Project2: KNN Classification with Different Distance Metrics

Jieyu Li, Dongyue Li, Hongbin Chen, Haoxuan Wang
Course: CS245 - Data Science Principles

***Abstract*— This is a report on the experiments we have done when using KNN (K-Nearest Neighbots) with different distance metrics to do image classification. We experimented on different distance metrics and evaluated their performance on the AwA2 dataset. We also tried different metric learning methods such as LMNN, LFDA and LSML to help learn a better distance metric for classification. We further discuss the different obtained results and analyz the reasons for their different performances.**

The following text describes our experiments and results in detail.

I. Introduction

A. Project Description

In this project, we tend to use KNN for image classfication based on the deep learning features extracted from the AwA2 dataset. Different simple distance metrics are required to be used and compared. To further improve the KNN performance, metric learning approachs are adapted to help us classify the samples. Based on the experiment result observations, discussions are made in order to analyz how and why the algprithms perform in such different ways.

B. Dataset

The AwA2 (Animals with Attributes) dataset consists of 37322 images of 50 animal classes with pre-extracted deep learning features for each image. We split the images in each category into 60% for training and 40% for testing (which is the same as $Project1$) and shuffle them.

C. Experiment Settings

1) Python3.6
2) Sklearn, Metric_Learn, Pandas, Seaborn, Matplotlib

II. Classfication with Different Neighbors Under Different Distance Metrics

The performance of KNN on a particular dataset mainly depends on our choice of distance metric and the value of K. We choose our distance metric from Euclidean distance, Manhattan distance, Chebyshev distance and Cosine distance, and our choice of K is a value from $1, 3, 5, 8, 10, 15, 20, 30, 40, 50, 100$. After running the experiments, we obtain the following results:

Classification Accuracy				
K Value	Euclidean	Manhattan	Chebyshev	Cosine
1	87.59	87.44	75.97	88.31
3	88.37	87.87	77.02	89.17
5	88.94	88.08	78.50	89.84
8	88.84	87.85	78.56	90.19
10	88.85	87.63	78.45	90.24
15	88.53	86.97	78.36	90.07
20	87.96	86.42	77.97	89.78
30	87.36	85.76	77.01	89.54
40	86.90	84.91	76.10	89.16
50	86.45	84.30	75.08	89.00
100	84.47	81.84	72.29	87.62

TABLE I
Direct KNN

From the tabular, we can find that:

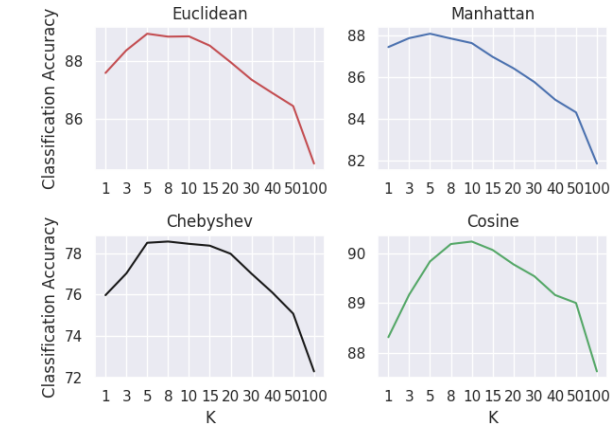
- Under the same value of K, the best performance always comes from Cosine distance.
- Under the same distance metric, the best K is around the value of 5 (10 for Cosine distance).

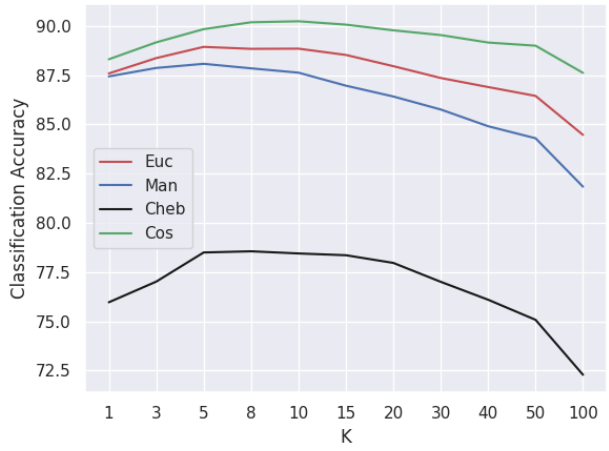
We further analyze the reason behind these phenomenons in the following sections.

