K-Nearest Neighbors (KNN) Classification with Different Distance Metrics

Sijia Li 518030910294

Abstract—Metric learning is a typical task in machine learning, which is usually combined with many familiar metric-based methods (such as KNN, K-means, etc.) to achieve classification or clustering. In this project, we conduct different distance distance metrics and metric learning method on the K-Nearest Neighbors (KNN) Classification task and compare their performances.

Index Terms—K-Nearest Neighbors Classification, Distance Metric, Metric Learning

1 K-Nearest Neighbors Classification

In K-Nearest Neighbors Classificatio task, the training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point. In this project, we test different distance metrics and metric learning methods on the task of K-Nearest Neighbors Classification. Figure 1 is an example of k-NN classification quoted from the wikipedia website [1].

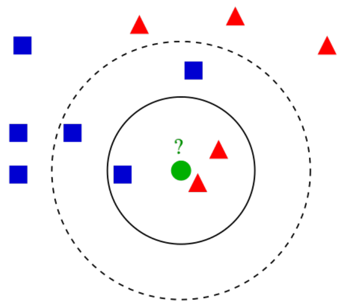

Fig. 1. KNN Classification Example: The test sample (green dot) should be classified either to blue squares or to red triangles. If k = 3 (solid line circle) it is assigned to the red triangles because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the blue squares (3 squares vs. 2 triangles inside the outer circle)

2 Distance Metrics

We often need to define the distance between two samples or two distributions, and this is the distance metrics. In this section, we introduce several commonly used distance metrics.

2.1 Minkowski Distance

The Minkowski distance of order p (where p is an integer) between two points $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n) \in R^n$ is defined as

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

2.2 Euclidean Distance

Euclidean distance between two points in Euclidean space is the length of a line segment between the two points. It can be calculated from the Cartesian coordinates of the points using the Pythagorean theorem. It is a specialization of Minkowski distance where p = 2.

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

2.3 Chebyshev Distance

Chebyshev distance is a metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension. It is a specialization of Minkowski distance where $p = \infty$.

$$D(X,Y) = max_i |x_i - y_i|$$

2.4 Manhattan Distance

Manhattan distance between two points is the sum of the absolute differences of their Cartesian coordinates. It is a specialization of Minkowski distance where $p = 1$.

$$D(X,Y) = \sum_{i} |x_i - y_i|$$

2.5 KNN accuracy with different distance metrics

The datastset we use is Animals_with_Attributes2 (AwA2) dataset. This 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.

We test three situations of minkowski distance: eudlidean, chebyshev, manhattan distance metrics on KNN classification task with diffenrent parameter K in Table 1.

As the result shows, eudlidean and chebyshev distance reach the best performace when $K = 5$ with 0.89396 and

0.79034 respectively, while manhattan distance reaches the highest accuracy when $K = 7$ with 0.88686. The best performance appears then we take eudlidean distance and $K = 5$. Information Throretic Metric Learning

TABLE 1
KNN accuracy with different distance metrics

K	euclidean	chebyshev	manhattan
1	0.88271	0.76958	0.87708
2	0.86643	0.74647	0.86329
3	0.88941	0.77507	0.88278
4	0.88961	0.78197	0.88177
5	***0.89396***	***0.79034***	0.88526
6	0.89269	0.78813	0.88492
7	0.89236	0.79007	***0.88686***
8	0.89095	0.78853	0.88472
9	0.89129	0.79014	0.88425
10	0.88887	0.78847	0.88331

3 Metric Learning

3.1 PCA

On typical method of metric learning is projecting different samples or distributions to common space. Thus, I conduct experiments on different dimension reduction methods such as PCA and LDA which have been introduced in project 1.

For PCA, I set the dimension as 45 and kernel as linear in Table 2. With eudlidean distance, the accuracy reaches the best with 0.886864 when $K = 11$. With chebyshev distance, the accuracy reaches the best with 0.845870 when $K = 10$. With manhattan distance, the accuracy reaches the best with 0.886597 when $K = 9$.

