Metric Learning Approaches for Face Identification

COMPUTER VISION (CSE578) COURSE PROJECT

AKSHAY BANKAR (2019201011)

ANGAD YENNAM (2019900082)

APRIL 2020

Github Repository

Contents

List of Figures

List of Tables

Abstract

The project's focus is understanding and implementation of methods proposed for face identification by Matthieu Guillaumin et al. in their paper "Is that you? Metric Learning Approaches for Face Identification". Face identification is the problem of determining whether two face images depict the same person or not. The authors propose two methods for learning robust distance measures: (a) a logistic discriminant approach which learns the metric from a set of labelled image pairs (LDML) and (b) a nearest neighbour approach which computes the probability for two images to belong to the same class (MkNN). The results of these methods are compared with state-of-the-art methods of Largest Margin Nearest Neighbors (LMNN) and Information-Theoretic Metric Learning. We understand and implement these methods as well. The evaluation is done on Labelled Faces in Wild (LFW) dataset. In addition to this, we show the results on a relatively smaller Olivetti faces dataset as well.

We observed that the LDML method proposed out-performs the LMNN and ITML in case of Olivetti faces dataset but gives sub-optimal results on LFW dataset. This owes to the fact that the LWF dataset is challenging, having a big variety in pose, expression, lighting. All these methods showed improvement in their performance when used with M-KNN as classification technique as compared to KNN on both the datsets. As proposed by the authors, we use ROC-AUC curve for measuring performance of the algorithms.

1   Introduction

1.1   What is metric learning

Distance metric learning aims at automatically constructing task-specific distance metrics from (weakly) supervised data, in a machine learning manner. The learned distance metric can then be used to perform various tasks (e.g., k-NN) classification, clustering, information retrieval, data visualization).

1.2   Mahalanobis Distances

Given a real-valued parameter matrix L of shape (num_dims, n_features) where n_features is the number features describing the data, the Mahalanobis distance associated with L is defined as :

$$D(x, x') = \sqrt{(Lx - Lx')^\top (Lx - Lx')}$$

A Mahalanobis distance is a Euclidean distance after a linear transformation of the feature space defined by L (taking L to be the identity matrix recovers the standard Euclidean distance). Mahalanobis distance metric learning can thus be seen as learning a new embedding space of dimension num_dims.

Let X be a set and $d : X^2 \to R$ a function with the following properties:

(i) $d(x, y) \geq 0$ for all $x, y \in X$ i.e distances are always positive.

(ii) $d(x, y) = 0$ if and only if $x = y$ i.e two points are zero distance apart if and only if they are the same point.

(iii) $d(x, y) = d(y, x)$ for all $x, y \in X$ i.e the distance from X to Y is the same as the distance from Y to X.

(iv) $d(x, y) + d(y, z) \geq d(x, z)$ for all $x, y, z \in X$ i.e the distance from A to B via C is at least as great as the distance from A to B directly (This is called the *triangle inequality* after the result in Euclidean geometry that the sum of the lengths of two sides of a triangle is at least as great as the length of the third side).

If a mapping satisfies the first three properties but not the fourth, it is called a pseudo-metric. Then we say that d is a *metric* on X and that (X, d) is a *metric space*.

We obtain a family of metrics over X by computing Euclidean distances after performing a linear transformation x = L x. It is common to express squared distances under the metric in terms of the square matrix: $M = L'L$. Any matrix M formed in this way from a real-valued matrix L is guaranteed to be positive semidefinite (i.e., to have no negative eigenvalues). In terms of the matrix M, squared distances are denoted by $D(\ xi,\ xj) = (\ xi\ xj)'M(\ xi\ xj)$ and pseudometrics of this form are refered as Mahalanobis metrics.

