Project 2: KNN with Different Distance Metrics

Xu Jiayi , Gao Yifeng, Tang Qidong, Bi Wendong

I. KNN

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It is a non-parametric, lazy and supervised learning algorithm [1] . In this project, KNN serves as the classifier for image classification, in which we explore the effect of distance metrics.

Non-parametric means that it does not make any assumptions on the underlying data distribution. It is a good property because most of the data in the real world does not obey the typical theoretical assumptions (as in linear regression models, for example). A type of lazy learning algorithm means that off-line training is not needed. In test phase, the KNN classifier directly searches through all the training examples by respectively calculating their distances between the testing example in order to identify the testing example's nearest neighbors and produce the classification output.

II. Distance Metrics

In KNN, the distance between two data points is decided by a similarity measure (or distance function) where the Euclidean distance is the most widely used distance function. However, the real world dataset such as the AwA2 in this project is not always Euclidean and we believe some other distance metrics may perform better in particular circumstance.

A. Mahalanobis Distance

The Mahalanobis distance [2] is the distance between two points in multivariate space. Mahalanobis distance takes into account the correlations of the dataset, and is thus unitness and scale-invariant. The Mahalanobis distance is defined as a dissimilarity measure between two random vector $\vec{X}$ and $\vec{Y}$ of the same distribution with the covariance matrix Σ:

$$d(\vec{X},\vec{Y}) = \sqrt{(\vec{X}-\vec{Y})^T\Sigma^{-1}(\vec{X}-\vec{Y})} \tag{1}$$

In particular, as Σ is a unit matrix, Mahalanobis distance is degenerated into Euclidean distance.

The Mahalanobis distance has the following properties:

- It accounts for the fact that the variances in each direction are different.
- It accounts for the covariance between variables.

B. Minkowski Distance

The Minkowski distance [3] is a metric measured in a normalized vector space. It is a generalized metric distance defined as :

$$d(\vec{X},\vec{Y}) = \left((\vec{X}-\vec{Y})^p\right)^{1/p} = \left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{1/p}$$

For different value of p, Minkowski distance has different particular meanings [3].

1) **Manhattan Distance**
As $p=1$, Minkowski distance is specific to Manhattan distance.

$$d(\vec{X},\vec{Y}) = |(\vec{X}-\vec{Y})| = \sum_{i=1}^{n}|x_i-y_i|$$

Manhattan distance is the distance between two points measured along axes. As shown in the left half of Fig. 1, three routes from origin to destination are the same long. Manhattan distance is also known as the L^1 norm illustrated in the right half of Fig. 1.

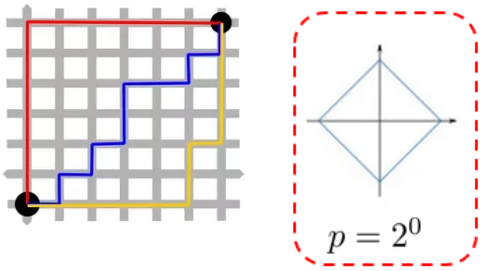

Fig. 1. Illustration of Manhattan Distance / L^1 norm

2) **Euclidean Distance**
As $p=2$, Minkowski distance is specific to Euclidean distance.

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

Euclidean distance [4] is the "ordinary" straight-line distance between two points in Euclidean space. The Euclidean distance between points X and Y is the length of the line segment connecting them, which is also known as the L^2 norm illustrated in Fig. 2.

3) **Chebyshev Distance**
As $p=\infty$, Minkowski distance is specific to Chebyshev distance.

$$d(\vec{X},\vec{Y}) = \max_i(|x_i-y_i|)$$

Chebyshev distance [5] between two vectors is the greatest of their differences along any coordinate dimension.

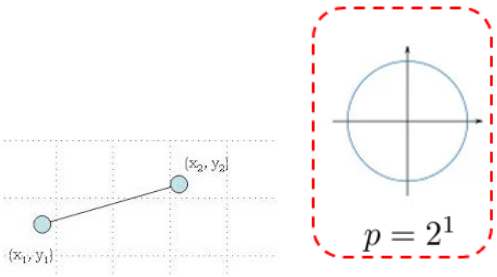

Fig. 2. Illustration of Euclidean Distance / L^2 norm

As shown in the left half of Fig. 3, all 8 adjacent cells from the given point can be reached by one unit, so Chebyshev distance is also called chessboard distance. It is also known as the L^∞ norm illustrated in Fig. 3.

