Pattern Recognition Coursework 2

Spyridon Couvaras (CID: 01119198, login: sc8415), Renke Wang (CID: 01762400, login: rw319)
Department of Electrical and Electronical Engineering, Imperial College London

I. Distance Metrics

A. Data Setup

The setup includes 520 face images, each with dimension R^{2576} corresponding to 52 classes (identities). The data are separated into testing and training data. The first 320 images (identities 1-32) are used as the training set and the rest 200 images (identities 33-52) are used as a testing set. The feature vectors in the non-normalized case are the original non-normalized data. While in normalised case, the feature vectors to data normalized to its norm. Having this setup, k-nearest neighbour (kNN) is performed as a classification algorithm for retrieving images from similar classes.

B. Baseline: Standard Non-Learned Distance Matrices Comparison

For non-learned matrices, we test for Euclidean Distance (L_2), Cosine Distance(L_2) Manhattan Distance (L_1), Chebyshev Distance (L_∞) and χ-Square Distance, a refined version of Euclidean Distance. We performed kNN on both normalized test data and non-normalized test data. @rank1 accuray, @10 accuracy and mAP are provided in Table I. We compared the @rankK CMC curves for normalized features and non-normalized features in Appendix A.

TABLE I
Retrieval Accuracy for Non-Normalized Data
(NN=Non-Normalised, N=Normalised)

	Distance Matrice	@rank1	@rank10	mAP
NN	Euclidean Distance	0.6350	0.9450	0.2268
NN	Manhattan	0.6850	0.9350	0.2412
NN	Chebyshev	0.1555	0.6350	0.0669
NN	χ-Square	0.7050	0.9500	0.2533
NN	Cosine	0.6800	0.9450	0.2294
N	Euclidean Distance	0.6800	0.9450	0.2294
N	Manhattan	0.7550	0.9550	0.2626
N	Chebyshev	0.1500	0.5650	0.0618
N	χ-Square	0.7400	0.9500	0.2560
N	Cosine	0.6800	0.9450	0.2294

We see that in general kNN performs better on normalized data, this is due to after normalization, the similarity of different images characterizes more about their structural contents.

Comparing mAP of different matrices, we see that Chebyshev Distance seems to have a much worse result than the others, this is due to the fact that it takes into account only the dimension with largest difference and hence not a good measure when we have data in high dimension. And surprisingly, Manhattan Distance performs the best for normalized feature vectors, followed by χ-Square and Euclidean Distance. χ-Square outperforms Euclidean Distance because it weights the dimensions inversely to the total of the values of the dimensions.

In non-normalized data, χ-Squared Distance is the best in terms of mAP. Cosine Distance performs better than Euclidean Distance, because it is an angle based measure of dissimilarity. And it performs exactly as Euclidean Distance in case of normalized data. In general, in the context of high dimensionality, it may be preferable to use lower k when we choose different distance matrices (L_k). [1]

C. Experiment 1: Standard Non-Learned Distance on Histogram

In the baseline section, normalized feature vectors seemed to have an overall better performance compared to the non-normalized feature vectors. For this reason, in this experiment, histograms of pixel intensities of the normalized feature vector images are obtained and used as feature representations of the original image. Each feature vector was separated into four segments, the first segment has the first 644 features, the second segment 645 features, the third segment 645 features and the fourth segment 642 features. Then the histogram representation was calculated for each of the segments and the results from all segments were concatenated together to form a new feature vector. This outperforms non-concatentaed histogram, and we report comparison on concatenated histogram and non-concatenated histogram using different distance matrices in the Appendix B.

TABLE II
Scores for Different Distance Metric for Normalized Histogram Data for n_{Bins}=120

Distance Metric	@rank1	@rank10	mAP
Euclidean	0.3500	0.7650	0.1156
Manhattan	0.4250	0.8400	0.1393
Chebyshev	0.1800	0.6400	0.07425
χ-Squared	0.0550	0.235	0.031
Earth Movers	0.2950	0.715	0.0978

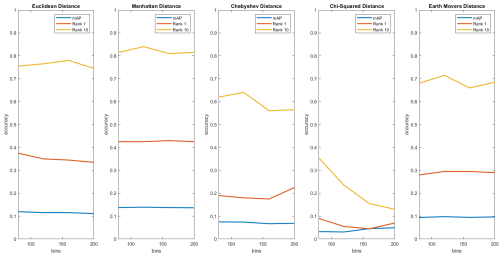

Fig. 1. Performance for varying number of bins

Fig. 1 shows the mAP, @rank1 and @rank 10 performance using different distance metrics for a varying number of bins. As the graph shows the best performing metric is Manhattan distance while the worst performing metric was χ-squared. This is because χ-squared distance is appropriate to use when have categorical data for two independent variables. And this is not the case in our experiment for histogram. Compared to Table I, histogram representation is not efficient to substitute the image pixel intensities as the performance decreases drastically. In Appendix B, we show different distance matrices performance (@rankK accuracy) with varied bin numbers.

