KNN Classification with Different Distance Metrics

Jizheng Chen 520021911182

April 23, 2023

1 Introduction

K-Nearest Neighbors (KNN) is an effective algorithm used for both classification and regression tasks, where the K refers to the number of nearest neighbors to consider when doing clustering work. For a given input dataset, the algorithm finds the K nearest data points in it and uses their labels (for classification) or values (for regression) to determine the label or value of the input data point.

As in the KNN training process the algorithm needs to find the nearest data points, a certain distance metric is required to measure the similarity and distance between data points. The distance between data points is typically calculated using **Euclidean distance** or **Manhattan distance**, although other distance metrics can also be used. The value of K is usually chosen based on cross-validation, as is used in this project.

One of the advantages of KNN is its simplicity and ease of implementation. However, KNN also has some drawbacks including:

- KNN can be computationally expensive on large datasets, for the reason that calculating distances between many data points cost much computation time.
- When the data is high-dimensional, the algorithm does not perform well due to the curse of dimensionality.

The above two drawbacks is verified through experiments, and to deal with the problems, **Metric Learning** is applied. Metric learning deals with the problem of learning a distance metric between data points in a dataset. The goal of metric learning is to find a distance metric that maps similar data points close to each other in the feature space, while mapping dissimilar data points far away from each other.

There are several approaches to metric learning, including supervised and unsupervised methods. In supervised metric learning, labeled training data is used to learn a distance metric that maximizes the inter-class distance and minimizes the intra-class distance. Examples of supervised metric learning methods include the Mahalanobis distance, the Large Margin Nearest Neighbor (LMNN) algorithm, and the Siamese Neural Network.

In unsupervised metric learning, only the unlabeled data is available, and the goal is to learn a distance metric that preserves the local and global structure of the data. Examples of unsupervised metric learning methods include the Neighborhood Component Analysis (NCA), the Information Theoretic Metric Learning (ITML) algorithm, and the Distance Metric Learning via Gradient Descent (DMLGD) algorithm.

The rest of the report will be illustrated as follows: Section 2 gives some background knowledge of metric learning method used, and Section 3 elaborates the experiment details as well as the results get. Finally Section 4 gives a conclusion of this whole project.

2 Background Knowledge

2.1 Metric Learning

Many approaches in machine learning require a measure of distance between data points. Traditionally, practitioners would choose a standard distance metric (Euclidean, City-Block, Cosine, etc.) using

a priori knowledge of the domain. However, it is often difficult to design metrics that are well-suited to the particular data and task of interest.

Distance metric learning (or simply, 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).

2.2 Mahalanobis Distances

Many metric learning algorithms aims to learn so-called Mahalanobis distances. Given a real-valued parameter matrix L of shape (num_dims, n_features) where n_features is the number features describing the data, the Mahalanobis distance associated with L is defined as follows:

$$D(x, x') = \sqrt{(Lx - Lx')^T (Lx - Lx')} \tag{1}$$

In other words, a Mahalanobis distance is a Euclidean distance after a linear transformation of the feature space defined by L (taking L 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.

2.3 Neighborhood Components Analysis

NCA is a distance metric learning algorithm which aims to improve the accuracy of nearest neighbors classification compared to the standard Euclidean distance. The algorithm directly maximizes a stochastic variant of the leave-one-out k-nearest neighbors (KNN) score on the training set. It can also learn a low-dimensional linear transformation of data that can be used for data visualization and fast classification.

NCA uses the decomposition $M = L^T L$ and define the probability p_{ij} that x_i is the neighbor of x_j by calculating the softmax likelyhood of the **Mahalanobis distance** by:

$$p_{ij} = \frac{exp(-||Lx_i - Lx_j||_2^2)}{\sum_{l \neq i} exp(-||Lx_i - Lx_l||_2^2}, p_{ii} = 0 \tag{2}$$

Then the probability that x_i will be correctly classified by the stochastic nearest neighbors rule is:

$$p_i = \sum_{j: j \neq i, y_j = y_i} p_{ij} \tag{3}$$

The optimization problem is to find matrix L that maximizes the sum of probability of being correctly classified:

$$L = argmax \sum_i p_i \tag{4}$$

2.4 Metric Learning for Kernel Regression

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:

$$k_{ij} = \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{d(x_i, x_j}{\sigma^2}) \tag{5}$$

where $d(x, y)$ is the squared distance under some metrics, here in the fashion of Mahalanobis, it should be $d(x_i, x_j) = ||L(x_i - x_j)||^2$, , the transition matrix L is derived from the decomposition of Mahalanobis matrix $M = L^T L$.

