MS BIG DATA 2015
Facial Recognition Challenge Report

François Blas

June 22, 2015

Contents

Chapter 1

Challenge Description

1.1 Goal of the Challenge

The data in this challenge are provided by Morpho, Safran group company that works especially on facial recognition. The challenge is on face recognition: in particular, the goal is to develop a system that is able to predict whether two face images correspond or not to the same person, even if they have not been seen in learning. For this, we will use a distance measure (or similarity) between the images if the distance is less than a certain threshold τ, it is predicted that this is the same person. In order to get a good performance, it is necessary to use of metric learning techniques to have a distance measure best suited to the problem that a simple distance (Euclidean one for example).

1.2 Datasets

For reasons of confidentiality, Morpho does not directly provide the data as images, but in the form of extracts descriptors from images (the extraction procedure descriptors is not disclosed). The data are separated into two bases: an easy base and a difficult base. For this challenge, there will thus have a leaderboard for each database with a specific performance criterion at each base (see below).

1.2.1 Easy dataset

There is therefore 312 022 training examples, each a vector of 1500 features. The number of class (people) in the training set is 203513. Thus, there is on average very few examples per class (one in many cases)! However, as the goal is to predict whether or not two images belong to the same class, we can usually be reduced to a binary problem on the pairs (with a positive or negative label associated with each pair of images) and avoid the problem of having to make predictions for a very large number of classes. The performance criterion on this is a particular point of the ROC curve: the false negative rate of 0.1% of false positives. This corresponds to setting the threshold low enough τ so as to have very few false positives (that is to say, cases where it is predicted that the two images are the same person, whereas this is not the case). The score then gives the proportion of positive pairs that have not been detected: the lower it is, the more performance is good.

1.2.2 Hard dataset

The data from the hard base are the same size as the base. There are 252813 training examples distributed in 8966 classes. The average number of examples per class is much larger than in the easy basis. The performance criterion is also a point on the ROC curve: but this time it is the Equal Error Rate (EER), corresponding to the point where the false positive rate equals the false negative rate.

Chapter 2

The approach

Here is describe the method used in this challenge. After analyzed the datasets and create some useful functions to manipulate them (like generates pairs and select some random features), I have first tried to compute some simple distance measure by replacing the euclidean one given by the starter kit. This first part is describe on the chapter 'Simple metrics'.
On the second part, I have tried to compute some metric learning methods. The goal is to find a good metric, mainly used in the field of facial recognition and also find the right parameters to compute it. First thing I have tried is the LMNN method, which is quite used. Then, I have used ITML, which is an other method quite close of the LMNN but with a special regularization term. I have finished by using some stochastic descent gradient and an approximation of a special gradient for cosine measure. This second part is describe on the chapter 'Metric Learning'.
All results are discussing on the chapter 'Results'.

Chapter 3

Simple Metrics

Because we are dealing with distance measure it's interesting to simply test some of them before computing some metrics learning methods. Here are describe some alternatives to Euclidean distance which can be use to measure the similarity (or dissimilarity) of dataset's pairs.

3.1 Cosine

$$Cosine(u,v) = 1 - \frac{u.v}{\|u\|_2 \|v\|_2} \tag{3.1}$$

for 2 vectors u and v. $u.v$ represent the dot product between u and v

3.2 Canberra

$$Canberra(u,v) = \sum_i \frac{|u_i - v_i|}{|u_i| + |v_i|} \tag{3.2}$$

for 2 vectors u and v.

3.3 Bray-Curtis

$$BrayCurtis(u,v) = \frac{\sum_i |u_i - v_i|}{\sum_i |u_i + v_i|} \tag{3.3}$$

for 2 vectors u and v.

Chapter 4

Metric Learning

Changing the distance measure is great but it's not good enough, so we have to use some metrics learning methods to optimize our distance measure. The main goal here is to build a measure containing a dynamic similarity matrix. By training an algorithm on the train dataset we can compute a similarity matrix and have a distance measure that fit differently regarding the method. In the following sections we are exploring some interesting methods use in the facial recognition field to build this similarity measure.

