Metric Learning Approaches for Face Identification [2mm]

Computer Vision (CSE578) Course Project

Akshay Bankar (2019201011)

Angad Yennam (2019900082)

April 2020

Contents

List of Figures

List of Tables

Abstract

1 Introduction

1.1 What is metric learning

Distance metric learning (or simply, 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 follows:
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, $y \in X$.
(ii) $d(x, y) = 0$ if and only if $x = y$.
(iii) $d(x, y) = d(y, x)$ for all x, $y \in X$.
(iv) $d(x, y) + d(y, z) \geq d(x, z)$ for all x, y, $z \in X$. (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.)

Then we say that d is a *metric* on X and that (X, d) is a *metric space*. You should imagine the author muttering under his breath
'(i) Distances are always positive.
(ii) Two points are zero distance apart if and only if they are the same point.
(iii) The distance from A to B is the same as the distance from B to A.
(iv) The distance from A to B via C is at least as great as the distance from A to B directly.'
if a mapping satisfies the first three properties but not the fourth, it is called a pseudometric.

We obtain a family of metrics over X by computing Euclidean distances after performing a linear transformation x = L x. These metrics compute squared distances as: DL(xi, xj) = kL(xi xj)k2 2,

It is common to express squared distances under the metric in Eq. (1) in terms of the square matrix: M = LL. (2) 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, we denote squared distances by DM(xi, xj) = (xi xj)M(xi xj), (3) and we refer to pseudometrics of this form as Mahalanobis metrics

2 Marginalized K-NN (Mknn)

The kNN classification is used to assign single data points 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,

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

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 our Marginalized kNN (MkNN) binary classifier for a pair of images (xi, xj) is based on how many positive neighbour pairs we can form from neighbours of xi and xj.

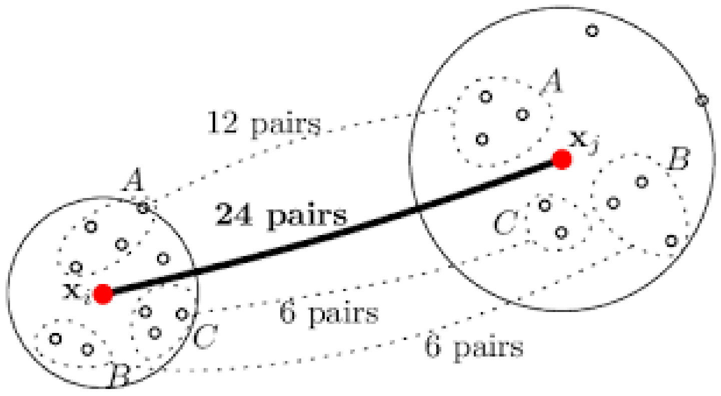


Figure 1: Schematic representation of k = 10 neighbourhoods for xi and xj, and the 24 neighbour pairs (out of 100) that have the same name and contribute to the score.

3 Large Margin Nearest Neighbor (LMNN)

The method attempts to lean a Mahalanobis distance metric of the form described in above section. 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.

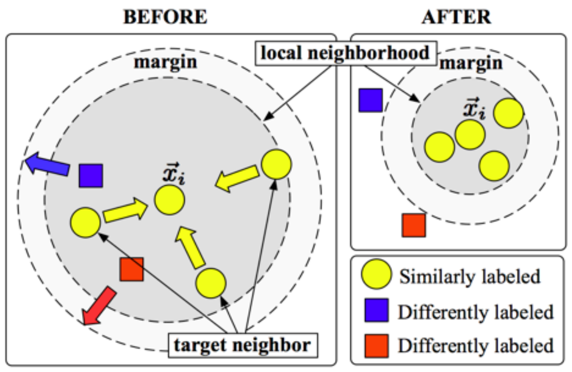


Figure 2: Illustration of one input's neighborhood before training (left) versus after training (right).

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

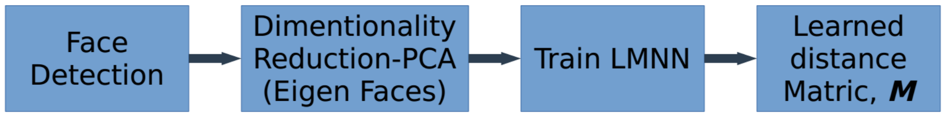
(2) $\xi_{ijl} \geq 0$

(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.


Face Detection
Dimentionality Reduction-PCA (Eigen Faces)
Train LMNN
Learned distance Matric, ***M***

3.4.1 Parameter setting

4 Information Theoretic Metric Learning (ITML)

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.

4.1 Implementation details

Refer 5.2 for details which has similar flow.