2.5 Local Fisher Discriminant Analysis

LFDA is a linear supervised dimensionality reduction method which effectively combines the ideas of Linear Discriminant Analysis and Locality-Preserving Projection . 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.

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

$$S^{(w)} = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}^{(w)}(x_i - x_j)(x_i - x_j)^T, \tag{6}$$

$$S^{(b)} = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}^{(b)}(x_i - x_j)(x_i - x_j)^T \tag{7}$$

where

$$W_{ij}^{(w)} = \begin{cases} 0, y_i \neq y_j \\ A_{ij}/n_l, y_i = y_j \end{cases}$$

$$W_{ij}^{(b)} = \begin{cases} \frac{1}{n}, y_i \neq y_j \\ A_{ij}(\frac{1}{n} - \frac{1}{n_l}), y_i = y_j \end{cases}$$

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

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

$$L_{LFDA} = argmax_L tr((L^T S^{(W)} L)^{-1} L^T S^(b) L \tag{8}$$

3 Experiments

This section elaborates the experiment settings and results we get from KNN and metric learning methods. As there are many data, the result table is put in appendix section, and in the experiment section I mainly show the results in graph form.

3.1 Experimental Settings and Details

3.1.1 Dataset

The KNN and metric learning methods are trained and evaluated on AwA2 dataset, which is a widely used benchmark dataset in the field of computer vision and machine learning. The dataset consists of 37322 images of 50 different animal classes, each annotated with 85 binary attribute labels. Here I directly used the features extracted by a Resnet101, which is a vector of length 2048 for each individual picture sample. The original dataset is divided into a training set consisting 60% of data, and a testing set of 40% data.

3.1.2 Metric Learning Details

In my series of experiments, three hyper-parameters are considered and explored: the **Neighbor number K** that decides the number of nearest points in KNN classification, the **Number of Components** in metric learning methods, and the **Distance Metric** applied in calculating the distance between two points. In the basic setting of KNN, the neighbor number k is set to 2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,30,50 to explore the detailed influence of k on classification performance, and number of components is set to 32,64,128,256. Manhattan and Euclidean distance are used to measure the difference between the two points.

3.2 Experiment Results

This section gives all my experiment results in both tabular and graph forms. One point worth mentioning is that, some metric learning methods consumes much time to learn, and running KNN on different parameter settings after metric learning is redundant work. To deal with it, I stored every transformed numpy array in a .npy file that corresponding KNN algorithms can directly run on, which saves me a lot of running time (Figure 12), the transformed data and all the codes can be found at https://github.com/Otsuts/PDS-projects

Four questions are raised to verify the performance of metric learning methods:

1. **Which distance measure method, (Manhattan distance or Euclidean) distance, can get better classification performance?**
2. **Can the three metric learning methods used outperform baseline KNN in accuracy? If so, which one of them performs best?**
3. **How will the performance change along with the number of neighbors K in each method?**
4. **How will the performance change along with the number of components in each metric learning method?**

The following parts will answer the four questions in an elaborated way.

3.2.1 Plain KNN

When no metric learning metric is applied, under Manhattan and Euclidean distance settings, the change of performance and the number of neighbors is shown at Figure 1 and Table 1.