TABLE 2
KNN accuracy with PCA

K	euclidean	chebyshev	manhattan
1	0.867171	0.810436	0.863755
2	0.849756	0.793355	0.849019
3	0.879161	0.826579	0.876080
4	0.881171	0.829727	0.878759
5	0.884587	0.837564	0.885860
6	0.884855	0.840780	0.884520
7	0.886396	0.842521	0.885190
8	0.884654	0.843861	0.885525
9	0.886329	0.843995	***0.886597***
10	0.884520	***0.845870***	0.884989
11	***0.886864***	0.844330	0.883850
12	0.885994	0.843861	0.884118
13	0.885793	0.845469	0.883917
14	0.883314	0.845670	0.881774
15	0.884520	0.843593	0.883180

3.2 LDA

I conduct experiments on LDA and set the dimension as 45. in Table 3

With eudlidean distance, the accuracy reaches the best with 0.922701 when $K = 20$. With chebyshev distance, the accuracy reaches the best with 0.909840 when $K = 12$. With manhattan distance, the accuracy reaches the best with 0.922700 when $K = 20$.

TABLE 3
KNN accuracy with LDA(dimension=45)

K	euclidean	chebyshev	manhattan
1	0.902539	0.889410	0.901266
2	0.896711	0.883582	0.897783
3	0.909907	0.900261	0.910711
4	0.912318	0.903008	0.911916
5	0.915801	0.905352	0.916873
6	0.916002	0.905955	0.917476
7	0.917878	0.907228	0.918548
8	0.917811	0.907161	0.919419
9	0.918883	0.909103	0.920557
10	0.919687	0.908768	0.920356
11	0.919486	0.909639	0.920892
12	0.919954	***0.909840***	0.920289
13	0.920892	0.908768	0.921160
14	0.920155	0.908902	0.920423
15	0.920155	0.909371	0.921629
16	0.921495	0.909371	0.921495
17	0.921830	0.909371	0.921830
18	0.922433	0.909371	0.922433
19	0.922567	0.909371	0.922567
20	***0.922701***	0.909371	***0.922770***
21	0.922165	0.909371	0.922165
22	0.922031	0.909371	0.922031

3.3 LFDA

LFDA(Local Fisher Discriminant Analysis) is a linear supervised dimensionality reduction method. It is particularly useful when dealing with multimodality, where one ore more classes consist of separate clusters in input space. [2] It is a function in the package *metric_learn* as below.

```
class metric_learn.LFDA(n_components=None,
k=None, embedding_type='weighted',
preprocessor=None)
```

I first use default parameter setting with no dimension reduction in Table 4 and find the performance is extremely poor. I guess this is because I neglect the process the dimension reduction and default is not suitable for metric learning.

TABLE 4
KNN accuracy with LFDA (defaul dimesion)

K	euclidean	chebyshev	manhattan
1	0.127001	***0.344631***	0.115212
2	***0.152857***	0.330498	***0.137518***
3	0.088619	0.332708	0.075893
4	0.092772	0.331971	0.080715
5	0.074419	0.329627	0.063835
6	0.075223	0.331435	0.065644
7	0.065778	0.329627	0.056266
8	0.067051	0.326412	0.058142
9	0.060687	0.324871	0.052716
10	0.061759	0.323866	0.054056
11	0.057405	0.320785	0.050640
12	0.058075	0.318709	0.052113
13	0.055663	0.317570	0.049635
14	0.056735	0.316699	0.051176
15	0.054257	0.315493	0.048898

Then I explore the performance of difference dimensions with euclidean distance in Table 5. The result improves and the highest classification accuray reaches when the dimension = 56 and K = 17.

3.4 ITML: Information Theoretic Metric Learning

ITML [3] 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. This algorithm can handle a wide variety of constraints and can optionally

TABLE 5
KNN accuracy with LFDA (euclidean, different dimesions)

