MACHINE LEARNING AND PATTERN RECOGNITION REPORT

Fingerprint Spoofing Detection

Antonio Iorio
Who?
Where?
When?

Contents

Introduction The project task consists of a binary classification problem. The goal is to perform fingerprint spoofing detection, i.e. to identify genuine vs counterfeit fingerprint images. The dataset consists of labeled samples corresponding to the genuine (True, label 1) class and the fake (False, label 0) class. The samples are computed by a feature extractor that summarizes high-level characteristics of a fingerprint image. The data is 6-dimensional.

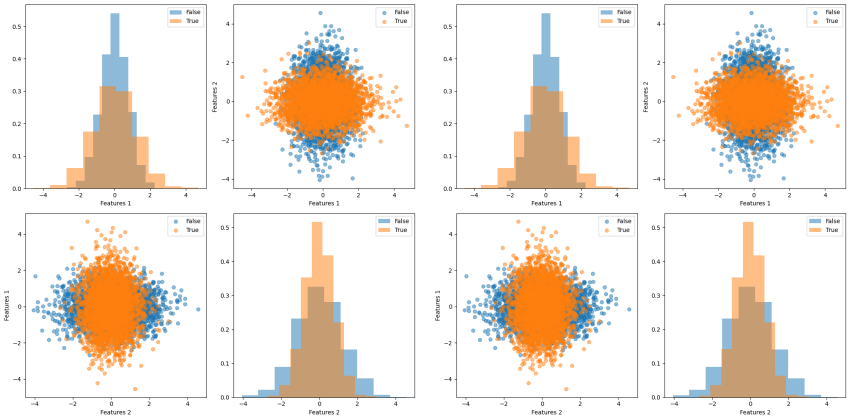
1 Dataset Analysis

In our analysis process, one first begins to represent what are the data related to the various features and how among the various features the data are distributed by making a visual representation in pairs of features.

1. Starting with the analysis of the first two features and creating a histogram and a scatter Figure 1, one can see:

   - Both features overlap
   - Follow a Gaussian distribution
   - Feature 1 has a peak at [-0.213, 0.276] and it is worth 0.541 for the false class, instead for Feature 2 the peak at [-0.402, 0.165] and it is worth 0.516 for the true class.

   Seeing Figure 1 again, it would appear that Figure 1a and Figure 1b appear to be visually the same but in b they are represented centred which as can be seen is quite similar because Feature 1 has $\mu = 0.00170711$ and $\sigma^2 = 1.00134304$, instead Feature 2 has $\mu = 0.00503903$ and $\sigma^2 = 0.9983527$



(a) Without mean (b) With mean

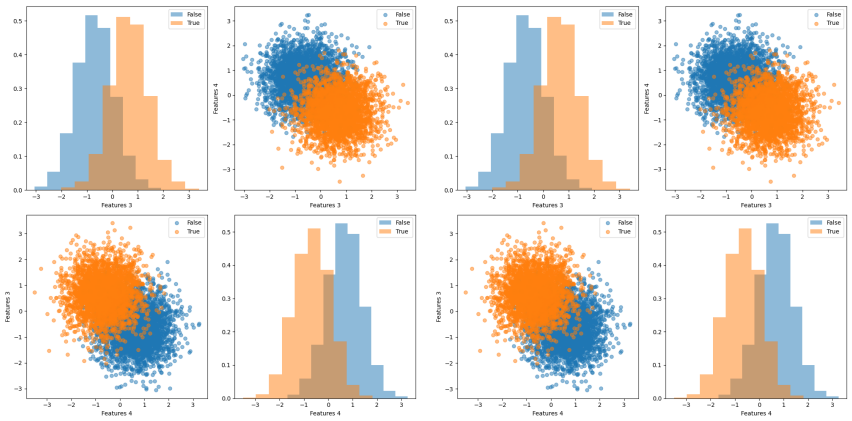
Figure 1: Feature 1 vs Feature 2 - Without centering the data relative to the average (a) and with centering the data relative to the average (b)

2. For features 3 and 4, observed in Figure 2, on the other hand, they have:

   - do not overlap like the previous two
   - Follow a Gaussian distribution but the true and false labels are centred at different points

- Feature 3 has a peak at [-1.063, -0.568] and it is worth 0.517 for the false class, instead for Feature 4 the peak at [0.290, 0.783] and it is worth 0.525 for the false class.