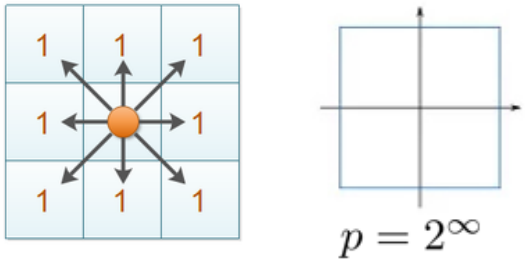

Fig. 3. Illustration of Chebyshev Distance / L^∞ norm

C. Cosine Distance

Cosine distance [6] is a measure of similarity between two non-zero vectors within an inner product space that measures the cosine of the angle between them. Given two vector $\vec{X}$ and $\vec{Y}$, Cosine distance is defined as:

$$d(\vec{X}, \vec{Y}) = \frac{\vec{X} \cdot \vec{Y}}{||\vec{X}||||\vec{Y}||} = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}}$$

Cosine similarity is always used to measure how similar the documents are irrespective of their size. It is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity. However, cosine distance is not suitable in this project which we will discuss detailly later.

III. Metric Learning

When two different samples are alike, we can use the Distance Metrics described above to distinguish them. But what if two samples are heterogeneous? Here comes Metric Learning Methods. In thoughts of Metric Learning, we first use a projection metric P to project two different samples, for example x and y, into common space:

$$dist(x, y) \rightarrow dist(Px, Py) \tag{2}$$

We then calculate the distance in such common space:

$$dist(Px, Py) = ||Px - Py|| = \sqrt{(x^T - y^T)P^T P(x - y)} \tag{3}$$

We donate the metric $P^T P$ as M, and then we use metric learning methods to learn this metric. Finally it equals to compute the Mahalanobis distance of x and y, and we need to learn its covariance metric donated by M.

$$dist_M(x, y) = dist(Px, Py) = \sqrt{(x^T - y^T)M(x - y)} \tag{4}$$

Next we need to set a goal of learning M to enhance cluster coherence and separation. Different metric learning methods have different goal and constraints. Here we will introduce 9 metric learning methods as follows.

A. Large Margin Nearest Neighbor (LMNN)

LMNN [7] learns a Mahalanobis distance metric in the KNN classification settings using semi-definite programming method. The learned metric attempts to keep k-nearest neighbors in 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.

$$\begin{aligned} \min \quad & \sum_{ij} n_{ij}(\boldsymbol{x_i} - \overrightarrow{x_j})^T M(\boldsymbol{x_i} - \boldsymbol{x_j})^T + c\sum_{ij} n_{ij}(1 - y_{ij})\xi_{ijl} \\ s.t. & (\boldsymbol{x_i} - \boldsymbol{x_l})^T M(\boldsymbol{x_i} \quad -\boldsymbol{x_l}) - (\boldsymbol{x_i} - \boldsymbol{x_j})^T M(\boldsymbol{x_i} - \boldsymbol{x_j}) \\ & \geq 1 - \xi_{ijl} \\ & \xi_{ijl} \geq 0 \\ & M \succeq 0 \end{aligned}$$

B. Neighbourhood Components Analysis (NCA)

NCA Algorithm [8] is to maximize the objective function Eq. (5) using a gradient based optimizer such as deltabar-delta or conjugate gradients.

$$q_{ij} = \frac{exp(-||Px_i - Px_j||^2)}{\sum_{k \neq i} exp(-||Px_i - Px_k||^2)}, \; p_{ii} = 0;$$

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

$$f(P) = \sum_i p_i \tag{5}$$

Of course, since the cost function is not convex, some care must be taken to avoid local maximum during training. We compute $g(P)$ in Eq. (6) instead of $f(P)$.

$$g(P) = \sum_i \log(p_i) \tag{6}$$

C. Local Fisher Discriminant Analysis (LFDA)

LFDA [9] 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 core optimization problem of LFDA is solved as a generalized eigenvalue problem.

let $S^{(w)}$ and $S^{(b)}$ be the within-class scatter matrix and the between-class scatter matrix defined by

$$S^{(w)} = \sum_{i=1}^{l} \sum_{j:y_j=i} (x_j - \mu_i)(x_j - \mu_i)^T \tag{7}$$

$$S^{(b)} = \sum_{i=1}^{l} n_i(\mu_i - \mu)(\mu_i - \mu)^T \tag{8}$$

Then the FDA transformation matrix T_{FDA} is defined as follows:

$$T_{FDA} = \underset{T \in R^{d \times m}}{\arg\max} \quad tr[(T^T S^{(w)} T)^{-1} T^T S^{(b)} T] \tag{9}$$

By this way, T is determined so that between-class scatter is maximized while within-class scatter is minimized. It's known that T_{FDA} is given by

$$T_{FDA} = (\phi_1 | \phi_2 | \cdots | \phi_m) \tag{10}$$

where $\{\phi_i\}_{i=1}^{d}$ are the generalized eigenvectors associated to the generalized eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ of the following generalized eigenvalue problem:

$$s^{(b)}\phi = \lambda S^{(w)}\phi \tag{11}$$

D. Metric Learning for Kernel Regression (MLKR)

MLKR is an algorithm of 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.