2   Marginalized K-NN (Mknn)

The kNN classification is used to assign single data point xi to one of a fixed set of k classes associated with the training data. The probability of class c for xi is

$$p(y_i = c|\mathbf{x}_i) = n_c^i/k,$$

where n(i,c) is the number of neighbours of xi of class c. But the aim here is to predict whether a pair of images (xi, xj) belongs to the same class or not. To adapt to such clustering marginal probability is computed that assigns xi and xj to the same class using a kNN classifier, which is given by,

$$p(y_i = y_j|\mathbf{x}_i, \mathbf{x}_j) = \sum_c p(y_i = c|\mathbf{x}_i)p(y_j = c|\mathbf{x}_j)$$
$$= k^{-2}\sum_c n_c^i n_c^j.$$

The method can be viewed as a nearest neighbor classifier in the implicit binary labelled set of N^2 pairs. In this set, we need a measure to define neighbours of a pair which can be done by taking all pairs of k neighbours of xi with all of the k neighbours of xj. The probability for the positive class given by this classifier for a pair is then determined by the number of positive and negative neighbour pairs. Hence the score of Marginalized kNN (MkNN) binary classifier for a pair of images (xi, xj) is based on how many positive neighbour pairs can be formed from neighbours of xi and xj.
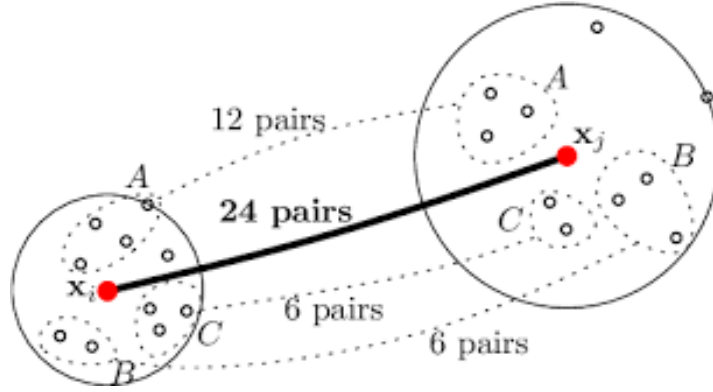

Figure 1: Schematic representation of k = 10 neighbourhoods for xi and xj, and the 24 neighbour pairs (out of 100) that have the same name and contribute to the score.

3   Large Margin Nearest Neighbor (LMNN)

The method attempts to learn a Mahalanobis distance metric of the form described in above section1. The model has three essential properties : (i) Its convex loss function, (ii) its goal of margin maximization, (iii) the constraints on the distance metric imposed by accurate kNN classification.

3.1   Terminology

Target neighbors: Each input xi has k nearest neighbors that share its same label yi. These establish a perimeter based on Mahalanobis distance.

Imposters : These are differently labeled inputs in the training set that invade perimeter set by the target neighbors.

$$\|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2 \leq \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 + 1.$$

The model is based on two idealizations: : first, that each training input xi should share the same label yi as its k nearest neighbors (target neighbors); second, that training inputs with different labels(imposters) should be widely separated.

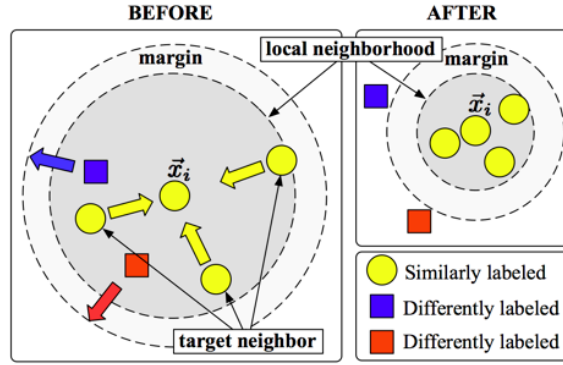Figure 2: Illustration of one input's neighborhood before training (left) versus after training (right).

3.2   Loss function

The loss function consists of two terms, one which acts to pull target neighbors closer together, and another which acts to push differently labeled examples further apart.
The first term in the loss function penalizes large distances between each input and its target neighbors.

$$\varepsilon_{\text{pull}}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2.$$

The second term in the loss function penalizes small distances between differently labeled examples.

$$\varepsilon_{\text{push}}(\mathbf{L}) = \sum_{i,j \rightsquigarrow i} \sum_{l} (1 - y_{il}) \left[1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2 \right]_+$$

where the term $[z]+ = \max(z, 0)$ denotes the standard hinge loss.