Seeing Figure 2a and Figure 2b, data are already similar because, the mean calculated with reference to the two classes is close to 0, in fact: Feature 3 has $\mu = -0.00560753$ and $\sigma^2 = 1.0024818$, instead Feature 4 has $\mu = 0.00109537$ and $\sigma^2 = 0.99029389$



(a) Without mean (b) With mean

Figure 2: Feature 3 vs Feature 4 - Without centering the data relative to the average (a) and with centering the data relative to the average (b)

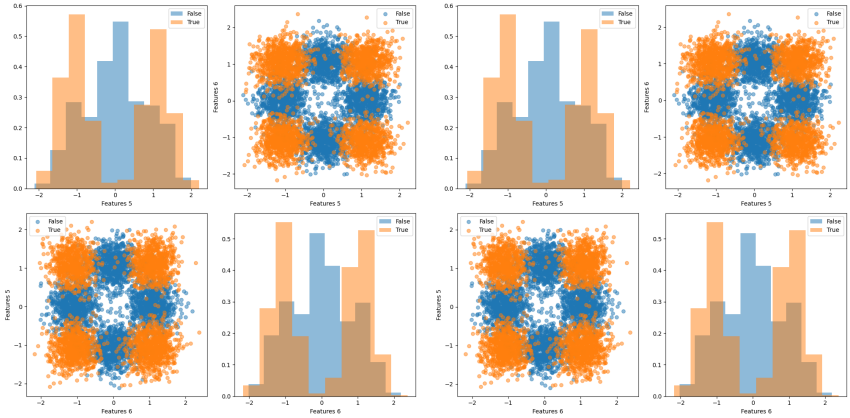
3. For features 5 and 6, observed in Figure 3, one can see:

   - Do not totally overlap
   - For both features, the true labels don't follow a Gaussian distribution as opposed to the false ones, which could be more approximate
   - Feature 5 has a peak at [-1.211, -0.783] and it is worth 0.572 for the true class, instead for Feature 6 has a peak at [-1.273, -0.817] and it is worth 0.553 for the true class.

   Also in this other case, Figure 3a and Figure 3b are similar because again the average is close to 0. Feature 5 has $\mu = -0.00700025$ and Feature 6 has $\mu = 0.00910515$

2 Dimensionality Reduction

Before proceeding with classification, two techniques of dimensionality reduction PCA and LDA can be analysed. The goal is to find a subspace of the feature space that preserves most of the useful information, that is, mapping from the *n–dimensional* feature space to *m–dimensional* space, with $m \ll n$



(a) Without mean (b) With mean

Figure 3: Feature 5 vs Feature 6 - Without centering the data relative to the average (a) and with centering the data relative to the average (b)

2.1 PCA

is an unsupervised technique. Where starting from a dataset $X = \{x_1, \ldots, x_k\}$ and calculated average. It starts with the empirical covariance matrix:

$$C = \frac{1}{K} \sum (x_i - \bar{x})(x_i - \bar{x})^T \tag{1}$$

We compute the eigen-decomposition of $C = U \Sigma U^T$ and project the data in the subspace spanned by the m columns of U corresponding to the m largest eigenvalues.

$$y_i = P^T (x_i - \bar{x}) \tag{2}$$

where P is the matrix of the m columns of U associated to the m highest eigenvalues of C. A cross-validation approach can be used to figure out the optimal value of m to be selected. To evaluate each eigenvalue, one would have to calculate the variance corresponding to the axis. The percentage can be calculated as the rate between the sum of the m eigenvalues and the sum of all of them. In Figure 4 we can see how it changes in the project. A good m, corresponds to that value which allows a percentage greater than 95%, so in our case we would need all 6 features.