A. Relationship Between Accuracy and K Value

First, we plot the graph for the relationship between classification accuracy, K value and distance metrics.

Fig. 1. Relationship Between Acc, K, Distance Metric





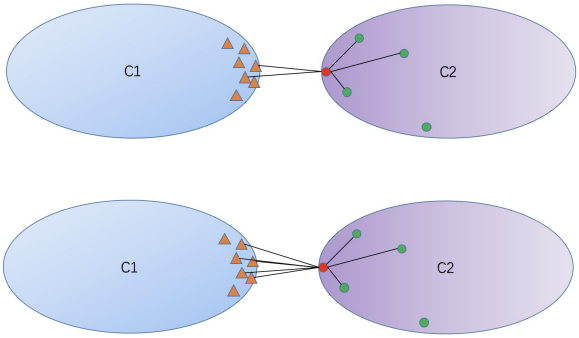


Fig. 2. Above two figures illustrate the condition when K should be smaller

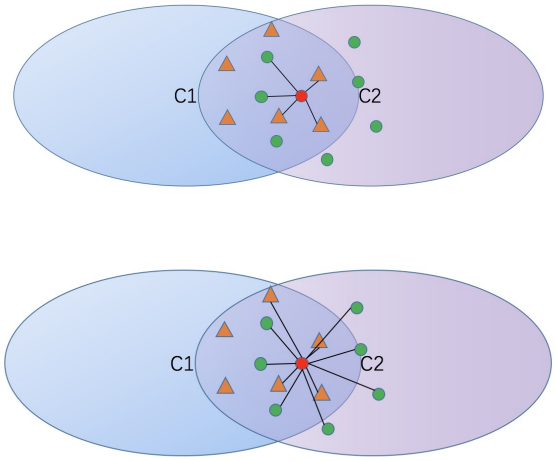


Fig. 3. Above two figures illustrate the condition when K should be larger

Under the same distance metric, the accuracy changes with different K values, and from the curves, we can intuitively infer that there exists a K value that guarantees best accuracy. Thus, we try to analyze this inference through the following graphs. In the 4 drawings, $C1$ and $C2$ represent two different classes, the red point represent the data sample that we are classifying using KNN, yellow triangles represent data samples that belong to class $C1$, and green dots represent data samples that belong to class $C2$. Points that are connected to the red point indicate the data samples that are nearest when doing KNN clustering. Our goal is to correctly classify the red point into class $C2$.

- Circumstance 1: As shown in Fig.2., the upper graph is the condition when $K = 5$ and the lower graph is the condition when $K = 8$. When $K = 5$, the red point could be correctly classified (with 3 nearest points in $C2$ and only 2 nearest data samples in $C1$). However, when K gets larger and becomes 8, the red point would be classified into class $C1$ (with 3 nearest points in $C2$ and 4 nearest points in $C1$). Thus, taking a K whose value is too large might be misleading to the result.
- Circumstance 2: As shown in Fig.3, the upper graph is the condition when $K = 5$ and the lower graph is the condition when $K = 11$. When $K = 5$, the nearest points all fall in the overlapping area of the two classes, and classifying the red point becomes hard and ambiguous. However, if K is increased to 11, then the red point would have better chance of getting in touch with more data samples that

are in the same cluster, and KNN would have higher probability of correctly classifying data samples.
- From the above discussion, we conclude that K's value should be an intermediate value, not too large nor too small. Our observation also agree with this conclusion.