4.1 Stochastic Gradient Descent

Let's define all variables and objects used in this section : We are dealing with a training set $D_n = (x_i, y_i)_{i=1}^n \in (X, Y)^n$ where $X \subset \mathbb{R}^p$ and $Y = 1, ..., C$. Here we are interested in the distance family of type 'Mahalanobis' which takes the following form :

$$Mahalanobis(u, v) = d_M(u, v) = (u - v)^T M(u - v) \tag{4.1}$$

for 2 vectors u and v. With M is the cone of symmetric matrices pxp positive semidefinite. We can notice that if M=I, then Mahalanobis is the Euclidean distance.

$$Euclidean(u, v) = \|u - v\|_2 \tag{4.2}$$

The goal is to optimize the matrix M using the training set to obtain a distance adapted to the problem considered. To do that we are dealing with examples pairs (x_i, x_j), $1 < i, j < n$ and we associate them to a specific label y_{ij} as follow :

$$y_{ij} = \begin{cases} 1 & \text{if} \quad y_i = y_j \\ -1 & \text{else} \end{cases} \tag{4.3}$$

Then the metric learning problem is defined as the optimization problem like so :

$$min_M = \frac{1}{n^2} \sum_{i,j=1}^{n} l(d_M(x_i, x_j), y_{ij}) \tag{4.4}$$

Where l is a loss function on the pairs. The algorithm used is a stochastic gradient descent with a Hinge loss function which is very popular in metric learning. The loss function is defined as follow :

$$l(d_M(x_i, x_j), y_{ij}) = max(0, 1 + y_{ij}(d_M(x_i, x_j) - 2)) \tag{4.5}$$

So the gradient is computed like so :

$$\nabla_M l(d_M(x_i,x_j),y_{ij}) = \begin{cases} 0 & \text{if} \quad 1+y_{ij}(d_M(x_i,x_j)-2) \leq 0 \\ y_{ij}(x_i-x_j)(x_i-x_j)^T & \text{else} \end{cases} \tag{4.6}$$

4.2 Large Margin Nearest Neighbor

Here we are considering an other method called Large Margin Nearest Neighbor (LMNN). The main goal is to learn a pseudometric under which all data instances in the training set are surrounded by at least k instances that share the same class label. If this is achieved, the leave-one-out error is minimized.
Let's define variables and objects used in this section : We are dealing with a training set $D_n = (x_i, y_i)_{i=1}^n \in (X,Y)^n$ where $X \subset \mathbb{R}^p$ and $Y = 1,...,C$. Again we are interested in the distance family of type 'Mahalanobis'. The method define 2 types of special data points : target and neighbors and impostors.
Target neighbors Tn_i are selected before learning and are the points that should become nearest neighbors under the learned metric. Each instance x_i has exactly k different target neighbors within D_n, which all share the same class label y_i.
An impostor of a data point x_i is another data point x_j with a different class label ($y_i \neq y_j$) which is one of the nearest neighbors of x_i. During learning the algorithm tries to minimize the number of impostors for all data instances in the training dataset. We can reformulate the optimization of the problem as an instance of

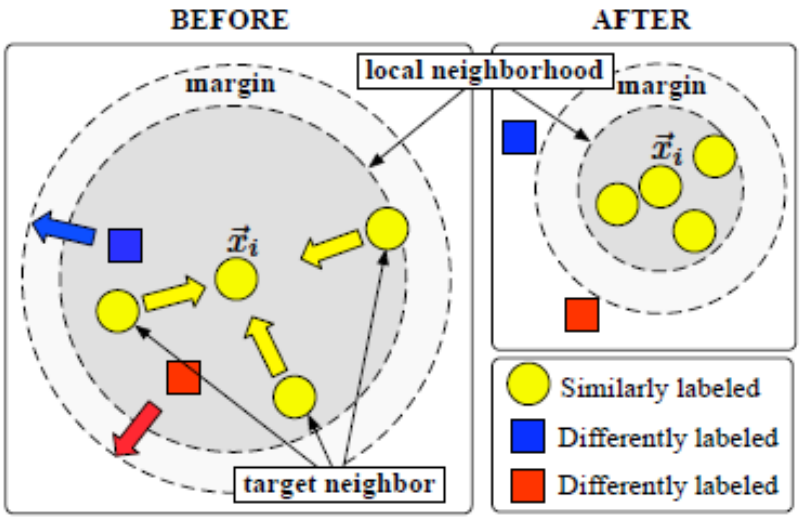

Figure 4.1: Effect of the LMNN method

semidefinite programming. A semidefinite program (SDP) is a linear program with the additional constraint that a matrix whose elements are linear in the unknown variables is required to be positive semidefinite. The algorithm has 2 objectives : for every data point x_i the target neighbors should be close and the impostors should be far away.
To achieve the first goal we have to minimize the average distance between instances

