# Fundamentals in Statistical Pattern Recognition

Project Report

June 10th, 2015

Student:

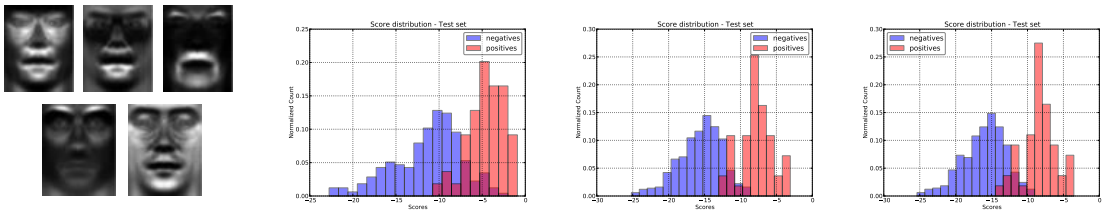
Ana Lucia Ureche

# 1. Introduction

The aim of this project is to develop a face recognition tool, capable of distinguishing a person from a set of existing users. The system is tested on the AT&T database of facial images. The approached is based on three steps: (1) creating a background model, (2) enrolling a new user and developing an individual model based on the background knowledge, (3) probing a new user by comparing the set of images with the models already enrolled in the database.

**Limitations:** The problem is formulated in a restrictive manner. There are two major limitations that turn the apparent nature of multi-class classification into a problem of discriminating one particular user from the rest, in a one-to-one comparison.
1. The training set is not presented in classes. However when applying LDA we assume that each class has exactly 10 images.
2. For the enrollment procedure the data is never presented as a whole set, but one chunk at a time.

**Assumptions:** We formulate the following assumptions:
1. The Training Set, Development Set and Testing Set are disjunctive
2. The background model serves as ground truth for human characteristics.
3. The training set used for obtaining the background model is generic enough.




**(a)** first 5 eigen faces **(b)** 5 principal components **(c)** 100 principal components **(d)** 1000 principal comp.

**Figure 1:** Recognition rates on the test set using the originally provided code (i.e. using only PCA), but varying the number of eigen vectors kept. Increasing the number of principal components does not improve significantly the results.

# 2. Approach

We take an iterative approach for analyzing the problem and developing a solution. We test multiple hypothesis, starting from the limitations and assumptions mentioned above. The implementation is based on existing source code which provided the functionality of performing PCA on a pixel-level on the existing images.

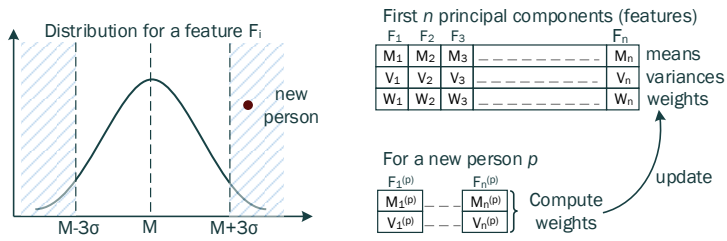
---

**Hypothesis 1.** There are specific characteristics of a person that differentiate it from the group. These represent dominant features that are far away from the group mean, and repeatable (or stable) features that have a very small variance compared to the average group variance.

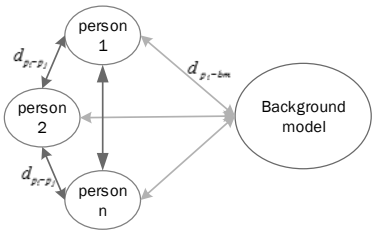
---

The first eigen components retained when computing the background model (which also retain the highest percent of variance), represent common features of the human faces. As seen in Figure 1 they mostly retain information about the shape of the face, mouth, eyebrows and nose. However when enrolling a new individual, projecting the data onto the same components, gives a basis for comparison, but it does not help in discriminating how different an individual is from the rest of the "crowd".

