Principles of Data Science Project 2
K-Nearest Neighbors Classification with Different Distance Metrics

Junhao Dai daijunhao@sjtu.edu.cn

April 19, 2024

1 Introduction

1.1 Experimental Configuration

The Python version used for this experiment is 3.11.5, with the following library versions:

- NumPy version: 1.26.2
- Pandas version: 2.0.3
- Scikit-learn version: 1.3.0
- metric-learn version: 0.7.0

Additionally, it's worth noting that this experiment was conducted without the use of servers; all programs were run on a laptop with a CPU model AMD Ryzen 7 5800H with Radeon Graphics, operating at 3.20 GHz.

1.2 My Work

In this project, we are required to explore the optimal distance metrics method to comlpete the task of data classification by using K-Nearest Neighbors Classification. Firstly we reuse the Animals with Attributes (AwA2) dataset in Project 1. We use the k-fold cross-validation method to find the optimal hyperparameter, which in this experiment is K in KNN, i.e., the number of k nearest neighbors.

Secondly, we train the KNN model using unused distance metrics, and use the classification accuracy of the model on the test set as the performance evaluation criterion, in order to compare the advantages and disadvantages of different distance metrics.

Then we use the metric learning approach to learn a transformation function that maps the data points from the original vector space to a new vector space where the similar points are closer and the non-similar points are farther away, and the metric is more in line with the task, thus allowing the KNN to perform better.

2 Experiments and Results

In this section, I will provide a detailed description of my experiment's methodology and corresponding results. Firstly we show the result of the baseline, and then we measure different distance metrics, which are Euclidean, Manhattan, and Minkowski distance. After that, in the metric learning part of this experiment, we use MMC, ITML and LMNN methods respectively to preprocess the dataset.

2.1 Baseline

K-fold cross-validation is a technique used to evaluate the performance of machine learning models. In this method, the original dataset is randomly divided into K subsets, called " folds." Then, the model is trained K times, each time using K-1 folds of data to train the model and the remaining one fold to evaluate the model' s performance. In this experiment, we set the parameter k (i.e., the number of folds) to 5.

In our experiment, in order to find the optimal hyperparameter K, we set the range of k to be from 2 to 49. For each k, we use the training set to fit the KNN model, with the default metric being minkowski. The results are shown in Figure 1 and Table 1.

Table 1: Cross-validated Accuracy on Training Data for Different K Values

k	cv_score (%)	k	cv_score (%)	k	cv_score (%)
2	86.44	18	88.43	34	87.26
3	88.55	19	88.47	35	87.17
4	88.59	20	88.24	36	87.09
5	89.05	21	88.28	37	87.11
6	88.70	22	88.09	38	87.07
7	**89.12**	23	88.07	39	87.06
8	88.96	24	87.97	40	87.01
9	89.05	25	87.95	41	86.93
10	88.97	26	87.89	42	86.85
11	89.09	27	87.85	43	86.82
12	88.95	28	87.73	44	86.85
13	88.92	29	87.74	45	86.82
14	88.89	30	87.59	46	86.85
15	88.86	31	87.50	47	86.82
16	88.66	32	87.42	48	86.85
17	88.56	33	87.39	49	86.82

From Table 1 we can get that the optimal K is equal to 7 and the best baseline result is obtained as shown in Table 2:

Table 2: Results of the best K value in KNN

Best K	Best Acc(%)
7	87.93

2.2 Simple Distance Metrics

In machine learning tasks, usually different vectors represent different samples, and a small distance between two samples is understood to mean that the similarity between the two samples is high, and conversely if the distance between two samples is large, the similarity between them is low.