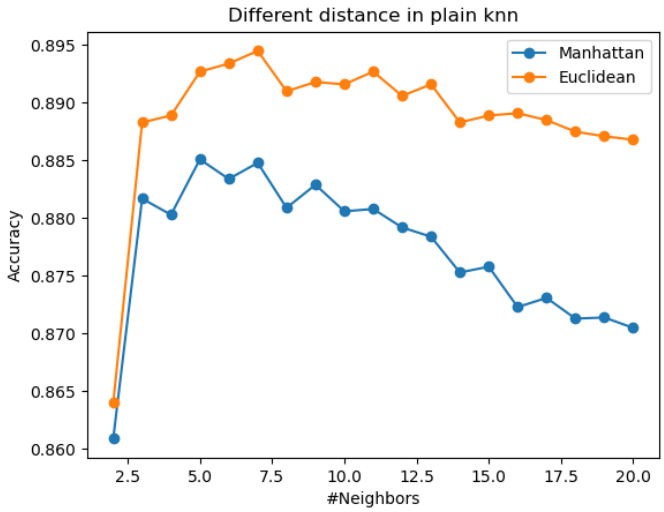

Figure 1: Plain KNN results using two distance metrics

From Table 1 and Figure 1 we can draw several conclusions:

- The best accuracy is 89.45 when K = 7 and using Euclidean distance. This performance is superior to SVM when using all the data without doing dimensionality reduction.
- As for the two distance metrics, Euclidean distance gets a better performance compared with Manhattan distance, possible reason behind that may be that the measuring way of Euclidean distance is more suitable to the original data distribution.
- When the number of neighbors is small, the performance is not good, and when the number of neighbors gets too large, performance will also drop. The best K=7 implies that the original data may form to be clusters with 7 data points inside.

3.2.2 NCA Metric Learning Method

Performance using NCA metric with different components and K under Manhattan distance is at Table 3, and the results using Euclidean distance is at Table 2, all the results of NCA are shown in Figure 2 and Figure 3 . From it we can draw the following conclusions:

- Generally Euclidean distance get better results than Manhattan distance, but best result is 91.47 when K=14, component=256 and using Manhattan distance.
- All the two metric learning methods get a better performance compared with baseline KNN, verifying the contribution of NCA.
- We get similar finding of K's change to the baseline KNN, i.e. performance increases and then decreases when K changes from 2 to 50.
- The bigger the number of components gets, the better the performance is as to the NCA case.

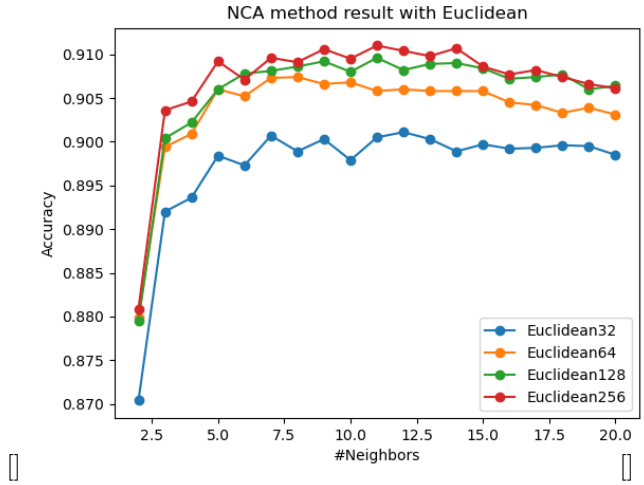

Figure 2: Euclidean NCA

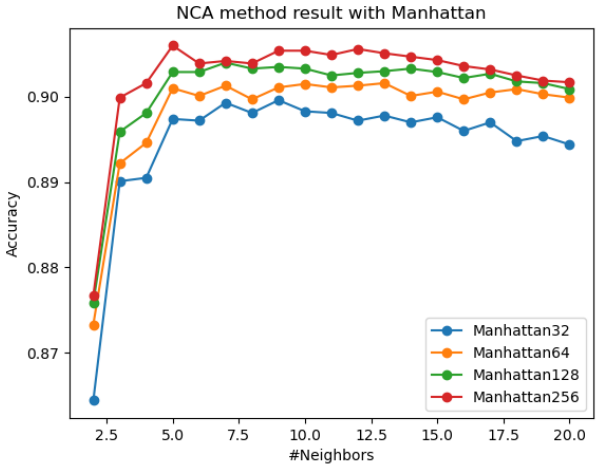

Figure 3: Manhattan NCA

Figure 4: NCA results

3.2.3 LFDA Metric Learning Method

Performance using LFDA metric with different components and K under Manhattan distance is at Table 5, and the results using Euclidean distance is at Table 4, all the results of LFDA are shown in Figure 5 and Figure 6 . From it we can draw the following conclusions:

- Still Euclidean distance get better results than Manhattan distance, and best result is 92.20 when K=9, component=64 and using Euclidean distance.
- All the two metric learning methods get a better performance compared with baseline KNN, verifying the superiority of LFDA to both baseline KNN and NCA.
- We get similar finding of K's change to the baseline KNN, i.e. performance increases and then decreases when K changes from 2 to 50.
- Different from the conclusion in NCA, the best number of components lies in 64 or 128, and when the components gets too big, performance deceases.

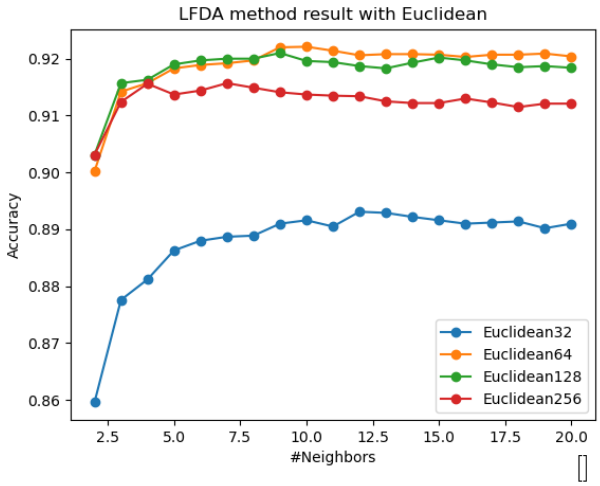

Figure 5: Euclidean LFDA

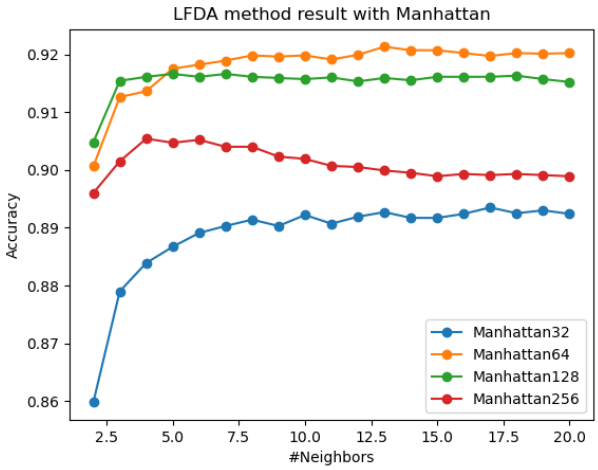

Figure 6: Manhattan LFDA

Figure 7: LFDA results

3.2.4 MLKR Metric Learning Method

Performance using MLKR metric with different components and K under Manhattan distance is at Table 7, and the results using Euclidean distance is at Table 6, all the results of MLKR are shown in Figure 8 and Figure 9 . From it we can draw the following conclusions:

- Still Euclidean distance get better results than Manhattan distance, and best result is 90.74 when K=8, component=128 and using Euclidean distance.
- All the two metric learning methods get a better performance compared with baseline KNN, verifying the superiority of MLKR, but MLKR can not outperform NCA and LFDA in final performance.
- We get similar finding of K's change to the baseline KNN, i.e. performance increases and then decreases when K changes from 2 to 50.
- Different from the conclusion in NCA, the best number of components lies in 64 or 128, and when the components gets too big, performance deceases.

4 Conclusion

In this section I summarizes all the findings in above sections as Figure 11 to answer the four questions raised previously.

