Pattern Recognition 2018 - Coursework 2

Clive Toh Soon, Wong
Imperial College London
CID: 01044264 cw3915@ic.ac.uk

Zheng Yang, Lee
Imperial College London
CID: 01042500 zyl115@ic.ac.uk

Summary

The objective of this coursework is to learn a metric that excels in person re-identification (re-ID). We explored different metric learning methods, namely Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Siamese Network with Contrastive Loss (SNCL) and Infomation-Theoretic Metric Learning (ITML). These metrics were evaluated with k-nearest neighbour and k-means, and the results were compared against the baseline. The transformed data for each metric is 3D-plotted for qualitative comparison and insights. The experimental result shows that ITML performs the best with $acc_{rank1} = 47.14\%$*,* $acc_{rank5} = 67.14\%$*,* $acc_{rank10} = 76.14\%$ *and* $acc_{kmeans} = 70.01\%$ *when tested with the specified evaluation protocol.*

1. Introduction

This project aims to perform metric-learning in person re-identification (re-ID). It is done by training the CUHK03 datasets, which is a very popular dataset containing 13,164 images of 1,360 pedestrians. The k-Nearest Neighbour (k-NN) classifier and k-means are used to evaluate the performance for different metric learning methods. The training is done using **Python** with libraries such as **numpy**, **pandas**, **matplotlib**, **sklearn** and **metric-learn**.

1.1. Dataset

Images in this dataset were captured with use of two cameras, and each identity was photographed a number of times. Therefore, each identity (person) is represented in the dataset approximately 8-9 times. This dataset is further split into query set, gallery set and training set. Query and gallery sets make up as the test set, which is a disjoint set of training set, i.e the labels in the test set will not be present in the training set.

The features of the dataset were extracted from images by training ResNet50 on the training set and then passing both training and test sets through exactly the same network. This has implication in training and validation, which will be discussed in detail in Section 3.

2. Formulation of the Learning Problem

Since the features of the sample have been extracted and trained, it is aimed to learn the Mahalanobis distance $d(x, y)$ such that it maps the features to a space that would result in a better performance (in terms of speed and accuracy).

The distance metric $d(x, y)$ is formulated as:

$$\begin{aligned} d_A(x_1, x_2) &= (x_1 - x_2)^T A(x_1 - x_2) \\ &= (x_1 - x_2)^T G^T G(x_1 - x_2) \\ &= (Gx_1 - Gx_2)^T (Gx_1 - Gx_2) \\ &= \|Gx_1 - Gx_2\|_2^2 \end{aligned} \tag{1}$$

The goal is to learn the transformation matrix $A = G^T G$, which is positive-definite, such that it transforms the features space x closer to the related groups (or further away from dissimilar groups), in hope that such transformation will result in higher accuracy. Section 4.2 will lay out the details of the optimisation problem to achieve the goal.

3. Baseline Approach

The baseline model is to make $A = I$ (identity matrix), which makes the distance metric an euclidean distance. It is initially thought to randomly allocate 100 labels from the training set as the validation set (consists of query and gallery sets). However, as the features of the training set have already been trained, the rank-1 k-NN and the k-means validation accuracy are extremely high (close to 100%). This proves to be difficult and unreliable to learn the distance metric from the training set. This shows that the training set is already very well trained and optimised, hence the validation set extracted is no longer representative of the test set. Since the goal is to learn the transformation matrix A (or G) and the validation set is not representative of the test set, the test set is used to evaluate the performance of the distance metric trained with the training set.

Classification is done using k-NN by applying query set onto gallery set to determine the top k groups (or labels) the query set is closest to with the Mahalanobis distance defined in Equation 1. For k-means, classification is done by first forming N number of clusters, where N is the number of labels present in the set, and then linear assignment is used to map the clusters to the labels provided.

