REPORT MACHINE LEARNING AND PATTERN RECOGNITION

Fingerprint spoofing detection

Alessia Intini
s309895
Politecnico di Torino
2023/24

MLPR Project:
Fingerprint spoofing detection

Contents

MLPR Project:
Fingerprint spoofing detection

The goal of project is to perform a binary classification on fingerprint spoofing detection, that is 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 data is 6-dimensional.

1 Dataset Analysis

1.1 Training and evaluation sets

The datasets provided contain 6000 samples; the first 6 values in each row represent the features, while the last value is the label. Specifically they are:

- Training Set: 2990 samples beloging to the Fake class (label 0) and 3010 samples belonging to the Genuine class (label 1)
- Evaluation Set: 3010 samples beloging to the Fake class (label 0) and 2990 samples belonging to the Genuine class (label 1)

We will use the Training Set to perform all the analysis, during this phase the dataset is divided to use about 60% of it as the training dataset and the remaining 40% for validation. After this step, the most promising models were chosen and the evaluation dataset was used to evaluate them and make the final considerations.

1.2 Features analysis

We can start to analize all features, here is the graph of each feature, through these histograms it is shown how each feature is distributed:

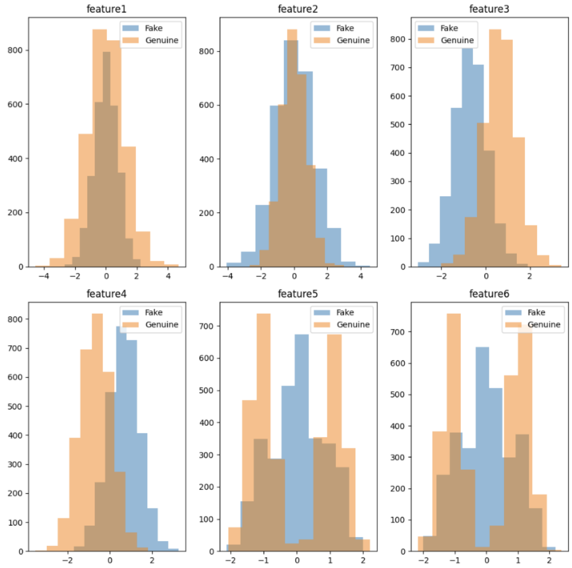

Figure 1: All features

By plotting a histogram for each feature, separately for the Fake and Genuine classes, it is possible to show whether the samples follow a Gaussian distribution and to what extent. Thus, we can say that some features follow a Gaussian distribution. Furthermore, it can be seen that features 4 and 5 may be the most discriminating features based on their ability to separate the data of the two classes.

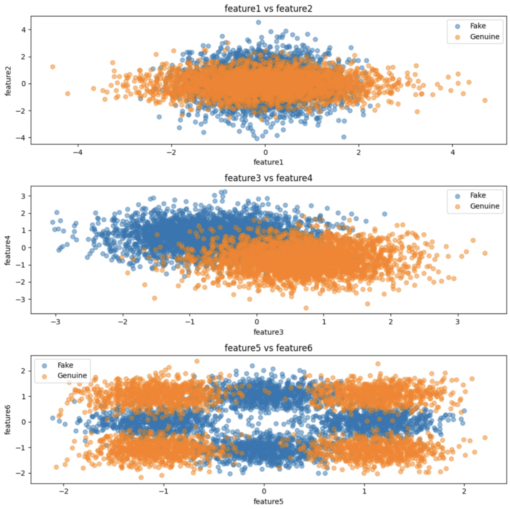

Figure 2: Distribution of feature pairs

Feature	Mean	Variance
1	0.00170711	1.00134304
2	0.00503903	0.9983527
3	-0.00560753	1.0024818
4	0.00109537	0.99029389
5	-0.00700025	1.00119747
6	0.00910515	0.99722374

Figure 3: Mean and variance for each features

After making these calculations we can conclude that the data are already centered since all features have mean close to zero.
Analyzing the pairwise averages and variances of the features, we can make some observations:

- Feature 1 and Feature 2: These two features have averages very close to zero, indicating that their values are similarly distributed around zero. However, the variance of Feature 2 is slightly lower than that of Feature 1, indicating that the values of Feature 2 are slightly less dispersed than those of Feature 1.

