Politecnico di Torino

Fingerprint Spoofing Detection

Gabriele Ferro, s308552
&
Filippo Greco, s309529

July 2023

Contents

Chapter 1

Introduction

1.1 Problem Overview

The task of this project is to build a classifier that will detect whether a fingerprint's image represents an authentic fingerprint or a spoofed one. The classifier will be chosen among the most common Machine Learning ones, selecting the one that gives best results.
The training set contains over 2300 samples and it's important to mention that it's imbalanced, containing more spoofed images than authentic ones. Each of the spoofed samples is generated by one out of 6 different techniques not known before hand so this samples are only considered as one class with different distributions inside of it.
Each sample is described by 10 features and a label (1 for authentic, 0 for spoofed).

The evaluation phase will be performed on the evaluation set, which contains around 7700 samples. Again this set is unbalanced with the same manner as the training set. In the last chapter a comparison between the performances over these two sets is made, to understand better if the analysis conducted in this review was effective.

The target application has a working point described by the triplet ($\pi_T = 0.5, C_{fn} = 1, C_fp = 10$): two different costs are used for the errors due to the security vulnerabilities that are involved in a misclassification.

For this reason it is possible to define the effective prior

$$\tilde{\pi} = \frac{\pi_T C_{fn}}{\pi_T C_{fn} + (1 - \pi_T) C_{fp}} = 0.09$$

that leads to the equivalent working point $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.09, 1, 1)$

1.2 Feature Analysis

The next images shows the behaviour of the distribution of each feature (one feature per plot) compared between the two classes. The orange distributions represent the authentic fingerprints (class 1), meanwhile the blue are for the spoofed ones (class 0).

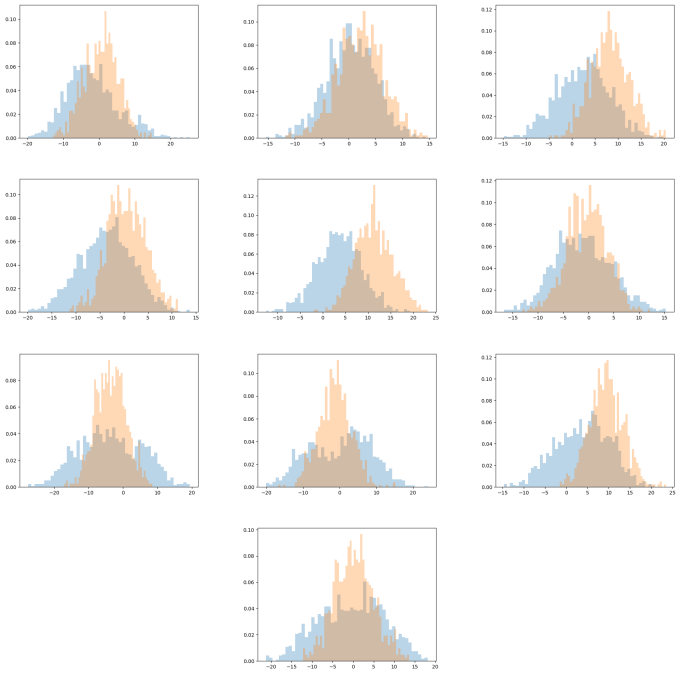


Figure 1.1: Features Distributions

On one hand the plots show that each feature of class 1 (authentic) follow a Gaussian density distribution, but this is not true for class 0 (spoofed) instead: this is due to the fact that the samples relative to the spoofed fingerprints are generated with 6 different techniques, each one of them representing a different way of trying to fake the fingerprint.
The more discriminant feature seems to be the fifth (second of the second row in Figure 1.1), even if the two distributions are majorly overlapped it separates the two classes better than the other features.

Here again we can see, with the help of some scatter plots, the distribution of the samples projected on the combination of two features each time.
The fifth feature (fifth row of Figure 1.2) shows again an higher degree of separation with respect to the other features.

The blue points (representing the spoofed fingerprints images) are always more sparse than the orange ones. Demonstrating again that these samples belong to different clusters that don't share the same pattern on the features.

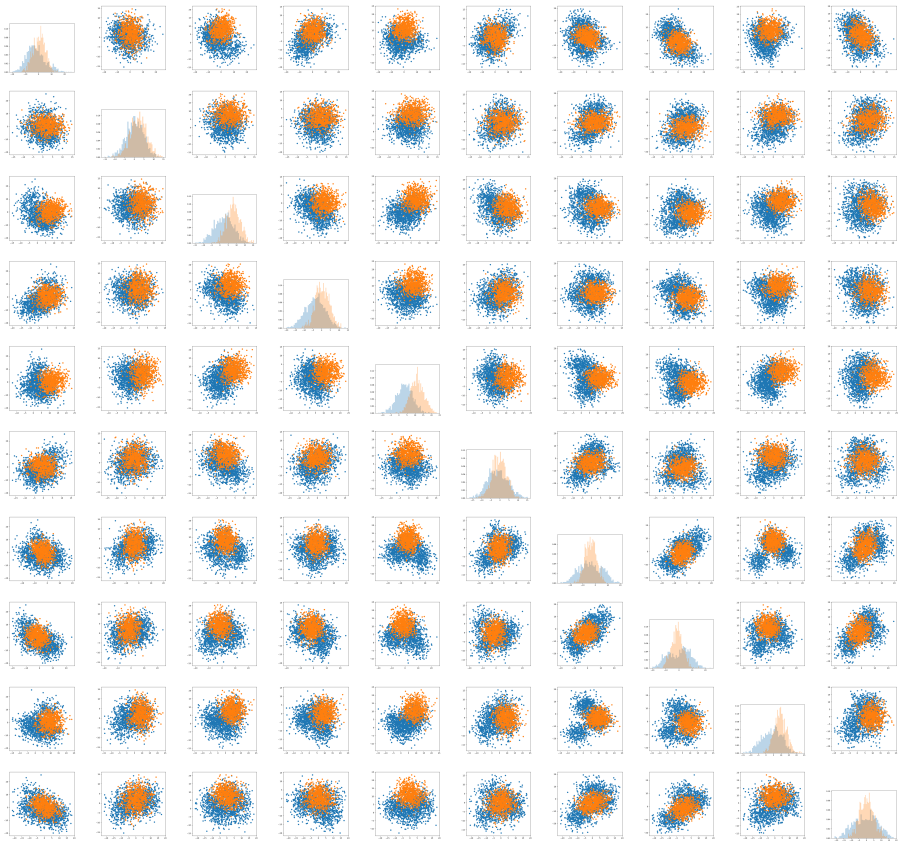


Figure 1.2: Scatters of pair of features

The plot in Figure 1.3 shows the projection of the dataset on the direction obtained by LDA: we obtain only one direction because LDA (Linear Discriminant Analysis) is able to discriminate the C-1 directions that better separate the feature space, where C is the number of classes. Working on a dual class problem brings us a single direction.

