Challenge MDI 343

Alexandre Rouxel

Data fusion algorithms - 5 février 2018

Notations

Let's call X the score matrix of face recognition algorithms, it is the features. And let 's call Y the decisions such that $Y = 1$ if the images pair belong to the same person and -1 otherwise. The problem to solve is equivalent to a classification problem. In fact, the goal is to find a threshold T such that the two images are considered to belong to the same person if :

$$\sum_{i=1}^{P} x_{ij}\theta_i \geqslant T,$$

and to different persons if

$$\sum_{i=1}^{P} x_{ij}\theta_i < T$$

This is équivalent to :

$$\sum_{i=0}^{P} x_{ij}\theta_i > 0$$

With $x_{0j} = 1$ and $\theta_0 = -T$. So we want to find $\hat{\theta}$ such that :

$$sign(X\hat{\theta}) = Y, y_i \in \{-1,1\}\ ,\ X = [1,x_1, x_2 \dots x_P], \qquad (1)$$

The fusion matrix, M defined in the provided notebook is built by reorganizing the estimated values of $\hat{\theta}$. The coefficient θ_i gives the weight set to the quadratique score built by a linear combination of at most two scores . In order to generate quadratique combinaison of features I use the scikitlearn method : *PolynomialFeatures()*, it generates 120 features from scores of 14 algorithms. The reordering of the coefficients from $\hat{\theta}$ to M is performed by the method *BuildM()*.

Data cleaning

The first step consists in cleaning the data. In fact, several lines of X contains negative and infinite values, these lines are dropped out, as the scores have to be positive.

Problem statement

Some constraints are added, $\hat{\theta}$ has to fill the running time requirements. Let i denote the running time of the algorithm i in milliseconds ($i = 1,\dots,14$). Then, there is a given threshold,T_r , such that the total running time of the combined algorithms does not exceed T_r :

$$\sum_{i \in C} t_i \leq T_r \qquad (2)$$

where $C \subset \{1,\dots,14\}$ is the set of algorithms combined. So each non nul component of $\hat{\theta}$ adds one or two elements in C. This contraint can be reformulated as a constraint on sparseness of $\hat{\theta}$.

Joint features selection and fusion optimization

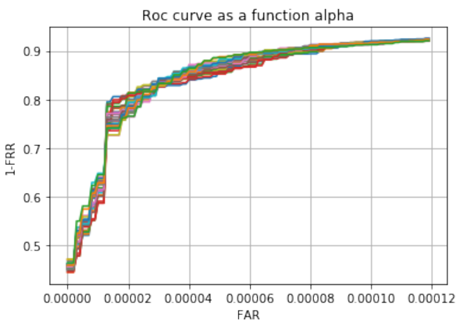
Since a null coefficient in $\hat{\theta}$ means that one algorithm combinaison is not used, to find a sparse $\hat{\theta}$ minimizing the classification error defined in (1), I optimise the following criteria :

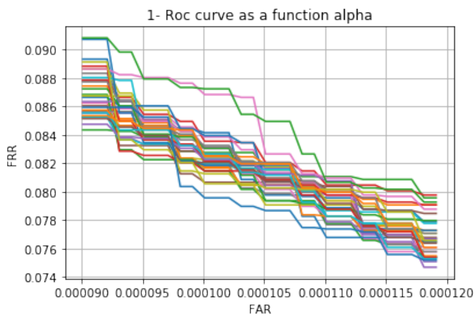
$$E(\theta,\alpha) = \frac{1}{n}\sum_{i=1}^{n}(1 - y_i(\theta^T x_i))_+ + \alpha \ \| \ \theta \ \| \ _1 \text{, with } (z)_+ = max(0,z) \quad (3)$$

I use the Scikitlearn build in method : *SGDClassifier()* to optimize $E(\theta,\alpha)$. The parameter α is used to tune the sparseness of $\hat{\theta}$. I tune the hyper-parameter α adaptively either by measuring the FRR at 0.01 % FAR or by optimizing Area Under ROC criteria with a grid search criteria.