B. Relationship Between Accuracy and Distance Metrics

From TABLE I and Fig.1., we find that:

- Chebyshev performs worse than the 3 others. To explain this, we guess that: (1) The differences are originated from the data dimension. The reduction process and arguments for this can be seen in section IV A. (2) Chebyshev distance is similar to spectural norm, it makes the feature space sparse, and information of other dimensions are lost. (3) Chebyshev and Manhattan distance all depend on the perpendicular bases. If a feature is evenly distributed between the bases, then taking the Chebyshev distance is definitely invalid as choosing any dimension if "unfair" for the others. There are no effective approaches to validate or deal with the first two guesses. But we can change the direction of the coordinates to prove the third guess. Thus, PCA is used to rotate the sample space, and detailed experiments are done in section IV B.
- Cosine distance is better than other distances. The reason: Actaully their performances are similar, with only difference of around 2% at most, thus we regard this as the consequence of randomness during optimization.

III. Adapting Different Metric Learning Methods

We experimented on three different metric learning methods: LMNN (Large Margin Nearest Neighbor), LDFA (Local Fisher Discriminant Analysis), and LSML (Least Squares Metric Learning). While their optimizations details and performances are different, they are based on the same initial idea: Learn a best projection space that can maximize the difference between different classes and minimize the dfference between the same class data samples. Experiment details and observations are shown in the following sections.

A. LMNN

The basic idea of LMNN is to "compress" the data samples from the same class together and at the same time, "push" the other data samples away. It is essentially based on the theory of nearest neighbor, and adopts the Euclidean distance while optimizing. Below shows our experiment results:

Classification Accuracy				
K Value	Euclidean	Manhattan	Chebyshev	Cosine
1	88.76	87.78	74.19	88.61
3	89.52	88.74	74.81	89.95
5	90.48	89.49	76.22	90.59
8	90.60	89.36	76.83	90.91
10	90.52	89.60	76.79	91.00
15	90.66	89.38	76.61	91.06
20	90.37	88.98	76.10	90.96
30	89.91	88.54	75.42	90.71
40	89.57	87.93	74.55	90.61
50	89.20	87.47	73.95	90.27
100	87.71	85.87	71.40	89.61

TABLE II
LMNN+KNN

To get better intuition from the data, we plot the corresponding curves in Fig.4.

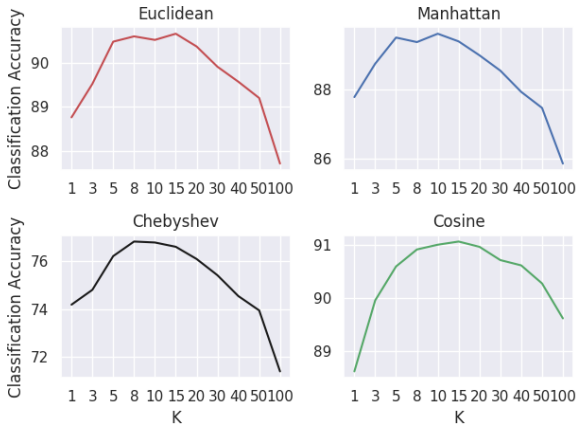
By observing the results, we obtain the following conclusions and present our reasons:

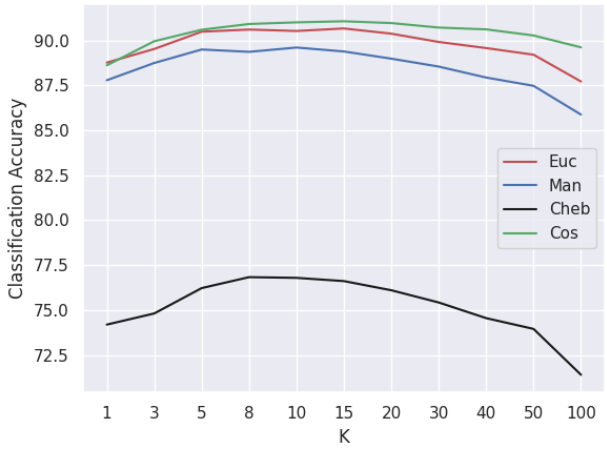
- The best value for K has become larger for all distance metrics (from around 5 to around 15). We think it might be due to two reasons: (1) Since LMNN also takes K as its input parameter, we think the best K is a balanced value to make both LMNN and KNN perform well; (2) LMNN is similar to KNN, thus we are actually doing KNN twice. Thus LMNN might have first clustered some points into super-nodes, and KNN would then cluster these nodes into the final classes. As a result K would be larger.
- Using LMNN improved our classification accuracy slightly for Euclidean distance, Manhattan distance and Cosine distance, but the accuracy decreased for Chebyshev distance. But we have not found a proper explanation to this phenomenon.