- Feature 3 and Feature 4: Feature 3 has a negative mean, while Feature 4 has a positive mean. This might indicate that the values of Feature 3 tend to be lower than those of Feature 4. Also, the variance of Feature 3 is slightly higher than that of Feature 4, indicating that the values of Feature 3 are more dispersed than those of Feature 4.

- Feature 5 and Feature 6: Feature 5 has a negative mean, while Feature 6 has a positive mean. This might indicate that the values of Feature 5 tend to be lower than those of Feature 6. Also, the variance of Feature 5 is slightly higher than that of Feature 6, indicating that the values of Feature 5 are more dispersed than those of Feature 6.

2 Dimensionality reduction

Dimensionality reduction is useful in this context to compute a mapping from an n-dimensional feature space to an m-dimensional space; it is applied because it can compress information, remove noise and simplify classification.

2.1 PCA

PCA is an unsupervised dimensionality reduction technique that, given a centered dataset $X = \{x_1, \dots, x_k\}$, it aims to find the subspace of R^n that allows to preserve most of the information, i.e. the directions with the highest variance. We start by calculating the covariance matrix:

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

After that we can compute the eigen-decomposition of $\mathbf{C} = \mathbf{U}\Sigma\mathbf{U}^T$, where $\mathbf{U}$ is the matrix of eigenvectors and Σ is the diagonal matrix of eigenvalues. Now we can project the data into the new subspace, it is spanned by the m columns of U corresponding to the m values of the highest eigenvalues.

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

P is the matrix corresponding to the m columns of U associated with the m highest eigenvalues of C. At this point to choose which is the best value of m we need to check how much total variance in the data we are able to retain using different values of m. To do this evaluation, it was decided to represent on a graph the variance of the data as m increases, i.e., we exploited that each eigenvalue corresponds to the variance along the corresponding axis, and the eigenvalues are the elements of the diagonal of the matrix Σ . The percentage will be calculated as the ratio of the sum of the first m eigenvalues to the sum of all eigenvalues.

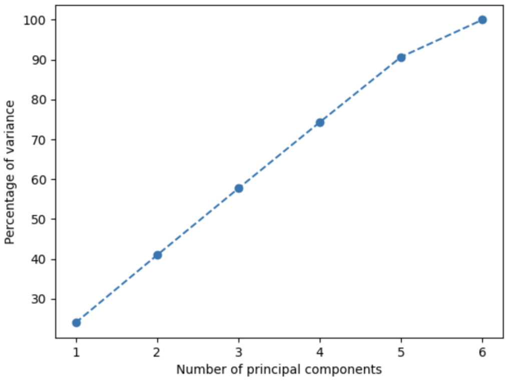

Figure 4: Variance of data for each m values

It is clear from Figure 4 that values of $m < 5$ rapidly decrease the amount of variance retained in the data. Removing a feature is probably the best choice, since this way the percentage of variance remains fairly high.