The two distribution seem to be discretely separable by a linear separation rule, but more inspection on the data should be done. For example applying PCA, as we are going to show in the next page, will reveal some more details.

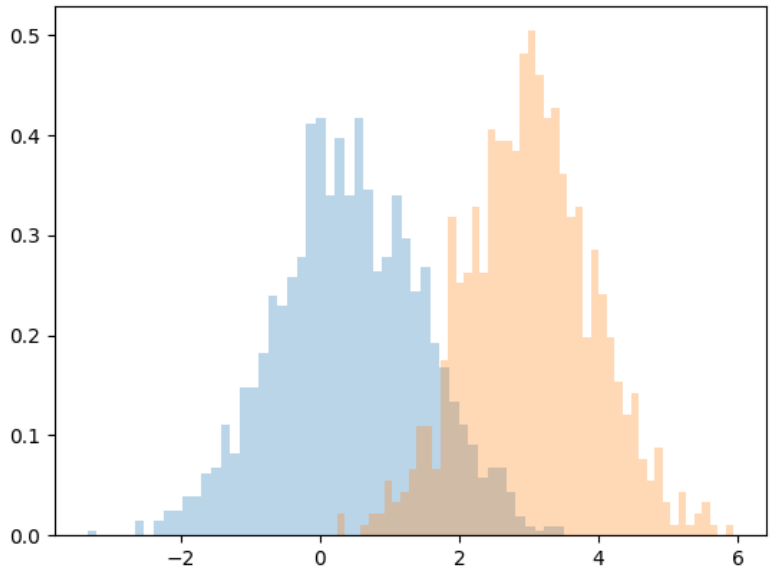


Figure 1.3: LDA Samples Projection

Figure 1.4 is the result of PCA: the first two figures shows the projections of our samples on the two principal components, the last image, instead, shows a scatter over these two dimensions.

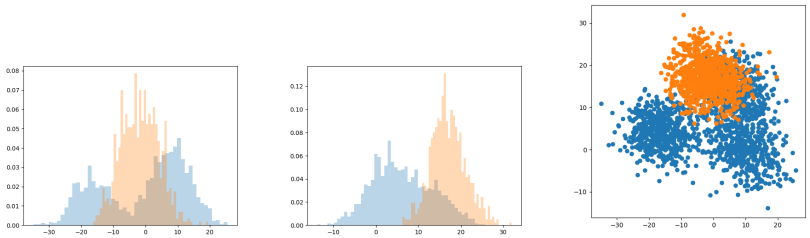


Figure 1.4: PCA Analysis

It looks like the two principal components of our dataset are not so well separated, using a linear model in this case will hardly produce a good outcome, even though LDA showed us a discrete result; A non-linear/non-Gaussian classifier can gives us a better solution for our application.

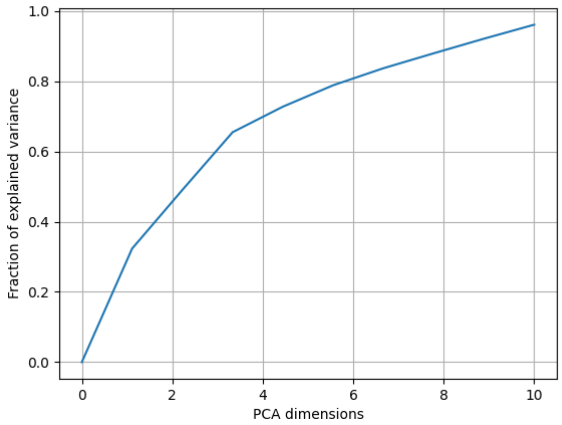
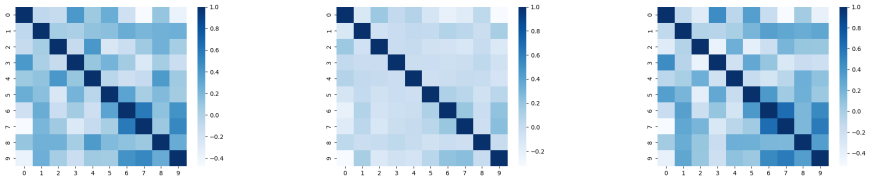


Figure 1.5: Percentages of explained variance with PCA

The plot 1.5 shows that with at least 8 directions we maintain around 90% of the dataset variance. In the following discussion also 6 and 7 will be used as number of directions preserved by the PCA to further study the behaviour.

The last useful analysis on the features shows, through the help of Heat maps, the correlation between the features. It gives a clear understanding on if the reduction of dimensions will lead to an improvement on the model, giving the possibility to remove the dimensions that are highly correlated (do not gives information to help discriminate between classes). It also helps decide if Naive-Bayes could be used to simplify the model or not.



(a) Dataset (b) Authentic (c) Spoofed

Figure 1.6: Heat maps - Pearson Correlation

They show feature correlation taking into account the whole dataset, only authentic fingerprint samples, only spoofed fingerprint samples respectively from the left to the right. It is important to notice that the authentic samples have not a great degree of correlation on the features, meanwhile the spoofed ones behave differently, again because of the sub clusters inside this class, and shows much more correlation between features like 6 and 7 telling us it can benefit from dimensionality reduction.

Reducing the dimension of the feature space could be an option but given the low number of starting features it is possible that this leads to loss of useful information and consequentially, it could worsen our classifier.

1.3 Classification approach

The report contains the results, in particular the values of minDCF, for each of the following classifier:

1. **Multivariate Gaussian Classifier**
   (a) MVG with Full Covariance
   (b) MVG with Naive-Bayes assumptions
   (c) MVG with Tied Covariance Matrix
2. **Logistic Regression**
   (a) Linear Logistic Regression
   (b) Quadratic Logistic Regression
3. **Support Vector Machine**
   (a) Linear SVM
   (b) SVM with Polynomial Kernel (degree 2 and 3)
   (c) SVM with Radial Basis kernel Function (RBF)
4. **Gaussian Mixture Model**
   (a) GMM using LBG algorithm

Each model has been trained using **K-fold**, with K=10: this help us simulating more training and validation data in order to better sharpen the model parameters. **minDCF** has been computed using the triplet mentioned above representing our working point:

$$(\tilde{\pi}, C_{fn}, C_{fp}) = (0.09, 1, 1)$$

For a deeper analysis also $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.5, 1, 1)$ and $(\tilde{\pi}, C_{fn}, C_{fp}) = (0.9, 1, 1)$ are considered as tested applications.

