Fingerprint Spoofing

Project Report for "Machine Learning and Pattern Recognition" course

Silvio Chito, s309619
Andrea Draperis, s298717

Date: July 3rd, 2023

Politecnico di Torino

Turin, Italy

1 Introduction

1.1 Abstract

The goal of the application is to build a model that solves the task of classifing authentic fingerprint images from spoofed ones. In the following sections, we'll use different Machine learning models and decide which one performs the best, by discussing the corresponding pros and cons.

1.2 Problem Overview

The given dataset is made of datapoints that represent fingerprint images by means of embeddings (low-dimensional representation). Every fingerprint is, in practice, a 10-dimensional continuos-valued vector of real numbers that are obtained by mapping images to a low-dimensional manifold. An authentic fingerprint is labeled as 1, while the spoofed one with 0. The embedding components do not have a physical interpretation. The spoofed fingerprint samples belong to one of 6 possible sub-classes, corresponding to different spoofing approaches, but the actual spoofing method (sub-class label) is not available. Moreover, the target application takes in consideration the fact that the prior probabilities for the two classes are the same, but this does not hold for the costs that the application pays for misclassification. The training set consists of 2325 samples, whereas the test set contains 7704 samples (the datasets are imbalanced, with the spoofed fingerprint class having significantly more samples).

1.3 Feature Analysis

Below are histograms for each of the 10 features.

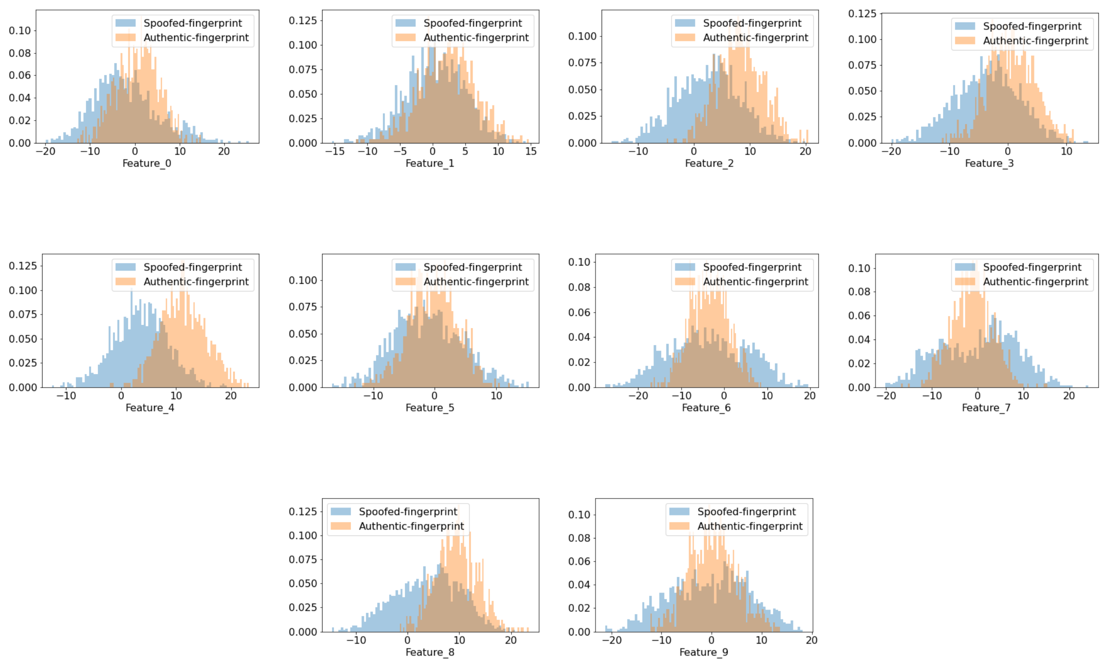


Figure 1: Histograms of the features

We can notice that they have approximatly a Gaussian shape, which means that they could potentially satisfy the assumption made by same classifiers (Gaussian classifiers) discussed in the next paragraphs

