	0.901
20	**0.929**	0.901
25	0.928	0.900

Table 2: Accuracy of LMNN with regard to different K

From the experimental results, first comparing the results after LDA processing, we found that the performance of LMNN is not significantly improved compared to the standard distance measurement. We speculate that this may be the effect of LDA. Because the purpose of LDA is to reduce the distance within a group and increase the distance between groups, while LMNN is also to bring the target neighbors closer and push away samples of different types. This can also explain why when the preprocessing method is changed to PCA, LMNN has a significant improvement compared to the traditional distance measurement.

3.4 NCA

NCA is a supervised algorithm. Experimental result has been shown in Table 4. When training the NCA model, it will only force data points in the same category to be closer, without retaining local information. This makes it even on the data set preprocessed by LDA to maintain the accuracy increase when the value of k is large, and it still does not converge when $k = 25$, which is obviously better than other algorithms. But limited to this, on the data set preprocessed by PCA, its performance is slightly worse than that of LMNN.

K	NCA_LDA	NCA_PCA
2	0.904	0.864
5	0.920	0.895
8	0.921	0.897
10	0.922	**0.898**
15	0.922	0.898
20	0.923	0.896
25	**0.924**	0.895

Table 3: Accuracy of NCA with regard to different K

3.5 LFDA

As we analyzed in Section 2.4, LFDA is derived from LDA, and the main idea of LFDA is to introduce local property into LDA. The experimental results are shown in Table 4.

K	LFDA_LDA	LFDA_PCA
2	0.905	0.869
5	0.924	0.894
8	0.925	0.893
10	0.927	**0.894**
15	0.927	0.894
20	**0.9283**	0.892
25	0.928	0.890

Table 4: Accuracy of LFDA with regard to different K

After comparison, we found that the performance of LFDA was inferior to that of KNN after LDA pretreatment. We suspect that this may be because LFDA is designed for multi-modality tasks, where one type of sample may form several separate clusters. But in our data set, it is rare to have multiple clusters of the same kind. Although LDA is usually used as a dimensionality reduction method, it does not focus on the individual clusters in the group, but pays more attention to the compactness within the group and the separation between the groups. This can also explain the poor performance of LFDA after PCA pretreatment

3.6 MLKR

The experimental results of MLKR are shown in Table 5.

From the experimental results, we found that the overall performance of MLKR is average, but it is worth noting that the k value when MLKR achieves the best accuracy is very small, which is outstanding in all algorithms. This is due to its direct minimization of $leaveone - out$ regression error to learn the distance function. In addition, it is worth mentioning that the MLKR algorithm converges very quickly. When $k = 2$, its accuracy rate is as high as 91.04%

3.7 MMC

The experimental results are shown in Table 6. In contrast, the accuracy of MMC is significantly lower than that of other methods. By definition, the purpose of MMC is to minimize the sum of

K	MLKR_LDA	MLKR_PCA
2	0.9104	0.864
5	0.919	**0.896**
8	0.921	0.896
10	0.921	0.894
15	0.922	0.893
20	**0.923**	0.890
25	0.922	0.890

Table 5: Accuracy of MLKR with regard to different K

Mahalanobis distances between samples of the same cluster and to make the sum of Mahalanobis distances between samples of different clusters greater than a threshold. Therefore, we speculate that MMC may be affected by its assumptions about unimodal distributions of all categories. In other words, the tradeoff between the "compactness" of the cluster and the overall compactness of the MMC algorithm does not adapt to our data set, and after MMC, the cluster is more dispersed, resulting in an increase in classification error.

K	MMC_PCA
2	0.674
5	0.731
8	0.736
10	0.736
15	**0.737**
20	0.731
25	0.730

Table 6: Accuracy of MMC with regard to different K

3.8 ITML & SDML

K	ITML_PCA	SDML_PCA
2	0.839	0.846
5	0.872	0.874
8	0.875	0.875
10	0.876	0.874
15	0.875	**0.877**
20	0.873	0.873
25	0.870	0.870

Table 7: Accuracy of ITML and SDML with regard to different K

Experimental result has been shown in Table 7. As we know, both ITML and SDML are weakly supervised learning methods. Limited by this, the performance of both are slightly worse than the supervised learning method under the same data set. This is why we only select Learning on pairs for experiments in the weakly supervised learning part (qualitatively speaking, weakly supervised learning methods all sacrifice a certain accuracy rate in exchange for smaller constraints on the

data set). But at the same time, we also noticed that SDML outperforms ITML on all quantitative constraint sets, which shows that sparse constraints can indeed improve performance.

4 Conclusion

In this project report, we tried a total of 13 distance metrics, including 4 simple metrics and 9 metric learning algorithms (7 of which were tested). On the one hand, simple metrics are used to evaluate the performance of the KNN classifier, and on the other hand, the pros and cons of various metric learning algorithms are explored. Based on the above results, we can draw the conclusion:

1. Considering performance and complexity, Euclidean distance is the best choice;
2. The performance of most metric learning algorithms is similar or even inferior to Euclidean distance, but considering Computational complexity, the metric learning algorithm has been greatly improved;
3. The metric learning algorithm may not perform well due to the noise in the data set;
4. The weakly supervised metric learning algorithm requires a decrease in the data set, but the accuracy rate Poor performance.

As a prospect for future work, we think we can consider providing more computing resources to further explore how noise affects the performance of metric learning algorithms. Can you first try to remove the noise in the process with the hope that the performance of the metric learning algorithm can be improved

5 Contribution

5.1 Hou Shengyuan(518021910604)

(1) Related work part: MMC,STML,IDML.
(2) Experimental part: do experiment on all methods(Simple distance, LMNN, NCA, LFDA, MLKR, MMC, STML, IDML). Also do some visualization.
(3) Other: Typesetting of the whole report except for citation label and literature survey.

5.2 Yan Binghao(518021910753)

(1) Related work part: Simple distance, LMNN, NCA, LFDA, MLKR.
(2) Experimental part: write all methods' experimental part(Simple distance, LMNN, NCA, LFDA, MLKR, MMC, STML, IDML) report and do some visualization.
(3) Other:write introduction and abstract.

5.3 Wu Shiqu(518021910665)

(1) Related work part: Provide material for Yan Binghao in related work part.
(2) Experimental part: program code for all methods(Simple distance, LMNN, NCA, LFDA, MLKR, MMC, STML, IDML).
(3) Other: **Coordination and leadership**, completed most of the code, literature survey.

References

1. Yang, Liu, and Rong Jin. "Distance metric learning: A comprehensive survey." Michigan State Universiy 2.2 (2006): 4.
2. Weinberger, Kilian Q., and Lawrence K. Saul. "Distance metric learning for large margin nearest neighbor classification." Journal of machine learning research 10.2 (2009).
3. Goldberger, Jacob, et al. "Neighbourhood components analysis." Advances in neural information processing systems 17 (2004): 513-520.
4. Sugiyama, Masashi. "Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis." Journal of machine learning research 8.5 (2007).
5. Weinberger, Kilian Q., and Gerald Tesauro. "Metric learning for kernel regression." Artificial intelligence and statistics. PMLR, 2007.
6. Xing, Eric P., et al. "Distance metric learning with application to clustering with side-information." NIPS. Vol. 15. No. 505–512. 2002.
7. Davis, Jason V., et al. "Information-theoretic metric learning." Proceedings of the 24th international conference on Machine learning. 2007.
8. Qi, Guo-Jun, et al. "An efficient sparse metric learning in high-dimensional space via l 1-penalized log-determinant regularization." Proceedings of the 26th Annual International Conference on Machine Learning. 2009.