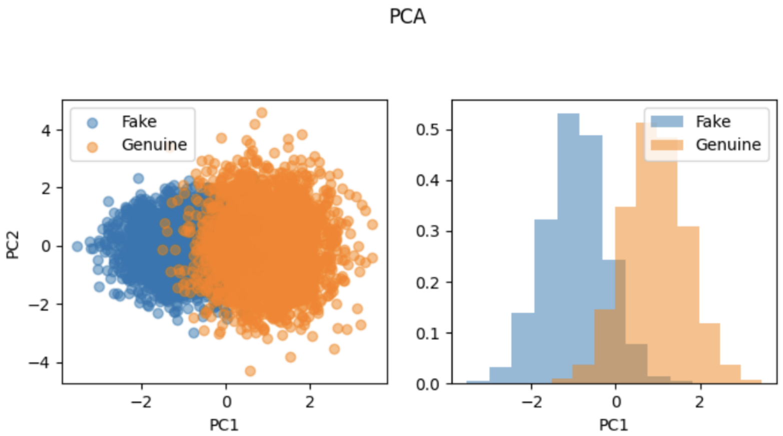

Figure 5: PCA for the two first components

After the analysis done to choose which m is best to use, we can represent the first two principal components found by calculating the PCA with m=6, visualizing these components can help to understand which features contribute most to the variance in the data. It can actually be observed that there is better separation between classes by applying PCA and choosing an m that has good variance of the data.

2.2 LDA

LDA is a supervised dimensionality reduction technique that aims to find the subspace that maximizes the separation between classes.
To compute the transformation matrix W we need to compute between-class and within-class covariance matrices.

$$\mathbf{S}_b = \frac{1}{N}\sum_{c=1}^{K} n_c(\mu_c - \mu)(\mu_c - \mu)^T \tag{3}$$

$$\mathbf{S}_w = \frac{1}{N}\sum_{c=1}^{K}\sum_{i=1}^{n_c}(x_{c,i} - \mu_i)(x_{c,i} - \mu_i)^T \tag{4}$$

where μ_c is the mean of the c-th class, μ is the mean of all the data, and n_c is the number of samples in the c-th class.
The LDA directions can be computed solving the generalized eigenvalue problem $S_b w = \lambda S_w w$, usually to solve the generalized eigenvalue problem we transform the S_w matrix into an identity matrix and then solve the eigenvalue problem for the other S_b matrix. This process allows us to find the discriminant vectors w that maximize the separation between the classes.
In LDA, the maximum number of discriminant directions or dimensions that can be obtained is always equal to C-1 where C is the number of classes. This is because the goal of the LDA is to find the directions that maximize the separation between the classes. In our case having only two classes the LDA can find at most one direction that maximizes the separation between these two classes.

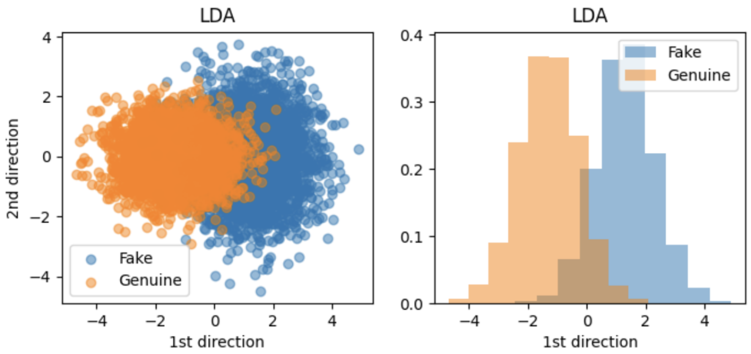

Figure 6: LDA for first direction

Looking at the results of LDA, in Figure 6, we can see that it cannot find an optimal direction that gives us a better separation of classes than the results shown previously.

2.3 Classification

Subsequent to applying dimensionality reduction techniques we used LDA to perform classification, in fact after finding the hyperplane on the training data and projecting the validation data into the new space we used different thresholds to perform classification. There are two different methods that have been used to calculate the threshold value:

- **First Threshold**: The first method calculates the average of the LDA values for the two classes and uses the average of these two averages as the threshold. This method is simpler, but does not take into account the distribution of the data or the classification error rate. In fact, it may not provide an optimal threshold if the data are not well balanced or if the distributions of the two classes overlap significantly.

- **Second Threshold**: The second method uses a more sophisticated approach to determine the threshold. Instead of simply calculating the mean, a range of possible threshold values are examined and the one that minimizes the classification error rate is chosen. This method is more computationally heavy, but can lead to a more accurate threshold and a lower classification error rate.