B. LDFA

LDFA is based on FDA (which is a kind of Linear Discriminate Analysis method), but introduced

Fig. 4. Classification Results Using LMNN





local information into its computation. Originally, we do not consider the distances between data samples in the same class, but LDFA take the local structure into account, and compute the relative distance between the data samples in the same class. In this way, we are able to discriminate between different clusters in the same class. Below shows our experiment results.

To make the data more observable, we plot corresponding curves, as shown in Fig.5.

Our observations are listed below:

- The best K for the 4 distances vary greatly from each other, which is different from LMNN.
- Manhattan distance performs the best when $K = 1$, and accuracy falls with th increase of K. Also, its average performance is also the lowest. This implies that Manhattan distance and LDFA does not cooperate well with each other.
- Chebyshev distance performed the best, which is quite surprising. Euclidean distance and Cosine distance all performed slightly worser. These conclusions all indicate that LDFA might be especially suitable for Chebyshev.

Classification Accuracy				
K Value	Euclidean	Manhattan	Chebyshev	Cosine
1	84.95	67.64	87.74	88.92
3	84.94	65.61	89.22	89.33
5	85.16	63.87	89.87	89.71
8	85.18	62.24	90.22	89.20
10	84.99	61.40	90.15	89.08
15	84.52	59.98	90.29	88.86
20	84.34	58.59	90.24	88.64
30	83.88	56.32	90.12	88.25
40	83.52	54.97	90.07	87.72
50	83.29	53.78	89.83	87.32
100	81.70	50.03	89.58	85.88

TABLE III
LDFA+KNN

C. LSML

We use the supervised-LSML, which generates some constraints randomly from the labeled training data. Each constraint indicates relation that $d_M(a, b) < d_M(a, c)$, where a, b belong to same class and a, c belong to different classes. LSML learns to satisfy constraints as more as possible by applying square loss to the constraints that haven't been satisfied.

Classification Accuracy				
K Value	Euclidean	Manhattan	Chebyshev	Cosine
1	15.61	16.71	19.82	82.58
3	9.96	10.86	18.60	84.25
5	7.99	9.00	20.00	85.86
8	6.90	7.88	20.74	86.35
10	6.19	7.03	20.62	86.67
15	5.28	5.83	20.28	86.42
20	5.18	5.49	20.30	86.23
30	4.86	5.07	19.63	84.97
40	4.68	4.98	19.01	83.54
50	4.50	4.88	18.40	82.00
100	3.82	4.42	16.58	73.78

TABLE IV
LSML+KNN

- From the graphs, we can observe that Euclidean distance, Manhattan distance and