2.2 LDA

is a supervised technique. To find a direction that has the best separation between classes, we measure spread between classes in terms of class covariance. The objective is to maximize the $between - class$ variability over $within - class$ variability ratio for the transformed samples:

$$\max_w \frac{w^T S_B w}{w^T S_W w} \tag{3}$$

where:

$$S_B \triangleq \frac{1}{N} \sum_{c=1}^{K} n_c (\mu_c - \mu)(\mu_c - \mu)^T \tag{4}$$

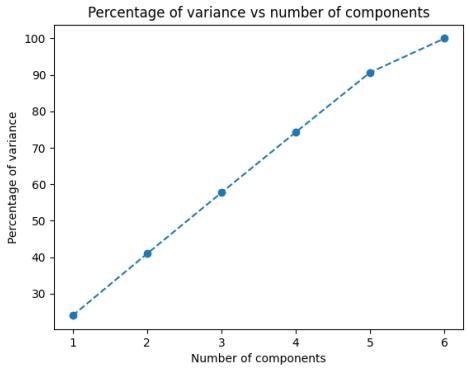


Figure 4: Cross validation for PCA impact evaluation

$$S_W \triangleq \frac{1}{N} \sum_{c=1}^{K} \sum_{i=1}^{n_c} (x_{c,i} - \mu_c)(x_{c,i} - \mu_c)^T \tag{5}$$

μ is dataset mean μ_c is class mean

2.3 Our project

PCA and LDA are applied to the dataset, in particular, m = 6 is used, and in Figure 5 we can observe what are the outcomes for the indicated directions.

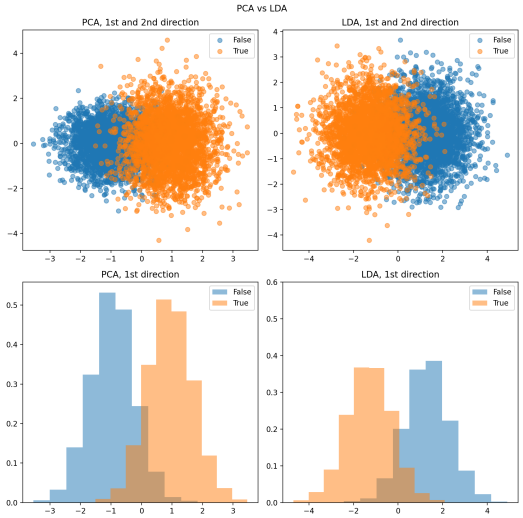


Figure 5: Comparing resulta between PCA and LDA

At a later stage, they were used to carry out a classification. The available dataset was divided into two sub-portions one for training and the other for validation. In Table 1 we can see the error of the classification, errors made in the classification when varying m (only for some values) and the threshold were reported

Method	Num Samples	Error	Error Rate
LDA - First threshold	2000	186	9.30%
LDA - Second threshold	2000	186	9.30%
PCA (m=5) + LDA - First threshold	2000	186	9.30%
PCA (m=5) + LDA - Second threshold	2000	185	9.25%
PCA (m=6) + LDA - First threshold	2000	186	9.30%
PCA (m=6) + LDA - Second threshold	2000	184	9.20%

Table 1: Table showing the results of the LDA and PCA + LDA method for classification.

3 Multivariate Gaussian Density

Multivariate Gaussian Density is an extension of the Gaussian Density to multiple dimensions. It is used to describe the distribution of a vector of random variables in a $multi-dimensional$ space, and it could be defined as:

$$N(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{M}{2}} |\Sigma|^{\frac{1}{2}}} \exp^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{6}$$

where M is the size of the feature vector x, and $|\Sigma|$ is the determinant of Σ. Since the computation of the exponential could cause problems, the logarithm is applied, so from Equation 6 we get Equation 7:

$$\log N(x \mid \mu, \Sigma) = -\frac{M}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) \tag{7}$$