Table 2 shows the baseline rank-1, rank-5 and rank-10 accuracy of validation set and test set. Rank-k accuracy

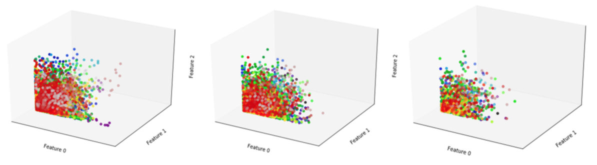

Figure 1. Plot of training set, gallery set and query set respectively

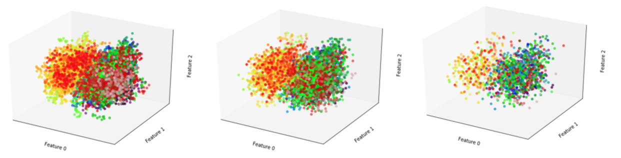

Figure 2. Plot of training set, gallery set and query set respectively on PCA reduced space

is defined as the presence of the actual label in its k nearest neighbours predicted, extracted from the cumulative matching characteristic (CMC) score [1]. The 3D-plot of the first three features with their respective labels (represented by different colours) is shown in Figure 1.

4. Improved Approach

4.1. Principal Component Analysis

PCA is a technique used for identification of a smaller number of uncorrelated variables known as principal components from a larger set of data [2]. It is done by computing the eigenvectors and eigenvalues of the covariance matrix $S = \frac{1}{N}VV^T$, where N is the number of samples, V is the matrix of the difference between each image's feature (x_n) and the respective average feature of all images ($\overline{x}$).

When operating at the original space with 2048 features, the evaluation using k-NN takes 4 mintutes 22s to complete with an rank-1, 5, and 10 accuracy of 47%, 66.86%, 74.93% respectively. With dimension reduction using PCA by picking the top 50 features, the time taken is reduced drastically to 4.87s with the rank-1, 5 and 10 accuracy of 46.9%, 67.5%, and 74.86%. With the same reduction, the evaluation time of k-means is reduced from 5 min 31s to 33.8s with a slight improvement of accuracy from 68.74% to 69.26%. This substantial improvement in the evaluation speed outweighs the slight drop in accuracy. PCA is important for further learning as the reduced dimension will greatly improve the evaluation speed without compromising the accuracy. Therefore, following Section 4.2 most training will be operating on PCA reduced space.

The result is summarised in Table 2. The 3D plot of the first three features of the training, gallery and query set on PCA reduced space (top 50) with their respective labels is shown in Figure 2.

4.2. Metric Learning

4.2.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) [3] is a supervised dimensionality reduction technique. It generates a matrix that maps the labelled data points to a subspace that maximises ratio of between-class variance to within-class variance. The data points will be more linearly separable in the resulting subspace, which generally leads to a better classification performance.

The objective of LDA is to maximise the following function,

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \tag{2}$$

where w is the projection subspace, S_B is the between-class scatter matrix, and S_W is the within-class scatter matrix. Using Lagrange reduction, this is equivalent to solving:

$$max_w w^T S_B w \quad \text{subject to} \quad w^T S_W w = k \tag{3}$$

The solution of this function can be found by solving the following eigenvalue problem,

$$S_w{}^{-1} S_B w = \lambda w \tag{4}$$

where w and λ is the eigenvector and eigenvalue of the matrix $S_w{}^{-1}S_B$. The transformation matrix, G (recall $A = G^T G$) can be generated by selecting the first m eigenvectors with the biggest eigenvalues,

$$G = [w_1, w_2, w_3, ..., w_m] \tag{5}$$

The transformation matrix G is generated by first reducing the dimension of data from 2048 to 1000 using PCA, and then selecting the first 765 eigenvectors of LDA, which is the theoretical limit (766 classes in training set) as the maximum rank of G is equal to $c - 1$, where c is the number of classes. The use of PCA before LDA ensures the non-singularity of S_W and increases the speed of the algorithm. A distance metric can be trained using LDA. First, the transformation matrix G is generated using the training set. The features of the test set are then mapped onto the LDA space, i.e Gx_n. Figure 3 shows the training data's distribution in the new feature space (only the first three features). With PCA-LDA, the rank-1, 5, 10 accuracy for k-NN are 44.79%, 65.93%, and 74.36%, and 65.95% for k-means.