Fig. 5. Classification Results Using LDFA

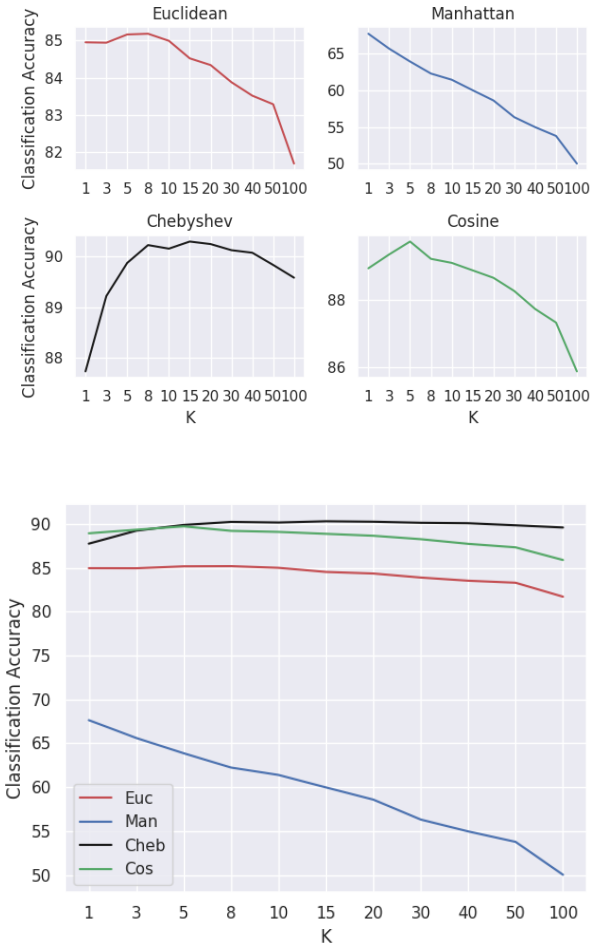
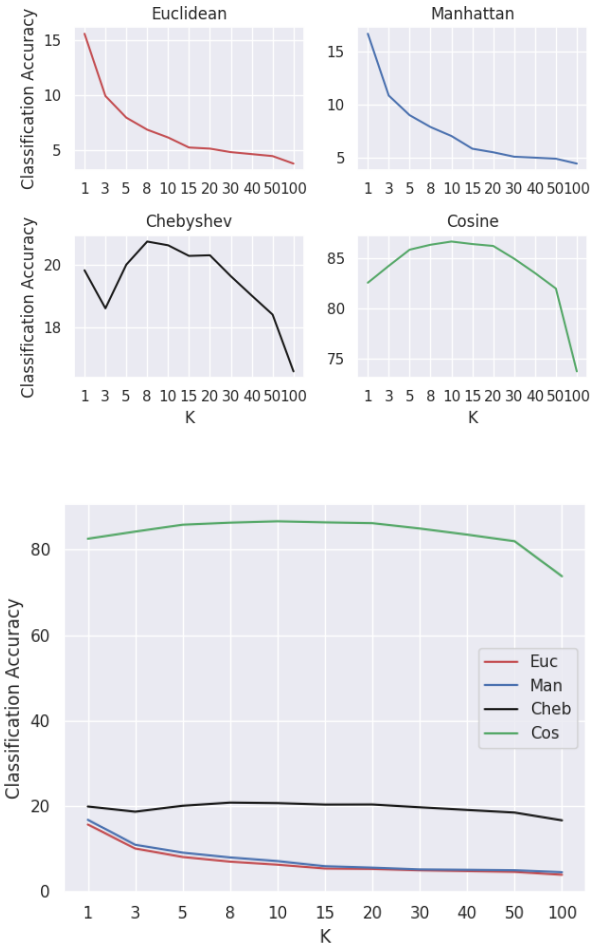


Fig. 6. Classification Results Using LSML



Chebyshev distance all performed very badly. We guess the reason might be the projection matrix we have learned probably have projected the original data samples into a more ambiguous space, and classification becomes more difficult. We also changed the number of constraints, but the performance did not improve.

- Euclidean distance and Manhattan distance's classification accuacry all decreased with K, we guess that this might be due to the fact that: LSML always tries to keep the distance between data samples of the same class smaller than the distance between different classes. In this way, if the data samples are evenly distributed, and the decision boundaries are ambiguous, the taking $K = 1$ is definitely the best choice for classification.
- Also, LSML may not be consistent with our data. By the outstanding performance of cosine distance and the relative bad performance of Manhattan distance and Chebyshev distance, we make the assumption that the data samples are distributed near the surface of a sphere, with the feature vectors arranged in radial pattern. During the process of learning, LSML may have projected the vectors to some other coordinates, such as the polar coordinate, achieving a worse performance on most distance measures.