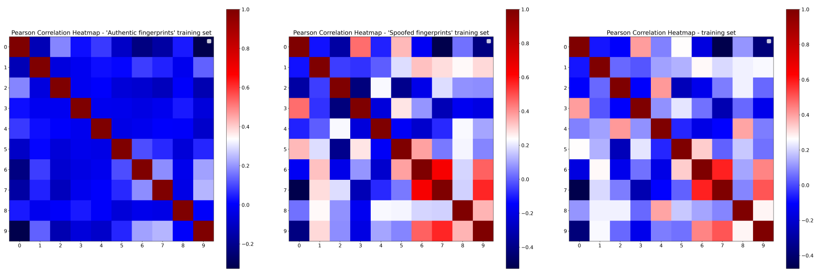
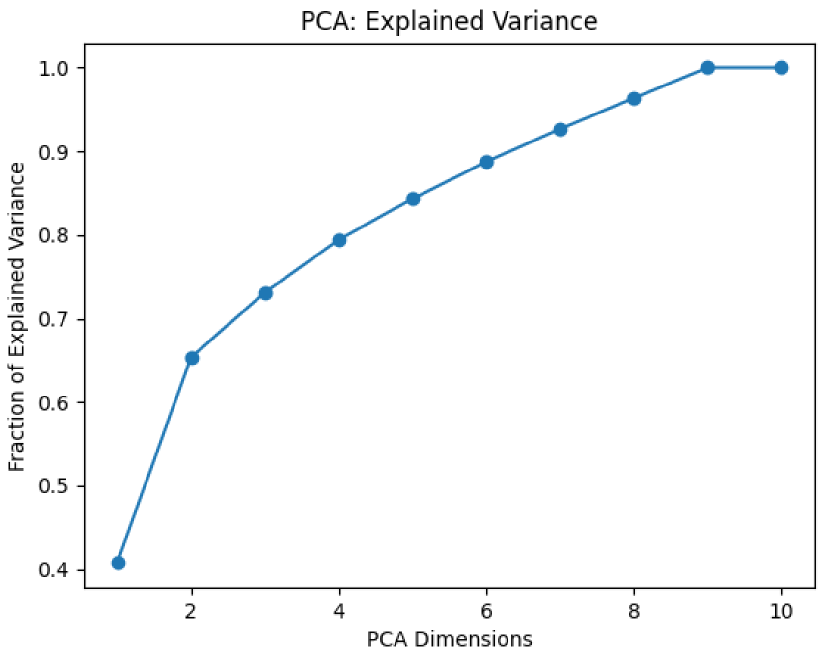


Figure 2: Histograms of the features

Looking at the heatmaps, it can be deduced that almost all features are poorly correlated and therefore it would not be useful to exploit feature dimensionality reduction techniques (PCA), but we will check this assumption later. By plotting the fraction of explained variance for each principal component, we can notice indeed that we can still keep most of the information by reducing the number of dimensions.



2 Classification and Validation

2.1 Introduction

In order to write the report, the collected results have been taken as the output of the following classifying models' list:

- Gaussian classifier
  - Multivariate Gaussian classifier (MVG)
  - MVG + Diagonal Covariance
  - MVG + Tied Covariance
  - MVG + Tied Covariance with Diagonal Covariance
- Logistic Regression
  - Prior weighted logistic regression
  - Quadratic logistic regression
- Support Vector Machine (SVM)
  - Linear SVM
  - Quadratic SVM (polynomial kernel function with degree=2)
  - SVM with Radial Basis kernel Function
- Gaussian Mixture Model (GMM)
  - Gaussian Mixture Models (GMM)
  - GMM + Diagonal Covariance
  - GMM + Tied Covariance

For what concerns validation, it is necessary to make the following clarifications:

- To search the best hyperparamters configuration for each model and to asses the effect of PCA, we use the K-fold cross-validation which has K=10.
- The result reported in this phase are minDCF. We verify the DCF in a second phase.
- Every model was trained with the effective prior π_T=$\frac{1}{11}$ as a project choice. In fact: (π,Cfn,Cfp)=(0.5,1,10)=($\frac{1}{11}$,1,1)
- When we apply preprocessing techniques such as z-score normalization, we apply them after PCA (if we use it), because our features do not have geometric meanings ; In fact our features consist in embeddings of fingerprint images, so we think that is better.

