Pattern Recognition - Coursework 2

Koral Hassan, Mohika Gupta
Imperial College London
Kensington, London SW7 2AZ
kbh15@ic.ac.uk, mg5215@ic.ac.uk

Abstract

The CUHK03 dataset is used to experiment with identity matching. First, the multi-class classification problem is formally defined. Then, some baseline classifiers are implemented and compared against improved approaches. Distance metric learning is utilised and the challenges each model solves are discussed. Among the different methods are k-NN, k-means, MMC and LMNN.

1. The Problem

The aim is to be able to identify a new image of a person from a new angle. We possess previous images of the same person taken from a different angle. We also possess images of other people from the same angles. Therefore, we wish to successfully match the new image to the old images of the same person. Furthermore, we want the model to work for any previously unencountered group of people.

This functionality is similar to Google Photos going through a gallery and distinctly categorizing the different people that appear in the pictures.

The model should not learn the characteristics of specific people in the training dataset. Rather, it should learn to relate any new image to any previous group of images. To this end, we will be using two different sets of people with no overlap for our training and testing datasets.

2. The Dataset

The CUHK03 dataset containing 13,164 images of 1,360 pedestrians is used. Images in this dataset were captured with the use of 2 cameras and each person photographed approximately 8-9 times. Along with this, a file containing the feature vectors representing each image in the dataset was used. Each image in the dataset has corresponding 'camID' and 'labels' label where 'camID' refers to the camera used to take the image and 'labels' is used to identify pedestrians.

2.1. Pre-processing

The MATLAB file provided contained data labels and the indices to use to split the array containing the features vectors into training, query and gallery subsets. Since Python was used to process this data, all the indices provided had to be offset by -1 before use.

Once the features were split into training, query and gallery, they were standardized with respect to the training data. The mean μ and standard deviation σ of the training features was calculated. Then every sample x_i in the dataset was standardized using these values as: $z_i = \frac{x_i - \mu}{\sigma}$. The motivation behind standardization of the dataset was to make the the individual features looks more like standard Gaussian distributed data, with mean zero and unit variance, to ensure an estimator can learn from all features as expected.

2.2. Validation

A part of our training set was initially set aside for the purposes of validation. We selected 115 unique identities out of the 767 found in the training set (15%). However, the set could not validate any results. The models' accuracies on the validation set were highly correlated to the accuracies on the training set. That is to say, the validation set was incapable of providing any additional valuable insights. For the same reasons, the validation accuracy was so high that varying the hyper-parameters produced no significant difference.

The reason for the unusually high correlation was the feature extraction process. The extraction algorithm was developed on the entirety of the training set. Crucially, this set included both of our newly split training and validation sets. This apparently benign inclusion actually causes a data leakage, making our validation accuracies unreliably high. The same noise characteristics are present in the training and validation features since the extraction process took both into account. The validation set should have actually been set aside before the feature extraction process.

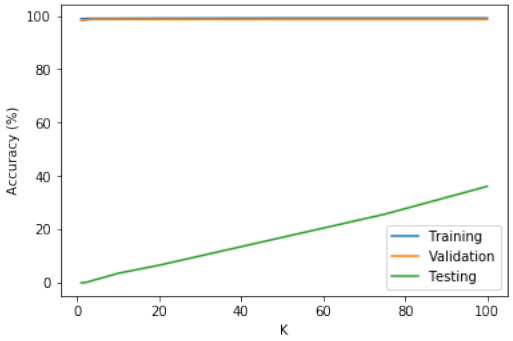

Figure 1. KNN and K-means Baseline Retrieval Accuracy

The aforementioned unreliability can be clearly demonstrated by comparing the training, validation, and testing accuracies, as was done in Figure 1. The testing set was not involved in the feature extraction process so gives an accurate representation of the out-of-sample error. The validation error is similar to the training error, when it should be similar to the testing error instead.

3. Baseline

3.1. K-Nearest Neighbours

The k-Nearest Neighbours algorithm (k-NN) is a non-parametric method used for classification and regression. [1] We will be using it for classification. The input consists of the k closest training examples in the feature space. The output is a class membership.

An object is normally classified by a majority vote of its neighbours, with the object being assigned the class most common among its k nearest neighbours. In our case, we accepted the classification as accurate so long as the correct label was within the set of neighbours.

3.2. K-means

k-means clustering is a method of vector quantization. It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, also called a cluster centre. This results in a partitioning of the data space into Voronoi cells.

The cluster centres are first distributed randomly. Then their positions are re-adjusted to better represent the mean point of the cluster members. This results in some members shifting from one cluster to another, so the re-adjusting is iterated several times to get the final position of the centres. Since the initial centres are randomly distributed and only find local optimums, this process is not deterministic. The training might be done multiple times with different initial states in order to get results closer to the global optimum.