D. Experiment 2: Dimensionality Reduction with PCA

In this experiment, we first reduce the dimension of the data by performing PCA on training data to find the subspace on which the data is projected. Based on dimension reduced data, we then further performed Euclidean Distance and Mahalanobis Distance on normalized features.

The results for Mahalanobis Distance are shown in Table III and the results for Euclidean Distance are shown in Table IV. The results for non-normalized features can be found in Appendix C.

TABLE III
SCORES FOR DIFFERENT DIMENSION CHOSEN FOR NORMALIZED FEATURE VECTORS WITH PCA-MAHALANOBIS

Dimension	@rank1	@rank10	mAP
Original Covariance (2576)	0.5900	0.8750	0.1938
16	0.5350	0.8850	0.1877
32	0.5900	0.9150	0.2098
64	0.6600	0.9350	0.2399
128	0.6850	0.9400	0.2467
256	0.6250	0.9150	0.2269

TABLE IV
SCORES FOR DIFFERENT NUMBER OF PRINCIPAL COMPONENTS CHOSEN FOR NORMALIZED FEATURE VECTORS WITH PCA-EUCLIDEAN

Number of Principal Components	@rank1	@rank10	mAP
16	0.5409	0.8804	0.1935
32	0.5850	0.9270	0.2097
64	0.6150	0.9450	0.2209
128	0.6451	0.9500	0.2270
256	0.6750	0.9500	0.2332

From Appendix A, we see that when number of principal components increases, the accuracy also increases due to more discriminant features are kept. Comparing the two distance metrics, Mahalanobis distance generally has a better performance as different weight is assigned to each feature, while Euclidean assigns the same weight for all features [2].

E. Experiment 3: Distance Metrices With PCA and PCA-LDA

For this experiment, we use normalized features since they tend to give a better performance as can be seen in Appendix A. For performing PCA only we set $M_{PCA} = 64$, since when $M_{PCA} = 64$, we already see fair performance in terms of both accuracy and precision and also for computation convenience.

We perform different non-learned standard matrices on normalized feature vectors using PCA representations. The results are shown in Table V. Here χ-Square Distance is not considered since for appropraite using χ-Square test, the features should be positive.

TABLE V
SCORES FOR DIFFERENT DISTANCE METRICES WITH PCA REPRESENTATIONS

Distance Metric	@rank1	@rank10	mAP
PCA-Euclidean	0.6150	0.9450	0.2209
PCA-Manhattan	0.6600	0.9350	0.2405
PCA-Chebyshev	0.3800	0.8400	0.1421
PCA-Mahalanobis	0.6600	0.9350	0.2399

We see that using PCA representations, Manhattan Distance performs the best followed by Mahalanobis Distance and then Euclidean Distance.

Then we perform PCA-LDA on normalized feature vectors. Setting $M_{PCA} = 64$ for comparison with performing PCA only and $M_{LDA} = 25$. We report @rank1, @rank10 and mPA for PCA-LDA-Euclidean with varied number of M_{LDA} in Appendix D, we see when $M_{PCA} = 25$, mPA is maximum for normalized features. we performed different standard non-learned distane matrices and the results are shown in Table VI.

Compared to Table V using PCA-LDA out-performs using PCA only. This is due to by LDA, we try to maximize distances between different classes and minimize the within-class distance.

TABLE VI
SCORES FOR DIFFERENT DISTANCE METRICES WITH PCA-LDA REPRESENTATIONS ON NORMALIZED DATA

Distance Metric	@rank1	@rank10	mAP
PCA-LDA-Euclidean	0.8150	0.9850	0.3664
PCA-LDA-Manhattan	0.8200	0.9800	0.3614
PCA-LDA-Chebyshev	0.6650	0.9650	0.3005
PCA-LDA-Mahalanobis	0.7900	0.9700	0.3213

Now, dimensionality reduction is performed for the histogram data obtained in Experiment 1. For this part the number of bins used for the histogram data is 120. In Appendix B we see that when $n_{bins} = 120$, the mPA is the highest for most of the distance matrices. Firstly, PCA is performed on histogram data considering M_{PCA} values $16, 32, 64$ and 100. The results were obtained for five different distance metrics (Euclidean, Manhattan, Chebyshev and Earths Mover). The results are shown in Appendix E. We see that in general Manhattan Distance performs the best followed by Euclidean Distances, Chebyshev Distances and Earths Mover. In Table VII we report @rank1, @rank10 and mAP for concatenated histogram using different distance metrices with $M_{PCA} = 16$.