First we can find that using Euclidean distance can generally get a better performance in all the methods used above. But the best result doesn't necessarily appear in Euclidean method (as is the case in NCA), that's probably because metric learning methods decreases the difference between the two distance metrics and learn a new distance that can better represent features.

Then, all the metric learning methods outperforms plain KNN, verifying the performance of used metric learning methods. And LFDA gets the best result, with an excellent time consumption. MLKR takes most time and the performance is just so so.

Finally, different methods have different most suitable K and N. For example NCA prefers a bigger N, while in LFDA and MLKR case a smaller N can get better results. Also, a K around 10 is the best in all the cases.

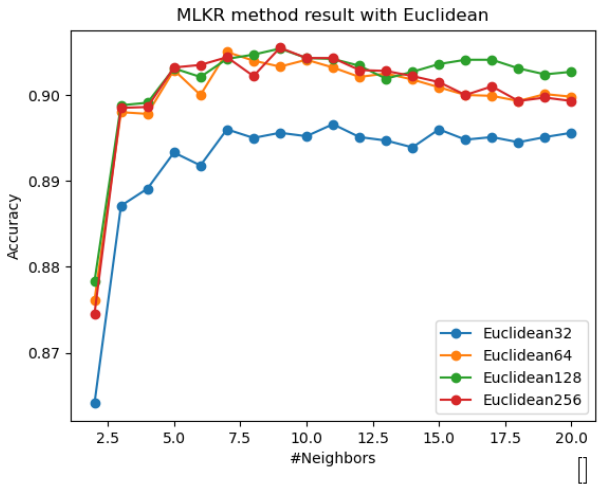

Figure 8: Euclidean MLKR

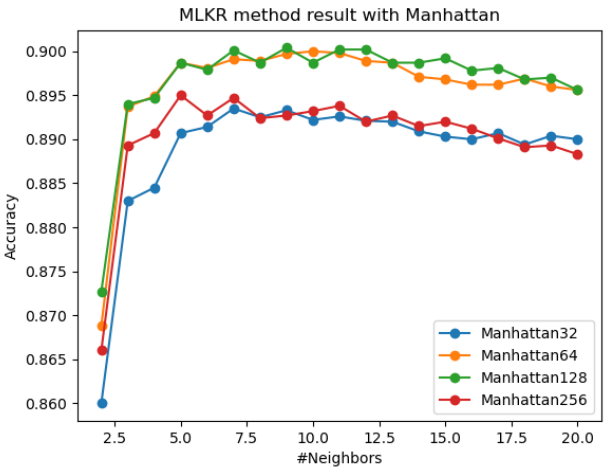

Figure 9: Manhattan MLKR

Figure 10: MLKR results

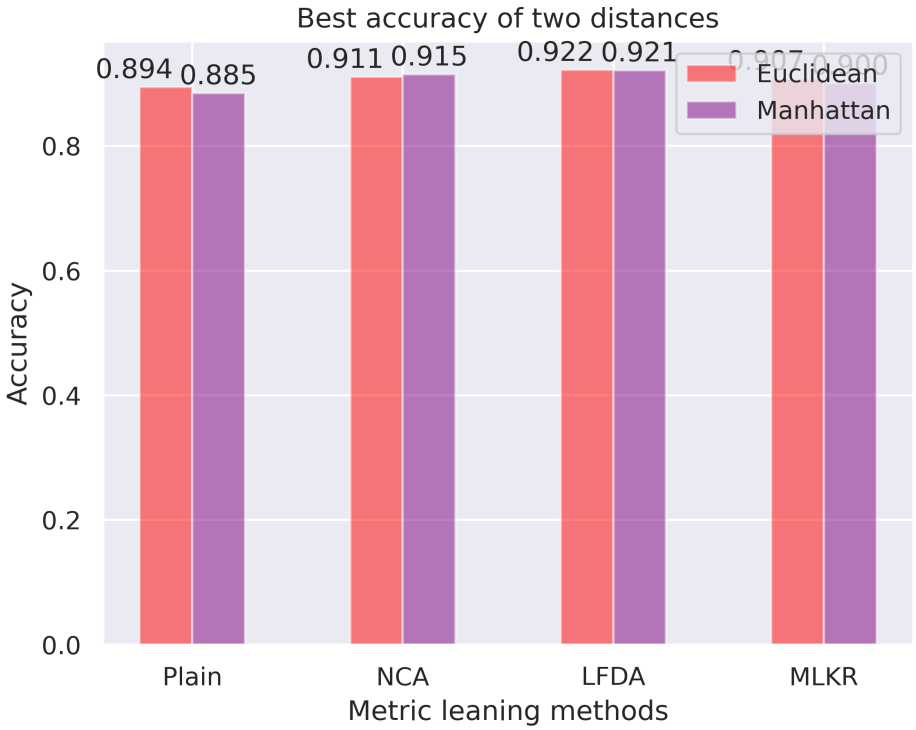

Figure 11: Final results

	2	3	4	5	6	7	8	9	10	11	12	13	14
Manhattan	86.09	88.17	88.03	88.51	88.34	88.48	88.09	88.29	88.06	88.08	87.92	87.84	87.53
Euclidean	86.40	88.83	88.89	89.27	89.34	**89.45**	89.10	89.18	89.16	89.27	89.06	89.16	88.83

	15	16	17	18	19	20	30	50
Manhattan	87.58	87.23	87.31	87.13	87.14	87.05	86.23	84.84
Euclidean	88.89	88.91	88.85	88.75	88.71	88.68	87.89	86.74

Table 1: Performance with two distance metrics and different K.