Some components, even if they are determined as important, may actually represent "noisy features". For example many photos are taken from different angles, so the orientation of the face might be extracted as a feature, but it is not a useful feature in differentiating a person from another. However if a person




**(a)** updating feature weights based on a person's characteristics




**(b)** the distance $d_{p_i - bm}$ increases, but not $d_{p_i - p_j}$ (between individuals)

**Figure 2:** Illustration of Hypothesis 1

**Algorithm 1** Weighted PCA

**Background model**()
*compute PCA projections*
*compute global feature_means, feature_sigmas*
**end**

**Enroll**(scaling factors $s$, feature gains $\theta$)
*do*
*init feature_weights = 1*
*compute local_features (pca projections)*
*compute local_means, local_sigmas*
**if** $abs(local\_means - feature\_means) > feature\_sigmas/s$ and $(local\_sigma - feature\_sigma) < local\_sigmas/s$ **then**
% Increase confidence in feature
$feature\_weights = feature\_weights * \theta$ *% Increase confidence in feature*
**Else** $feature\_weights = feature\_weights/\theta$ *% Decrease confidence in feature*
**end if**
*normalize and save weights*
**end**

**Probe**()
*load model and weights*
*compute projections and difference to model*
*scale output using the feature_weights*
**end**

typically wears glasses, or has a mole, this feature might help in distinguishing him from the rest. So in the first part of this work, we try to enhance such individual features.

So therefore we hypothesise that keeping more principal components might be beneficial, however depending on the person we should trust some features more than others. The hypothesis 1 is summarized in Figure 2, and the experiment for testing it is explained in Algorithm 1:

**Parameters:** There are 3 main parameters:
(1) $n_{pc}$ – the number of principal components kept after performing PCA
(2) $s$ – the scaling factor used for determining a confidence interval around the means
(3) $\theta$ – the gain for scaling the feature weights
The final weights obtained for each person, are normalized, to give a common basis for comparison.

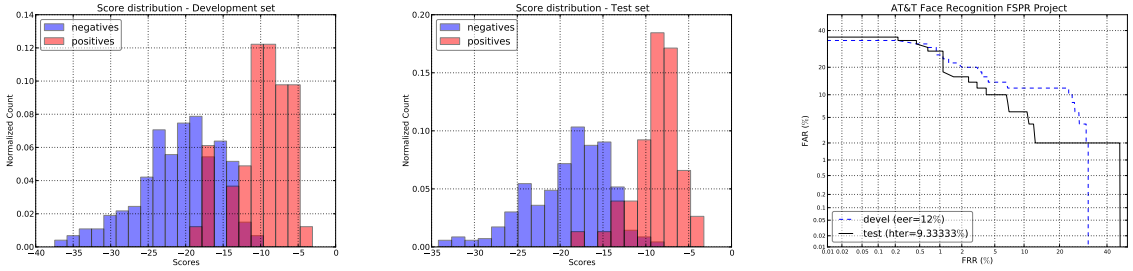
We also tested various possible ways of updating the feature weights:
(a) $feature\_weights = feature\_weights * \theta$ – a fixed boosting factor
(b) $feature\_weights = feature\_sigmas/local\_sigma$
(c) $feature\_weights = 1/local\_sigma$
The options (b) and (c) are based on the following observation of attributing "trust" to a variable:

| *local_variance* | *global_variance* | *trust_in_variable* |
|---|---|---|
| *high* | *low* | *very_low* |
| *low* | *high* | *very_high* |
| *high* | *high* | *no_change* |
| *low* | *low* | *no_change* |

**Discussion:** Figures 3 and 4 present comparative results between using the fixed and varying weight feature scaling. While better results seem to have been achieved by using the varying weights, still performing grid search can determine good values for $s$ and $\theta$ such that results become similar (less than 1% difference in the error rate). In Table 1 we present comparative results, for this and the methods used in the remaining of this report.