TABLE VII
CONCATENATED HISTOGRAM DATA PERFORMANCE FOR DIFFERENT NUMBER OF PCA COMPONENTS CHOSEN

Distance Metric	PCA Components	@rank1	@rank10	mAP
Euclidean	16	0.3750	0.7550	0.1248
Manhattan	16	0.3700	0.5400	0.1250
Chebyshev	16	0.3100	0.7200	0.1024
Earths Mover	16	0.1550	0.4850	0.0545

Next, both PCA and LDA is performed on the histogram data using the same number of bins. To examine the performance, $M_{PCA} = 60$ was chosen so there is room to examine a wide range of M_{LDA} values. The M_{LDA} values considered were: $5, 10, 15, 20, 25$ and 30. The results are shown in Appendix.D. Generally, the performance is better for bigger values of M_{LDA}, except the Earths Mover distance which performs better for smaller M_{LDA} values.

TABLE VIII
HISTOGRAM DATA PERFORMANCE FOR DIFFERENT NUMBER OF LDA COMPONENTS CHOSEN WHILE $M_{PCA} = 60$

Distance Metric	LDA Components	@rank1	@rank10	mAP
Euclidean	30	0.2500	0.7150	0.0918
Manhattan	30	0.2300	0.6550	0.0840
Chebyshev	30	0.2500	0.6650	0.0879
Chi-Squared	30	0.0500	0.3700	0.0249
Earths Mover	30	0.0900	0.4800	0.0433

F. Experiment 4

In this section we perform Mahalanobis metric learning considering two different distance metric techniques. The first technique is a learned metric called Relevant Component Analysis (RCA) and the second technique is also a learned metric called Large Margin Nearest Neighbors (LMNN).

The idea of RCA is to maintain the relation between points of the same class while separating neighboring points of different classes. RCA algorithm has as input the training data set $x \in R^{2576\times320}$ and the labels of each training data which are referred as "chunklets" C_j. For the training data, chunklet values range from $32-52$. As described in [3] the steps taken from the RCA algorithm are:

1) For each chunklet, we subtract the mean of the chunklet from all the vectors it contains
2) Covariance matrix of the centered data-points in chunklets is computed. Assuming, total k chunklets, total p vectors and that the index j corresponds to the j^{th} chunklet, we got:

$$C = \frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ji} - m_j)(x_{ji} - m_j)^T$$

3) Compute the inverse of the chunklet covariance matrix: $distance(x, y) = (x - y)^T C^{-1}(x - y)$

Large Margin Nearest Neighbor (LMNN) learns a psuedo-metric designed for k-nearest neighbor classification. The algorithm specifies k target neighbors, means k other neighbors that have minimal distance to the specific data point that we are looking at. The cost function for LMNN for the learned linear transformation L is composed by two terms, the first term penalizes large distances between each input and its target neighbors, the second term penalizes small distances between each input and other input that doesn't share the same label. Then the test example is classified by the hypothetical label that minimizes the combination of this two terms [7]. In Table IX, we report the scores for RCA and LMNN with $k = 9$. Since due to lack of regularization, LMNN is prone to overfit the training data, thus we also simulated LMNN result for other values of k and the results are shown in Appendix F. We also compared LMNN performed on normalized features and non-normalized features and report the result in Appendix F, it is clear that LMNN has a better performance on non-normalized features.

TABLE IX
PERFORMANCE OF NON-LEARNED AND LEARNED MAHALANOBIS DISTANCE METRICS

Algorithm	@rank1	@rank10	mAP
Original Mahalanobis	0.5900	0.8750	0.1938
PCA-Mahalanobis	0.6600	0.9350	0.2399
PCA-LDA-Mahalanobis	0.7900	0.9700	0.3035
RCA	0.7300	0.9750	0.2606
LMNN	0.8150	0.9800	0.3312

The results show that LMNN perform better than all non-learned matrices. Comparing RCA and LMNN, LMNN seem to have a better accuracy and mAP, since RCA cannot make use of the discriminative information provided by negative pairs.

II. CLUSTER BASED REPRESENTATIONS

A. Clustering

In this section clustering was performed before the kNN classification. Two different types of clustering methods are examined, k-means clustering (or Top-down) and agglomerative clustering (or Bottom-up).

The algorithms for impementing k-means and agglomerative clustering are given in Appendix G.