4.2.2 Siamese Network with Contrastive Loss

We propose a metric learning method that trains a neural network which maps data to a new feature space. It is not possible to represent this network as a matrix due to its

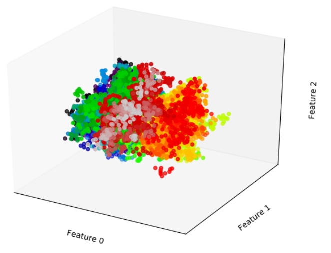

Figure 3. 3D plot of training data in LDA feature space

non-linearity, but it functions just as the transformation matrix G in the previous metric learning methods. This neural network is trained by feeding pairs of data points into a Siamese network (SNCL) [4]. The architecture of our Siamese network is shown in Figure 4.

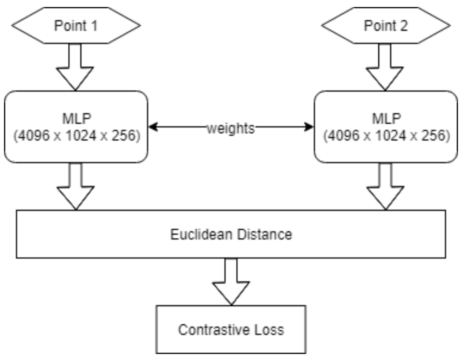

Figure 4. Architecture of a Siamese network

The network takes in 2 data points as input and feed them into two separate but identical neural network that shares the same weights and biases. The difference between the two outputs is then measured using euclidean distance. The loss of the Siamese network is calculated using a contrastive loss function,

$$L(W, Y, \overrightarrow{X_1}, \overrightarrow{X_2}) = \\ (1-Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2 \tag{6}$$

where W is the MLP network, Y is a binary label assigned to the pair (0 for same class and 1 for different class), $\overrightarrow{X_1}$ is point 1, $\overrightarrow{X_2}$ is point 2, and D_W is the euclidean distance between two outputs of W. $m > 0$ is the margin that specifies how far the desired minimum distance is for points of different class. Training the Siamese network with this loss function will minimise the euclidean distance for similar points and push dissimilar points apart.

The distance metric is trained by sampling pairs of points from the training set and feeding them into this Siamese network. The weights of the two MLP networks are jointly updated after each batch of samples containing 256 pairs of randomly sampled pairs. The trained MLP network is then used to map data points to a new feature space. k-NN can be implemented in that new feature space using Euclidean distance as the metric. Similarly, k-means can be used to evaluate the test set with the new feature space. The network is trained using the following parameters: MLP dimension = (4096 x 1024 x 256), batch size = 256, ratio of dissimilar pairs to similar pairs = 3. Figure 5 shows the training data in the feature space of the MLP network after 3400 iterations.

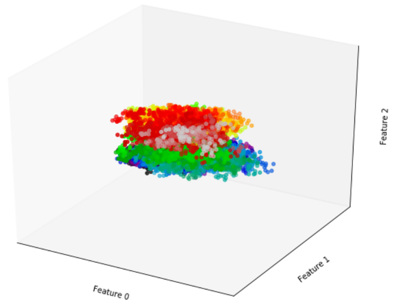

Figure 5. 3D plot of training data in MLP feature space (3400 iterations

The network is trained and the model is saved for every 200 iterations. Figure 6 shows the accuracy of the network for different number of iterations. The MLP network performed best after 3400 iterations, with $acc_{rank1} = 40.07\%$, $acc_{rank5} = 59.86\%$, and $acc_{rank10} = 69.64\%$. The accuracy slowly drops beyond this point due to overfitting, as the network starts to learn a metric that is specific to the training set and generalises badly with other images.