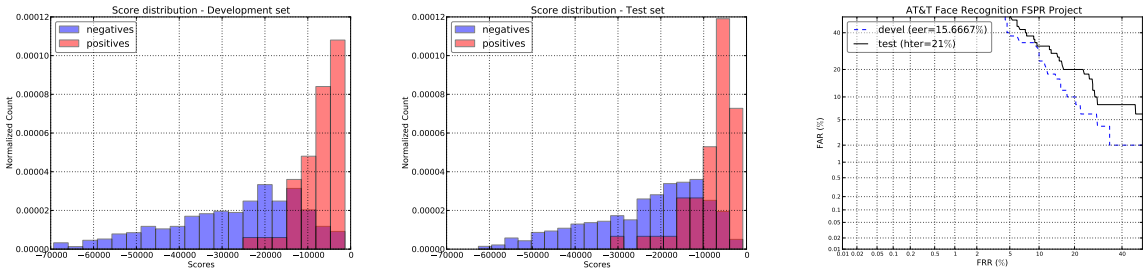
The proposed method of scaling the features achieves an increase in the distance $d_{p_i - bm}$ between a person $p_i$ and the background model $bm$, but it does not insure an increase in the distance between individuals $d_{p_i - p_j}$, see Figure 2(b).

We chose to compare the proposed method of scaling the feature weights, with performing LDA after projecting the data using PCA. This method gives slightly better results, however it makes use of additional information (i.e. the class labels for the training set). This is due to the fact that when projecting the data using LDA we make the assumption that every class has exactly 10 images, even if the background_model function, never receives labeled data.




**(a)** development set scores **(b)** test set scores **(c)** FRR vs. FAR

**Figure 3:** Results obtained from the pca + fixed weights feature scaling method. Parameters used: $s = 2$; $\theta = 2$

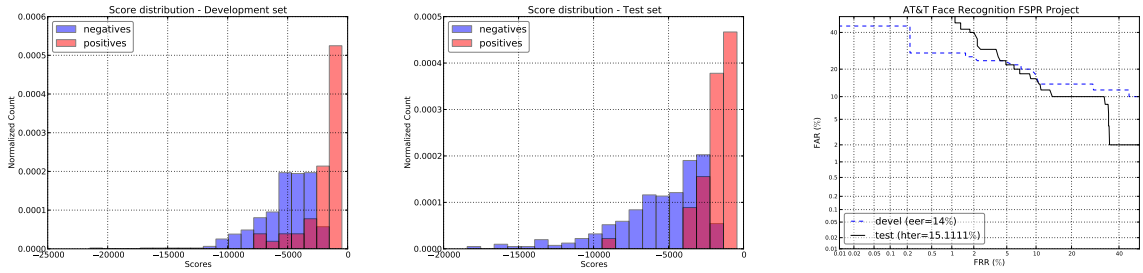



**(a)** development set scores **(b)** test set scores **(c)** FRR vs. FAR

**Figure 4:** Results obtained from the PCA and varying feature weights update method ($feature\_weights = 1/local\_sigma$)

**Hypothesis 2.** Further encoding the obtained features for each user in a GMM and computing the log–likelihood of a new input image to each model might improve the recognition rates.

We constructed a GMM model, trained with Expectation Maximization (EM), for each user, based on the extracted PCA features. We tested the performance with and without feature scaling. Am additional




**(a)** development set scores **(b)** test set scores **(c)** FRR vs. FAR

**Figure 5:** Results obtained when using fixed weights and training a GMM model for each individual

parameter was added with respect to the previous hypothesis, $mc$, the number of mixture components. Increasing $ms$ did not aim an improvement in the results, therefore we only report scores for $mc = 1$. In figure 5 we present results obtained for GMM and fixed weights.

Results are shown in Table 1. Performing GMM on the PCA projections only decreased the number of false recognition rates to half, when compared to simple PCA, however it increased the total error rate by almost twice. This happened because the false acceptance rate increased.