B. Experiment 5

In this experiment, both agglomerative and k-means clustering were performed on the normalised test data. For assigining an overall label to a cluster, majority voting method was used. This means that a cluster is assigned the label that the majority belongs to. The result is shown in Fig. 2, where K-means clustering method is superior to the agglomerative clustering method, for all number of clusters considered.

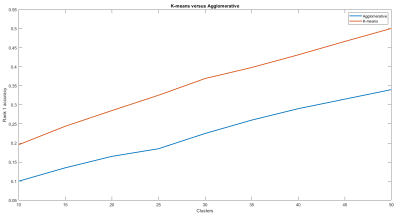

Fig. 2. Comparison between Agglomerative and K-means clustering for a varying number of clusters

C. Experiment 6

1) Vectors of distances to cluster centers as feature vectors: In this section, K-means clustering was implemented on the Training data for different number of clusters. After implementing the K-means clustering on the training data, the distance of each testing feature vector to the K cluster centers was computed. The new testing feature vector matrix has dimension $R^{k\times 200}$, where k is the number of clusters. The results are shown in Table X.

TABLE X
PERFORMANCE USING VECTORS OF DISTANCES TO CLUSTER CENTERS AS FEATURE VECTORS FOR DIFFERENT DISTANCE METRICS

Distance Metric	Clusters	@rank1	@rank10	mAP
Euclidean	20	0.3550	0.7360	0.1390
Manhattan	20	0.3500	0.7250	0.1300
χ-Square	20	0.3440	0.7490	0.1358

Fig. 14 in Appendix H shows that there is an increase in the performance with the increase of cluster numbers. From the three distance metrics, Manhattan shows to have a better overall performance, but not as good as the performance of the original feature vectors.

2) Vectors of distances to cluster centers as feature vectors: Having the previous setup where the feature vector was the distance of the original feature vector to each cluster center k, the softmax probability of inverse distances is used as the new representation. The equation of softmax probability is given by [6], so in our case the formula used is:

$$\sigma(i) = \frac{e^{1/distance_i}}{\sum_{j=1}^{K} e^{1/distance_j}} \tag{1}$$

TABLE XI
PERFORMANCE USING SOFTMAX PROBABILITY OF INVERSE DISTANCES REPRESENTATION FOR DIFFERENT DISTANCE METRICS

Distance Metric	Clusters	@rank1	@rank10	mAP
Euclidean	20	0.3690	0.7520	0.1404
Manhattan	20	0.3530	0.7400	0.1391
χ-Square	20	0.3370	0.7420	0.1322

The results show that the new representation has very similar performance to the previous one but there is slight increase in the performance of Euclidean and Manhattan Distance when 15 or 20 clusters are used (See Table XVIII and XIX in Appendix H)

3) Fisher Vectors: Fisher vectors are used for encoding images by computing the first and second order differences between feature vectors and Gaussians. GMM is a probability desnsity function represented as a linear combination of multiple Gaussian distribution and it has the form:

$$p(x|\lambda) = \sum_{i=1}^{N} w_k p_k(x|\lambda) \tag{2}$$

Where N is the number of Gaussians, $\lambda = (\mu_k, w_k, \Sigma_k)$ are the GMM parameters, corresponding to the mean, the prior probability ($w_k = \frac{elements\ belong\ to\ cluster\ k}{total\ training\ elements}$) and the diagonal covariance matrix of each Gaussian [5]. In equation (2), $p_k(x|\lambda)$ is the distribution of Gaussian k, $\mathcal{N}(x|\mu_k, \Sigma_k)$. Afterwards, we get the occupancy probability, which corresponds to the probabilty that testing image x_i is generated by the k^{th} Gaussian [4] :

$$\gamma_k(x_i) = \frac{w_i p_i(x_t|\lambda)}{\sum_{j=1}^{N} w_j p_j(x_t|\lambda)} \tag{3}$$

Having the following we can calculate v_k and u_k for every testing image x_i:

$$v_k = \frac{1}{M\sqrt{w_k}} \sum_{i=1}^{M} \gamma_k(x_i) \frac{x_i - \mu_k}{\sigma_i} \tag{4}$$

$$u_k = \frac{1}{M\sqrt{2w_k}} \sum_{i=1}^{M} \gamma_k(x_i) (\frac{x_i - \mu_k}{\sigma_i} - 1)^2 \tag{5}$$

So every testing image can be represented by a Fisher Vector V_{fisher} with dimesion $R^{2K\times D}$ where K corresponds to the number of Gaussians-Clusters considered.

$$V_{fisher} = [v_1^T, u_1^T, v_2^T, u_2^T, ..., v_K^T, u_K^T] \tag{6}$$

PCA was applied on the data with $M_{PCA} = 30$ for two reasons, the first reason was to reduce the dimension of Fisher Vectors from $R^{2K\times 2576}$ to $R^{2K\times 30}$ and the second reason was to prevend having 0 determinant of covariance matrix Σ_k for any of the K clusters when calculating $\mathcal{N}(x|\mu_k, \Sigma_k)$. To improve the performance, u_k and v_k are L_2 normalised for improving the performance [8].