3.3   Convex optimization

The loss function defined above is not convex in the matrix elements of the linear transformation L. To minimize this loss function, approach is prone to being trapped in local minima.
This is overcome by reformulating the above equation as an optimization over positive semi-definite matrices. From section 1, D(x, x') denotes the squared distance with respect to the Mahalanobis metric M. Therefore, by substituting for L by D(x, x') in the loss function we get,

$$\varepsilon(\mathbf{L}) = (1 - \mu)\, \varepsilon_{pull}(\mathbf{L}) + \mu \varepsilon_{push}(\mathbf{L}).$$

The loss function is now expressed over positive semi-definite matrices M ¿ 0,as opposed to real-valued matrices L. Hence this loss function is a piece-wise linear, convex function of the elements in the matrix M.

$$\varepsilon(\mathbf{M}) = (1 - \mu) \sum_{i,j \rightsquigarrow i} \mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) + \mu \sum_{i,j \rightsquigarrow i} \sum_{l} (1 - y_{il}) \left[1 + \mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) - \mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_l) \right]_+$$

The above optimization can be formulated as an SDP (semi-definite program), which is a linear program that incorporates an additional constraint of that requires the matrix to be positive semi-definite i.e to only have non-negative eigenvalues. To formulate the above equation as an SDP introduce non-negative slack variables for all triplets of target neighbors (j ¿i) and impostors xl. The slack variable is used to measure the amount by which the large margin inequality of the imposters is violated. Using the slack variables to monitor these margin violations, the SDP obtained is:

Minimize $(1 - \mu) \sum_{i,j \rightsquigarrow i} (\vec{x}_i - \vec{x}_j)^\top \mathbf{M} (\vec{x}_i - \vec{x}_j) + \mu \sum_{i,j \rightsquigarrow i,l} (1 - y_{il}) \xi_{ijl}$ **subject to:**

(1) $(\vec{x}_i - \vec{x}_l)^\top \mathbf{M}(\vec{x}_i - \vec{x}_l) - (\vec{x}_i - \vec{x}_j)^\top \mathbf{M}(\vec{x}_i - \vec{x}_j) \geq 1 - \xi_{ijl}$
(2) $\xi_{ijl} \geq 0$
(3) $\mathbf{M} \succeq 0.$

This SDP form can be solved by standard solver packages.

3.4 Implementation details

The training images are read and given to face detector module which returns the face ROI. This is done using OpenCV's Face detection implementation which is based on Viola-Jones algorithm. The detected faces are of variable size. These are resized to 150x150. The resized face images are flattened into 1-D array. On this, dimensionality reduction is performed using PCA such that 95% of the information is retained. The reduced eigen faces thus obtained are given to LMNN module which learns the distance metric.

4 Information Theoretic Metric Learning (ITML)

4.1 Formulation

An information-theory based distance metric learning algorithm. Given an initial metric, it learns the nearest metric that satisfies some similarity and dissimilarity constraints. The closeness between the metrics is measured using the Kullback-Leibler divergence between the corresponding gaussians. Given pairs of similar points S and pairs of dissimilar points D, it formulates distance metric learning problem as,

$$\min_{A} \quad \mathbf{KL}(p(\boldsymbol{x}; A_0) \| p(\boldsymbol{x}; A))$$
$$\text{subject to} \quad d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq u \qquad (i, j) \in S,$$
$$d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \ell \qquad (i, j) \in D.$$

where the distance measure between two Mahalanobis distance functions parameterized by A0 and A by the (differential) relative entropy between their corresponding multivariate Gaussians:

$$\mathbf{KL}(p(\boldsymbol{x}; A_o) \| p(\boldsymbol{x}; A)) = \int p(\boldsymbol{x}; A_0) \log \frac{p(\boldsymbol{x}; A_0)}{p(\boldsymbol{x}; A)} \, d\boldsymbol{x}.$$

LogDet divergence : The LogDet divergence is a Bregman matrix divergence generated by the convex function (X) = log detX defined over the cone of positive-definite matrices, and it equals (for n n matrices A, A0),

$$D_{\ell d}(A, A_0) = \text{tr}(AA_0^{-1}) - \log \det(AA_0^{-1}) - n.$$

It has been shown that the differential relative entropy between two multivariate Gaussians can be expressed as the convex combination of a Mahalanobis distance between mean vectors and the LogDet divergence between the covariance matrices.

$$\mathrm{KL}(p(\boldsymbol{x}; A_0) \| p(\boldsymbol{x}; A)) = \frac{1}{2} D_{\ell d}(A, A_0)$$