Each model is trained using different values for **PCA** and uses cross-validation for the decision of **hyper-parameters**.

Finally, for **Logistic Regression** and **SVM**, the prior-weighted version of the models have been tried

Chapter 2

Multivariate Gaussian Model

2.1 Model

We focus now on training a Multivariate Gaussian Model, assuming that each class will have a distribution of the form:

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

We will also consider different degrees of PCA, and try using Naïve-Bayes assumption and a Tied Covariance Model.

Given the distribution of our features using a **Tied Covariance** should be worse than a Full Covariance Gaussian Model since the two classes don't have the same shape when plotted, thus having different covariance matrix.

As far as **Naïve-Bayes** we can see that the authentic fingerprints class has almost a diagonal covariance matrix but the spoofed fingerprints class shows some correlation between features, this suggest that Naïve-Bayes can provide discrete, yet not optimal, results

2.2 Results

After training the model with the combinations described above, they produced the following results shown in tabular form. They reflect what is said earlier having the standard Gaussian Model performing the best minDCF with respect to the other two types, in particular by applying a PCA on 7 dimensions.
The result is not optimal because the features are not Gaussian Distributed.

The results relative to the Naïve-Bayes shows that using a diagonal covariance matrix will gives good result using exactly 9 dimensions in the PCA. The result is not as good as what the normal Gaussian produce (with the full covariance matrix) because the spoofed fingerprints' features are slightly dependent.

In the end the Tied is performing poorly, in respect to the other two assumptions, due to what is said before: the covariance matrices of normal and spoofed samples have a different behaviour, that could be because of the different method used to generate the spoofed ones.

Considering the other applications tested the same PCA value for Full Covariance and Naïve-Bayes models perform well, meanwhile the Tied one for $\tilde{\pi} = 0.9$ is performing a bit worse if we consider applying PCA.

Table 2.1: minDCF of Multivariate Gaussian Model for different priors

Model	PCA	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Full Covariance	-	0.335	0.106	0.191
	9	0.338	0.105	0.194
	8	0.336	0.104	0.190
	7	**0.333**	0.106	0.188
	6	0.335	0.105	0.193
Naïve-Bayes	-	0.471	0.134	0.242
	9	**0.380**	0.109	0.167
	8	0.390	0.108	0.174
	7	0.388	0.110	0.174
	6	0.387	0.109	0.172
Tied	-	0.476	0.170	0.380
	9	0.469	0.170	0.391
	8	0.474	0.168	0.393
	7	**0.467**	0.168	0.394
	6	0.471	0.170	0.395

Chapter 3

Logistic Regression

3.1 Model

We now take into account Logistic Regression classifiers, both linear and the quadratic approach will be tested, with the latter one potentially being our best option due to the non-linear separation rules needed in order to better discriminate the two classes. The quadratic classifier is possible through a feature expanded space obtained in this fashion:

$$\Phi(\boldsymbol{x}) = \begin{bmatrix} vec(\boldsymbol{x}\boldsymbol{x}^T) \\ \boldsymbol{x} \end{bmatrix}$$

Even if we expect the quadratic to be better than the linear we still include in the report the result of both of them for completeness. In both of the models we introduced the regularization term, through the hyper-parameter λ: it represents the regularization term (penalty) for the norm of w; when its value grows up the model become less strict giving generally worst results for the application, meanwhile diminishing it will lead to poor classification accuracy for unseen data. Its value can be chosen using cross-validation.
More specifically we want to minimize the **Empirical Risk Function**:

$$R(\boldsymbol{w}, b) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-z_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)})$$

With z_i equals to -1 for spoofed samples and 1 for authentic ones.

Since the Regularized formulation of Logistic Regression is non invariant to linear transformation on features, we also try to implement some pre-processing of the data.

Finally, is worth trying to simulate an empirical prior π_T that reflects our application prior, using a prior-weighted version of the model:

$$R(\boldsymbol{w}, b) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{\pi_T}{n_T}\sum_{i|z_i=1} l(z_i s_i) + \frac{1-\pi_T}{n_F}\sum_{i|z_i=-1} l(z_i s_i)$$

where $l(x)$ is the logistic loss and it is defined as:

$$\log(l(x) = 1 + e^{-x})$$

For points correctly classified the logistic loss decrease to 0 proportionally to how far the point is to the separation surface. On the other hand for wrong classifications the logistic loss grows towards infinity, increasing consequentially the value of $R(\boldsymbol{w}, b)$.
Our goal is to find a separation surface described by $\boldsymbol{w}$ and b that best separates the two classes which indeed corresponds to minimizing the **Empirical Risk Function**.

3.2 Results

As shown in Figure 3.1 the previously exposed prediction was right, the minDCF for the linear logistic regression is a little bit high, even considering combinations of PCA and z-norm attesting similar results (with PCA ones attesting a lower result).
It also seems that using z-norm regularization is worst with a growing λ but achieve the best overall result for $10^{-5} < \lambda < 1$.

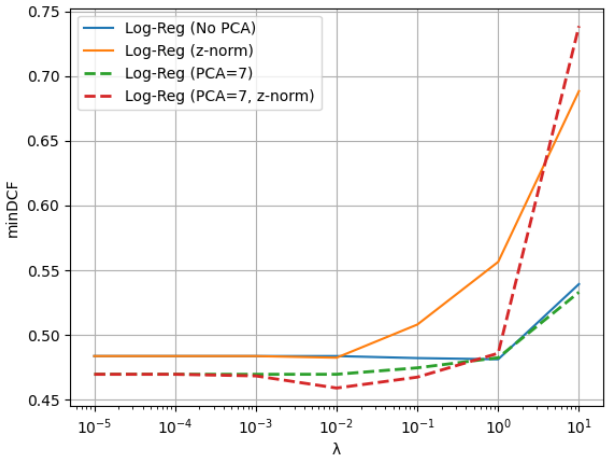


Figure 3.1: Linear Logistic Regression

The results with quadratic separation rules are, as predicted, better then the linear model even without pre-processing on the dataset (no PCA nor z-norm) as shown by the graph of Figure 3.2.
Enabling PCA brings to a better results in each of the tested cases (dimensions in range [6, 9]) with the best result associated with 7 dimensions.

Applying pre-processing strategies to the best PCA result (7) leads to the results shown by Figure 3.3. Pre-processing doesn't bring any overall advantage to the model, but surprisingly with z-norm, whitening and l2-norm, as presented from the red line, it achieves the lowest minDCF, for $\lambda = 10^{-4}$, of around 0.25.