Among the distance metrics, the most familiarly known as the Euclidean distance, the Euclidean distance between two vectors x and y is defined as:

$$d(x,y) = \sqrt{\Sigma_i (x_i - y_i)^2} \tag{1}$$

The Manhattan distance between two vectors x and y is defined as:

$$d(x,y) = \Sigma_i |x_i - y_i| \tag{2}$$

In addition to this commonly used distance metric is the Minkowski distance, which is defined as the pth order distance between two vectors and can be written as:

$$d(x,y) = (\Sigma_i |x_i - y_i|^p)^{\frac{1}{p}} \tag{3}$$

Obviously, the Euclidean distance is a special case of the Min distance when p=1. When p=2, it is the Manhattan distance, which is an algorithm that subtracts the absolute values of the corresponding points.

In this section, we set the metrics of KNN model as Euclidean distance, Manhattan distance and Minkowski distance, respectively, and set the range of k as [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19], and we use the accuracy of the KNN model's classification on the test set as the criterion of the utility assessment, and we get the results of the Minkowski distance in Figure 2, Table 3; the results of the Euclidean distance in Figure 3, Table 3; the results of the Manhatan distance in Figure 4, Table 3.

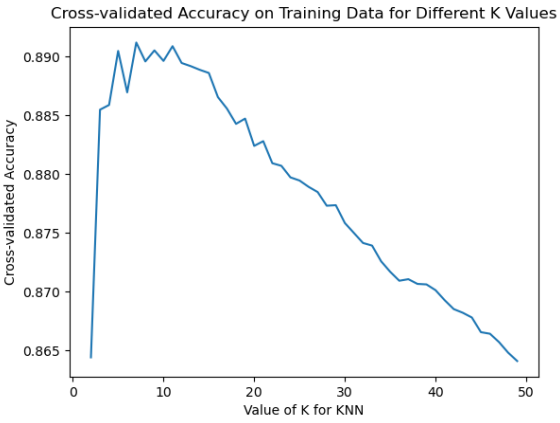


Fig 1: Cross-validated Accuracy on Training Data for Different K Values.

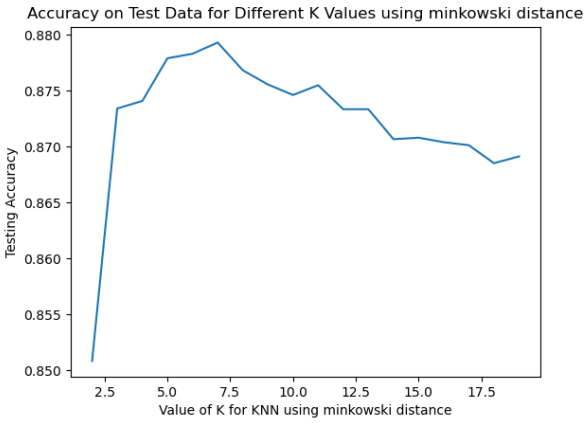


Fig 2: The results of Minkowsi metrics.

From Table 3, it can be seen that all three metrics perform best and have the highest accuracy when k=7. minkowski and Euclidean distance results are the same, and it is speculated that the default minkowski model of KNN model is the Euclidean distance, meanwhile the Manhattan distance performs a little bit worse. Compared to the Euclidean distance, the Manhattan distance

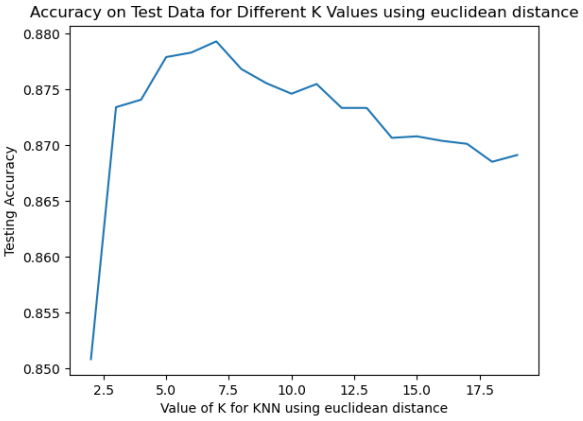


Fig 3: The results of Euclidean metrics.