Hence the optimization problem is re-posed as

$$
\begin{aligned}
\min_{A \succeq 0, \boldsymbol{\xi}} \quad & D_{\ell d}(A, A_0) + \gamma \cdot D_{\ell d}(\mathrm{diag}(\boldsymbol{\xi}), \mathrm{diag}(\boldsymbol{\xi}_0)) \\
\text{s. t.} \quad & \mathrm{tr}(A(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T) \le \xi_{c(i,j)} \quad (i,j) \in S, \\
& \mathrm{tr}(A(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T) \ge \xi_{c(i,j)} \quad (i,j) \in D,
\end{aligned}
$$

In some cases, there may not exist a feasible solution. To prevent such a scenario from occurring, slack variables are incorporated into the formulation to guarantee the existence of a feasible A.

4.2   Implementation details

Refer 5.2 for details which has similar flow.

5   Logistic Discriminant Metric Learning (LDML)

A distance metric learning algorithm that maximizes the likelihood of a logistic based probability distribution.The model is based on the idea that the distance between images in positive pairs to be made smaller than the distances corresponding to negative pairs, and obtain a probabilistic estimation of whether the two images depict the same object.

5.1   Formulation

Using the Mahalanobis distance between two images, the method models the probability pn that pair n = (i, j) is positive, i.e. the pair label tn is 1, as:

$$p_n = p(y_i = y_j | \mathbf{x}_i, \mathbf{x}_j; \mathbf{M}, b) = \sigma(b - d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j))$$

where

$$\sigma(z) = (1 + \exp(-z))^{-1}$$

is the sigmoid function and b a bias term. The bias works as a threshold value and is learned together with the metric parameters. Now the probability can be re-written as:

$$:p_n \stackrel{-}{=} \sigma(b - W^\top X_n)$$

where W is the vector containing elements of M and Xn the entries of $x - x'.\mathrm{T}x - x'$. The parameters of the model are then optimized using the maximum log-likelihood. The log-likelihood, L, can be written as:

$$\mathcal{L} = \sum_n t_n \ln p_n + (1 - t_n) \ln(1 - p_n)$$
$$\nabla \mathcal{L} = \sum_n (t_n - p_n) X_n,$$

This log-likelihood is smooth and concave. Thus, can be solved by gradient ascent.

5.2 Implementation details

The training images are read and given to face detector module which returns the face ROI. This is done using OpenCV's Face detection implementation which is based on Viola-Jones algorithm. The extracted face is the given to shape predictor module which performs 68-point landmark detection. This uses module in Dlib which is implementation of a random forest classifier proposed by Kazimi et al for face landmark detection. Once the landmark points are obtained, 9 of these points corresponding to eye, nose and lip corners are selected as keypoints for SIFT feature extraction. Corresponding to each point, we get 128-length feature vector at three scales. Thus, for a face image we obtain a 128x9x3 = 3496-length feature vector. Dimensionality reduction of these feature vectors of train set is performed with number of components equal to 35 as proposed in the paper. This reduced feature vectors obtained are given to LDML/ITML module which learns the distance metric.
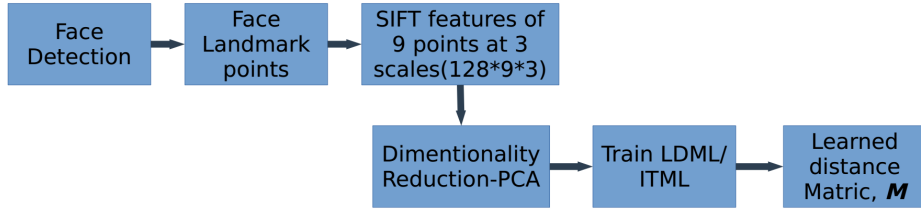
6 Results

6.1 Dataset

The Olivetti faces dataset : The dataset contains ten different images of each of 40 distinct subjects. Relatively simpler dataset as the variation is mostly in the lighting and facial expressions.

Labelled Faces in Wild (LFW) : This dataset is a collection of JPEG pictures of famous people

collected over the internet. There are a total of 13233 images and 5749 people in the database. Each image is a 250x250 jpg.

6.2   Performance measure

The results are reported based on the operating points of the ROC curves.