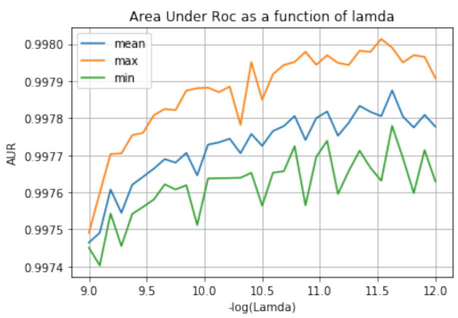
Tuning of α by Area Under ROC curves optimisation

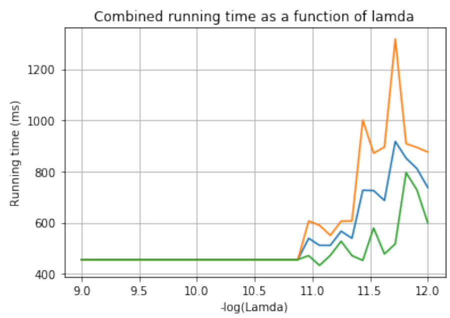
Roc curves are defined by the True Positive Rate,TPR, as a function of False Positive Rate, FPR with our notation, FPR = 1 - FRR and FAR= FPR. The Figure 1 shows the Roc Curve and Figure 2 shows a zoom in the area of interest with FRR instead of FPR.





I use the Area Under Roc curve to optimize my parameter α. It is easier to compute and to plot as a function of α.





Those two figures give the optimal α around $\alpha \approx exp(-10.75)$. The index of the selected algorithms are [8 10 12 14] the computation time is 456 ms. Once I have my optimal subset of algorithms I run a new optimization on the same criteria (3) but now with a constrain on norm L2 and not L1 as the spareness on $\hat{\theta}$ is no more required.

$$E(\theta,\alpha) = \frac{1}{n}\sum_{i=1}^{n}(1 - y_i(\theta^T x_i))_+ + \alpha \ \| \ \theta \ \| \ _2^2 \qquad (4)$$

The criteria (3) and (4) are based on Hinge metric which is a convex approximation of the accuracy but unfortunately we want to optimize the FRR at FAR = 1e-4 which is not a convex criteria. That's why I can not guaranty that the minimal is reached with a such algorithm. The submission of this approach gives a score around 0.07.

PCA based threshold tuning

I developed an algorithm based on PCA. The goal is to find θ maximizing the dot product with scores when $Y = 1$, and minimizing it when $Y = 0$. To do so I define a matrix Z_β built with X such that :

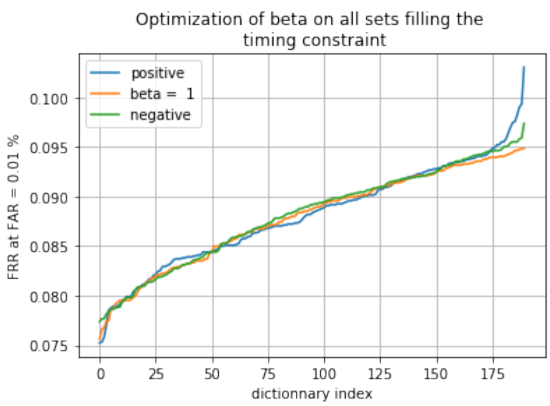
$$z^{\beta}_{i,j} = x_{i,j} \text{ if } y_j = 1$$
$$z^{\beta}_{i,j} = -\beta x_{i,j} \text{ if } y_j = 0$$

β is a parameter to tune the AUR, if $\beta = 0$, I minimize FRR, and if $\beta \gg 1$, I minimize the FAR, the algorithm will then maximize $J(\theta, \beta)$:

$$J(\theta, \beta) = \left\| Z_\beta \theta \right\|_2^2, \ \theta \text{ fills the timing contraint defined by } (2)$$

The nice property is that θ that maximizes $J(\theta, \beta)$ is the principal eigen vector of $R_\beta = Z_\beta Z_\beta^T$. By principal eigen vector I mean eigen vector associated to the greatest eigen value. To avoid the constraint on the sparseness of θ, I use a systematic search approach.

1. *Generate a dictionary D, such that the running time of each set d, $d \in D$, fills the timing constraint*
2. *For each $d \in D$*
For each β
Compute $\hat{\theta}_\beta =$ principal eigen vector of $R_\beta = Z_\beta Z_\beta^T$ on a training set
Compute $FRR(\hat{\theta}_\beta)$ on test set
3. *Choose* $\hat{\theta}_\beta = \mathbf{ArgMin} FRR(\hat{\theta}_\beta)$



The best set of algorithms found is [8,9,10,11,14] with $\beta = 1$
The submission score of this algorithm is around 0.065.