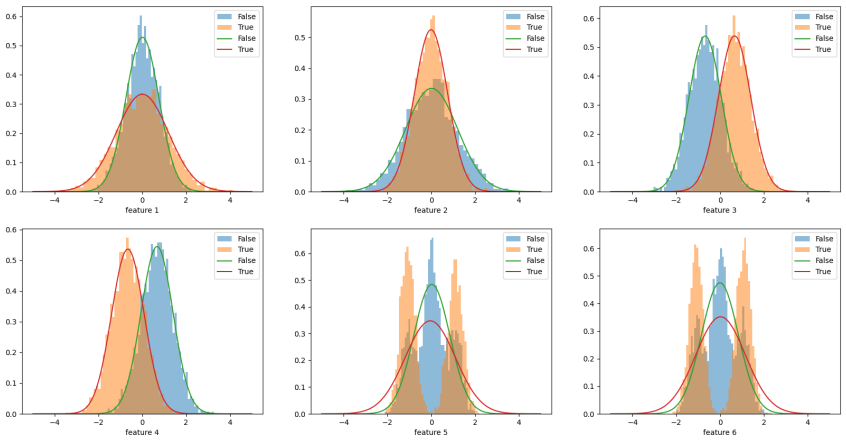


Figure 6: Gaussian Density

Thus, applying Gaussian probability density to the 6 features in our dataset, the following results in Figure 6, are obteinded.
It can observe that **features 1 - 2 - 3 - 4** fit the Gaussian distribution, the histogram of the data is well approximated by the Gaussian curve. For **features 5 - 6** this does not happen and therefore could not be a good model.

4 Classification Models Analysis

To perform the classification, the dataset must first be divided into two sub-portions, the training and validation sub-portions.

4.1 Gaussian models

Since, it is dealing with a binary classification task, it will assign a probabilistic score to each sample in terms of the class-posterior log-ratio:

$$\log r(x_t) = \log \frac{P(C = h_1 \mid x_t)}{P(C = h_0 \mid x_t)} \tag{8}$$

Analysing Equation 8 in more detail, it becomes:

$$\log r(x_t) = \log \frac{f_{X|C}(x_t \mid h_1)}{f_{X|C}(x_t \mid h_0)} + \log \frac{P(C = h_1)}{P(C = h_0)} \tag{9}$$

The first addend of the equation is called the *llr* or *log-likelihood ratio* and an optimal decision is given by Equation 10.

$$\log r(x_t) \gtrless 0 \tag{10}$$

Considering $P(C = h_1) = \pi$ and $P(C = h_0) = 1 - \pi$, from Equation 9 and Equation 10, it is possible to write that the class assignment is based on Equation 9 and Equation 10, to obtain Equation 11.

$$llr(x_t) = \log \frac{f_{X|C}(x_t \mid h_1)}{f_{X|C}(x_t \mid h_0)} \gtrless -\log \frac{\pi}{1 - \pi} \tag{11}$$

4.1.1 Multivariate Gaussian Classifier

The first classifier is given by the empirical mean and covariance of each class,

$$\mu_c^* = \frac{1}{N_c} \sum_{i|c_i=c} x_i \ , \quad \Sigma_c^* = \frac{1}{N_c} \sum_{i|c_i=c} (x_i - \mu_c^*)(x_i - \mu_c^*)^T \tag{12}$$

4.1.2 Naive Bayes Gaussian Classifier

This model makes an important assumption that simplifies the number of parameters to be estimated, it assumes that the features are independent given their class. This causes the covariance matrix to be a diagonal matrix, consequently, matching MVG with the diagonal covariance matrix. However, the assumption of independence may be too restrictive and lead to inferior performance if the features are indeed correlated.

$$\mu_{c,[j]}^* = \frac{1}{N_c} \sum_{i|c_i=c} x_{i,[j]} \ , \quad \sigma_{c,[j]}^2 = \frac{1}{N_c} \sum_{i|c_i=c} (x_{i,[j]} - \mu_{c,[j]}^*)^2 \tag{13}$$

Features	Model	Error Rate (%)
	no PCA	
1 to 6	MVG	7.00
1 to 6	Naive Bayes	7.20
1 to 6	Tied Covariance	9.30
1 to 4	MVG	7.95
1 to 4	Naive Bayes	7.65
1 to 4	Tied Covariance	9.50
1 - 2	MVG	36.50
1 - 2	Naive Bayes	36.30
1 - 2	Tied Covariance	49.45
3 - 4	MVG	9.45
3 - 4	Naive Bayes	9.45
3 - 4	Tied Covariance	9.40
	PCA m = 5	
1 to 6	MVG	7.10
1 to 6	Naive Bayes	8.75
1 to 6	Tied Covariance	9.30
	PCA m = 6	
1 to 6	MVG	7.00
1 to 6	Naive Bayes	8.90
1 to 6	Tied Covariance	9.30

Table 2: Table showing the results of the Error Rate for different Models and Features.