AUC-ROC curve : It is a performance measurement for classification problem. ROC is a probability curve and AUC represents degree or measure of separability i.e it tells how much model is capable of distinguishing between classes. The ROC curve is plotted with true positive rate (TPR) against the false positive rate (FPR).

6.3   Results

6.3.1   Performance on Olivetti faces dataset

Method	NN-method	ROC classification results			
		3-NN	5-NN	7-NN	10-NN
LMNN	K-NN	85.28	81.45	81.47	74.14
	M-KNN	86.38	92.47	94.69	92.94
ITML	K-NN	89.11	90.41	87.23	89.28
	M-KNN	95.20	96.31	97.14	96.20
LDML	K-NN	82.39	72.85	71.83	70.15
	M-KNN	96.20	95.85	96.21	94.43

Table 1: ROC classification result comparison of LMNN, ITML and LDML on Olivetti faces dataset
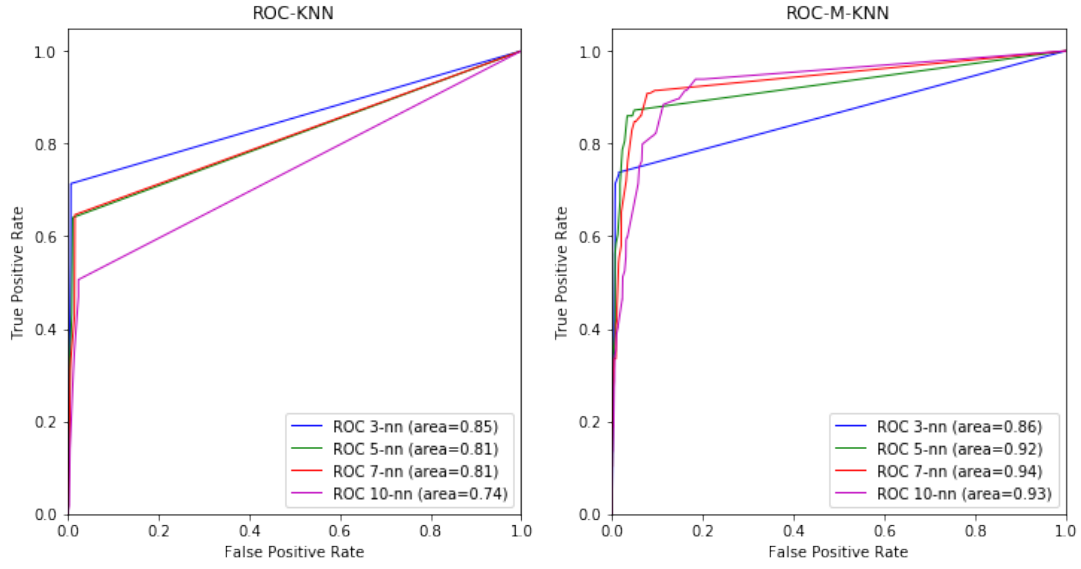
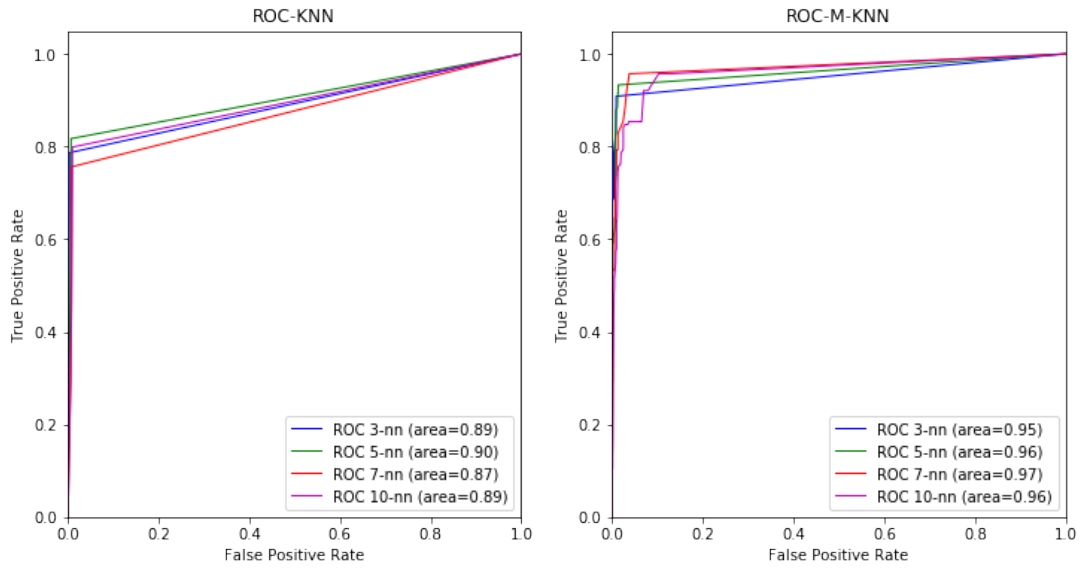Figure 3: ROC classification curve for LMNN method with KNN and M-KNN