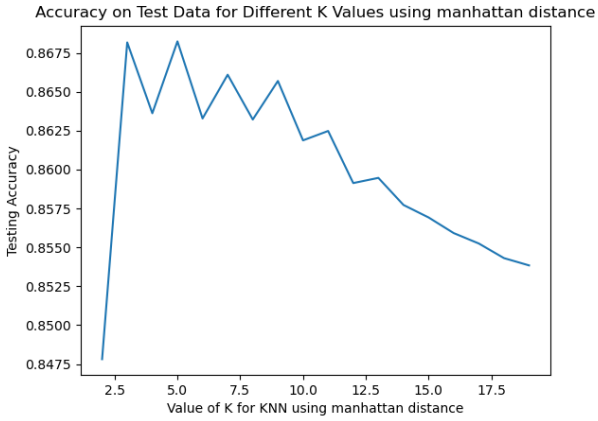


Fig 4: The results of Manhattan metrics.

is less sensitive to outliers because it does not square the differences. This can make it more suitable for certain datasets or problems where the presence of outliers may have a significant impact on the performance of the model.

Table 3: Accuracy for Different Distance Metrics

k	Euclidean	Manhattan	Minkowski	k	Euclidean	Manhattan	Minkowski
2	85.08	84.78	85.08	11	87.55	86.25	87.55
3	87.34	86.82	87.34	12	87.33	85.91	87.33
4	87.41	86.36	87.41	13	87.33	85.95	87.33
5	87.79	86.82	87.79	14	87.07	85.77	87.07
6	87.83	86.33	87.83	15	87.08	85.69	87.08
7	**87.93**	**86.61**	**87.93**	16	87.04	85.59	87.04
8	87.68	86.32	87.68	17	87.01	85.52	87.01
9	87.55	86.57	87.55	18	86.85	85.43	86.85
10	87.46	86.19	87.46	19	86.91	85.38	86.91

2.3 Metrics Learning

Distance metric learning aims at automatically constructing task-specific distance metrics from (weakly) supervised data, in a machine learning manner. The learned distance metric can then be used to perform various tasks (e.g., k-NN classification, clustering, information retrieval).

In the metric-learn package, all algorithms currently implemented learn so-called Mahalanobis distances. Given a real-valued parameter matrix of shape (`num_dims, n_features`) where `n_features` is the number features describing the data, the Mahalanobis distance associated with is defined as follows:

$$D\left(x, x'\right) = \sqrt{\left(Lx - Lx'\right)^T \left(Lx - Lx'\right)} \tag{4}$$

In other words, a Mahalanobis distance is a Euclidean distance after a linear transformation of the feature space defined by (taking to be the identity matrix recovers the standard Euclidean distance). Mahalanobis distance metric learning can thus be seen as learning a new embedding space

of dimension num_dims. Note that when num_dims is smaller than n_features, this achieves dimensionality reduction. In this experiment we used two global supervised learning methods, MMC and ITML, respectively, and a local supervised learning method, LMNN.

2.3.1 MMC

Mahalanobis Metric Learning for Clustering (MMC)[3] minimizes the sum of squared distances between similar points, while enforcing the sum of distances between dissimilar ones to be greater than one. This leads to a convex and, thus, local-minima-free optimization problem that can be solved efficiently. The algorithm aims at minimizing the sum of distances between all the similar points, while constrains the sum of distances between dissimilar points:

$$\min_{\mathbf{M}\in\mathbb{S}_+^d} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in S} d_{\mathbf{M}}\left(\mathbf{x}_i,\mathbf{x}_j\right) \qquad \text{s.t.} \qquad \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in S} d_{\mathbf{M}}^2\left(\mathbf{x}_i,\mathbf{x}_j\right) \geq 1 \tag{5}$$

We set the range of k to be [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19] as well, the metric of the KNN model to be euclidean, set the maximum number of constraints for the MMC model to be 200, and set the diagonal to be true to compute the diagonal elements, otherwise the model will take a very long time to converge on the events. The results obtained are shown in Table 4 and Figure 5, and it is easy to find that the best results are obtained when k is also equal to 7, which is consistent with the results of baseline. The highest accuracy reaches 88.10%, which is slightly higher than baseline, indicating that the mapping of MMC to the training set is favorable for KNN performance improvement.