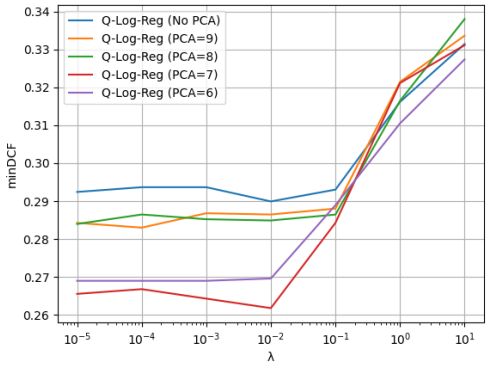


Figure 3.2: Quadratic Logistic Regression for different PCAs

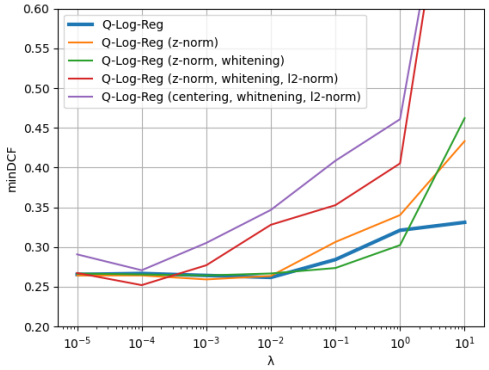


Figure 3.3: Quadratic Logistic Regression for different pre-processing strategies, PCA=7

Considering the weighted prior version, the model is computed simulating an empirical prior equals to the effective prior of different tested applications.
The result which perform best is the same as before, without any improvement introduced by the prior weighting.

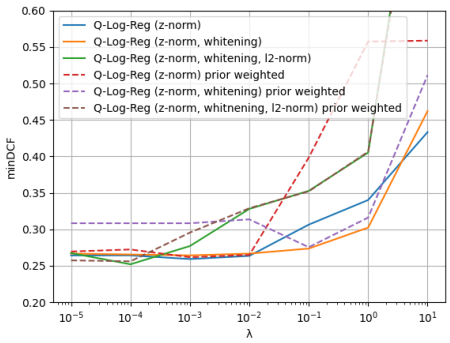


Figure 3.4: Quadratic Logistic Regression, PCA=7, prior weighted comparison

Table 3.1 shows all the data collected before and confirms what we discussed, attesting to the model with empirical prior probability of the dataset, pre-processed with Z-norm, whitening and L2-norm the best result in term of minDCF for the target application.
As far as the other tested working points ($\tilde{\pi} = 0.5$ and $\tilde{\pi} = 0.9$) it seems that a prior weighted version of the model is beneficial (results highlighted in orange).

Table 3.1: Logistic Regression with PCA=7, $\lambda = 10^{-4}$

Prior π_T	pre-proc	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Emp. (Fold)	-	0.267	0.094	0.208
Emp. (Fold)	[1]	0.264	0.094	0.201
Emp. (Fold)	[2]	**0.251**	0.106	0.290
$\pi_T = 0.09$	[1]	0.272	0.092	0.199
$\pi_T = 0.09$	[2]	0.256	0.105	0.291
$\pi_T = 0.5$	[1]	0.272	0.090	0.208
$\pi_T = 0.5$	[2]	0.256	0.107	0.295
$\pi_T = 0.9$	[1]	0.272	1.000	9.000
$\pi_T = 0.9$	[2]	0.256	1.000	9.000

[1] z-norm
[2] z-norm + whitening + l2-norm

Chapter 4

Support Vector Machine

4.1 Model

In this chapter we will try to use SVMs, which are classifiers that look for maximum margin separation hyperplanes by solving the primal or the dual formulation of the problem. The soft margin approach is considered.

The primal formulation is expressed as the **minimization** of:

$$\widehat{J}(\widehat{\boldsymbol{w}}) = \frac{1}{2}\|\widehat{\boldsymbol{w}}\|^2 + C\sum_{i=1}^{n}\max(0, 1 - z_i(\widehat{\boldsymbol{w}}^T\widehat{\boldsymbol{x}}_i))$$

The dual one is instead the **maximization** of:

$$J^D = (\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{H}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{1}$$

$$s.\,t.\ \ 0 \leq \boldsymbol{\alpha}_i \leq C\ ,\ \forall i \in \{1...n\}\ ,\ \ \sum_{i=1}^{n}\boldsymbol{\alpha}_i z_i = 0$$

Anyway, in order to solve the problem using L-BFGS-B we will use a slightly modified version in which H is replaced with $\widehat{H}$, and each element of the latter is defined as

$$\widehat{H}_{ij} = z_i z_j \widehat{x}_i^T \widehat{x}_j$$

with

$$\widehat{x}_i = \begin{bmatrix} x_i \\ K \end{bmatrix}\ ,\ \ \widehat{\boldsymbol{w}} = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix}$$

Where K will be set to $K = 1$ and mitigates the effect of the bias term when normalizing $\widehat{\boldsymbol{w}}$, since $\|\widehat{\boldsymbol{w}}\|^2 = \|\boldsymbol{w}\|^2 + b^2$.

Once found $\boldsymbol{\alpha}$ we are able to compute $\widehat{\boldsymbol{w}}^*$ with the following relation:

$$\widehat{\boldsymbol{w}}^* = \sum_{i=1}^{n} \boldsymbol{\alpha}_i z_i \widehat{\boldsymbol{x}}_i$$

That will help us giving scores to unlabeled samples through:

$$s(\boldsymbol{x}_t) = \widehat{\boldsymbol{w}}^T \widehat{\boldsymbol{x}}_t$$

Moving to kernel SVM the following kernel functions are considered, which corresponds respectively to the **Polynomial kernel** function of degree d and the **Radial Basis Function**.

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1^T \boldsymbol{x}_2 + 1)^d$$
$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$$

More specifically we are going to use d=2 and d=3.
The hyper-parameter γ is chosen using cross-validation in order to find the best value. With low value of γ the support vectors points influence other points belonging to a wider area, meanwhile having a large value of γ support vectors tends to have negligible influence on points that are not close.

Finally we also mention that the two previous kernel functions are slightly modified as well to work on the modified version of the problem using in both cases

$$\widehat{k}(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(\boldsymbol{x}_1, \boldsymbol{x}_2) + \xi \ , \ \ \xi = K^2 = 1$$

Where ξ is used to regularize the bias, since in the non-linear SVM is not enough to simply extend the feature vectors as we previously did, but we need to consider this constant value for our kernel function.