3.3. Results

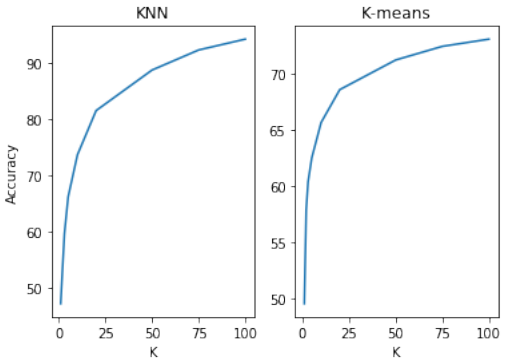

Figure 2. KNN and K-means Baseline Retrieval Accuracy

4. Improved Approach

4.1. Distance Metric Learning

Metric learning aims to find a distance function over datapoints. A metric or distance function has to obey four axioms: non-negativity, identity of indiscernibles, symmetry and subadditivity. These axioms are mathematically defined in Appendix A.

We aim to learn a metric which minimizes the distance between datapoints in related groups, and maximizes the distance between datapoints in unrelated groups.

4.2. Mahalanobis Distance Learning

The general form for the Mahalanobis pseudo metrics is

$$\mathrm{d_M}(\mathbf{x}_1 - \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T M(\mathbf{x}_1 - \mathbf{x}_2) \quad (1)$$

$$= \| \mathbf{x}_1 - \mathbf{x}_2 \|_M^2 \quad (2)$$

If M can be written in the form $\mathrm{M} = \mathrm{L}^{\mathrm{T}}\mathrm{L}$ then the Mahalanobis distance is just the squared Euclidean distance after applying the linear transformation L

$$d_A(\mathbf{x}_1 - \mathbf{x}_2) = \| \mathrm{L}(\mathbf{x}_1 - \mathbf{x}_2) \|_2^2 \quad (3)$$

5. Principal Component Analysis

Principal Component Analysis (PCA) computes an orthogonal transformation, in order to project data onto a set of linearly uncorrelated variables. These variables are called principal components.

Principal components are chosen such that they maximize the variance between data points. The basis of the principal component space is formed from the M eigenvectors that correspond to the M largest eigenvalues of the covariance matrix of the training data. In other words, each

succeeding component has the highest variance possible under the constraint that it is orthogonal to the preceding components.

$$\max_{L} Tr(L^T C L) \text{ subject to: } LL^T = I \quad (4)$$

PCA is used for reducing the dimensionality of the feature space. This is desirable for avoiding overfitting and reducing computational load. It avoids overfitting because we increase the number of datapoints per dimension in our dataset (since the size of the training set stays constant). It reduces computational load since we are losing information, but we are doing it with an objective function loses only the most irrelevant information.

Need input data, ML problem, objective function (can find), loss function (Subject to the constraint that L defines a projection matrix, see Lin Alg?)???

Where the optimization has a closed form solution of the the M eigenvectors that correspond to the M largest eigenvalues of the co-variance matrix.

- Unsupervised metric

6. Mahalanobis Metric Learning for Clustering

Mahalanobis Metric learning for Clustering (MMC) was designed to learn a metric to improve the performance of clustering algorithms such as K-means, it achieves this by minimizing the distances between similarly labeled inputs, and maximizing the distance between differently labeled inputs. This is formulated as the minimization of the sum of the squared distances between pairs of similar points, whilst constraining the the sum of squared distances between pair of different points to be greater than some margin.

6.1. Problem Formulation

Define the training subset as the set $\mathcal{X}$ such that data points $(x_i, x_j) \in \mathcal{X}$ then we can construct the sets $\mathcal{S}, \mathcal{D}$ containing pairs of points (x_i, x_j) such that

$$(x_i, x_j) \in \mathcal{S} \text{ if } x_i, x_j \text{ are similar} \quad (5)$$

$$(x_i, x_j) \in \mathcal{D} \text{ if } x_i, x_j \text{ are 'dissimilar' or not in } \mathcal{S}. \quad (6)$$

Optimization problem:

$$\min_{M} \sum_{(x_i,x_j)\in\mathcal{S}} \| x_i - x_j \|_M^2 \quad (7)$$

$$\text{s.t.} \sum_{(x_i,x_j)\in\mathcal{D}} \| x_i - x_j \|_M^2 \geq 1, \quad (8)$$

$$\mathrm{M} \leq 0 \quad (9)$$

Where equation 7 ensures M does not project the data set to a single point.