K	dim=32	dim=40	dim=48	dim=56	dim=64
1	0.873937	0.889075	0.898587	0.900797	0.895907
2	0.868913	0.887333	0.895103	0.894367	0.890884
3	0.884453	0.902740	0.908366	0.907562	0.904749
4	0.885793	0.900931	0.907964	0.910309	0.906692
5	0.890750	0.903945	0.912452	0.915065	0.911381
6	0.889142	0.903476	0.910242	0.914462	0.911916
7	0.891888	0.905486	0.914529	0.917074	0.914127
8	0.890884	0.905285	0.913591	0.916069	0.914730
9	0.892424	0.907362	0.915467	0.917878	0.914730
10	0.892424	0.907696	0.915667	0.917007	0.914663
11	0.893161	0.908500	0.916337	0.918749	0.915801
12	0.893831	0.908299	0.917007	0.918682	0.915132
13	0.893831	0.909706	0.917878	0.917878	0.916069
14	0.893697	0.910175	0.917677	0.917409	0.915868
15	0.894233	***0.910309***	***0.918213***	0.918079	***0.916873***
16	0.893630	0.909103	0.917476	0.918146	0.915333
17	***0.894501***	0.910309	0.918079	***0.919352***	0.916337
18	0.894434	0.909304	0.917744	0.918414	0.915868
19	0.894434	0.910175	0.917543	0.918950	0.915333
20	0.893898	0.909505	0.917476	0.918615	0.915132

incorporate a prior on the distance function. Unlike some other methods, ITML does not rely on an eigenvalue computation or semi-definite programming. [2] It is a function in the package *metric_learn* as below (supervised form).

```
class metric_learn.ITML_Supervised(gamma=1.0,
max_iter=1000, convergence_threshold=0.001,
num_constraints=None, prior='identity',
verbose=False, preprocessor=None, random_state=None)
```

I conduct experiments with euclidean, chebyshev, manhattan distance and euclidean after LDA dimension reduction in Table 6. With eudlidean distance, the accuracy reaches the best with 0.853239 when $K = 5$. With chebyshev distance, the accuracy reaches the best with 0.619733 when $K = 8$. With manhattan distance, the accuracy reaches the best with 0.859803 when $K = 1$. With eudlidean distance and LDA dimension reduction, the accuracy reaches the best with 0.919620 when $K = 17$.

TABLE 6
KNN accuracy with ITML

K	euclidean	chebyshev	manhattan	euclidean & LDA
1	0.850894	0.603925	***0.859803***	0.899859
2	0.834081	0.579945	0.841115	0.893831
3	0.850894	0.600777	0.859066	0.909170
4	0.849756	0.609351	0.855784	0.910376
5	***0.853239***	0.615982	0.857258	0.914194
6	0.851832	0.619131	0.852033	0.914864
7	0.850894	0.619666	0.854176	0.916069
8	0.847076	***0.619733***	0.852368	0.916136
9	0.846942	0.618997	0.853775	0.917275
10	0.844598	0.618394	0.850291	0.916270
11	0.844062	0.618461	0.849889	0.918012
12	0.840981	0.619131	0.847612	0.917811
13	0.841383	0.617925	0.847143	0.917543
14	0.840512	0.617925	0.845803	0.918883
15	0.839239	0.615045	0.846406	0.918883
16	0.837230	0.614643	0.842990	0.919352
17	0.837497	0.612767	0.841918	***0.919620***
18	0.835622	0.609552	0.840981	0.918615
19	0.834483	0.608815	0.839708	0.918816
20	0.833278	0.607676	0.838167	0.918280

3.5 LSML: Least Squared-residual Metric Learning

LSML [4] proposes a simple, yet effective, algorithm that minimizes a convex objective function corresponding to the sum of squared residuals of constraints. This algorithm uses the constraints in the form of the relative distance comparisons, such method is especially useful where pairwise constraints are not natural to obtain, thus pairwise constraints based algorithms become infeasible to be deployed. Furthermore, its sparsity extension leads to more stable estimation when the dimension is high and only a small amount of constraints is given. [2] It is a function in the package *metric_learn* as below (supervised form).

```
class metric_learn.LSML_Supervised(tol=0.001,
max_iter=1000, prior='identity',
num_constraints=None, weights=None,
verbose=False, preprocessor=None, random_state=None)
```

I conduct experiments with euclidean, chebyshev, manhattan distance and euclidean after LDA dimension reduction in Table 7. With eudlidean distance, the accuracy reaches the best with 0.882912 when $K = 7$. With chebyshev distance, the accuracy reaches the best with 0.694621 when $K = 6$. With manhattan distance, the accuracy reaches the best with 0.877956 when $K = 5$. With eudlidean distance and LDA dimension reduction, the accuracy reaches the best with 0.920222 when $K = 13$.