2.2 Multivariate Gaussian Classifier

Now, we focus on Multivariate Gaussian Classifiers (MVG), in particular: Full Covariances, Tied Covariance, Diagonal Covariances. These are generative models that work well if for each class the data are gaussian distributed:

$$X|C = c \sim N(\mu_c, \Sigma_c)$$

Tied MVG assumes that each class has its own mean μ_c as the other MVG models, but the same covariance matrix for all the classes:

$$X|C = c \sim N(\mu_c, \Sigma)$$

The diagonal model assumes that covariance matrices are diagonal matrices.

2.2.1 Expectations

Since our histograms have a similar gaussian flow, we expect that MVG performs well on our data. Regarding what we have said about Pearson correlation in the previous paragraph, also Naive Bayes could performs quite well. Our histograms show also that for each class, data are spreaded differently so they will have different covariance matrixes. We hypothesise that MVG + Tied Covariance will perform not as well as the other ones.

2.2.2 Results

Table 1: Gaussian Classifiers (minDCF)

Gaussian Classifiers (minDCF)			
MVG	MVG+Diag	MVG+Tied	MVG+Tied+Diag
NO PCA			
0.33	0.46	0.47	0.55
PCA(10)			
0.33	0.38	0.47	0.56
PCA(9)			
0.33	0.39	0.46	0.56

As we expected, the best results were brought from MVG. So, we can deduce that the models that perform better on our dataset are quadratic. We can notice that the results concerning MVG are the same; therefore we have decided to choose as candidate model for gaussian classifiers the MVG with PCA=9, in order to have a simpler model that tries to avoid overfitting.

2.3 Logistic Regression

Since our project decision is to use effective prior as prior we have implemented prior-weighted logistic regression to avoid using empirical prior.

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

This version of Logistic Regression applies linear separation rules, but it is possible to define non-linear separation rules by building a certain expanded feature space defined as:

$$\phi(x) = \begin{pmatrix} \text{vec}(xx^T) \\ \\ \mathbf{x} \end{pmatrix}$$

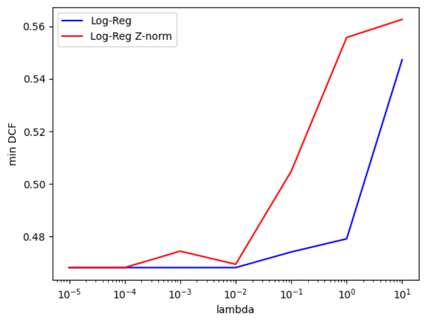
We can thus train the LR model defined above, but this time using the feature vectors $\phi(x)$ rather than x. It allows computing linear separation rules for $\phi(x)$, and this corresponds to estimating quadratic separation surfaces in the original space.

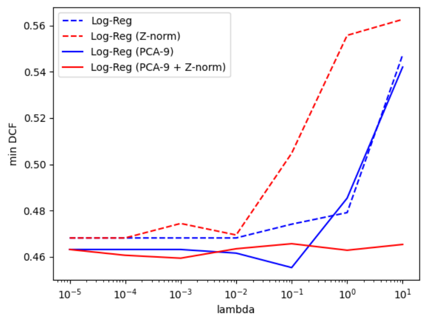
2.3.1 Expectations

Having said that for the gaussian classifiers the best results were obtained with MVG (which is a quadratic model), we expect that quadratic LR perform better than linear version, because our data seems to be better separated by quadratic rules. Moreover, the scale of the features does not play a crucial role in logistic regression's performance. So, applying z normalization, which primarily aims to address scale differences in the data, would not have a significant impact on the performance of logistic regression.