The MLKR consists of setting initial values of θ, and then adjusting the values using a gradient descent procedure

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

where ϵ is an adaptive step-size, and the loss function L is the cumulative leave-one-out quadratic regression error:

$$L = \sum_i (y_i - \hat{y_i}^2)$$

$$\hat{y_i} = \frac{\sum_{j \neq i} y_j k_{ij}}{\sum_{j \neq i} k_{ij}}$$

$$k_{ij} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{d(\overrightarrow{x_i}, \overrightarrow{x_j})}{\sigma^2}}$$

$$d(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} - \overrightarrow{x_j})^T \ M \ (\overrightarrow{x_i} - \overrightarrow{x_j}) \tag{12}$$

$$M = A^T A$$

where the i^{th} row of A is the vector $\sqrt{\lambda_i}\overrightarrow{v_i}^T$. Where $\overrightarrow{v_i}$ is the i^{th} eigenvector and λ_I is the i^{th} eigenvalue.

E. Information Theoretic Metric Learning (ITML)

ITML [10] 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:

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

In the objective function above, $\boldsymbol{M}$ represents the metric to be learned, $\boldsymbol{M_0}$ 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. $\boldsymbol{S}$ represents all pairs of similar points and $\boldsymbol{D}$ represents all pairs of dissimilar points. The objective function is to minimize the KL divergence between $\boldsymbol{M}$ and $\boldsymbol{M_0}$ to avoid overfitting, and make the learned metric $\boldsymbol{M}$ to satisfy the threshold.

Since the "closeness" between $\boldsymbol{M}$ and $\boldsymbol{M_0}$ 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.

F. Least Squares Metric Learning (LSML)

LSML [11] is a simple, yet effective algorithm. It learns a Mahalanobis metric from a given set of relative comparisons. This is done by formulating and minimizing a convex loss function that corresponds to the sum of squared hinge loss of violated constraints.

Suppose $\boldsymbol{X} = [\boldsymbol{x_1}, \cdots, \boldsymbol{x_n}]$ is the set of data points. A set of relative comparisons is given in the the form of:

$$\mathcal{C} = \{(\boldsymbol{x_a}, \boldsymbol{x_b}, \boldsymbol{x_c}, \boldsymbol{x_d}) : d(\boldsymbol{x_a}, \boldsymbol{x_b}) < d(\boldsymbol{x_c}, \boldsymbol{x_d})\} \tag{13}$$

The loss for each comparison $d(\boldsymbol{x_a}, \boldsymbol{x_b}) < d(\boldsymbol{x_c}, \boldsymbol{x_d})$ is defined as:

$$L(d(\boldsymbol{x_a}, \boldsymbol{x_b}) < d(\boldsymbol{x_c}, \boldsymbol{x_d})) = H(d_M(\boldsymbol{x_a}, \boldsymbol{x_b}) - d_M(\boldsymbol{x_c}, \boldsymbol{x_d})) \tag{14}$$

, where $H(\cdot)$ is the squared hinge function defined as:

$$H(x) = \begin{cases} 0 & if \quad x \leq 0 \\ x^2 & if \quad x > 0 \end{cases} \tag{15}$$

The loss function is easy to understand because $d_M(\boldsymbol{x_a}, \boldsymbol{x_b}) - d_M(\boldsymbol{x_c}, \boldsymbol{x_d}) \leq 0$ means that the comparison relationship between $\boldsymbol{x_a}, \boldsymbol{x_b}, \boldsymbol{x_c}, \boldsymbol{x_d}$ remains correct in the projected space, while $d_M(\boldsymbol{x_a}, \boldsymbol{x_b}) - d_M(\boldsymbol{x_c}, \boldsymbol{x_d}) > 0$ means the projection matrix $\boldsymbol{M}$ makes mistakes which should suffer a loss.