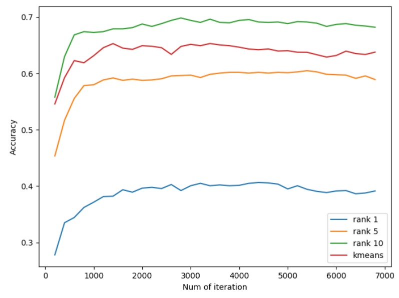

Figure 6. Rank 1, 5, 10 and k-means accuracy of MLP for different number of iterations

4.2.3 Information-Theoretic Metric Learning

Information-Theoretic Metric Learning (ITML) is a method to minimise the differential relative entropy between two multivariate Gaussians under constraints on the distance function, which can be formulated into a Bregman optimisation problem by minimising the LogDet divergence subject to linear constraints [5]. The metric learning problem is summarised as following:

$$min_A KL(p(x; A_0)||p(x; A)) \tag{7}$$

subject to

$$\begin{aligned} d_A(x_i, x_j) \leq u \quad (i, j) \in S, \\ d_A(x_i, x_j) \geq l \quad (i, j) \in D \end{aligned}$$

where S and D refer to the set of similar points and dissimilar points respectively. $KL(p(x; A_0)||p(x; A)) = \int p(x; A_0) \log \frac{p(x;A_0)}{p(x;A)} dx$ is a measure of "closeness" between two Mahalanobis distance functions. The Mahalanobis matrix A is regularised to be as close as possible to a given Mahalanobis distance function, parameterized by A_0. The measure of "closeness" between A and A_0 is quantified via a natural information-theoretic approach, in which the Mahalanobis distance parameterized by A is expressed as its corresponding multivariate Gaussian $p(x; A) = \frac{1}{Z} \exp\left(-\frac{1}{2} d_A(x, \mu)\right)$, where μ = Gaussian mean, and Z = normalising constant.

The learning algorithm is executed with the help of **metric-learn** package [6]. The matrix A is trained with the PCA reduced dimension (top 50) training sets. The algorithm essentially brings similar points closer together while "pushing" dissimilar points apart. It tries to find a feasible solution of problem (7) by introducing a slack variable γ to ensure the algorithm converges. By varying γ from 0.1 to 1.0, the result is tabulated in Table 1. The 3D plot of the transformed training sets by ITML is shown in Figure 7.

γ	Rank-1 (%)	Rank-5 (%)	Rank-10 (%)
0.2	47.14	67.14	76.14
0.4	46.14	67.43	76.14
0.6	46.43	67.29	75.36
0.8	45.57	66.64	75.64
1.0	46.21	66.79	75.00
5.0	46.36	66.64	75.64
10.0	45.14	66.79	75.14

Table 1. Rank 1, 5, and 10 accuracy of metrics A learned from ITML

From the table, it is obvious that at $\gamma = 0.2$, the accuracy is generally improved compared to the baseline. However, it is observed that ITML is quite robust against the slack variable γ and the number of dimensions, with the accuracy fluctuating between ±1%. The plot of the accuracy against the number of features is shown in Figure 8. The convergence time is also significantly faster compared to other distance metric algorithms such as LMNN and MMC.