2.3.2 Results

We start plotting the minDCF w.r.t. different values of the regularization term λ, in order to discover the best configuration. We decide to use also the PCA as a pre-processing technique, given the observation made in the previuos paragraph.



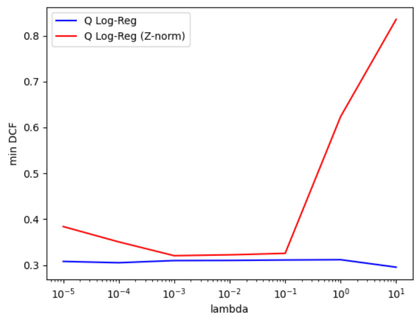


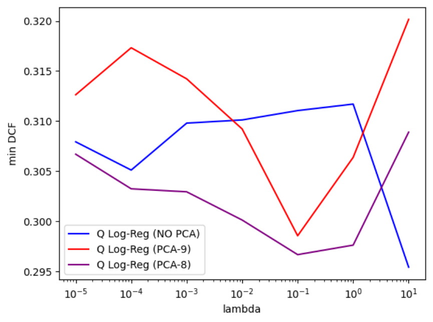
Looking at the graphs, we can see that tuning the lambda hyper parameter is useful, bringing benefits for values around 0.1 ; The use of PCA with 9 components also brought us benefits in minimising the value of the minDCF, while as far as z normalisation is concerned, we confirmed the above, obtaining worse results. In fact, even applying PCA together with z normalisation, we obtain worse results than with PCA alone (but better that the z normalization alone). Below are the relative minDCF values for the various cases (λ is fixed to 0.1):

Linear Logistic Regression (minDCF)		
λ	RAW	Znorm
	NO PCA	
0.00001	<u>0.46</u>	<u>0.46</u>
0.0001	<u>0.46</u>	<u>0.46</u>
0.001	<u>0.46</u>	0.47
0.01	<u>0.46</u>	<u>0.46</u>
0.1	0.47	0.50
1	0.47	0.55
10	<u>0.54</u>	0.56
	PCA(9)	
0.00001	0.46	0.46
0.0001	0.46	0.46
0.001	0.46	<u>0.45</u>
0.01	0.46	0.46
0.1	<u>0.45</u>	0.46
1	0.48	0.46
10	0.54	0.46

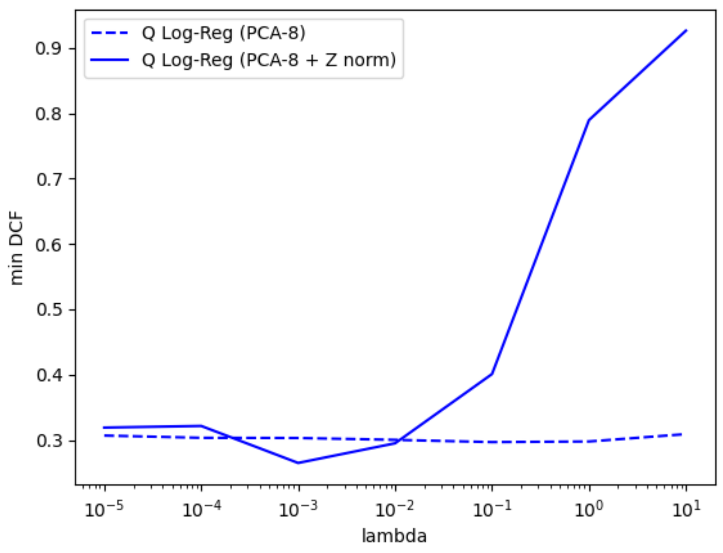
We can conclude by saying that the results obtained are of the order of those produced for the Gaussian tied models (as we expected, since both have linear classification rules). Therefore, we try the quadratic models, together with PCA practices and z-normalization, which we have seen can be useful.