x ans their target neighbors $\sum_{i,j\in Tn_i} D_M(x_i,x_j)$
To achieve the second one we have to constrained impostors x_k to be one unit further away than target neighbors x_j (so pushing them out of the local neighborhood of x_i). The resulting inequality constraint is defined as follow :

$$\forall_{i,j\in Tn_i,k,y_k\neq y_i} d_M(x_i,x_j)+1 \leq d_M(x_i,x_k) \tag{4.7}$$

The final optimization problem is then defined as so :

$$\begin{aligned} min_M \sum_{i,j\in Tn_i} D_M(x_i,x_j) + \sum_{i,j,k} \xi_{ijk} \\ \forall_{i,j\in Tn_i,k,y_k\neq y_i} d_M(x_i,x_j)+1 \leq d_M(x_i,x_k) \\ d_M(x_i,x_j)+1 \leq d_M(x_i,x_k)+\xi_{ijk} \\ \xi_{ijk} \geq 0 \\ M \succeq 0 \end{aligned} \tag{4.8}$$

Here ξ_{ijk} represent the slack variables and absorb the amount of violations of the impostor constraints. Although, The last constraint $M \succeq 0$ indicates that the matrix M is required to be positive semidefinite. SDPs tend to suffer from high computational complexity, this particular SDP instance can be solved very efficiently due to the underlying geometric properties of the problem. In particular, most impostor constraints are naturally satisfied and do not need to be enforced during runtime.

4.3 Information Theoretic Metric Learning

Here we are considering an other method called Information Theoretic Metric Learning (ITML). To optimize the matrix M we are here regularizing it such that it is as close as possible to a known prior M_0. This closeness is interpreted as a Kullbach-Leibler divergence between the two Gaussian distributions corresponding to M and M_0. Typically, the other contraints will be of the form $d_M(x_i,x_j) \leq u$ for positive pairs and $d_M(x_i,x_j) \geq l$ for negative pairs. The trade-off between satisfying the constraints and regularization is controlled in the objective function using an additional parameter. The regularizer is defined as so :

$$r(M) = tr(M) - logdet(M) \tag{4.9}$$

It can be viewed as a special case of the LogDet divergence which is defined by :

$$D_{ld}(M,M_0) = tr(MM_0^{-1}) - logdet(MM_0^{-1}) - d \tag{4.10}$$

Where d is the dimensionality of the data. The LogDet divergence has various properties, many of which are useful in metric learning contexts like connections to multivariate Gaussians :
Consider a multivariate Gaussian parametrized by mean μ and precision matrix M: $p(x,\mu,M) = \frac{1}{Z} exp(d_A(x,\mu))$
Then for two multivariate Gaussians of the same mean, with precision matrices M_0 and M, we have:

$$KL(p(x,\mu,M_0)(x,\mu,M)) = \frac{1}{2} D_{ld}(M,M_0) \tag{4.11}$$

Where KL is the Kullback–Leibler divergence.
So we can sum up the learning problem as a constrained optimization problem such as :

$$\begin{aligned} min_M r(M) = tr(M) - logdet(M) \\ such\,that,\ d_M(x_i, x_j) \leq u \\ and\ d_M(x_i, x_j) \geq l \end{aligned} \tag{4.12}$$

4.4 Quasi Cosine Similarity

Cosine distance is an efficient metric for measuring the similarity of descriptors in classiffication task. However, the cosine similarity metric learning (CSML) is not widely used due to the complexity of its formulation and time consuming. Here we are considering Quasi Cosine Similarity Metric Learning (QCSML) to make it easier. Let's first defined the CSML. Considering variables and objects used in this section : We are dealing with a training set $D_n = (x_i, y_i)_{i=1}^n \in (X, Y)^n$ where $X \subset \mathbb{R}^p$ and $Y = 1, ..., C$. We can decompose M with Cholesky formula as so : $M = WW^T$ and we also assume we have known the prior knowledge about the relationship constraining the similarity or dissimilarity between pairs of points.