4.2 Results

Linear SVM does not give us exciting results, anyway applying pre-processing seems to help the classifier (specifically using z-norm, whitening and l2-norm). PCA is not improving what we already achieved without it. Nevertheless the results are almost identical, so it's worth using it, due to the boost of the computation that leverages dimensionality reduction.

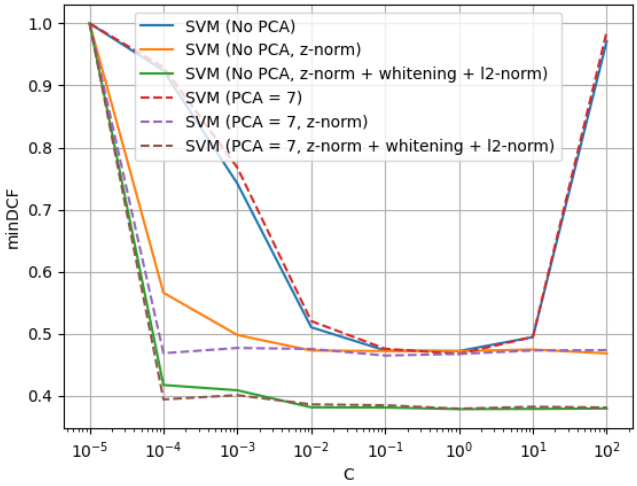


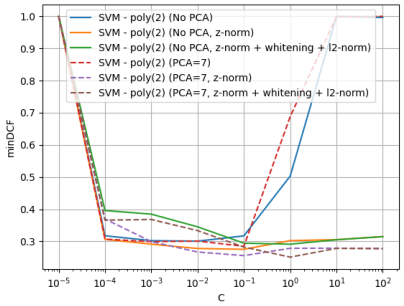
Figure 4.1: Linear SVM

Considering all the applications shown in Table 4.1 the best result seems to be for C equals 1 or 10 without PCA, nevertheless the other values for the hyper-parameter C does not worsen the model too much.

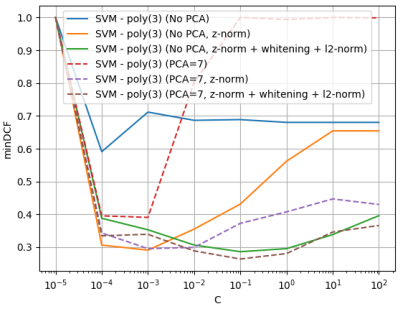
Table 4.1: Linear SVM with z-norm, whitening, l2-norm

$\tilde{\pi}$	C	No PCA	PCA = 7
$\tilde{\pi} = 0.09$	C=0.01	0.381	0.386
	C=0.1	0.381	0.385
	C=1	**0.379**	0.380
	C=10	**0.379**	0.382
	C=100	0.380	0.381
$\tilde{\pi} = 0.5$	C=0.01	0.154	0.147
	C=0.1	0.152	0.147
	C=1	0.149	0.145
	C=10	0.151	0.146
	C=100	0.150	0.144
$\tilde{\pi} = 0.9$	C=0.01	0.425	0.397
	C=0.1	0.412	0.397
	C=1	0.410	0.405
	C=10	0.425	0.419
	C=100	0.426	0.418

Moving on to kernel SVM and visualizing Polynomial results is clear that in both tested case (degree 2 and 3) the minDCF is attesting at a lower value than before.
Table 7.1 reveals that the best result between this two different degrees is attested by the polynomial of degree 2 with PCA and all the pre-processing techniques listed before. This confirms that the samples are better discriminated by quadratic separation surfaces.



(a) Degree 2



(b) Degree 3

Figure 4.2: SVM polynomial kernel function

Table 4.2: minDCF comparison for different degrees of polynomial kernel

d	pre-proc	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
2	[1]	0.256	0.087	0.194
	[2]	**0.248**	0.100	0.301
3	[2]	0.264	0.096	0.277

[1] PCA=7, z-norm
[2] PCA=7, z-norm + whitening + l2-norm

Considering the RBF as kernel it is visible that it doesn't provide better results than the Polynomial kernel SVM model.
Figure 4.4 considers also $\lambda = 10^{-5}$ as an option since the previous analysis shown in Figure 4.3 suggests it can lead to a new minimum. PCA does not improve minDCF and applying any pre-processing strategy is even worsening the model.
However from Figure 4.3 and Table 4.3 is deducible that the best value for λ is 10^{-3}.

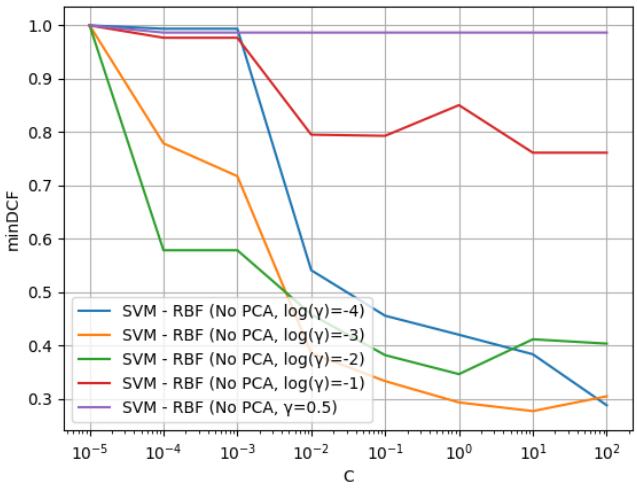
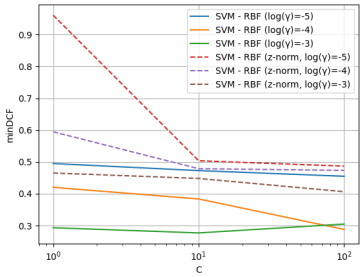
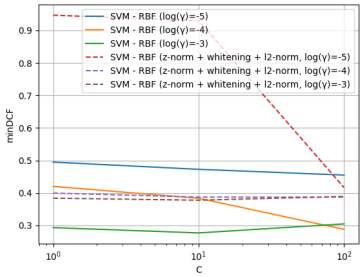


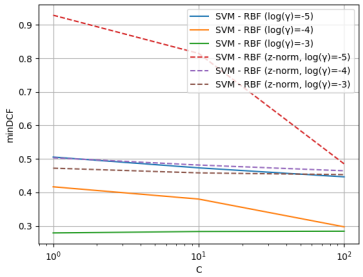
Figure 4.3: RBF kernel SVM - No PCA



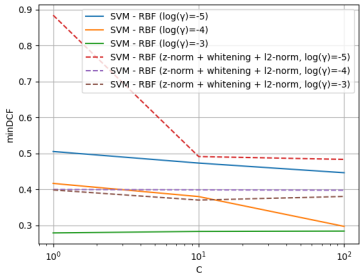
(a) No PCA, z-norm



(b) No PCA, *



(c) PCA=7, z-norm



(d) PCA=7, *

Figure 4.4: RBF kernel SVM (*= z-norm + whitening + l2-norm)