Table 4: Accuracy of MMC_euclidean and ITML_euclidean

MMC_euclidean				ITML_euclidean			
k	Accuracy	k	Accuracy	k	Accuracy	k	Accuracy
2	85.20	11	87.69	2	81.34	11	82.42
3	87.34	12	87.47	3	83.26	12	82.07
4	87.40	13	87.59	4	82.83	13	82.07
5	87.94	14	87.36	5	83.26	14	81.79
6	87.92	15	87.45	6	82.95	15	81.73
7	**88.10**	16	87.27	7	**83.08**	16	81.53
8	87.90	17	87.19	8	82.61	17	81.36
9	87.84	18	87.06	9	82.62	18	81.20
10	87.82	19	87.02	10	82.35	19	81.20

2.3.2 ITML

Information Theoretic Metric Learning (ITML) [1] 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. Given a Mahalanobis distance parameterized by M , its corresponding multivariate Gaussian is denoted as:

$$p\left(\mathbf{x};\mathbf{M}\right)=\frac{1}{Z}\exp\left(-\frac{1}{2}d_{\mathbf{M}}\left(\ \mathbf{x},\mu\ \right)\right)=\frac{1}{Z}\exp\left(-\frac{1}{2}\left(\left(\mathbf{x}-u\right)^{T}\mathbf{M}\left(\mathbf{x}-\mu\right)\right)\right) \tag{6}$$

where Z is the normalization constant, the inverse of Mahalanobis matrix $\mathbf{M}^{-1}$ is the covariance of the Gaussian.

We also set the maximum number of constraints to 200, but note that the ITML model does not have a diagonal parameter, and the rest of the settings are the same as MMC, and the results are shown in Table 4 and Figure 6. From the results, it is easy to see that ITML performs much worse than the baseline and MMC models, while the performance degradation is extremely severe in the case of larger K values. The best result is still the 83.08% accuracy when k is taken to be 7.

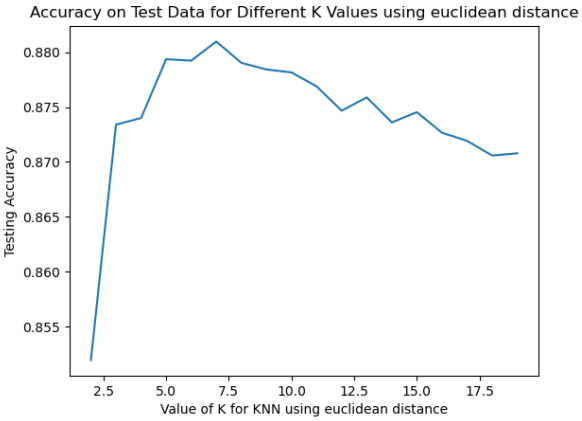


Fig 5: The results of MMC_euclidean metrics.

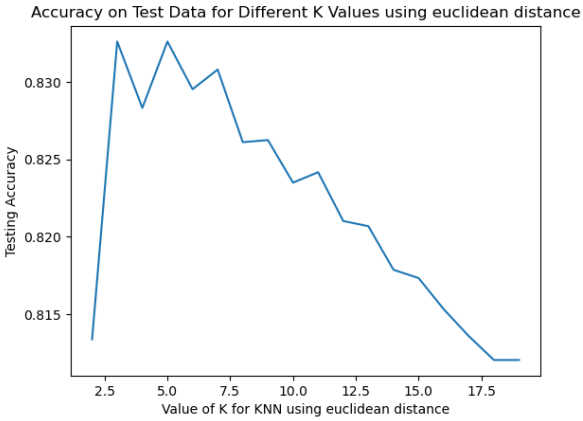


Fig 6: The results of ITML_eculidean metrics.

It is found that when the training samples are high-dimensional data, the metric matrix obtained by the ITML algorithm cannot effectively suppress the noise, and also has the disadvantages of low solving efficiency and being easily affected by the lack of training data.