TABLE XII
PERFORMANCE USING FISHER VECTOR REPRESENTATION FOR DIFFERENT DISTANCE METRICS

Distance Metric	Clusters	@rank1	@rank10	mAP
Euclidean	20	0.5910	0.9340	0.2367
Manhattan	20	0.6050	0.9260	0.2376
χ-Square	20	0.5950	0.9330	0.2338

Table XII shows that Fisher Vectors are overall a better representation of the data compared to the previous two representations examined as they extract more dense features from the images. Also, there is an increase in the performance of the Euclidean Distance is compared with the performance of the original feature vectors (see Table I).

REFERENCES

1 Charu C. Aggarwal, Alexander Hinneburg and Daniel A. Keim, *On the Suprising Behavior of Distance Matrices in High Dimensional Space*

2 Matthias Wolfel and Hazim Kemal Ekenel *FEATURE WEIGHTED MAHALANOBIS DISTANCE: IMPROVED ROBUSTNESS FOR GAUSSIAN CLASSIFIERS*

3 Aharon Bar-Hiller, Tomer Hertz, Noam Shental, Daphna Weinshall, *Learning a Mahalanobis Metric with Side Information*

4 Florent Perronin, Christopher Dance, *Fisher Kernels on Visual Vocabularies for Image Categorization*

5 R. Gopinath, C.Santhosh Kumar, K.I. Ramachandran, *Fisher vector encoding for improving the performance of fault diagnosis in a synchronous generator*

6 Michalis K. Titsias, *One-vs-Each Approximation to Softmax for Scalable Estimation of Probabilities*

7 Kilian Q. Weinberger, John Blitzer and Lawrence L K. Saul, *Distance Metric Learning for Large Margin Nearest Neighbor Classification*

8 Florent Perronnin, Jorge Sánchez, Thomas Mensink *Improving the Fisher Kernel for Large-Scale Image Classification*

III. APPENDIX

A. @rankK Accuracy Curve for Normalized and Non-Normalized Features

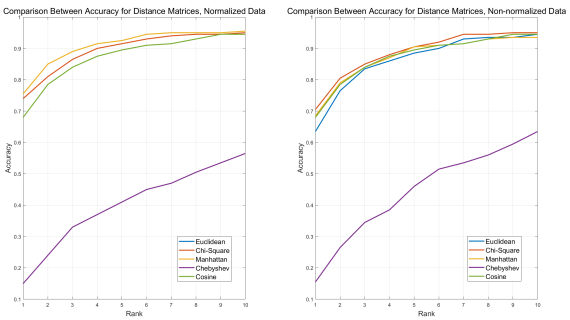

Fig. 3. @rankK Accuracy Comparison between Distance Matrices on Normalized and Non-Normalized Data

B. Experiment with Histogram

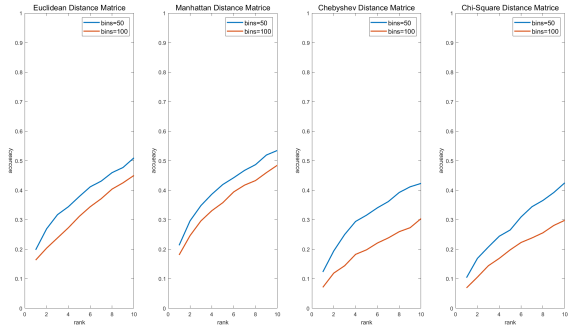

Fig. 4. @rankK Comparison between Distance Matrices on Normalized Non-Concatenated Histogram with Varied Bins

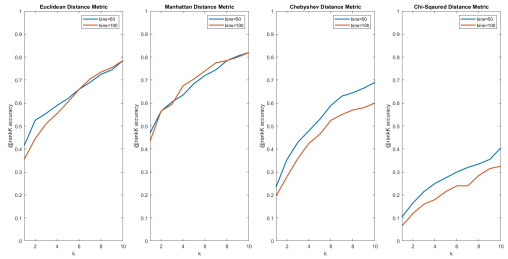

Fig. 5. Comparison between Distance Matrices on Normalized Concatenated Histogram with Varied Bins