Figure 4: ROC classification curve for ITML method with KNN and M-KNN
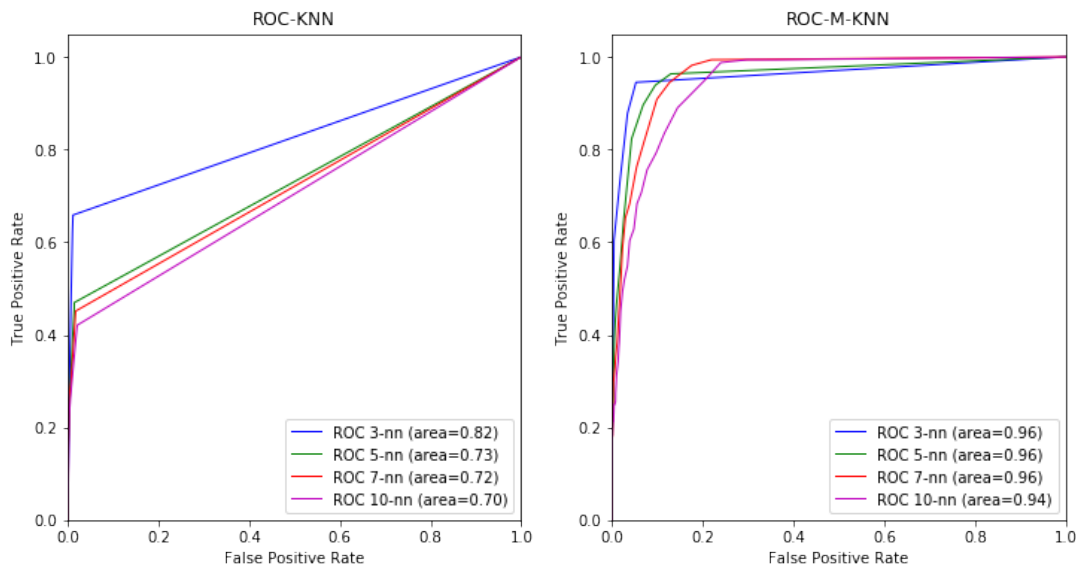
Figure 5: ROC classification curve for LDML method with KNN and M-KNN

6.3.2 Performance on Labelled Faces in Wild dataset

Method	NN-method	ROC classification results			
		3-NN	5-NN	7-NN	10-NN
LMNN	K-NN	59.01	59.23	57.23	57.70
	M-KNN	66.63	70.07	71.10	69.55
ITML	K-NN	59.62	62.03	62.98	65.41
	M-KNN	70.48	85.38	82.65	84.10
LDML	K-NN	50.76	52.25	51.90	52.09
	M-KNN	60.05	62.21	60.43	67.29