TABLE 7
KNN accuracy with LSML

K	euclidean	chebyshev	manhattan	euclidean with LDA
1	0.872865	0.672851	0.870520	0.902673
2	0.854980	0.648068	0.854243	0.897917
3	0.877219	0.676469	0.873736	0.909907
4	0.877420	0.681961	0.873669	0.912653
5	0.881707	0.690066	***0.877956***	0.914596
6	0.882377	***0.694621***	0.874740	0.916069
7	***0.882912***	0.694219	0.876415	0.918213
8	0.881908	0.694219	0.874272	0.917945
9	0.881171	0.694554	0.875276	0.918749
10	0.880032	0.692411	0.872463	0.919084
11	0.877755	0.692143	0.872463	0.919352
12	0.876281	0.690669	0.870052	0.919954
13	0.875879	0.690066	0.869516	***0.920222***
14	0.874807	0.689664	0.866769	0.919553
15	0.875209	0.688861	0.867171	0.919954

3.6 MMC: Mahalanobis Metric for Clustering

MMC [5] 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. [2] It is a function in the package *metric_learn* as below.

```
class metric_learn.MMC(max_iter=100, max_proj=10000,
convergence_threshold=0.001, init='identity',
diagonal=False, diagonal_c=1.0, verbose=False,
preprocessor=None, random_state=None)
```

I conduct experiments with euclidean, chebyshev, manhattan distance and euclidean after LDA dimension reduction in Table 8. With eudlidean distance, the accuracy reaches the best with 0.880233 when $K = 5$. With chebyshev distance, the accuracy reaches the best with 0.700784 when $K = 11$. With manhattan distance, the accuracy reaches the best with 0.873468 when $K = 7$. With eudlidean distance and LDA dimension reduction, the accuracy reaches the best with 0.917275 when $K = 13$.

TABLE 8
KNN accuracy with MMC

K	euclidean	chebyshev	manhattan	euclidean with LDA
1	0.872530	0.674258	0.867573	0.897716
2	0.852904	0.654163	0.848885	0.893898
3	0.874606	0.676335	0.870520	0.908701
4	0.873803	0.686248	0.869516	0.909639
5	***0.880233***	0.693148	0.873066	0.911849
6	0.878425	0.695157	0.870989	0.912988
7	0.878090	0.696564	***0.873468***	0.914462
8	0.878492	0.698439	0.869315	0.914395
9	0.878625	0.699243	0.870989	0.915801
10	0.876750	0.699712	0.867372	0.916337
11	0.877286	***0.700784***	0.867372	0.916136
12	0.876750	0.698439	0.866568	0.917141
13	0.875477	0.696564	0.865631	***0.917275***
14	0.873803	0.695626	0.864425	0.917275
15	0.873267	0.695425	0.864425	0.916940

4 Conclusion and Summary

In this project, we review the definition of three typical distance metrics: euclidean distance, chebyshev distance and manhattan on KNN classification task. In our dataset, euclidean distance always achieves the best performance while chebyshev distance's performance is poor. In different problem settings and datasets, we should choose the proper distance metric method. Besides, we conduct experiments with different metric learning methods: PCA, LDA, LFDA, ITML, LSML, MMC. We also find that in most cases, dimension reduction can not only lower the memory cost, but also improve the classification accuracy.

References

[1] "K-nearest neighbors algorithm," [EB/OL], https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

[2] "Scikit-learn document," [EB/OL], http://contrib.scikit-learn.org/metric-learn/.

[3] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, ser. ACM International Conference Proceeding Series, Z. Ghahramani, Ed., vol. 227. ACM, 2007, pp. 209–216. [Online]. Available: https://doi.org/10.1145/1273496.1273523

[4] E. Y. Liu, Z. Guo, X. Zhang, V. Jojic, and W. Wang, "Metric learning from relative comparisons by minimizing squared residual," in *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, M. J. Zaki, A. Siebes, J. X. Yu, B. Goethals, G. I. Webb, and X. Wu, Eds. IEEE Computer Society, 2012, pp. 978–983. [Online]. Available: https://doi.org/10.1109/ICDM.2012.38

[5] S. Xiang, F. Nie, and C. Zhang, "Learning a mahalanobis distance metric for data clustering and classification," *Pattern Recognit.*, vol. 41, no. 12, pp. 3600–3612, 2008. [Online]. Available: https://doi.org/10.1016/j.patcog.2008.05.018