The summed loss function for all constraints (comparisons) is as:

$$L(\mathcal{C}) = \sum_{(\boldsymbol{x_a},\boldsymbol{x_b},\boldsymbol{x_c},\boldsymbol{x_d})\in\mathcal{C}} \omega_{a,b,c,d} H(d_M(\boldsymbol{x_a},\boldsymbol{x_b}) - d_M(\boldsymbol{x_c},\boldsymbol{x_d}))$$
$$= \sum_{(\boldsymbol{x_a},\boldsymbol{x_b},\boldsymbol{x_c},\boldsymbol{x_d})\in\mathcal{C}} \omega_{a,b,c,d} H(\sqrt{(\boldsymbol{x_a}-\boldsymbol{x_b})^T M(\boldsymbol{x_a}-\boldsymbol{x_b})} - \sqrt{(\boldsymbol{x_c}-\boldsymbol{x_d})^T M(\boldsymbol{x_c}-\boldsymbol{x_d})}$$

G. Sparse Determinant Metric Learning (SDML)

SDML [12] exploits the sparsity nature underlying the intrinsic high dimensional feature space to do metric learning. The sparsity prior of learning distance metric serves to regularize the complexity of the distance model, so that it only requires a smaller number of examples to learn a well posed metric from the perspective of machine learning theory

We can minimize the off-diagonal l_1-norm:

$$||M||_{1,off} = \sum_{i\neq j} |M_{i,j}| \tag{16}$$

to pursue a sparse solution.

We can minimize the log-determinant divergence function:

$$D_g(\boldsymbol{M}||\boldsymbol{M}_0) = tr(\boldsymbol{M}_0^{-1}\boldsymbol{M}) - logdet\boldsymbol{M} \tag{17}$$

where $\boldsymbol{M}_0$ is a prior. In this way, the Mahalanobis matrix $\boldsymbol{M}$ will be regularized as close as possible to the prior $\boldsymbol{M}_0$.

We can minimize the loss function defined on sets of similarity $\mathcal{S}$ and dissimilarity $\mathcal{D}$:

$$\mathcal{L}(\mathcal{S},\mathcal{D}) = \frac{1}{2}\sum_{i,j=1}^{n} ||\boldsymbol{A}^T x_i - \boldsymbol{A}^T x_j||^2 \boldsymbol{K}_{i,j}$$
$$= \sum_{i,j=1}^{n} (x_i^T \boldsymbol{A}\boldsymbol{A}^T x_i - x_i^T \boldsymbol{A}\boldsymbol{A}^T x_j)\boldsymbol{K}_{i,j}$$
$$= \sum_{i,j=1}^{n} (x_i^T \boldsymbol{M} x_i - x_i^T \boldsymbol{M} x_j)\boldsymbol{K}_{i,j}$$

$$\boldsymbol{K}_{i,j} = \begin{cases} 1, & \text{if} \quad (x_i,x_j)\in\mathcal{S} \\ -1, & \text{if} \quad (x_i,x_j)\in\mathcal{D} \end{cases} \tag{18}$$

where $\boldsymbol{K}$ is a matrix which encodes the similarity and dissimilarity information. This loss function is meant to maintain the similarity information in the projected space.

Therefore, the overall objective function of SDML is:

$$\begin{aligned} \min \quad & tr(\boldsymbol{M}_0^{-1}\boldsymbol{M}) - logdet\boldsymbol{M} + \lambda||M||_{1,off} + \eta\mathcal{L}(\mathcal{S},\mathcal{D}) \\ s.t. \quad & \boldsymbol{M} \text{ is a positive semi-definite constraint.} \end{aligned}$$

where λ is a balancing parameter trading off between sparsity and the $\boldsymbol{M}_0$ prior, η is a positive balance parameter trading off between the loss function and the regularizer (sparsity and prior).

H. Relative Component Analysis (RCA)

RCA [13] learns a full rank Mahalanobis distance metric based on a weighted sum of in-class 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.

RCA follows the idea of [14], which states that when an input $\boldsymbol{X}$ is transformed into a new representation $\boldsymbol{Y}$, we should seek to maximize the mutual information $\boldsymbol{I}(\boldsymbol{X},\boldsymbol{Y})$ between $\boldsymbol{X}$ and $\boldsymbol{Y}$ under suitable constraints.

Suppose the original data points are $\{x_{j,i}\}_{j=1,i=1}^{k\ ,\ n_j}$, and the transformed data points are $\{y_{j,i}\}_{j=1,i=1}^{k\ ,\ n_j}$. The problem can be formed as:

$$\begin{aligned} \max \quad & \boldsymbol{I}(\boldsymbol{X},\boldsymbol{Y}) \\ s.t. \quad & \frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j} ||y_{ji} - m_j^y||^2 \leq K \end{aligned}$$

where m_j^y denotes the mean of points in chunklet j after the transformation, P denotes the total number of points in chunklets, and K is a constant.

I. Mahalanobis Metric Learning for Clustering (MMC)

MMC [15] minimizes the sum of squared distances between similar examples, while enforcing the sum of distances between dissimilar examples to be greater than a certain margin. This leads to a convex and, thus, local-minima-free optimization problem that can be solved efficiently. However, the algorithm involves the computation of eigenvalues, which is the main speed-bottleneck. Since it has initially been designed for clustering applications, one of the implicit assumptions of MMC is that all classes form a compact set, i.e., follow a unimodal distribution, which restricts the possible use-cases of this method. However, it is one of the earliest and a still often cited technique.