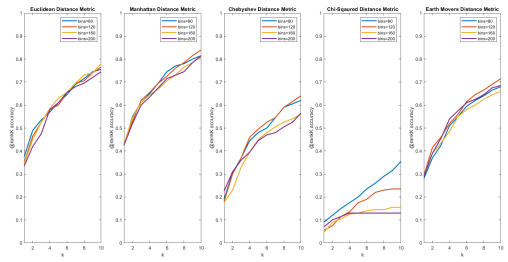

Fig. 6. @rankK Accuracy Performance of Different Distance Metrics with Varied Number of Bins of Concatenated Histogram

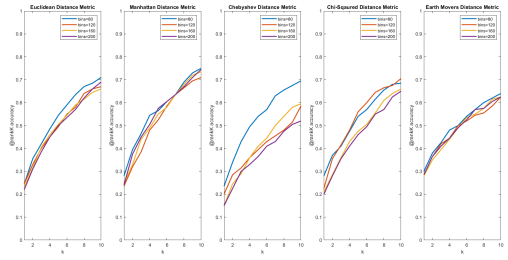

Fig. 7. @rankK Accuracy Performance of Different Distance Metrics with varied Number of Bins of Non-Concatenated Histogram

C. Dimension Reduction with PCA

TABLE XIII
SCORES FOR DIFFERENT NUMBER OF PRINCIPAL COMPONENTS CHOSEN FOR NON-NORMALIZED FEATURE VECTORS FOR PCA-EUCLIDEAN

Number of Principal Components	@rank1	@rank10	mAP
16	0.5100	0.9100	0.1802
32	0.5550	0.9350	0.1994
64	0.6100	0.9400	0.2176
128	0.6050	0.9400	0.2221
256	0.6300	0.9400	0.2272

TABLE XIV
SCORES FOR DIFFERENT DIMENSION CHOSEN FOR NON-NORMALIZED FEATURE VECTORS WITH PCA-MAHALANOBIS

Dimension	@rank1	@rank10	mAP
Original Covariance (2576)	0.635	0.895	0.2027
16	0.5450	0.9050	0.1954
32	0.6250	0.9200	0.2142
64	0.6800	0.9650	0.2457
128	0.7100	0.9300	0.2491
256	0.6850	0.9300	0.2374

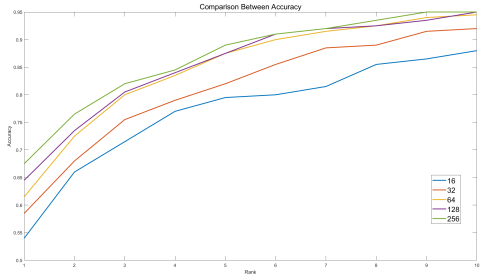

Fig. 8. @rankK Accuracy Comparison Using Different Number of Principal Components Using PCA-Euclidean

D. @rankK Accuracy and mAP for PCA-LDA-Euclidean with Varied M_{LDA}

TABLE XV
SCORES FOR DIFFERENT NUMBER OF LDA COMPONENTS CHOSEN FOR NORMALIZED FEATURE VECTORS USING PCA-LDA-EUCLIDEAN

Number of LDA Components	@rank1	@rank10	mAP
5	0.5050	0.9400	0.2604
10	0.7200	0.9600	0.3500
15	0.7650	0.9850	0.3728
20	0.7900	0.990	0.3811
25	0.8050	0.9950	0.3830
30	0.8300	0.9850	0.3808

TABLE XVI
SCORES FOR DIFFERENT NUMBER OF LDA COMPONENTS CHOSEN FOR NON-NORMALIZED FEATURE VECTORS USING PCA-LDA-EUCLIDEAN

Number of LDA Components	@rank1	@rank10	mAP
5	0.5100	0.9400	0.2559
10	0.6200	0.9700	0.3189
15	0.6750	0.980	0.3412
20	0.6950	0.9850	0.3438
25	0.7650	0.9850	0.3407
30	0.7950	0.9950	0.3438

TABLE XVII
SCORES FOR DIFFERENT DISTANCE METRICES WITH PCA-LDA REPRESENTATIONS ON NON-NORMALIZED DATA

Distance Metric	@rank1	@rank10	mAP
Cosine	0.8000	0.9700	0.3578
Manhattan	0.7550	0.9900	0.3225
Chebyshev	0.6050	0.9700	0.2557
χ-Squared	0.0300	0.5500	0.0729

E. PCA/PCA-LDA Representations mPA Performance Evaluation for Different Distance Metrices with Varied M_{PCA} and M_{LDA}