$$\begin{aligned} S : (x_i; x_j) \in S\, if\, x_i\, and\, x_j\, are\, similar \\ D : (x_i; x_j) \in D\, if\, x_i\, and\, x_j\, are\, dissimilar \end{aligned} \tag{4.13}$$

Then cosine similarity between two vectors can be defined as :

$$d_W^2(x_i, x_j) = \frac{(Wx_i)^T(Wx_j)}{\|Wx_i\|\|Wx_j|} \tag{4.14}$$

Given the similar sets S and dissimilar sets D, the objective function can be shown as :

$$max \sum_{(x_i,x_j)\in S} d_W^2(x_i, x_j) - \alpha \sum_{(x_i,x_j)\in D} d_W^2(x_i, x_j) \tag{4.15}$$

Then the gradient is computed as follow :

$$\frac{\partial d_W^2(x_i, x_j)}{\partial W} = \frac{1}{v(W)} \frac{\partial u(W)}{\partial W} - \frac{u(W)}{v(W)^2} \frac{\partial v(W)}{\partial W} \tag{4.16}$$

Where,

$$\begin{cases} \frac{\partial u(W)}{\partial W} &= W(x_i x_j^T + x_j x_i^T) \\ \frac{\partial v(W)}{\partial W} &= \frac{\|Wx_j\|}{\|Wx_i\|} Wx_i x_i^T - \frac{\|Wx_i\|}{\|Wx_j\|} Wx_j x_j^T \end{cases} \tag{4.17}$$

Let's simplify the problem, considering the cosine similarity like so :

$$d_W^2(x_i, x_j) \geq \frac{(Wx_i)^T(Wx_j)}{\|W\|^2\|x_i\|\|x_j|} \tag{4.18}$$

We can give the prior knowledge about the projection matrix W which is represented as : With some constraints we can rewritten the cosine similarity as so :

$$d_W^2(x_i, x_j) \geq (Wx_i)^T Wx_j \tag{4.19}$$

Then we define the label $y_i j$ represents the similarity or dissimilarity between a pair of two vectors (x_i, x_j). Therefore, we can denote a threshold $b \in \mathbb{R}$ that the pair is similar or different if the cosine similarity $d^2_W(x_i, x_j)$ is upon or below the threshold b. Therefore, these constrains can be defined as :

$$y_i j(d^2_W(x_i, x_j) - b) \geq 1 \tag{4.20}$$

Where $y_i j = 1$ if x_i and x_j are similar and $y_i j = -1$ otherwise. So the QCSML problem is sum up as follow :

$$\begin{aligned} min \sum_{i,j} max[1 - y_{ij}(d^2_W(x_i, x_j) - b), 0] \\ \|x_i\|_2 = 1,\ x_i \in S \cup D,\ i = 1, ..., n \\ Tr(WW^T) = 1 \end{aligned} \tag{4.21}$$

This method lead to the following algorithm

Algorithm 1: Quasi Cosine Similarity Metric Learning Optimization

Input : Train data: $(X, y), X = \{x_i\} \subseteq \mathbb{R}^d, \|x_i\| = 1, y_i \in \{-1, 1\}$
Output: The parameters:$\Theta = \{W, b\}$

begin
- Parameters initialization;
- `/* Initialize` W `*/`
- Set W_{init} =PCA_whitening(X);
- `/* Initialize` b `*/`
- (X_s, y_s) =sample(X,y);
- $\phi_s = WX_s$;
- $\text{score}_s = \tilde{d}^2_w(x_i, x_j) \quad \forall x_i, x_j \in X_s$;
- $b_{\text{init}} = \text{accuracy\_best}(\text{score}_s, y_s)$;
- `/* SGD iteration */`
- **for** $t = 1$ **to** n **do**
  - **switch** y_t **do**
    - **case** *positive*
      - $\text{score} = \tilde{d}^2_W(x_i, x_j) \quad \forall x_i, x_j \in X_t$;
      - **if** *score* $< b + 1$ **then**
        - $W_{t+1} = W_t - \alpha \frac{\partial \tilde{d}^2_W}{\partial W}$;
        - $b_{t+1} = b_t - \gamma_b$;
      - break;
    - **case** *negative*
      - $\text{score} = \tilde{d}^2_W(x_i, x_j) \quad \forall x_i, x_j \in X_t$;
      - **if** *score* $> b - 1$ **then**
        - $W_{t+1} = W_t + \alpha \frac{\partial \tilde{d}^2_W}{\partial W}$;
        - $b_{t+1} = b_t + \gamma_b$;
      - break;
- **return** $\Theta = \{W, b\}$;