MMC adopts a quite simple criterion for the desired metric. Suppose we have some set of points $\{x_i\}_{i=1}^m$, and then we can have 2 sets:

$$\mathcal{S}: \quad (x_i,x_j)\in\mathcal{S} \quad \text{if } x_i \text{ and } x_j \text{ are similar} \tag{19}$$

$$\mathcal{D}: \quad (x_i,x_j)\in\mathcal{S} \quad \text{if } x_i \text{ and } x_j \text{ are dissimilar} \tag{20}$$

MMC demands that pairs of points (x_i,x_j) in $\mathcal{S}$ have small squared distance between them, so that we have the objective function: $minimize_{\boldsymbol{A}} \sum_{(x_i,x_j)\in\mathcal{S}} ||x_i - x_j||_A^2$ ($\boldsymbol{A}$ is the parameter to transform points). This objective function is trivially solved with $\boldsymbol{A} =$, which is not useful, so a constraint is added: $\sum_{(x_i,x_j)\in\mathcal{D}} ||x_i - x_j||_{\boldsymbol{A}}^2 \geq 1$. Therefore, the objective function of MMC is:

$$\min \sum_{(x_i,x_j)\in\mathcal{S}} ||x_i - x_j||_A^2$$
$$s.t. \sum_{(x_i,x_j)\in\mathcal{D}} ||x_i - x_j||_{\boldsymbol{A}}^2 \geq 1$$
$$\boldsymbol{A} \text{ is positive semi-definite}$$

IV. Experiments

In this section, we will introduce the details about the experiment procedure, and the results of different methods will be displayed and compared.

A. About the Dataset

The dataset we use is Animals with Attributes (AwA2) dataset, which consists of 37322 images of 50 animals classes. The features we use for training are pre-extracted deep learning features.

B. Select the k Values of the Classifier

As a hyper parameter, the number of neighbors k significantly affect the performance of the KNN classifier on different feature space.

To decide the proper magnitude k, we use the features after PCA dimension reduction (to reduce the time complexity, especially when k is relatively large) and try different value of k. The further experiment also shows that PCA does not do much harm to the performance of the KNN classifier.

TABLE I
ACCURACY ON DIFFERENT k VALUES (PCA DIMENSION: 50)

Metric \ k	2	4	8	16	32	64
Euclidean	0.855	0.882	0.891	0.891	0.883	0.869
Manhattan	0.850	0.879	0.889	0.887	0.878	0.865
Chebyshev	0.822	0.856	0.864	0.866	0.858	0.843

The result shows that a relatively small value of k leads to a better performance.

An intuitive explanation to the result is that the KNN classifier need the nearest neighbor information to distinguish cluster properly. If k is too large, a new datapoint will easiler face the situation that its k nearest neighbors are from mixed classes, make the classification unclear.

Hence in the following experiments, we select the values of k in a smaller interval.

C. PCA's Effect on the Performance

Unless the dimension after PCA is too small, most of the information in the feature vector remains, especially partial order relation between distances among pairs of feature points. Hence KNN classifier's performance should not be affected too much if reasonable PCA dimension reduction is added.

The experiment also gives the same conclusion. compare to the original dimension (2048), 50 dimensions and 300 dimensions after PCA have similar performances, or even better than the original feature space.

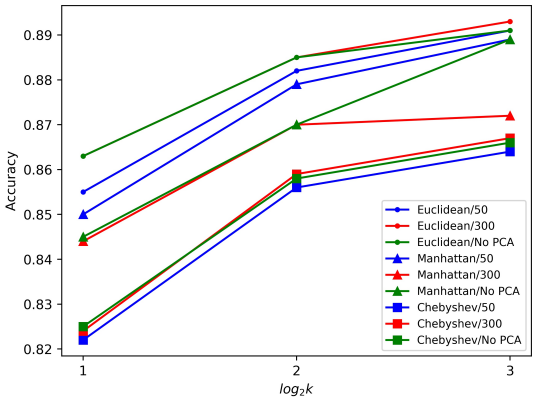

Fig. 4. Accuracy on different dimensions

TABLE II
ACCURACY ON DIFFERENT PCA DIMENSIONS

Metric/Dim \ k	2	4	8
Euclidean/50	0.855	0.9882	0.891
Euclidean/300	0.863	0.885	0.893
Euclidean/No PCA	0.863	0.0.885	0.891
Manhattan/50	0.850	0.879	0.889
Manhattan/300	0.844	0.870	0.872
Manhattan/No PCA	0.845	0.870	0.889
Chebyshev/50	0.822	0.856	0.864
Chebyshev/300	0.824	0.859	0.867
Chebyshev/No PCA	0.825	0.858	0.866

D. Distance Metrics

A few common distance metrics are tested in the KNN experiment, including Euclidean distance, Manhattan distance and so on.