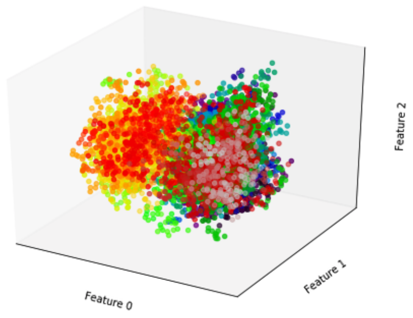

Figure 7. 3D plot of ITML transformed training set

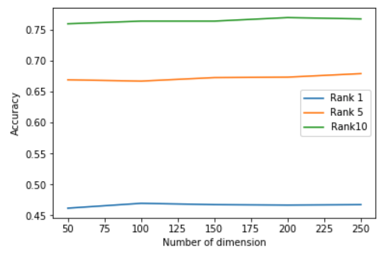

Figure 8. 3D plot of ITML transformed training set

5. Evaluation and Conclusion

Sets	Rank-1 (%)	Rank-5 (%)	Rank-10 (%)	k-means
Validation	99.50	99.50	99.50	98.00
Baseline	47.00	66.86	74.93	68.74
PCA	46.86	67.50	74.86	69.26
PCA-LDA	44.79	65.93	74.36	65.95
SNCL	40.07	59.86	69.64	65.36
ITML	47.14	67.14	76.14	70.01

Table 2. Rank 1, 5, 10 knn, and kmean accuracy for validation, baseline, PCA, PCA-LDA, SNCL and ITML set

All the metric learning approaches aforementioned essentially try to achieve a common goal: bring together similar points and "push" apart dissimilar points with different objective functions. This is evident in the 3D plot of the transformed features shown in Figure 3, 5 and 7, where similar points are observed to become "closer" but in different ways. Among the rest, ITML is found to be more robust against input parameters and achieve the best result, which is shown in Table 2. Since the features of the training set are well optimised and trained, taking a validation set from it is not representative of the test set, which is exemplified by the extremely high validation accuracy. It also makes improvement upon the already optimised training set difficult, depicted by the slight improvement or drops in of test accuracy due to overfitting.

References

[1] Tong Xiao. Evaluation metrics, 2017.

[2] Nick Burns. Pattern recognition via principal components analysis, 2017. URL `http://www.sqlservercentral.com/articles/R+Language/159578/`.

[3] Tae-Kyun Kim. Random sampling lda for face recognition, 2018.

[4] Siamese network (mnist). URL `https://github.com/leimao/Siamese_Network_MNIST`.

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

[6] metric-learn: Metric learning in python. URL `https://github.com/metric-learn/metric-learn`.

Appendix

CMC curve

The CMC curve for each metric is plotted up to rank 10 in Figure 9.

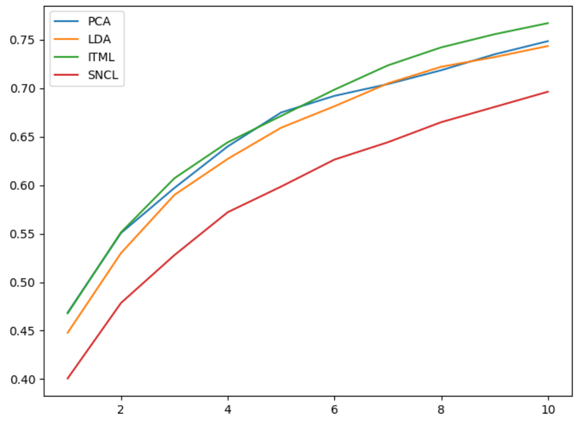

Figure 9. CMC curve of different metric (up to rank 10)

User Instructions

1. Clone repository from this Github link: https://github.com/CliveWongTohSoon/PatternRecognition2018.git. The code is contained in the Coursework2 directory.

2. The following python packages are needed: ujson, sklearn, scipy, matplotlib, tensorflow, pandas, numpy and metric-learn.

3. Download provided dataset CW2_data.zip and extract contents into a directory ’assets’. The assets directory should be in the Coursework2 directory.

4. Run baseline.py, pca.py, lda.py or itml.py to train and evaluate metric-learning methods of interest.

5. Run siamese_train.py to train neural network. Models will be saved in ’model’ and can be used for evaluation. The default save period is 200.

6. Run siamese_test.py to test neural network. Open the file and change the variable ’test_model’ to evaluate that particular model. The folder contains some trained models that can be used for testing.