Methods	**Number of errors**	**Error rate (%)**
LDA - First threshold	275	13.75
LDA - Second threshold	184	9.2
PCA(m=6)+LDA - First threshold	186	9.3
PCA(m=5)+LDA - First threshold	186	9.3
PCA(m=6)+LDA - Second threshold	184	9.2
PCA(m=5)+LDA - Second threshold	185	9.25

Table 1: LDA and PCA+LDA results with different threshold

As can be seen from the Table 1 there is not much real improvement by applying PCA as preprocessing before LDA and that the classification error turns out to be quite high so good classification is not performed using LDA or the combination of LDA with PCA.

3 Multivariate Gaussian Density

The multivariate Gaussian distribution is a generalization of the multidimensional Gaussian normal distribution. It is used to describe the probability of a vector of continuous random variables that can be correlated with each other.
The multivariate Gaussian distribution in d dimensions is characterized by a vector of averages μ and a covariance matrix Σ. The probability density function of the multivariate Gaussian distribution is given by:

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi)^{M/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{5}$$

Very often to prevent numerical problems due to exponentiation of large numbers, it is usually recommended to work on the logarithm of the density:

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

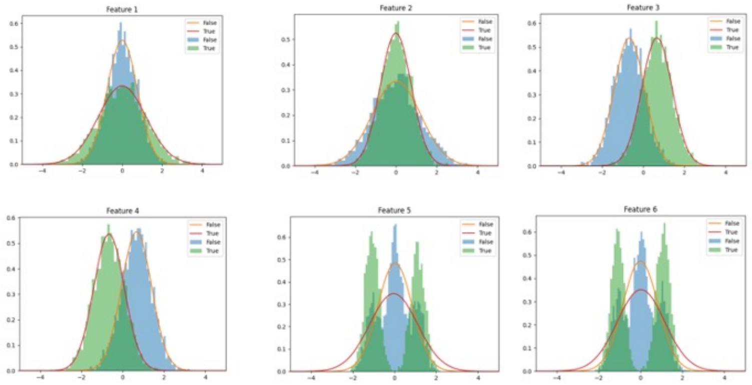

Figure 7: Gaussian density

The curve we can observe on the graph in Figure 7 represents the Gaussian probability density calculated from the data. If the data actually follow a Gaussian distribution, then the curve should fit the histogram of the data well. When this happens we can say that the Gaussian model is a good fit for the data while if the Gaussian curve does not fit the histogram well, then the Gaussian model may not be the most appropriate model for that data.
We can observe that feature 1, feature 2, feature 3 and feature 4 fit the model well and follow a Gaussian distribution.

4 Classification model analysis

To perform the classification obviously we had to divide the training dataset into a part for training and a part for validation.

4.1 Gaussian Models

The first class of models we will analyze is that of Gaussian generative models. A simple model is to assume that our data, given a class, can be described by a Gaussian distribution:

$$f_{x|C}(x|c) = \mathcal{N}(x|\mu_c, \Sigma_c) \tag{7}$$

Since this is a binary classification task, we will assign a probability score to each sample in terms of the log-posterior class likelihood ratio:

$$\log \frac{P(C=1|x)}{P(C=0|x)} = \log \frac{P(x|C=1)P(C=1)}{P(x|C=0)P(C=0)} \tag{8}$$

Table 2: Error Rates for Different Models and Features

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

4.1.1 Naive Bayes Classifier

4.1.2 Tied Bayes Classifier

4.1.3 Gaussian Models Comparison

4.2 Logistic Regression Classifier

4.2.1 Quadratic Logistic Regression(QLR)

4.3 SVM Classifier

4.3.1 Linear SVM

4.3.2 Kernel SVM

4.4 GMM Classifier

5 Score Calibration

5.1 Calibration Analysis on Selected Models

5.2 Calibrating Scores for Selected Models

6 Experimental Results

6.1 Calibration on evaluation score

6.2 Considerations

7 Conclusions