Note that the constraints (7) and (8) are convex, therefore this optimization problem is convex, allowing us to derive a global minimum. By restricting M to be a diagonal matrix, this optimization can be solved efficiently using the Newton-Raphson method.

6.2. Implementation

The following procedure was followed to implement MMC on the given dataset:

1. With the features of the training images as the input, optimize M as per the constraints defined in equations (8,9).

2. Using $M = L^T L$, perform the linear transform L on features of the query and gallery images.

3. Perform k-NN on the MMC transformed query and gallery features to evaluate the retrieval accuracy.

6.3. Initial Results

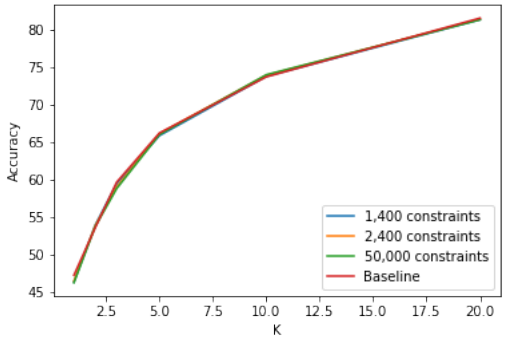

Figure 3. KNN + MMC compared to the baseline for a different number of constraints

The number of constraints is a measure of how many pairs of similar points are used to optimize M and is a parameter of the function used to implement MMC. Note how the use of MMC does not result in improvement on the baseline when evaluated using the kNN retrieval accuracy. This is further discussed in section 8.

6.4. K-means + MMC

As suggested in [2] we used MMC to learn a distance metric to bring similar points together, before applying the K-means algorithm described in section 3.2 to cluster the data. From this, we expect an improvement on baseline K-means, as MMC will pull similar points closer together, reducing the euclidean distance between then, leading to the generation of most informative cluster centres from the K-means algorithm.

6.5. Results of K-means + MMC

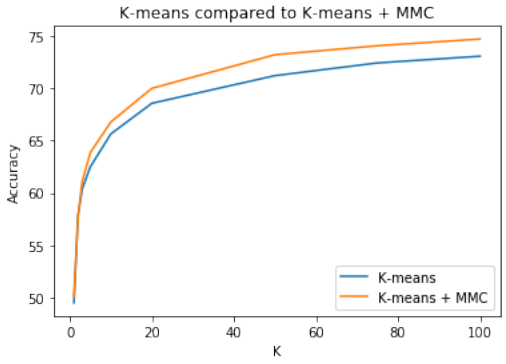

Figure 4. Improved Approach for K-means.

7. Large Margin Nearest Neighbour

The Large Margin Nearest Neighbour (LMNN) model has been specifically conceived to learn a Mahalanobis distance metric to improve the accuracy of kNN classification. The main difference between LMNN and MMC is that MMC learns a metric to minimize distances between all pairs of similarly labelled samples. In comparison, LMNN learns a metric that is optimized so that k-nearest neighbours belong to the same class, and samples from different classes are separated by a large margin.

7.1. Approach

kNN classification accuracy increases when data points which have the same label as the input point $\mathbf{x}_i$ are closer than differently labelled points. We define target neighbours (denoted $\mathbf{x}_j$) of $\mathbf{x}_i$ before training, where target neighbours are data points we desire to bring closest to $\mathbf{x}_i$ during training. We also define an imposter $\mathbf{x}_l$ as any differently labelled input such that

$$\| L(\mathbf{x}_i - \mathbf{x}_j) \|^2 \leq \| L(\mathbf{x}_i - \mathbf{x}_j \|^2 + 1 \tag{10}$$

During training, the imposter is pushed outside the margin defined.

7.2. Problem Formulation

The loss function formulated for LMNN in terms of the linear transformation L is not convex, therefore the results of an optimization function such as gradient descent will depend on initial estimates of L and may only find local minima rather than the global optimum. To overcome this, the problem is reformulated using Semi-definite Programming (SDP) over a positive semi-definite matrix $M = L^T L$. The cost function reformulated in terms of the Mahalanobis metric M is shown in equation (11), this function is now convex and can be efficiently optimized.

$$\epsilon(M) = (1-\mu) \sum_{i,j} d_M(\mathbf{x}_i, \mathbf{x}_j) + \mu \sum_{i,j} \sum_{l} (1 - y_{il})[1 + d_M(\mathbf{x}_i, \mathbf{x}_j) - d_M(\mathbf{x}_i, \mathbf{x}_l)]_+ \tag{11}$$

Where μ is a regularization parameter to prevent overfitting. Since this model is not sensitive to μ[3], we kept μ constant at 0.5.