Table 4.3: RBF kernel SVM best results

$\tilde{\pi}$	PCA	no pre-processing	z-norm	z-norm + whitening + l2
0.09	-	**0.277**	0.406	0.378
	7	0.283	0.453	0.371
0.5	-	0.080	0.135	0.148
	7	0.093	0.159	1.43
0.9	-	0.185	0.302	0.401
	7	0.179	0.348	0.395

In conclusion the best model found is the one that uses the polynomial kernel of degree 2 with PCA=7 and the other pre-processing strategies discussed before. Also in Table 4.4 we compare the version of the model with embedded empirical prior and the one resulting from a prior weighted version with π_T being the prior chosen for the weighting.
Additionally we take a look of different combination of $\tilde{\pi}$ and π_T.

Table 4.4: SVM polynomial kernel d=2 with PCA=7, z-norm + whitening + l2-norm

$\tilde{\pi}$	C	$\pi_T = Emp.(Fold)$	$\pi_T = 0.09$	$\pi_T = 0.5$	$\pi_T = 0.9$
0.09	C=0.1	0.282	0.283	0.289	0.334
	C=1	**0.248**	0.278	0.265	0.230
	C=10	0.278	0.272	0.279	0.360
	C=100	0.277	0.271	0.275	0.376
0.5	C=0.1	0.100	0.103	0.107	0.119
	C=1	0.106	0.108	0.103	0.111
	C=10	0.113	0.113	0.106	0.119
	C=100	0.117	0.119	0.104	0.131
0.9	C=0.1	0.301	0.337	0.290	0.323
	C=1	0.303	0.334	0.300	0.315
	C=10	0.312	0.322	0.292	0.378
	C=100	0.318	0.340	0.305	0.369

For our target application, and for a working point with $\tilde{\pi} = 0.5$ the prior weighted version is not improving what we obtain by using the empirical prior of the dataset. This is not true for application with $\tilde{\pi} = 0.9$, indeed in Table 4.4 the best result can be found for $\pi_T = 0.5$.

Chapter 5

Gaussian Mixture Model

5.1 Model

In this last chapter regarding classifiers we take a look at the Gaussian Mixture Model, which models a density that represents weighted sum of M Gaussians.

$$X \sim GMM(\boldsymbol{M}, \boldsymbol{S}, \boldsymbol{w}) \implies f_{\boldsymbol{X}}(\boldsymbol{x}) = \sum_{g=1}^{M} w_g \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$$

In order to choose the number of densities and their respective parameters we decided to use the LBG and EM algorithms that work as follow:

each new iteration of LBG algorithm constructs a GMM with $2G$ components from a GMM with G components, where the starting point is a single component GMM with $\text{GMM}_{init} = \big[(1, \boldsymbol{\mu}, \boldsymbol{\Sigma})\big]$; $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the empirical mean and covariance matrix of the set.

The split is performed in this fashion:

$$(w_g, \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g) \implies \left(\frac{w_g}{2}, \boldsymbol{\mu}_g + \boldsymbol{d}_g, \boldsymbol{\Sigma}_g\right), \left(\frac{w_g}{2}, \boldsymbol{\mu}_g - \boldsymbol{d}_g, \boldsymbol{\Sigma}_g\right)$$

Each time the split is performed we can use the EM algorithm that incrementally updates the GMM parameters until a point where the improvement of the log-likelihood $l(\boldsymbol{M}_t, \boldsymbol{S}_t, \boldsymbol{w}_t)$ between two iterations is less than a pre-fixed threshold, we will consider this threshold to be $\Delta_t = 10^{-6}$. The updates of the model parameters in the EM algorithm make use of these statistics:

$$Z_g = \sum_{i=1}^{N} \gamma_{g,i}, \quad \boldsymbol{F}_g = \sum_{i=1}^{N} \gamma_{g,i} \boldsymbol{x}_i, \quad \boldsymbol{S}_g = \sum_{i=1}^{N} \gamma_{g,i} \boldsymbol{x}_i \boldsymbol{x}_i^T$$

To obtain the new parameters:

$$\boldsymbol{\mu}_{g_{t+1}} = \frac{\boldsymbol{F}_g}{Z_g}, \quad \boldsymbol{\Sigma}_{g_{t+1}} = \frac{\boldsymbol{S}_g}{Z_g} - \boldsymbol{\mu}_{g_{t+1}}\boldsymbol{\mu}_{g_{t+1}}^T, \quad w_{g_{t+1}} = \frac{Z_g}{\sum_{g'=1}^{M} Z_{g'}}$$

We interchange the two algorithms until we reach a certain number of components chosen before hand (the number of components will be a power of 2: 1, 2, 4, 8, ...).

The computation with Tied and diagonalized covariance matrices is not considered for what was told and seen in the MVG chapter.

5.2 Results

From the study of the features done in the beginning it can be assumed that the best result will be found for 1 component for the true fingerprint class and 4 or 8 components for the spoofed class since there are 6 different spoofing techniques represented in our samples. In fact the results confirms the previous hypotheses showing the best components numbers to be 1 for class 1 and 4 for class 0 (note that it can't be 6 for what said in the model presentation, the model can be trained only using power of 2 number of components).