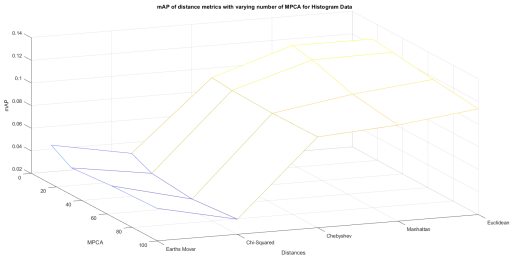

Fig. 9. mAP Performance Comparison between Distance Metrics for Different values of M_{PCA}

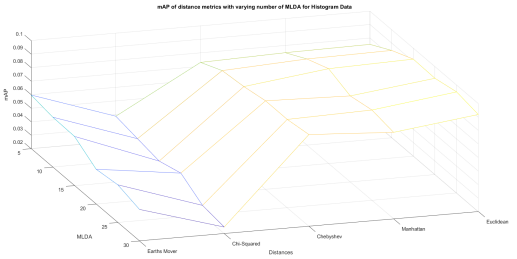

Fig. 10. mAP performance comparison between distance metrics for different values of M_{LDA}

F. Learning Matrices

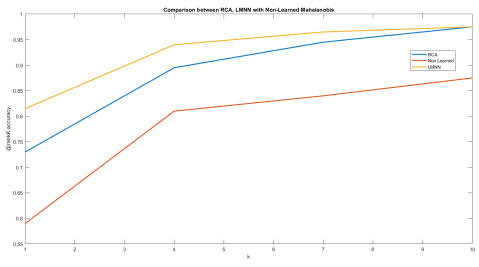

Fig. 11. @rankK Accuracy Comparison between RCA, LMNN and Original Non Learned Mahalanobis Distance

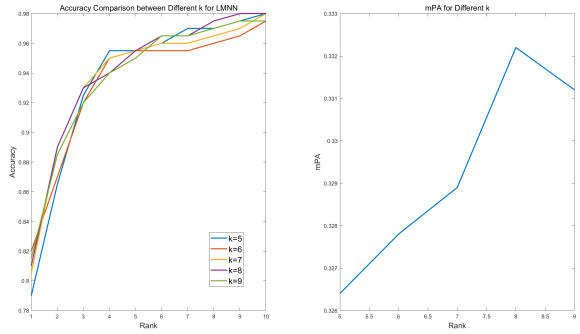

Fig. 12. @rankK Accuracy Comparison and mAP between Different k for LMNN

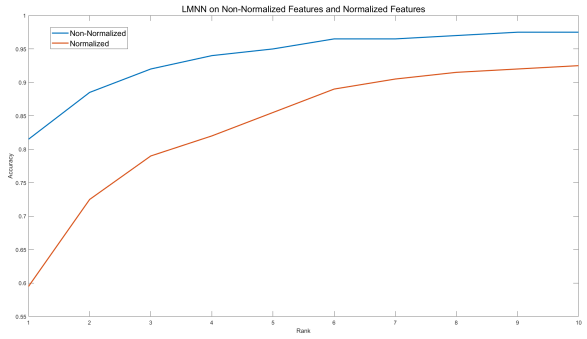

Fig. 13. @rankK Accuracy of LMNN on Normalized and Non-Nomarlized Features

G. Clustering algorithms

Algorithm 1: K-means Clustering

Input: Data, Number of Clusters(K)

Steps:

1. Chose K vectors from Data as cluster centers
2. For every vector in Data, find distance to K cluster centers
3. The cluster center that is closer to the vector corresponds to the cluster hat the vector is going to be assigned to
4. Calculate new cluster center
5. Repeat steps 2,3,4.

Algorithm 2: Agglomerative Clustering

Input: Data, Threshold Distance(T)

Steps:

1. Compute distance of every point with the other points
2. Get the smaller distance between two pairs 3. If distance is smaller than the threshold then merge the pair into one cluster
4. Replace those the pair with its average
5. Repeat steps 1,2,3,4

H. Cluster Based Representations

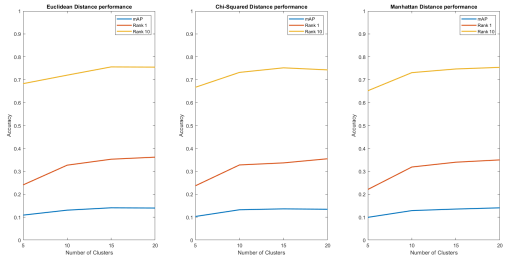

Fig. 14. Changing performance of vector distances to cluster centers representations for different number of clusters