Performing GMM after feature scaling in both fixed and various weights case, decreased the total error rate and the false acceptance rate, compared to the previous case.

| Measure | FAR [%] | FRR [%] | HTER [%] |
|---|---|---|---|
| PCA | 6 | 16 | 11 |
| PCA + fixed weights | 15.33 | 2 | 8.66 |
| PCA + varying weights | 9.77 | 8 | 8.88 |
| PCA + LDA | 10.22 | 2 | 6.11 |
| PCA + fixed weights + LDA | 16.66 | 0 | 8.33 |
| PCA + varying weights + LDA | 21.77 | 12 | 16.88 |
| PCA + GMM | 34 | 8 | 21.77 |
| PCA + fixed weights + GMM | 20.22 | 10 | 15.11 |
| PCA + varying weights + GMM | 15.77 | 10 | 12.88 |

**Table 1:** Comparative results for hypothesis 1 and hypothesis 2 on the test set, using $n_{pc} = 100$, $s = 2$, $\theta = 2$, $mc = 1$. In the case of varying weights, the following formula was used: $feature\_weights = 1/local\_sigma$.

# 3. Grid search for determining parameters

We performed automatic parameters estimation, by running grid search, on the PCA + fixed_weights setup. We estimated the number of principal components $n_{pc}$, the scaling factor (or threshold) $s$, and the gain (or boosting factor for scaling the feature weights) $\theta$. The parameter ranges scanned were:

| *parameter* | *min* | *max* | *step* |
|---|---|---|---|
| $n_{pc}$ | 5 | 100 | 5 |
| $s$ | 2 | 10 | 0.5 |
| $\theta$ | 2 | 10 | 0.5 |

# 4. Instructions to execute the code

Multiple files project_*.py are provided. They contain the separate functionality of each of the tested feature, as documented in Table 1 If run individually, they will reproduce the results shown in the table.

*project_pca_weights.py* – performs feature scaling. Parameters to be set: $n_{pc}$, $s$, $\theta$. Also fixed weights vs. varying weights.

*project_pca_lda.py* – contains the implementation for applying LDA to the PCA projections. A different LDA model is saved for each user in addition to the PCA model. Parameters to be set: $n_{pc}$

*project_pca_lda_weights.py* – in addition to the file above, this one also uses feature scaling. Parameters to be set: $n_{pc}$, $s$, $\theta$. Also fixed weights vs. varying weights.

*project_pca_gmm.py* – contains the implementation for applying GMM to the PCA projections. A different GMM model is saved for each user in addition to the PCA model. Parameters: $n_{pc}$, $mc$

*project_pca_gmm_weights.py* – additionally to the file above it also contains the functionality for computing the fixed and various feature scaling.

The folder grid_search contains the programs for automatically estimating the parameters. Execute runner.py. The file "results" contains the results of running the grid search in the following format, on columns: number of PCA components, threshold $s$, boost gain $\theta$, dev_far (Development False Accept Rate), dev_frr (Development False Reject Rate), eer (Equal Error Rate), test_far (Test False Accept Rate), test_frr (Test False Reject Rate), hter (Half-total Error Rate).

The folder tmp currently contains the output from running the *project_pca_weights.py* script. In case of running the variants with LDA orGMM additional files will be generated containing those models for each newly enrolled person.

# 5. Conclusions

In this project we tested 2 different hypotheses for performing face verification. We used 3 different machine learning techniques (PCA, LDA and GMM)and tested different parameter values.

Overall the results tend to sustain hypothesis 1 (claiming that an emphasis should be put on features that separate individuals from the group). However the results do not tend to support hypothesis 2 claiming that performing GMM will improve the results.

In conclusion while the proposed method increased the distance between a particular user and a generic model, still a better method should be designed for increasing the distance between users. This would increase the chances of correct face detection.