1) Minkowski Distance Metrics: Euclidean and Manhattan distance have a slightly better performance over Chebyshev distance.

A possible reason is that Chebyshev distance only keep the distance which is largest among all the dimensions, hence losing much information, while Euclidean and Manhattan distance reserve distance information in all dimensions.

TABLE III
ACCURACY ON MINKOWSKI DISTANCE METRICS (NO PCA)

Metric \ k	2	3	4	5	6	7	8
Euclidean	0.863	0.888	0.885	0.890	0.891	0.893	0.891
Manhattan	0.845	0.869	0.870	0.874	0.873	0.873	0.889
Chebyshev	0.825	0.855	0.858	0.864	0.865	0.867	0.866

2) Other Distance Metrics: In fact, the cosine similarity, as a distance metric, is also tested in the KNN experiment, but

with a extremely poor performance (with an accuracy lower than 1%).

An explanation is that cosine similarity regard the features space as a high-dimensional sphere, and all data points are projected to the surface of the sphere. The projection loses the information about the distance between the Origin point and the data point, only the direction remains.

The aftermath is, if multiple classes of data is in the similar directions in the high dimensional feature space but have different distances to the origin point, the cosine similarity will finds them overlapped and unable to give a clear classification.

E. Metric Learning Methods

Some metric learning methods are supervised that need labels to learn a projection matrix from the original feature space to the new feature space while some are unsupervised methods (e.g. covariance matrix) that do not need additional labels or information.

However, some weakly-supervised metric learning methods requires additional information about the dataset. For example, weakly-supervised SDML demands a connectivity graph between pairs of feature points, which is not available in our dataset.

Hence we experiment on the methods whose requirements do not exceed the labels of the dataset.

1) Covariance Matrix: Often regarded as a baseline of metric learning, this method does not "learn" anything, rather it calculates the covariance matrix of the input data [2].

After the projection the covariance in each dimension is scale-invariant, and correlations between dimensions are removed.

However, the scale-invariant property can introduce a problem: some the information in some dimensions is actually not as important as others, but the projection use covariance matrix ignore this possibility and exaggerate their importance.

The results shows that covariance matrix method have a worse performance than just using euclidean distance metric.

TABLE IV
ACCURACY ON COVARIANCE MATRIX METRIC LEARNING

k	2	3	4	5	6	7	8
CM	0.748	0.763	0.761	0.758	0.753	0.748	0.878
Euclidean	0.863	0.888	0.885	0.890	0.891	0.893	0.891

2) LMNN: LMNN uses labels information to learn a Mahalanobis distance metric that keep samples from different classes separated by a large margin [7].

In our dataset, LMNN performs slightly better than barely using Euclidean distance.

TABLE V
ACCURACY ON LMNN METRIC LEARNING

k	2	3	4	5	6	7	8
LMNN	0.864	0.891	0.894	0.899	0.899	0.903	0.886
Euclidean	0.863	0.888	0.885	0.890	0.891	0.893	0.891

3) LFDA: LFDA's main advantages is to deal with the situation that classes may consist of more separated classes [9].

The results shows that the performance with LFDA and without LFDA (Euclidean distance metric) is similar, which indicates than mutli-clusters classes may be rare in our dataset.

TABLE VI
ACCURACY ON LFDA METRIC LEARNING

k	2	3	4	5	6	7	8
LFDA	0.874	0.894	0.893	0.897	0.897	0.899	0.891
Euclidean	0.863	0.888	0.885	0.890	0.891	0.893	0.891

4) LSML: LSML try to keep the partial order relationship in the original features space when learning the new feature representation [11].

However, in our experiment, the new representation has a lower performance on the problem.

Maybe that's because the distance between $\boldsymbol{x_a}$ and $\boldsymbol{x_b}$ measured by $d(\boldsymbol{x_a}, \boldsymbol{x_b})$ cannot represent the actual relationship between $\boldsymbol{x_a}$ and $\boldsymbol{x_b}$. For example, we adopt euclidean distance to calculate the distance between 2 data points, but the dataset appears to have a "Swiss roll" pattern, thus the relative comparison got from euclidean distance is wrong. Then, maintaining the incorrect distance information in the new distance space will have bad effect on the classification accuracy. In addition, we are supposed to use metric learning to discover the hidden pattern of the dataset, rather than follow the relationship from the original distance space. LSML just changes a distance representation, and cannot guarantee high classification accuracy.