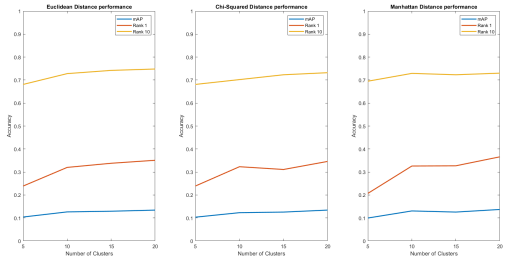

Fig. 15. Changing performances of softmax probability representations for different number of clusters

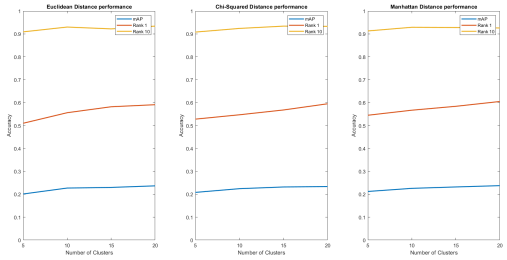

Fig. 16. Changing performance of Fisher Vector representations for different number of clusters

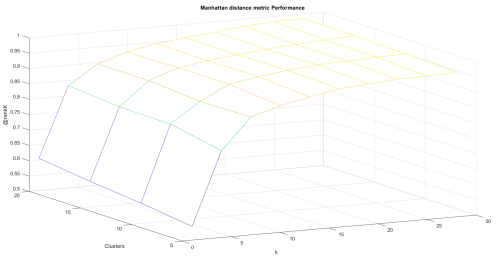

Fig. 17. Fisher Vector Performance using Manhattan distance metric

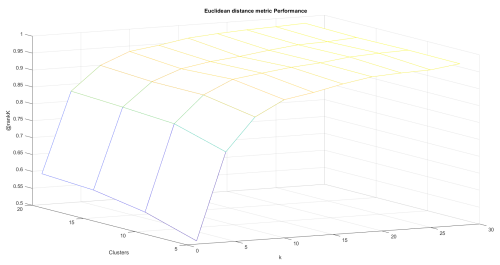

Fig. 18. Fisher Vector Performance using Euclidean distance metric

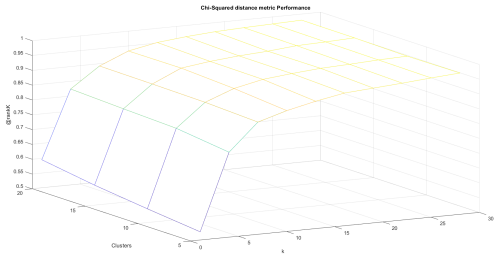

Fig. 19. Fisher Vector Performance using Chi-Squared distance metric

TABLE XVIII
PERFORMANCE USING VECTORS OF DISTANCES TO CLUSTER CENTERS AS FEATURE VECTORS FOR DIFFERENT DISTANCE METRICS

Distance Metric	Clusters	@rank1	@rank10	mAP
Euclidean	5	0.25	0.68	0.11
Manhattan	5	0.241	0.708	0.1111
Chi-Square	5	0.233	0.686	0.1037
Euclidean	10	0.32	0.735	0.1265
Manhattan	10	0.322	0.712	0.1235
Chi-Square	10	0.325	0.744	0.13
Euclidean	15	0.325	0.734	0.1276
Manhattan	15	0.342	0.75	0.1412
Chi-Square	15	0.35	0.761	0.1407

TABLE XIX
PERFORMANCE USING SOFTMAX PROBABILITY OF INVERSE DISTANCES REPRESENTATION FOR DIFFERENT DISTANCE METRICS

Distance Metric	Clusters	@rank1	@rank10	mAP
Euclidean	5	0.236	0.687	0.1073
Manhattan	5	0.218	0.663	0.1002
Chi-Square	5	0.25	0.692	0.1083
Euclidean	10	0.306	0.735	0.1238
Manhattan	10	0.31	0.734	0.1221
Chi-Square	10	0.301	0.741	0.1274
Euclidean	15	0.337	0.735	0.1315
Manhattan	15	0.335	0.716	0.1286
Chi-Square	15	0.347	0.743	0.1367

TABLE XX
PERFORMANCE USING FISHER VECTOR REPRESENTATION FOR DIFFERENT DISTANCE METRICS

Distance Metric	Clusters	@rank1	@rank10	mAP
Euclidean	5	0.51	0.909	0.2012
Manhattan	5	0.545	0.913	0.2122
Chi-Square	5	0.528	0.908	0.2082
Euclidean	10	0.556	0.93	0.2271
Manhattan	10	0.567	0.929	0.2258
Chi-Square	10	0.547	0.924	0.2244
Euclidean	15	0.582	0.922	0.2299
Manhattan	15	0.584	0.928	0.2319
Chi-Square	15	0.568	0.934	0.2318