Note that the cost function has two competing terms. The 1st term penalizes large distances between the input point and target neighbours, the second term penalizes cases where an imposter breaches the perimeter + margin separation distance defined by the input and target data points. The indicator variable y_{il} is defined as:

$$y_{il} = \begin{cases} 1, & \text{iff } y_i = y_l. \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The SDP optimization problem is to minimize:

$$(1-\mu) \sum\nolimits_{(i,j)} d_M(\mathbf{x}_i - \mathbf{x}_j) + \mu \sum\nolimits_{i,j,l} (1 - y_{il}) \xi_{ijl} \tag{13}$$

subject to:

$$d_M(\mathbf{x}_i - \mathbf{x}_l) - d_M(\mathbf{x}_i - \mathbf{x}_j) \geq 1 - \xi_{ijl} \tag{14}$$

$$\xi_{ijl} \geq 0 \tag{15}$$

$$\mathbf{M} \geq 0 \tag{16}$$

Where the slack variable ($\xi_{ijl} \geq 0$) is introduced to monitor the extent that equation (10) is violated.

7.3. Implementation

Before implementing LMNN on the data set, PCA was used to reduce the dimension of the inputs from 2048 to 512 ($\mathbf{x}_i \in \mathbb{R}^{2048} \rightarrow \mathbf{x}_i \in \mathbb{R}^{512}$) in order to reduce computation time. PCA takes the training, query and gallery features as input and returned the transformed training, query and gallery features ($\Omega_{trainingPCA}, \Omega_{queryPCA}, \Omega_{galleryPCA}$) as output.

The following procedure was followed to implement LMNN on the PCA transformed dataset:

1. With the input $\Omega_{trainingPCA}$, optimize M as per the constraints defined in equations (12,13,14).
2. Using $M = L^T L$, perform the linear transform L on $\Omega_{queryPCA}$ and $\Omega_{galleryPCA}$.
3. Perform k-NN on the LMNN transformed $\Omega_{queryPCA}$ and $\Omega_{galleryPCA}$ to evaluate the retrieval accuracy.

Method	Computation Time(s)
LMNN alone	419.6
PCA (M=512) + LMNN	38.45

Table 1. Comparison of LMNN computation time with and without PCA

7.4. Results

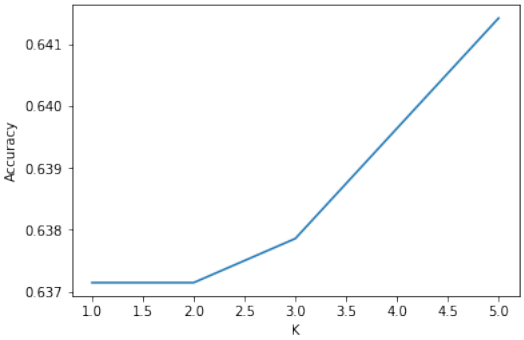

Figure 5. LMNN Accuracy when k-NN = 3.

8. Conclusions

8.1. Feature Extraction

The features were extracted from images by training ResNet50 on the training set and then passing both training and test sets through exactly the same network. [4] Since our validation set was extracted from our training set, its results were unreliable.

Another consequence of the feature extraction process was that it was very difficult to improve on the baseline method. The neural network was trained such that images with the same labels had very similar features extracted, and images with different labels had very different features extracted. By definition, this is optimizes the features for the use of euclidean as a metric. For this reason, our baseline methods performed very well.

8.2. Metric Learning

Discuss results

8.3. Training and Testing

Algorithms are trained using the features vectors split into the training set provided. For testing the query and gallery subsets were used.

9. Appendix

A. Properties of a Metric

For distance metric d;

$$d(\mathbf{x}, \mathbf{y}) \geq 0$$

$$d(\mathbf{x}, \mathbf{y}) = 0 \iff x = y$$

$$d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$$

$$d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$$

No. of Principal Components, M	39	63	116	200	245	363
NN	54.499	58.97	58.33	58.97	59.6	60.26
Alternative Method	73.07	73.07	73.07	73.07	73.07	73.04

Table 2. Accuracy(%) for NN and Alternative Method.

K-means - clusters generally modelled as normal or unimodal distributions - parametric - proves the importance of our per-processing steps.

- Account for images taken of same person with same camera in the gallery set

References

[1] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[2] M. I. J. Eric P. Xing, Andrew Y. Ng and S. Russell. Distance metric learning, with application to clustering with side-information.

[3] L. K. S. Kilian Q. Weinberger. Distance metric learning for large margin nearest neighbor classification, 2009.

[4] M. Nazarczuk. Suspiciously high validation error, 2018. Mentioned on the discussion boards.