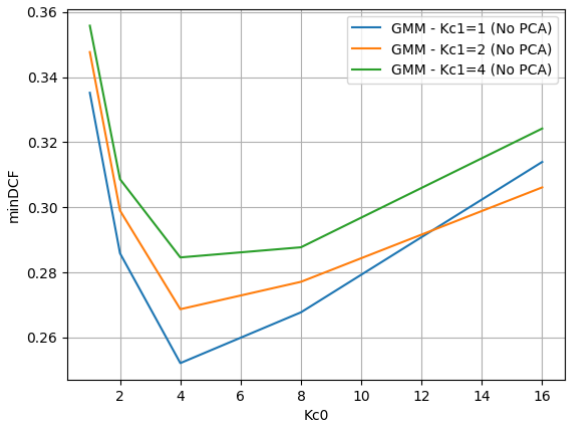
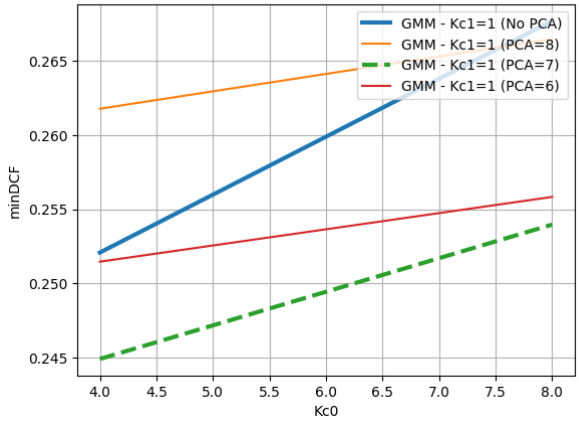
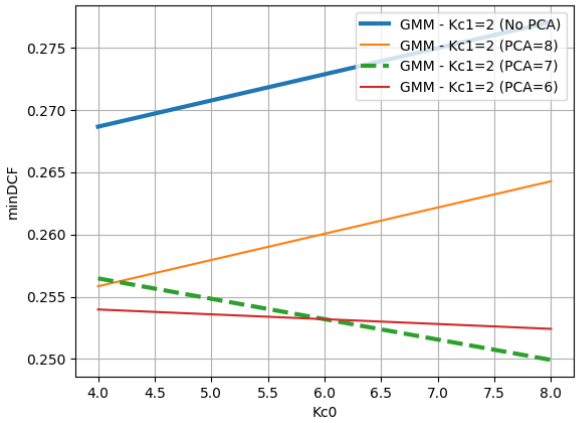


Figure 5.1: GMM without PCA

Introducing PCA brings an improvement, in particular for PCA=7 we achieve the best result as for the previously seen models. Only the range between 4 and 8 dimensions for class 0 is plotted to magnify the result and also the tests for class 1 with 2 dimensions is shown to further analyze any eventual improvement (even if the class sample all belongs to the same cluster).



(a) Kc1=1



(b) Kc1=2

Figure 5.2: minDCF of GMM for different PCA values

Table 5.1: minDCF of GMM for different configurations, PCA=7

Kc1	Kc0	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
1	**4**	**0.245**	0.079	0.159
1	8	0.254	0.084	0.160
2	4	0.256	0.082	0.144
2	8	0.250	0.085	0.149

Chapter 6

Calibration and Fusion

6.1 Score calibration

We now consider all the best models obtained so far. The overall best result is the GMM one, performing slightly better than the SVM with polynomial kernel function.

Table 6.1: minDCF of best models

Classifier	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Q-Log-Reg - PCA=7 - $\gamma = 10^{-4}$ - *	0.252	0.106	0.290
Poly(2) SVM - PCA=7 - C=1 - *	0.251	0.106	0.303
GMM - PCA=7 - Kc1=1, Kc0=4	**0.245**	0.079	0.159

* preprocessing: z-norm + whitening + l2-norm

Till now we only considered minDCF that is a metric obtained if we knew before-hand the optimal threshold, but of course due to score mis-calibration the actualDCF can be drastically different.
To obtain the actualDCF we use the threshold corresponding to the effective prior $\tilde{\pi}$.

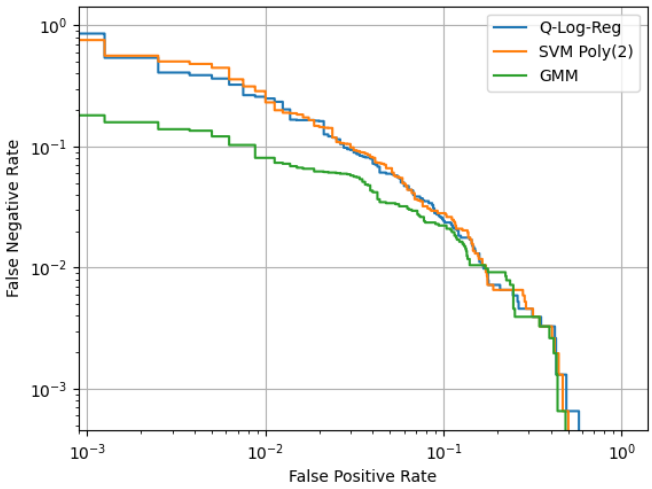


Figure 6.1: DET plot

From Figure 6.1 we can evaluate the performance of the considered models. It is visible that the GMM has the lowest FPR and FNR corresponding to the EER (Equal Error Rate, point where FNR=FPR) attesting the best performances on the training set.

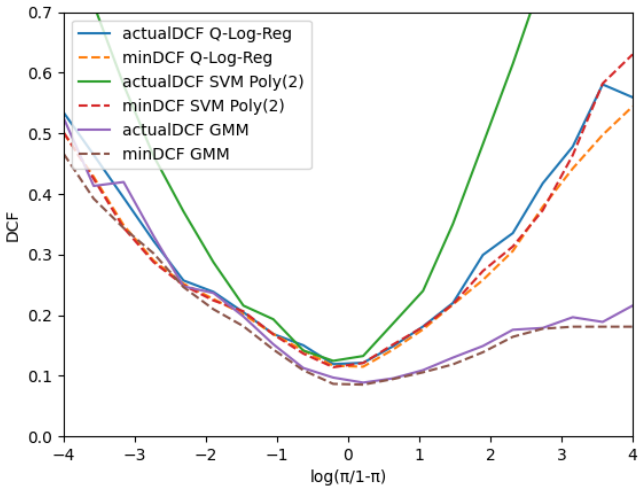


Figure 6.2: Bayes error plot for best models

Table 6.2: actualDCF for best models

Classifier	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Q-Log-Reg	0.304	0.120	0.320
Poly(2) SVM	0.499	0.115	0.408
GMM	**0.245**	0.088	0.177

Comparing the pairs of continuous/dotted lines it can be deducted that only the SVM can majorly improve after a calibration of the scores, compared to the others that performs identically when actualDCF is considered (in the case of GMM) or pretty much the same (for quadratic logistic regression).

At this point score calibration can help improving the SVM model and it is performed using a K-fold approach and by feeding the scores obtained by SVM to an MVG classifier to obtain the calibrated scores. MVG works as a calibration model since SVM can be calibrated with a generative probabilistic model.