Table 2: ROC classification result comparison of LMNN, ITML and LDML on LFW dataset
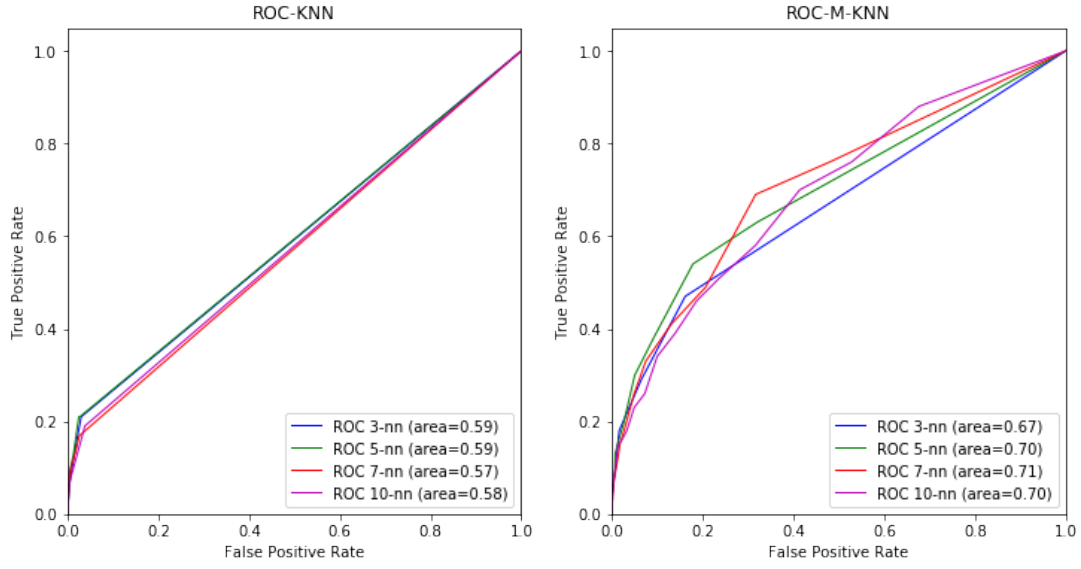
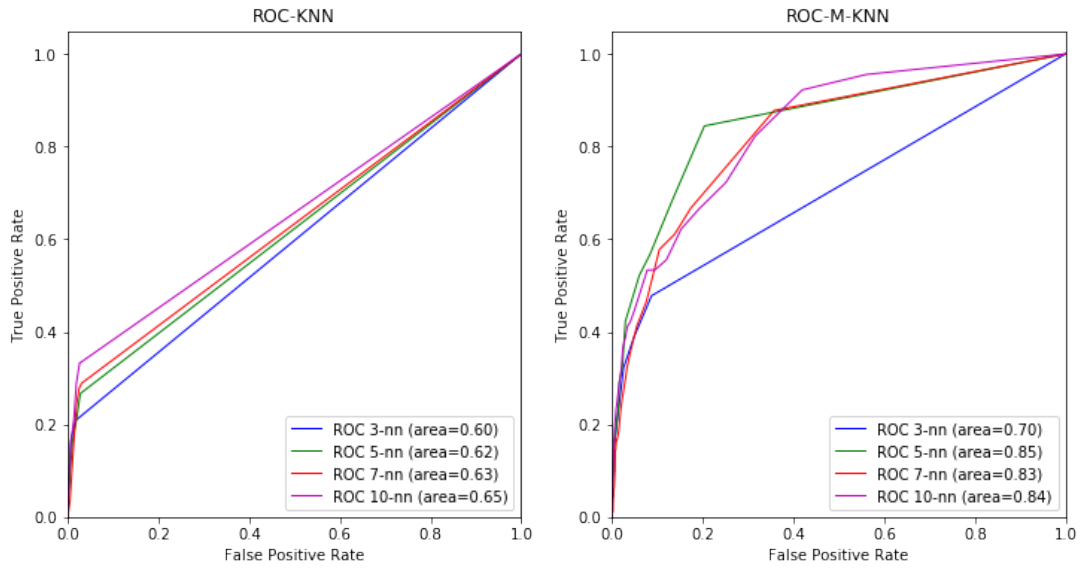Figure 6: ROC classification curve for LMNN method with KNN and M-KNN