2.3.3 LMNN

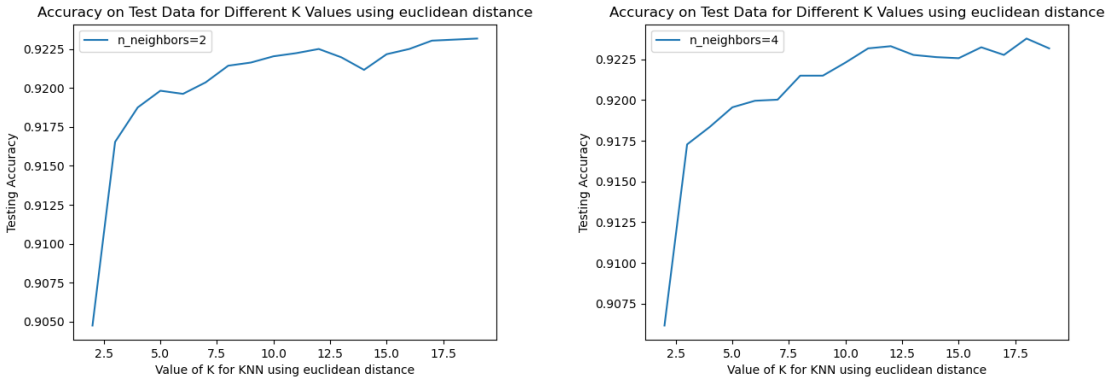
Large Margin Nearest Neighbor Metric Learning (LMNN)[2] learns a Mahalanobis distance metric in the kNN classification setting. The learned metric attempts to keep close k-nearest neighbors from the same class, while keeping examples from different classes separated by a large margin. This algorithm makes no assumptions about the distribution of the data. The distance is learned by solving the following optimization problem:

$$\min_{\mathbf{L}}\sum_{i,j}\eta_{ij}\left\|\mathbf{L}\left(\mathbf{x_i}-\mathbf{x_j}\right)\right\|^2+c\sum_{i,j,l}\eta_{ij}\left(1-y_{ij}\right)\left[1+\left\|\mathbf{L}\left(\mathbf{x_i}-\mathbf{x_j}\right)\right\|^2-\left\|\mathbf{L}\left(\mathbf{x_i}-\mathbf{x_l}\right)\right\|\right]_+ \tag{7}$$

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

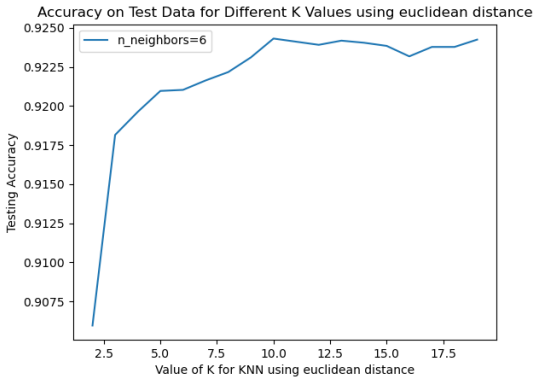
While training the LMNN model, the dataset is too high dimensional to be loaded in memory, so first we will use LDA to reduce the dimensionality of the original dataset. After testing, the higher the dimension, the higher the accuracy, so we choose the highest latitude within the feasible range, i.e., the dimension of the FEATURE minus 1, which is taken as 49 in this experiment. We

set the learning rate of LMNN to be 1e-6, the range of neighbor to be [2,4,6,8], and the range of k to be set as the same as MMC, and also use the accuracy of the KNN model as the evaluation criterion, and get the results as follows:

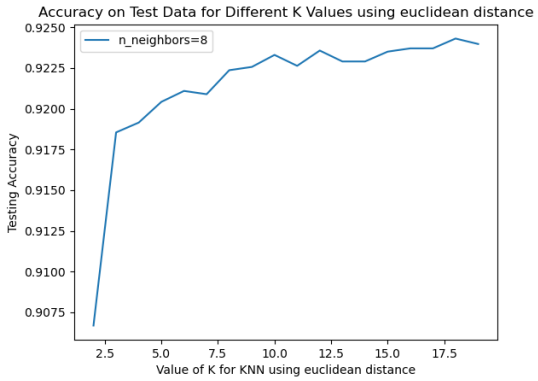


(a) neighbor = 2

(b) neighbor = 4



(c) neighbor = 6



(d) neighbor = 8

Figure 7: The results of LMNN for different neighbors.