IV. Further Experiments and Discussions

A. Insight Into Sample Data Dimension

We take the Euclidean distance's classification accuracy as the basis, Manhattan distance's accuracy is about 2% lower while Chebyshev distance'accuracy is about 10% lower. To get insight into why this is happening, we take point A and

point B in the data sample space, where $d_{Euc}(A) < d_{Euc}(B)$ and this is the correct case. To simulate the circumstance where accuracy might decrease, $d_{Man}(A) > d_{Man}(B)$ and $d_{Cos}(A) > d_{Cos}(B)$ are required. To deal with the problem, we derive the following equations, where: $M(x, y)$ denotes the Manhattan distance between point x and point y, $C(x, y)$ denotes the Chebyshev distance between point x and point y, O is the original point, r is the Euclidean distance from A to O, R is the Euclidean space from B to O, $f_i(x_1, x_2, ..., x_n)$ is the combination of different trigonometric functions with respect to i-dimension space(e.g. $\sum_{i=1}^{2} f_i(x) = cos(x) + sin(x)$), and n is the feature space dimension.

$$
\begin{aligned}
M(A,O) &= r\sum_{i=1}^{n-1} f_i(\alpha_1, \alpha_2, ..., \alpha_{n-1}) \in [r, \sqrt{n}r) \\
M(B,O) &= R\sum_{i=1}^{n-1} f_i(\beta_1, \beta_2, ..., \beta_{n-1}) \in [R, \sqrt{n}R) \\
C(A,O) &= rmax(f_i(\alpha_1, \alpha_2, ..., \alpha_{n-1})) \in [\frac{\sqrt{2}}{2}r, r] \\
C(B,O) &= Rmax(f_i(\beta_1, \beta_2, ..., \beta_{n-1}) \in [\frac{\sqrt{2}}{2}R, R]
\end{aligned} \tag{1}
$$

To make $d_{Man}(A) > d_{Man}(B)$ and $d_{Cheb}(A) > d_{Cheb}(B)$, we need

$$
\frac{M(B,O)}{M(A,O)} < 1, \frac{C(B,O)}{C(A,O)} < 1 \tag{2}
$$

to be always true when using Euclidean distance but false when using Manhattan or Chebyshev distance . And this implies that

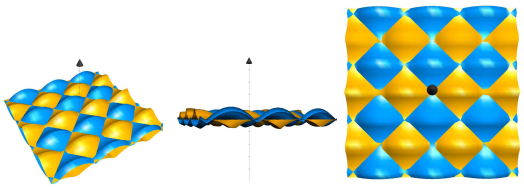
$$
1 < \frac{R}{r} < \sqrt{n} \tag{3}
$$

We calculate the relative probability for inequalities in (2). For simplification, we calculate the following function using expectation:

$$
g(x) = \frac{E(\frac{\sum_i f_i(\alpha_1, \alpha_2, ..., \alpha_{n-1})}{\sum_i f_i(\beta_1, \beta_2, ..., \beta_{n-1})})}{E(\frac{max(f_i(\alpha_1, \alpha_2, ..., \alpha_{n-1}))}{max(f_i(\beta_1, \beta_2, ..., \beta_{n-1}))})} \tag{4}
$$

Intuitively, it is hard to find $g(x)$'s value range, thus we plot $Man(x, y)$ and $Cheb(x, y)$ without regard to r and R, as shown in Fig.7.

Fig. 7. Distribution of Numerator and Denominator



(a) Perspective 1 (b) Perspective 2 (c) Perspective 3

We only plot a 2-dimensional feature in the graphs, with $f_1(\alpha) = sin\alpha$ and $f_2(\alpha) = cos\alpha$. We can see (by naked eyes) that the scale is $1:1$. But by our experiment, we expect to see a scale of $2\% : 10\% = 1 : 5$.

Thus, we can conclude from the above inductions that: When the feature dimension n is large, we expect more data samples to be misclassified due to much larger R than r. However, the $1:1$ scale eliminates this probability, and misclassifications are not related to the dimension of data. Further experiments using t-SNE to reduce the data diemnsion to 2 or 3 might help to prove this result, but limitation of time restricted us to do this.