	2	3	4	5	6	7	8	9	10	11	12	13	14
32	87.04	89.20	89.36	89.84	89.73	90.07	89.89	90.03	89.79	90.05	90.11	90.03	89.89
64	87.99	89.94	90.09	90.60	90.52	90.73	90.74	90.66	90.68	90.58	90.60	90.58	90.58
128	87.95	90.04	90.22	90.60	90.78	90.81	90.86	90.92	90.80	90.96	90.82	90.89	90.90
256	88.09	90.36	90.46	90.92	90.71	90.96	90.91	91.06	90.95	**91.10**	91.04	90.98	91.07

	15	16	17	18	19	20	30	50
32	89.97	89.92	89.93	89.96	89.95	89.85	89.89	89.05
64	90.58	90.45	90.42	90.33	90.39	90.31	89.89	89.10
128	90.84	90.72	90.74	90.77	90.60	90.64	90.27	89.46
256	90.86	90.77	90.82	90.74	90.66	90.61	90.20	89.47

Table 2: Performance using NCA metric with different components and K under Euclidean distance .

References

Appendix

In appendix all detailed results in tabular form are given.

	2	3	4	5	6	7	8	9	10	11	12	13	14
32	86.44	89.01	89.05	89.74	89.72	89.93	89.81	89.96	89.83	89.81	89.72	89.78	89.70
64	87.33	89.22	89.46	90.00	90.01	90.13	89.97	90.11	90.15	90.11	90.13	90.16	90.01
128	87.59	89.59	89.81	90.29	90.29	90.40	90.33	90.35	90.33	90.25	90.28	90.30	90.33
256	87.67	89.99	90.16	90.60	90.39	90.42	90.39	91.54	90.54	90.49	90.56	90.51	**91.47**

	15	16	17	18	19	20	30	50
32	89.76	89.60	89.70	89.48	89.54	89.44	89.24	88.75
64	90.06	89.97	90.05	90.09	90.03	89.99	89.55	88.62
128	90.29	90.22	90.27	90.18	90.16	90.09	89.77	88.99
256	90.43	90.36	90.32	90.25	90.19	90.17	89.69	88.93

Table 3: Performance using NCA metric with different components and K under Manhattan distance
.