Figure 4.2: QCSML algorithm

Chapter 5

Results

All results for metric learning was quite disappointing. Even some good approach describe in papers (like the cosine one), results on datasets (easy or hard) are not higher than 0.31 for hard and 0.24 for easy. One of the reason of these results is the

LMNN	ITML	SGD	QCSML	
0.27	0.26	0.36	0.24	Easy
0.31	0.35	0.37	0.34	Hard

sub-sampling used for these methods. Because we are dealing with large datasets, I have selected some random features (for example 700 column instead of keeping the 1500 given) and also performed a pairs selection. This sub-sampling impact a lot the accuracy of the similarity matrix generated into the algorithms. Another thing is the computing time, all the methods are quite slow to compute because we have to iterate a lot of times our gradients for example. With a lot of surprises, simple metrics like the Bray-Curtis one are quite good to score for the challenge. As we can see below, the Bray-Curtis score better than complex metric learning methods, but again we have to remind that ML were not trained on the entire dataset.

Euclidean	Canberra	Cosine	BrayCurtis	
0.56	0.32	0.35	0.21	Easy
0.38	0.34	0.35	0.31	Hard

Chapter 6

More Methods

6.1 Online Algorithm for Scalable Image Similarity

The Online Algorithm for Scalable Image Similarity learning (OASIS) learns a bilinear similarity measure over sparse representations. It is an online dual approach using the passive-aggressive family of learning algorithms with a large margin criterion and an efficient hinge loss cost. Experiments show that OASIS is both fast and accurate at a wide range of scales: for a data set with thousands of images, it achieves better results than existing state-of-the-art methods, while being an order of magnitude faster. It is a particular relevant measure for this challenge as it can compute a large dataset quickly and efficiently.

6.2 Logistic Discriminant based Metric Learning

Here we want to learn a Mahalanobis distance d_M that makes images of positive pairs closer than those of negative pairs.
$d_M(x_i, x_j) = (x_i x_j)^T M(x_i x_j)$, where M is a symmetric semi-definite positive matrix. This is equivalent to finding a separating ellipsoid in the space of data differences. We model the probability that pair (i, j) is positive with :

$$p(y_i = y_j | x_i, x_j, M, b) = \sigma(b d_M(x_i, x_j)) \tag{6.1}$$

where $\sigma(z) = (1 + exp(z))^1$ is the sigmoid function and b a bias term (the optimal distance threshold). This is a standard logistic regression model (linear in M) on which we perform maximum likelihood estimation via projected gradient descent, to enforce convex constraints on M (diagonality, SDP, ...).

Bibliography

Simple Metrics

Radhey Shyam and Yogendra Narain Singh *Identifying Individuals using Multimodal Face Recognition Techniques*

Firouz Abdullah Al-Wassai and N.V. Kalyankar *The Classification Accuracy of Multiple-Metric Learning Algorithm on Multi-Sensor Fusion*

Large Margin Nearest Neighbor

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

Matthieu Guillaumin, Jakob Verbeek and Cordelia Schmid *Is that you? Metric learning approaches for face identification*

Information Theoretic Metric Learning

Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra and Inderjit S. Dhillon *Information-Theoretic Metric Learning*

Matthieu Guillaumin, Jakob Verbeek and Cordelia Schmid *Is that you? Metric learning approaches for face identification*

Quasi Cosine Similarity

Hieu V. Nguyen and Li Bai *Cosine Similarity Metric Learning for Face Verification*

Xiang Wu, Zhi-Guo Shi and Lei Liu *Quasi Cosine Similarity Metric Learning*

More Methods

Gal Chechik, Varun Sharma, Uri Shalit and Samy Bengio *Large Scale Online Learning of Image Similarity Through Ranking*

Matthieu Guillaumin, Jakob Verbeek and Cordelia Schmid *Is that you? Metric learning approaches for face identification*

List of Figures