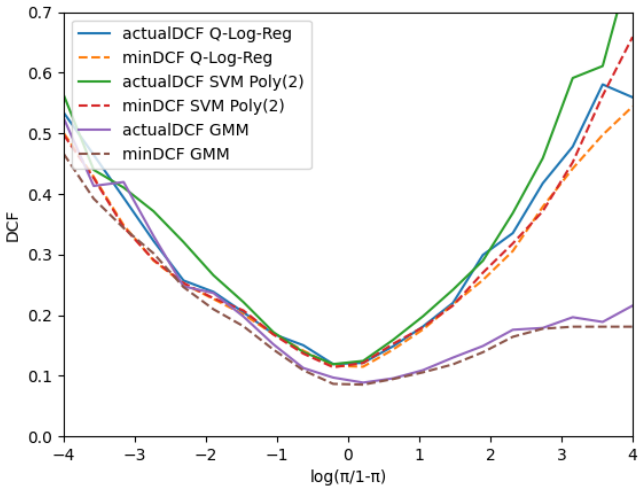


Figure 6.3: Bayes error plot for best models with calibration on SVM

After the calibration is noticeable that the SVM improved in terms of actualDCF at the working point going closer to the minimum one. The other models have the same plot as before since calibration was not strictly necessary.

Table 6.3: DCFs for SVM after calibration

	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
minDCF	0.252	0.105	0.306
actualDCF	0.325	0.109	0.332

From now on the calibrated model for SVM poly(2) will be used.

6.2 Fusion

Moving on we can now consider to use multiple models at once to combine their strong points and improve the classification.

Table 6.4: Fusion of the models

Fusion	minDCF			actualDCF		
	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
[1]	0.251	0.106	0.290	0.304	0.120	0.320
[2]	0.252	0.105	0.306	0.325	0.109	0.332
[3]	0.245	0.079	0.159	**0.245**	0.088	0.177
[1] + [2]	0.246	0.107	0.296	0.291	0.113	0.358
[1] + [3]	0.249	0.087	0.171	0.263	0.092	0.178
[2] + [3]	0.249	0.085	0.163	0.290	0.089	0.192
[1] + [2] + [3]	0.255	0.089	0.190	0.279	0.095	0.196

[1] Q-Log-Reg
[2] SVM Poly(2)
[3] GMM

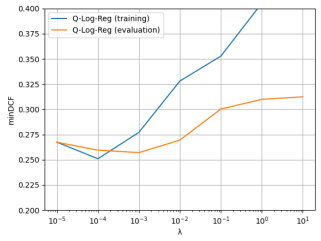
However it seems that all the combinations made with the best models achieved so far, do not deliver a better model than the GMM alone. Thus, from now on, we do not consider anymore fusion as an option for our classifier.

Chapter 7

Evaluation

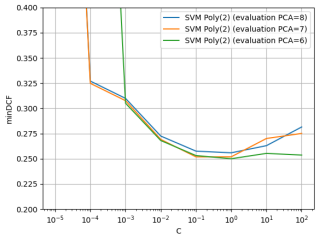
7.1 Scores on Evaluation Population

In this section we analyze the performances of our best classifiers on the evaluation set. Figures 7.1 shows the comparison between the the performance obtained over the training and evaluation set.
Given the collected results on the training set we expect the GMM to perform better than the other two chosen models.

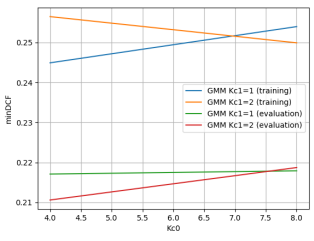


(a) Q-Log-Reg

(b) SVM Poly(2)



(c) SVM Poly(2), PCAs



(d) GMM

Figure 7.1: Comparison of performances between training and evaluation set

7.2 Final considerations

Quadratic Logistic Regression and **SVM Polynomial Kernel of degree 2** configurations were correctly chosen, indeed the minimum on the evaluation set is very close to the minimum of the training. Surprisingly **GMM** performances on the evaluation set are far better than what we expected, confirming GMM to be the best choice for our application.

In table 7.1 we report the numerical data for the **chosen configurations**, again for different applications.

Table 7.1: DCFs on the **chosen configurations**

Set	Classifier	minDCF			actualDCF		
		$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Training	[1]	0.251	0.106	0.290	0.304	0.120	0-320
	[2]	0.252	0.105	0.306	0.325	0.109	0.332
	[3]	0.245	0.079	0.159	0.245	0.088	0.177
Evaluation	[1]	0.259	0.102	0.357	0.271	0.103	0.359
	[2]	0.252	0.102	0.378	0.280	0.130	0.516
	[3]	0.217	0.071	0.165	0.223	0.073	0.172

[1] Q-Log-Reg: PCA=7, $\lambda = 10^{-4}$, z-norm + whitening + l2-norm
[2] SVM Poly(2): PCA=7, C=1, z-norm + whitening + l2-norm
[3] GMM: PCA=7, Kc1=1, Kc0=4

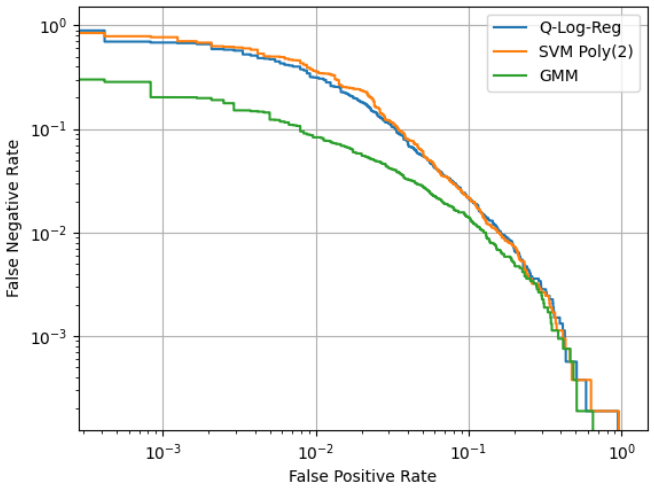


Figure 7.2: DET plot on evaluation set

As expected the GMM model still perform the same way even on the evaluation set. It is definitive that this model is by far the more accurate choice for our application.

In Table 7.2 instead we show the results obtained by using the **optimal configurations** on the evaluation set, that have slightly differences on the choice of hyper-parameters and pre-processing techniques.
Comparing the two tables together gives an estimate on how good our decisions were on choosing the configurations of the models.

Table 7.2: DCFs on the **optimal configurations**

Classifier	minDCF			actualDCF		
	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.09$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Q-Log-Reg	0.257	0.099	0.342	0.293	0.100	0.410
Poly(2) SVM	0.250	0.105	0.387	0.272	0.126	0.544
GMM	0.211	0.069	0.170	0.219	0.073	0.178

Q-Log-Reg: PCA=7, $\lambda = 10^{-3}$, z-norm + whitening + l2-norm
SVM Poly(2): PCA=6, C=1, z-norm + whitening + l2-norm
GMM: PCA=7, Kc1=2, Kc0=4