B. Resolving The Principle of Chebyshev Distance

The Chebyshev distance takes the maximum absolute difference of all dimensions, which in math is written as:

$$
d(x, y)_{cheb} = max_i|x_i - y_i| \tag{5}
$$

From the experiments conducted by KNN, LMNN+KNN, LDFA+KNN, we find that Chebyshev distance always performs the best among other distances. We guess that the poor performance might be due to its ignorance of other dimensions' information. Thus, we think if most of the information is accumulated into one dimension, then the performance for Chebyshev distance might be better.

To verify that our guess is correct, we perform PCA on the data first and then do KNN. Our PCA operation only changes the main distribution direction of the data distribution, but do not reduce the data dimension. We tried this idea on both pure KNN and KNN with LMNN. The experiment results are shown below.

To compare between different methods, we plot the following graphs.

Classification Accuracy				
K Value	Euclidean	Manhattan	Chebyshev	Cosine
1	87.58	78.99	84.53	87.84
3	88.36	77.77	85.92	88.94
5	88.93	76.40	86.99	89.30
8	88.84	74.60	87.40	89.33
10	88.85	73.44	87.34	89.41
15	88.52	70.96	87.19	89.62
20	87.95	68.73	87.05	89.17
30	87.36	64.72	86.60	89.01
40	86.90	61.53	86.04	88.64
50	86.44	58.84	85.62	88.19
100	84.46	48.99	83.91	86.77

TABLE V
PCA+KNN

Classification Accuracy				
K Value	Euclidean	Manhattan	Chebyshev	Cosine
1	88.78	86.52	85.55	88.43
3	89.53	86.80	86.99	89.41
5	90.48	87.41	88.11	89.81
8	90.61	87.24	88.59	90.13
10	90.53	87.14	88.75	90.17
15	90.66	86.66	88.86	90.31
20	90.37	86.29	88.86	90.09
30	89.90	85.46	88.59	89.89
40	89.57	84.43	88.21	89.54
50	89.20	84.54	88.06	89.18
100	87.71	80.43	86.49	88.26

TABLE VI
PCA+LMNN+KNN

Fig. 8. Comparison Between KNN and PCA+KNN

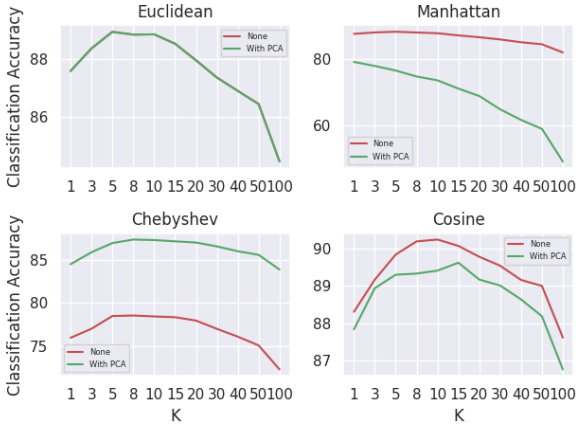
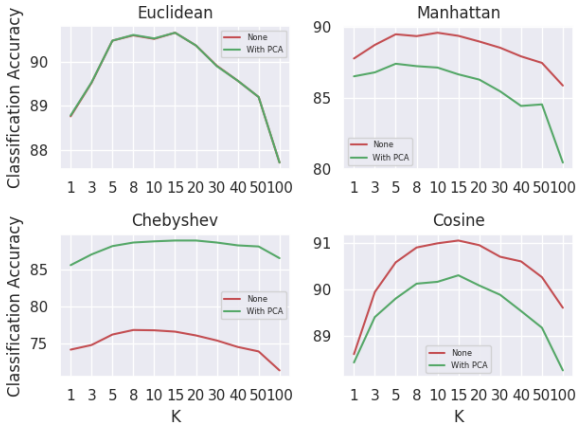