	2	3	4	5	6	7	8	9	10	11	12	13	14
32	85.96	87.76	88.12	88.63	88.80	88.87	88.89	89.10	89.16	89.05	89.31	89.29	89.22
64	90.02	91.42	91.57	91.83	91.89	91.92	91.97	**92.20**	92.21	92.14	92.06	90.08	92.08
128	90.31	91.57	91.63	91.90	91.97	92.00	92.00	92.10	91.96	91.94	91.87	91.83	91.93
256	90.31	91.24	91.56	91.37	91.44	91.57	91.49	91.41	91.37	91.35	91.34	91.25	91.22

	15	16	17	18	19	20	30	50
32	89.16	89.10	89.12	89.14	89.02	89.10	88.83	88.63
64	92.07	92.03	92.07	92.07	92.09	92.04	91.96	91.87
128	92.02	91.97	91.90	91.85	91.87	91.84	91.83	91.51
256	91.22	91.30	91.23	91.15	91.21	91.21	90.95	90.70

Table 4: Performance using LFDA metric with different components and K under Euclidean distance
.

	2	3	4	5	6	7	8	9	10	11	12	13	14
32	85.98	87.90	88.39	88.67	89.91	89.03	89.14	89.03	89.22	89.07	89.19	89.27	89.17
64	90.06	91.26	91.36	91.75	91.82	91.89	91.98	91.96	91.98	91.91	91.99	**92.13**	92.07
128	90.48	91.54	91.61	91.66	91.61	91.66	91.61	91.59	91.57	91.60	91.53	91.59	91.55
256	89.60	90.15	90.54	90.47	90.52	90.40	90.40	90.23	90.19	90.07	90.05	89.99	89.95

	15	16	17	18	19	20	30	50
32	89.17	89.24	89.35	89.25	89.30	89.24	89.04	88.83
64	92.07	92.02	91.97	92.02	92.01	92.02	91.87	91.76
128	91.61	91.61	91.61	91.63	91.57	91.52	91.45	90.96
256	89.89	89.93	89.91	89.93	89.91	89.89	89.43	89.00

Table 5: Performance using LFDA metric with different components and K under Manhattan distance
.

	2	3	4	5	6	7	8	9	10	11	12	13	14
32	86.41	88.71	88.91	89.33	89.18	89.60	89.50	89.56	89.52	89.66	89.51	89.4	89.39
64	87.61	89.80	89.78	90.18	90.00	90.50	90.40	90.33	90.41	90.32	90.21	90.25	90.18
128	87.83	89.88	89.91	90.31	90.21	90.42	**90.74**	90.54	90.43	90.42	90.34	90.19	90.27
256	87.45	89.85	89.86	90.32	90.35	90.44	90.22	90.55	90.43	90.43	90.29	90.28	90.22

	15	16	17	18	19	20	30	50
32	89.60	89.48	89.51	89.45	89.51	89.56	89.03	88.22
64	90.09	90.00	89.99	89.93	90.01	89.98	89.58	88.79
128	90.36	90.41	90.41	90.31	90.24	90.27	89.63	88.88
256	90.15	90.00	90.10	89.93	89.97	89.93	89.47	88.70

Table 6: Performance using MLKR metric with different components and K under Euclidean distance
.

	2	3	4	5	6	7	8	9	10	11	12	13	14
32	86.00	88.30	88.45	89.07	89.14	89.35	89.25	89.33	89.22	89.26	89.21	89.20	89.09
64	86.88	89.37	89.49	89.87	89.81	89.91	89.89	89.97	90.00	89.98	89.89	89.87	89.71
128	87.27	89.40	89.47	89.87	89.79	90.01	89.87	**90.04**	89.87	90.02	90.02	89.87	89.87
256	86.60	88.93	89.07	89.50	89.27	89.47	89.24	89.27	89.32	89.38	89.20	89.27	89.15

	15	16	17	18	19	20	30	50
32	89.03	89.00	89.07	89.94	89.04	89.00	88.73	87.77
64	89.68	89.62	89.62	89.69	89.60	89.56	89.39	88.51
128	89.92	89.78	89.81	89.68	89.70	89.56	89.18	88.47
256	89.20	89.12	89.01	88.91	88.93	88.83	88.21	87.29

Table 7: Performance using MLKR metric with different components and K under Manhattan distance
.

Figure 12: Npy files of transformed data.