The curves with different neighbor numbers are placed on the same graph for comparison and observation, and the results are shown in Figure 8:

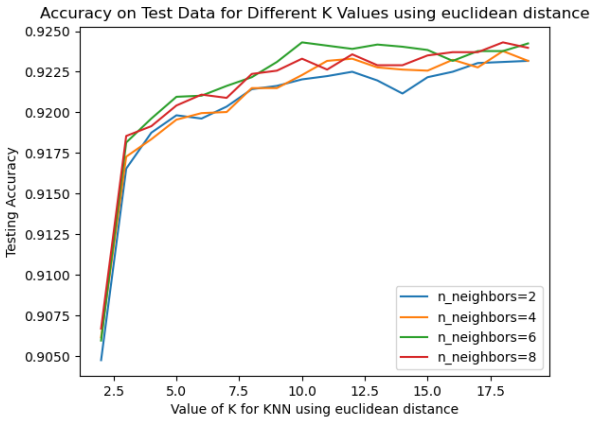


Figure 8: The results of LMNN for different neighbors in one Fig.

From the figure we can easily see that the results obtained after using LMNN are all better than baseline. when neighbor is equal to 6, we get the optimal results, combined with the results table 5, when k is equal to 10, LMNN gets the highest accuracy of 92.43%. Meanwhile the other three cases, when k reaches 19 the accuracy is still on the rise. The result here is not consistent with the conclusion that the optimal k is 7 obtained by baseline, which is presumed to be due to the dimensionality reduction operation of LDA, which makes the vector distance of the dataset change, and the optimal k changes accordingly.

Table 5: LMNN accuracy for different values of k and neighbors.

k	Neigh_2	Neigh_4	Neigh_6	Neigh_8	k	Neigh_2	Neigh_4	Neigh_6	Neigh_8
2	90.47	90.62	90.60	90.67	11	92.22	92.25	92.20	92.12
3	91.65	91.73	91.81	91.85	12	92.25	92.33	92.28	92.26
4	91.87	91.83	91.96	91.91	13	92.20	92.36	92.28	92.26
5	91.98	91.96	92.10	92.04	14	92.12	92.36	92.27	92.29
6	91.96	91.99	92.10	92.11	15	92.21	92.36	92.42	92.26
7	92.04	92.00	92.16	92.09	16	92.25	92.42	92.28	92.26
8	92.14	92.15	92.21	92.24	17	92.30	92.42	92.28	92.38
9	92.16	92.15	92.31	92.29	18	92.31	92.42	92.28	92.38
10	92.20	92.23	**92.43**	92.24	19	**92.32**	**92.42**	92.28	**92.40**

3 Conclusion

In the first part we use the k-fold cross-validation method to find the best k-parameters for the KNN model, in the second part we use different distance metrics to train the KNN model, and in the third part we use different metric learning methods to process the training set。

By testing and comparing the classification accuracy of the KNN models on the test set, we conclude that euclidean's KNN model is more accurate compared to Manhattan's KNN model. metric learning's approach, the MMC model causes a small increase in the model's accuracy, and instead of an increase, the ITML model has at larger values of k The training of LMNN model is much longer than MMC and ITML, but it can be completed in two and a half hours after LDA dimensionality reduction, and at the same time, the classification performance is greatly improved, compared with the baseline of 87.93% LMNN can reach 92.43% accuracy.

References

[1] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216, 2007.

[2] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

[3] Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15, 2002.