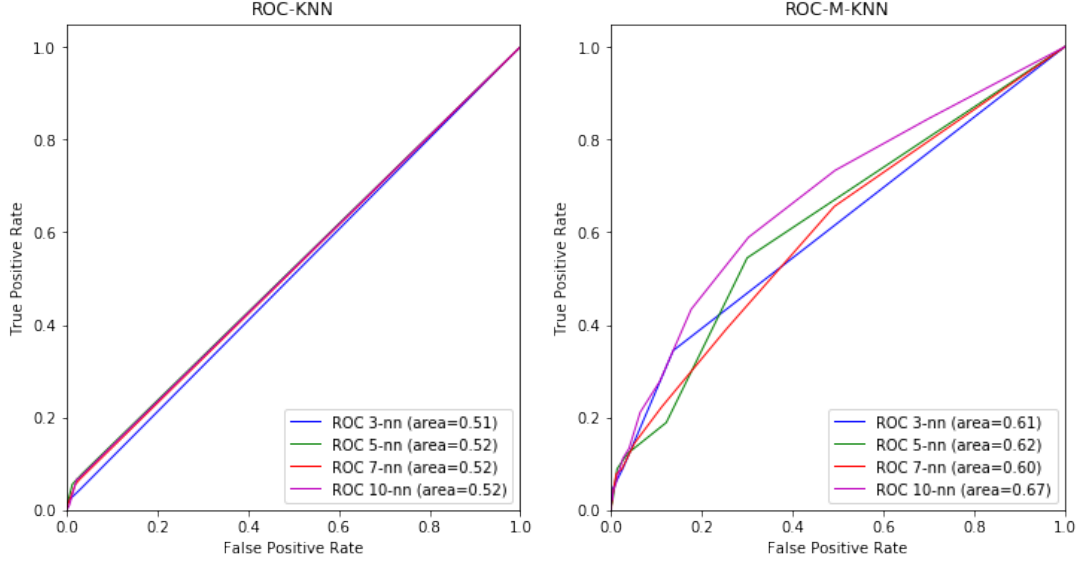Figure 7: ROC classification curve for ITML method with KNN and M-KNN

Figure 8: ROC classification curve for LDML method with KNN and M-KNN

6.4   Observations

The Olivetti faces dataset :

The results show that the proposed method of LDML out-performs the LMNN and ITML methods used for comparison. Also, the classification using proposed M-KNN improves results of all the methods as compared to when used with KNN.

The improvement in result can be described by the fact that the dataset is less challenging in terms of variations in various imaging aspects. As described below, the proposed method gives comparitively sub-optimal results on more challenging LFW dataset.

Labelled Faces in Wild dataset :

The perfomance obtained by the proposed LDML method on this dataset are not in line with the one's obtained in the paper. The dataset, in general, is challenging, having a big variety in pose, expression, lighting as compared to Olivetti faces dataset.

As with previous dataset, the M-KNN method shows consistent improvement in all methods' results when compared with K-NN as classification technique. This can be regarded to the fact that the score of our Marginalized kNN (M-KNN) binary classifier for a pair of images (x_i, x_j) is based on how many positive neighbour pairs we can form from neighbours of x_i and x_j. It is not "local" in the sense of usual K-NN classifiers as M-KNN measures the correspondence between two distinct local neighbourhoods.

7   Challenges

- The size of the training data affects LDML a lot. As the number of training pairs increase, the LDML seems to become intractable and gave convergence issue.

- Since SIFT features are used, these came out to be zero resulting in erroneous matrix operations.

- The LMNN implementation in pyDML was slow. Then used implementation from pyLMNN as separate code base.

- Lack of resources related to the formulation of the authors algorithms for understanding and implementing.

References

[1] Matthieu Guillaumin, Jakob Verbeek and Cordelia Schmid. *Is that you? Metric Learning Approaches for Face Identification.* 2009 IEEE 12th International Conference on Computer Vision.

[2] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. *Information-theoretic metric learning.* In ICML, 2007.

[3] K. Weinberger, J. Blitzer, and L. Saul. *Distance metric learning for large margin nearest neighbor classification.* In NIPS, 2006.

[4] P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* Proceedings of the 2001 IEEE Computer Society Conference on CVPR 2001.

[5] Vahid Kazemi and Josephine Sullivan. *One Millisecond Face Alignment with an Ensemble of Regression Trees.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[6] David G. Lowe *Distinctive Image Featuresfrom Scale-Invariant Keypoints.* CS Department, University of British Columbia Vancouver, January, 2004

[7] M. Everingham, J. Sivic, and A. Zisserman. *'Hello! My name is... Buffy' - automatic naming of characters in TV video.* In BMVC, 2006.

[8] Implementation of the Large Margin Nearest Neighbor algorithm.
`https://pypi.org/project/PyLMNN/`
`http://pylmnn.readthedocs.io/en/latest/`

[9] Distance Metric Learning Algorithms for Python, Juan Luis Suárez Díaz
`https://pypi.org/project/pyDML/`
`https://pydml.readthedocs.io/en/latest/`