4.1.3 Tied Covariance Gaussian Classifier

The assumption of the latter model consists of its own average for each class, but an equal covariance matrix for all classes.

$$\mu_c^* = \frac{1}{N_c} \sum_{i|c_i=c} x_i \ , \quad \Sigma^* = \frac{1}{N} \sum_c \sum_{i|c_i=c} (x_i - \mu_c)(x_i - \mu_c)^T \tag{14}$$

4.1.4 Gaussian Models Comparison

A threshold of 0 was used to perform our results, which means that $P(C = 1) = P(C = 0) = 1/2$. This model was applied and the outcomes can be seen in the Table 2.

If we go into the details of how the error rate changes as a function of the observed features, we can see that:

- **1 to 6**: in the case we consider all 6 features, the error rate is quite low and its range goes from 7.00% to 9.30%. This means that they all provide useful information.
- **1 to 4**: if we consider features from 1 to 4, we can see that the error rate increases slightly. This allow us to say that features 5 and 6 have useful but not fundamental information to change the outcome.
- **1 - 2**: features 1 and 2 have a rather high error rate, meaning that they don't contain relevant information.

- **3 - 4**: on the other hand, the latter two features considered have a rather low error rate, a value close to the case where all features are considered. This means that the information contained in these two features is relevant for making the classification.

Starting with the previous performance, it may be interesting to analyse how performance changes as three main parameters vary:

- $\tilde{\pi}$
- C_{fn}
- C_{fp}

	MVG	**Naive Bayes**	**Tied Covariance**
	Application $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.5, 1, 1)$		
actDCF	0.1399	0.1439	0.1860
minDCF	0.1302	0.1311	0.1812
	Application $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.9, 1, 1)$		
actDCF	0.4001	0.3893	0.4626
minDCF	0.3423	0.3509	0.4421
	Application $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.1, 1, 1)$		
actDCF	0.3051	0.3022	0.4061
minDCF	0.2629	0.2569	0.3628
	Application $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.5, 1, 9)$		
actDCF	0.3051	0.3022	0.4061
minDCF	0.2629	0.2569	0.3628
	Application $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.5, 9, 1)$		
actDCF	0.4001	0.3893	0.4626
minDCF	0.3423	0.3509	0.4421

Table 3: Table showing minDCF and actDCF for different models and application.

	MVG	Naive Bayes	Tied Covariance
Application $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.5, 1, 1)$			
no PCA			
actDCF	0.1399	0.1439	0.1860
minDCF	0.1302	0.1311	0.1812
PCA			
$m = 5$			
actDCF	0.1419	0.1749	0.1860
minDCF	0.1331	0.1737	0.1812
$m = 6$			
actDCF	0.1399	0.1780	0.1860
minDCF	0.1302	0.1727	0.1812
Application $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.9, 1, 1)$			
no PCA			
actDCF	0.4001	0.3893	0.4626
minDCF	0.3423	0.3509	0.4421
PCA			
$m = 5$			
actDCF	0.3980	0.4660	0.4626
minDCF	0.3512	0.4340	0.4451
$m = 6$			
actDCF	0.4001	0.4512	0.4626
minDCF	0.3423	0.4359	0.4421
Application $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.1, 1, 1)$			
no PCA			
actDCF	0.3051	0.3022	0.4061
minDCF	0.2629	0.2569	0.3628
PCA			
$m = 5$			
actDCF	0.3042	0.3930	0.4051
minDCF	0.2738	0.3545	0.3648
$m = 6$			
actDCF	0.3051	0.3920	0.4061
minDCF	0.2629	0.3535	0.3628

Table 4: Show minDCF and actDCF for different models and applications before and after applying PCA.