TABLE VII
ACCURACY ON LSML METRIC LEARNING

k	2	3	4	5	6	7	8
LSML	0.754	0.771	0.769	0.768	0.763	0.760	0.868
Euclidean	0.863	0.888	0.885	0.890	0.891	0.893	0.891

5) SDML Supervised: SDML use sets of similarity $\mathcal{S}$ and dissimilarity $\mathcal{D}$ as weak-supervised information to increase the distance between dissimilar samples and decrease the similar samples' distance [12].

To use the approach as a supervised algorithm, we just use the labels as a standard to differentiate: different classes means dissimilarity while sample class indicates similarity.

The result shows that the performance is worse than euclidean distance metric in our dataset.

From our point of view, there exist two reasons why SDML performs worse than Euclidean distance. First, SDML requires a prior Mahalanobis matrix $\boldsymbol{M_0}$. The choice of M_0 will influence the final classification accuracy. Second, SDML pursues a sparse solution, so that it may perform better than other distance metrics when using a smaller number of examples. However, when we have relatively sufficient examples, SDML cannot guarantee performing better.

TABLE VIII
ACCURACY ON SDML SUPERVISED METRIC LEARNING

k	2	3	4	5	6	7	8
SDML	0.770	0.817	0.811	0.840	0.834	0.830	0.802
Euclidean	0.863	0.888	0.885	0.890	0.891	0.893	0.891

6) RCA Supervised: RCA maximize the mutual information $\boldsymbol{I}(\boldsymbol{X}, \boldsymbol{Y})$ between the original representation $\boldsymbol{X}$ and new representation $\boldsymbol{Y}$ [13].

Weakly-supervised RCA algorithm need give the information that groups several points into chunklets. For we have the labels, points in the same classes can be grouped into the same chunklets.

TABLE IX
ACCURACY ON RCA SUPERVISED METRIC LEARNING

k	2	3	4	5	6	7	8
RCA	0.837	0.860	0.860	0.866	0.866	0.873	0.869
Euclidean	0.863	0.888	0.885	0.890	0.891	0.893	0.891

F. t-SNE visualization

A Mahalanobis distance metric in KNN classifier can be regard as a projection matrix from the original feature space to new feature space plus a Euclidean distance metric.

To give a clear visual comparison difference feature space after the projection with some metric learning methods. We implement t-SNE dimension reduction to show the data point in two dimension figures.

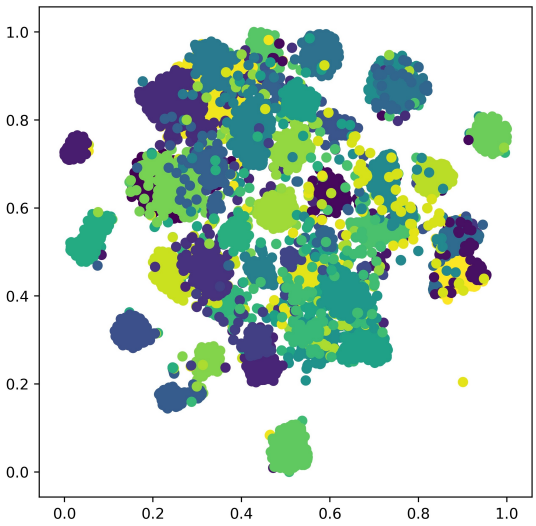

Fig. 5. t-SNE on the original feature space

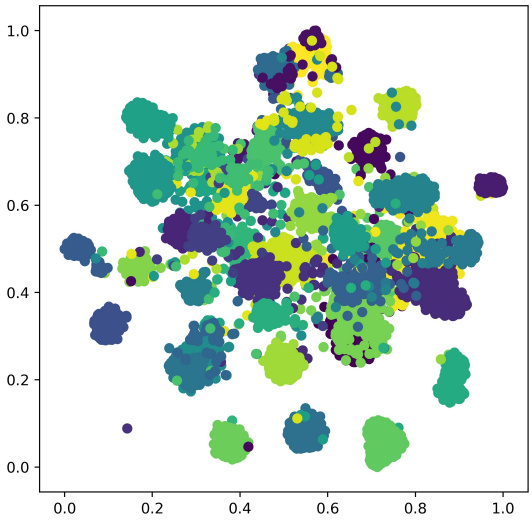

Fig. 6. t-SNE on the LMNN projected feature space

LMNN splits the clusters a bit more separately than the original feature space, which may benefit the performance of the KNN classifier.