4.1.1 Parameter setting

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:

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

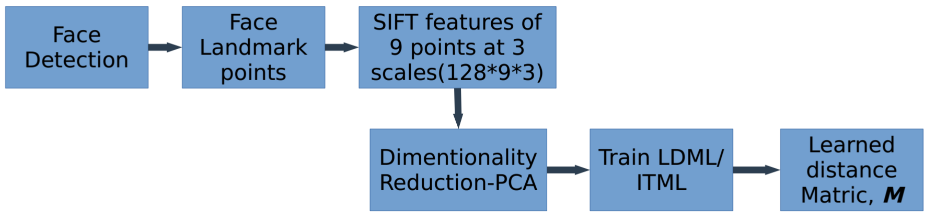
where W is the vector containing elements of M and Xn the entries of $x - x'.\mathrm{T}x - x'$. The parameters of the model are then optimized using the maximum log-likelihood. The log-likelihood, L, can be written as:

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

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.


Face Detection
Face Landmark points
SIFT features of 9 points at 3 scales(128*9*3)
Dimentionality Reduction-PCA
Train LDML/ ITML
Learned distance Matric, ***M***

5.2.1 Parameters setting

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

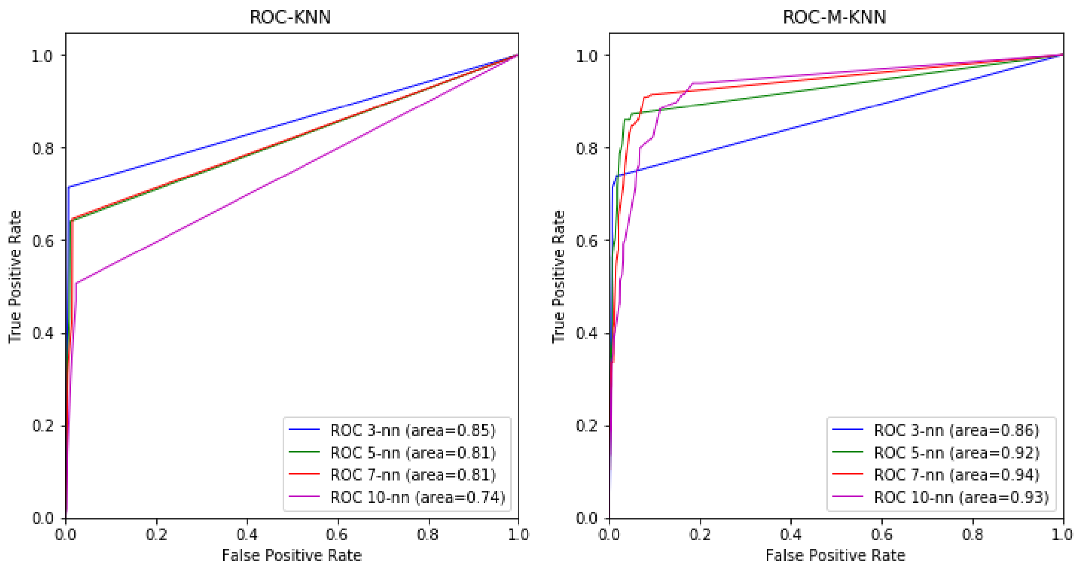


Figure 3: ROC classification curve for LMNN method with KNN and M-KNN

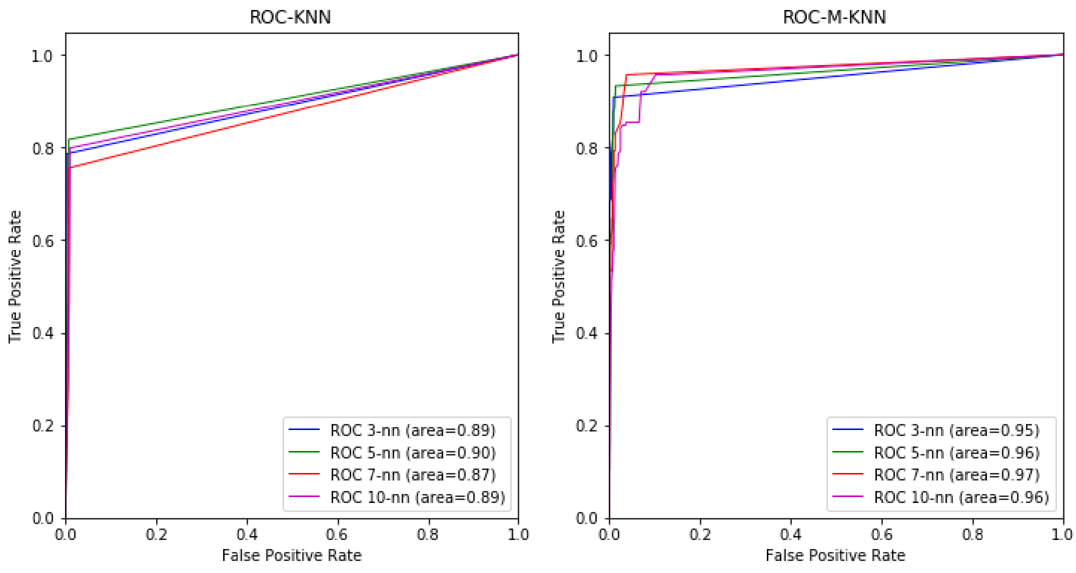


Figure 4: ROC classification curve for ITML method with KNN and M-KNN