Fig. 9. Comparsion Between KNN and PCA+KNN Under LMNN



From the tables and graphs, we can find that

- The classification performance using Chebyshev distance improved greatly, leading to an increase of accuracy around 10% for every value of K.
- Except for Chebyshev distance, the performance of other three distances either dropped or did not change. This observation fully proved that our guess is valid and correct in some ways.
- Also, using metric learning methods (at least LMNN) does not change the relationship between the different methods.
- Moreover, an interesting thing is that when we use Euclidean distance, the classification accuracy is surprisingly similar between methods that use PCA and methods that does not. This indicates that PCA does not effect the Euclidean distance between the samples much, and the data is rather distributed in a way that PCA prefers.

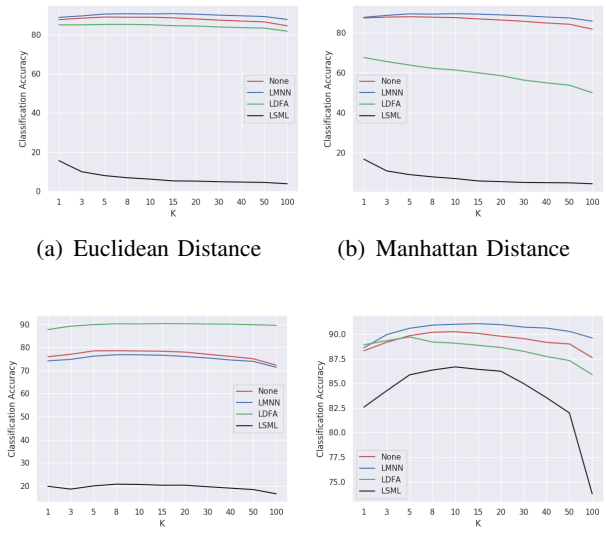
C. Insight Into Data Distribution: Why Cosine Distance Perform So Well

Realizing the distribution of initial data samples is important. We have guessed in the past sections that the data are distributed near the surface of a sphere due to the great performance of Cosine distance. Thus, we plot Fig.10. for comparison.

From the graphs, we observe that: Euclidean distance only perform badly on LSML, Manhattan distance perform badly on LDFA and LSML, Chebyshev distance perform badly on LSML. Though LSML using Cosine distance still has the worst performance, its absolute accuracy is high.

Since Cosine distance is unrelative with the vectors' length, thus the best guess is that the data

Fig. 10. Comparison Between Different Distances



(a) Euclidean Distance (b) Manhattan Distance

(c) Chebyshev Distance (d) Cosine Distance

samples are distributed near the surface of sphere. This assumption could also explain why Chebyshev distance has the relative worst preformance. Also, we can find from the graphs by using metric learning methods, our classification performace can somehow improve if the right method is used.

V. Conclusion

In this project, we explore the performance of KNN when doing classification. Since KNN uses distances to do clustering, we further explore how different distances effect its performance. To achieve better performance, we use metric learning methods to find the a good distance measure. Our best performance is achieved when using LMNN with Cosine distance, and we reach an accuracy of 91.06%. By observing our results of various experiments, we further discuss the function of Chebyshev distance, Cosine distance and the distribution of data samples. Due to our computational resources, we were not able to experiment on metric learning methods such as MMC, but we believe the above experiments and discussions are enough for us to get deep insight into KNN and different distance metrics.

References

[1] Sun S , Chen Q . HIERARCHICAL DISTANCE METRIC LEARNING FOR LARGE MARGIN NEAREST NEIGHBOR CLASSIFICATION[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2011, 25(07):1073-1087.

[2] Liu E Y , Guo Z , Zhang X , et al. MetricLearningFromRelative ComparisonsbyMinimizingSquaredResidual[C]// IEEE International Conference on Data Mining. IEEE Computer Society, 2012.

[3] Sugiyama M . Local Fisher Discriminant Analysis for Dimensionality Reduction. Department of ComputerScience, Tokyo Institute of Technology, 2-12-1-W8-74, O-okayama, Meguro-ku, Tokyo,152-8552,Japan