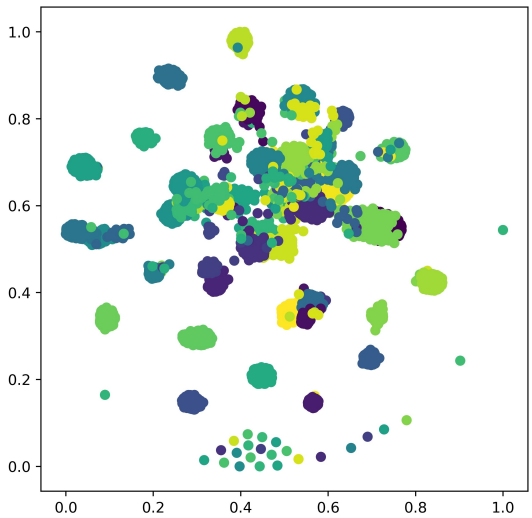

Fig. 7. t-SNE on the LFDA projected feature space

Fig. 7 shows that different clusters are far from each other and the each cluster shrinks down to a smaller size, indicating that LFDA method indeed maximizes the between-class scatter and minimize the within-class scatter.

G. Overview

After testing some metric learning, we find that some methods work well on our dataset, and give a better performance over the common distance metrics.

However, some distance methods perform worse than the common distance metrics like Euclidean distance metric. Instead of regard them as helpless, we think that those methods are not so suitable for the dataset, because they are often

designed to solve specific problems that common distance metric can not deal with well, for instance, if samples in the same classes are in separated clusters, LFDA is much better than Euclidean distance metric.

TABLE X
BEST ACCURACY ON EACH METHOD

Metric	Accuracy	k
Euclidean	0.893	7
Manhattan	0.889	8
Chebyshev	0.867	7
Covariance Metric	0.878	8
LMNN	0.903	7
LFDA	0.899	7
LSML	0.868	8
SDML	0.840	5
RCA	0.873	7

V. Conclusion

In this project, we use KNN to classify the AwA2 dataset with different distance metrics. We use K-fold to do cross validation, PCA to do dimension reduction (without losing performance). Altogether, we try 4 different distance metrics, Euclidean distance, Manhattan distance, Chebyshev distance, and Mahalanobis distance.

We implement 6 metric learning methods: covariance metric, LMNN, LFDA, LSML, SDML and RCA to learn new feature representations, which can be used as a Mahalanobis distance metric. We find that some metric learning methods such as LMNN and LFDA can actually learn a better representation of the feature space, which improves the accuracy of KNN classifier. We use t-SNE to visualize the learnt feature representation, and we can see that LMNN and LFDA can learn a feature space which separate different clusters and shrink them.

In conclusion, Euclidean distance is a quite simple yet well-performed distance metric. In addition, with appropriate metric learning method, we can improve the the feature space and benefit the follow-up tasks, such as classification.

REFERENCES

[1] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038 – 2048, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320307000027

[2] R. D. Maesschalck, D. Jouan-Rimbaud, and D. Massart, "The mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, pp. 1 – 18, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169743999000477

[3] A. Singh, A. Yadav, and A. Rana, "K-means with three different distance metrics," *International Journal of Computer Applications*, vol. 67, no. 10, 2013.

[4] P.-E. Danielsson, "Euclidean distance mapping," *Computer Graphics and image processing*, vol. 14, no. 3, pp. 227–248, 1980.

[5] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *City*, vol. 1, no. 2, p. 1, 2007.

[6] G. Qian, S. Sural, Y. Gu, and S. Pramanik, "Similarity between euclidean and cosine angle distance for nearest neighbor queries," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 1232–1237.

[7] K. C. Assi, H. Labelle, and F. Cheriet, "Modified large margin nearest neighbor metric learning for regression," *IEEE Signal Processing Letters*, vol. 21, no. 3, pp. 292–296, 2014.

[8] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in neural information processing systems*, 2005, pp. 513–520.

[9] 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.

[10] 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*. ACM, 2007, pp. 209–216.

[11] E. Y. Liu, Z. Guo, X. Zhang, V. Jojic, and W. Wang, "Metric learning from relative comparisons by minimizing squared residual," in *2012 IEEE 12th International Conference on Data Mining*. IEEE, 2012, pp. 978–983.

[12] G.-J. Qi, J. Tang, Z.-J. Zha, T.-S. Chua, and H.-J. Zhang, "An efficient sparse metric learning in high-dimensional space via l 1-penalized log-determinant regularization," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 841–848.

[13] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning a mahalanobis metric from equivalence constraints," *Journal of Machine Learning Research*, vol. 6, no. Jun, pp. 937–965, 2005.

[14] R. Linsker, "An application of the principle of maximum information preservation to linear systems," in *Advances in neural information processing systems*, 1989, pp. 186–194.

[15] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Advances in neural information processing systems*, 2003, pp. 521–528.