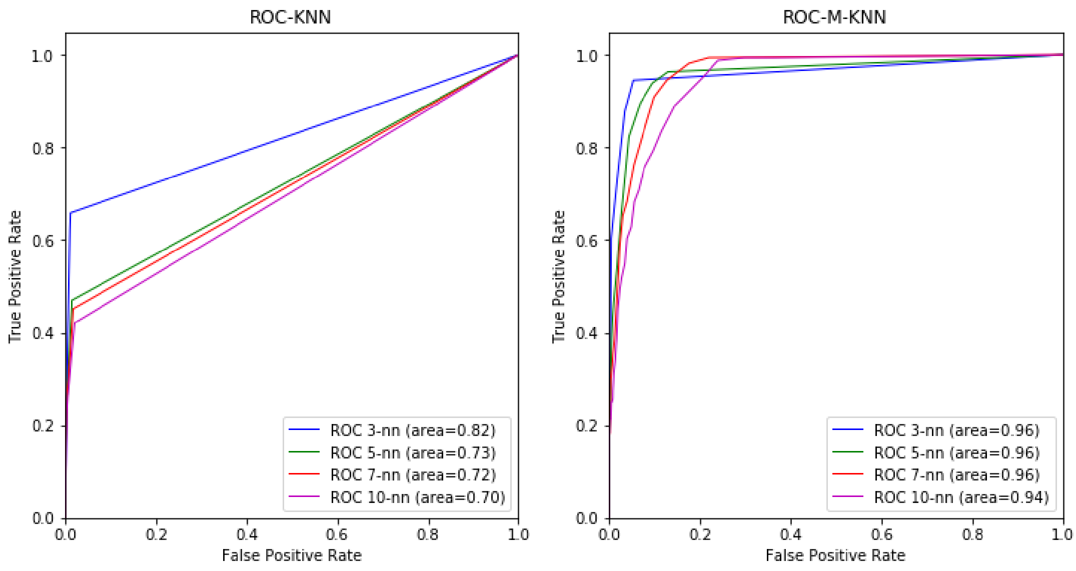


Figure 5: ROC classification curve for LDML method with KNN and M-KNN

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

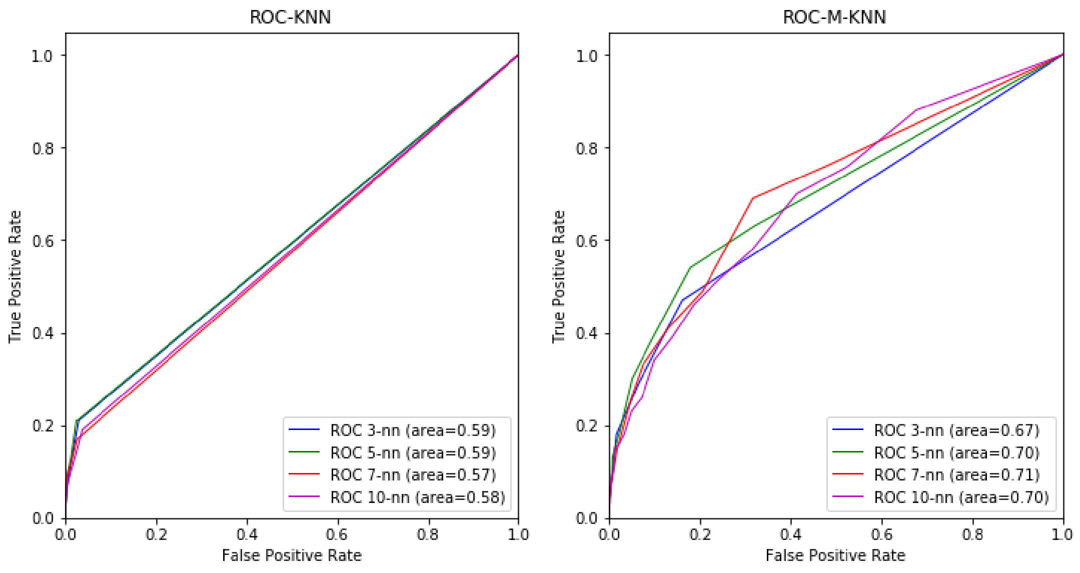


Figure 6: ROC classification curve for LMNN method with KNN and M-KNN

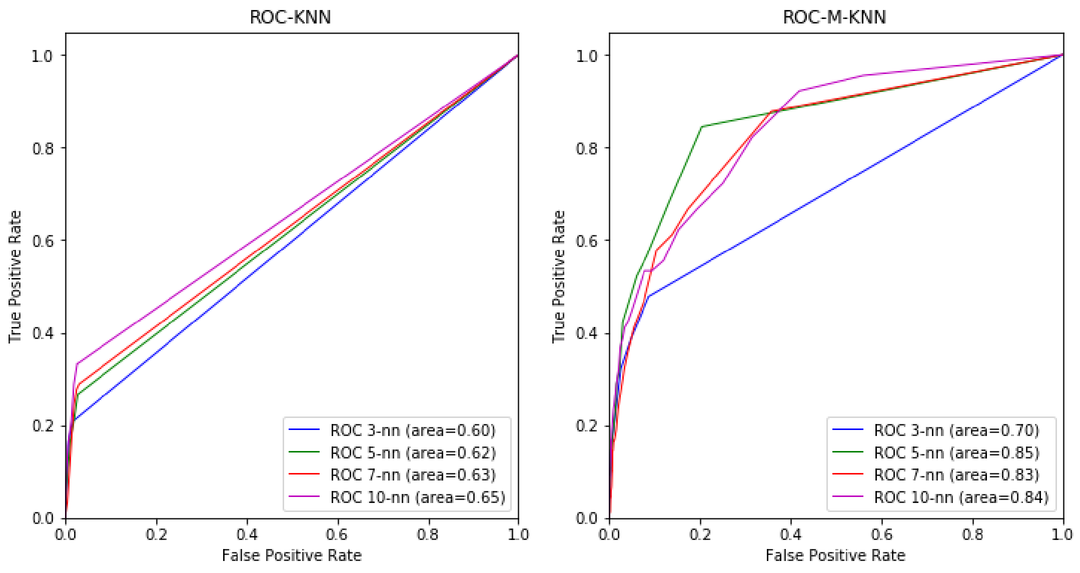


Figure 7: ROC classification curve for ITML method with KNN and M-KNN

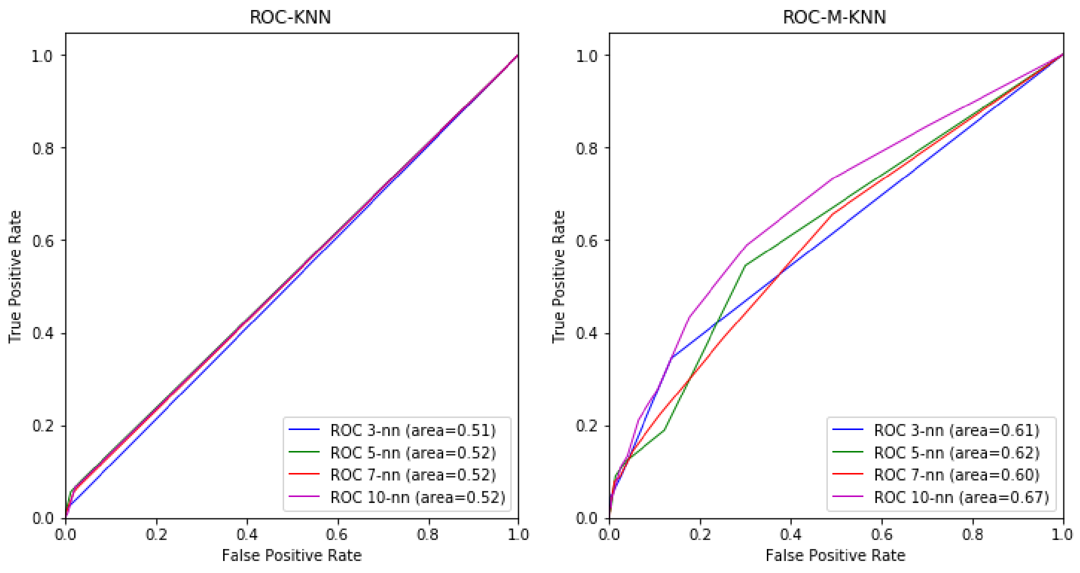


Figure 8: ROC classification curve for LDML method with KNN and M-KNN