2.3.3 Quadratic Logistic Regression





We can say that the quadratic model performs much better w.r.t. the linear model presented before, both with raw and pre-processed data, confirming our original expectation (by looking simply at the first graph). Since PCA worked well over the linear version of this model, we try to analyze the effects of PCA at several levels, by using a grid approach. PCA has given good results even with 8 components, therefore we decide to analyze also the effect that the z normalization could have (we exploit now a greedy approach, so without trying every possible combination)



Quadratic Logistic Regression (minDCF)		
λ	RAW	Znorm
	NO PCA	
0.1	0.31	0.32
	PCA(8)	
0.001	0.30	0.26

Indeed, the combination of PCA = 8 and z-normalization in this context gives the best result.

2.3.4 Conclusion

We can conclude that classes are better separated with quadratic decision rules. Now we'll test the remaining models.

2.4 Support Vector Machine

2.4.1 About SVM

The implemented SVM models are:

- **Linear SVM**: Obtained by solving the primal problem, expressed as the minimization of:

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

  where $\hat{x}_i = \begin{bmatrix} x_i \\ K \end{bmatrix}$ and $\hat{w} = \begin{bmatrix} w \\ b \end{bmatrix}$. The regularization term K mitigates the effects of the bias term due to the regularization of the norm of $\hat{w}$. In fact, it is equal to $\|\hat{w}\|^2 = \|w\|^2 + b^2$.

- **Quadratic SVM**: Obtained by solving the dual problem, expressed as the maximization of:

$$\alpha^T(1 - \frac{1}{2}\alpha^T H\alpha)$$

  subject to $0 \leq \alpha_i \leq C$ for $i = 1, \ldots, n$ and $\sum_{i=1}^{n} \alpha_i z_i = 0$. Directly solving this expression would increase the complexity of score computation. Hence, a kernel function is employed. The kernel function computes the dot product in the expanded space $\Phi(x_i)^T\Phi(x_j)$. Formally, the kernel function is defined as $k(x_1, x_2) = \Phi(x_1)^T\Phi(x_2)$. To add a regularized bias in the non-linear SVM, a constant value $\xi = K^2$ is added to the kernel function:

$$\hat{k}(x_1, x_2) = k(x_1, x_2) + \xi$$

  The kernel function can be computed in different ways. For the polynomial kernel of degree d, it is:

$$k(x_1, x_2) = (x_1^T x_2 + 1)^d$$

- **SVM with Radial Basis Kernel Function**: Obtained by solving the same dual problem as above, but in this case, the kernel function is defined as:

$$k(x_1, x_2) = e^{-\gamma\|x_1 - x_2\|^2}$$

  where γ is a hyper-parameter.

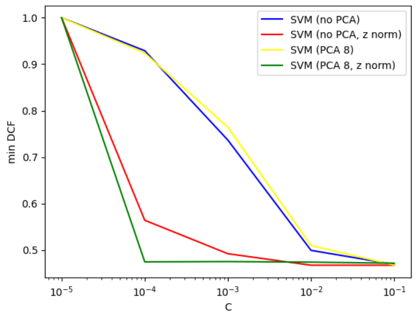
2.4.2 Expectations

As before, we expect that quadratic separation surfaces will perform better than linear ones for the same reasons already cited.

2.4.3 Results

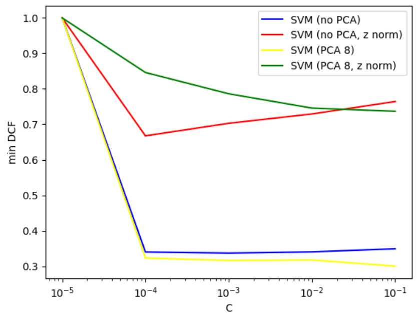
2.4.4 Linear SVM

Since we do not expect the linear svm to give any benefits, we simply test it with various C and only K = 0. Below are graphs showing that the minDCF does not improve on previous quadratic models.

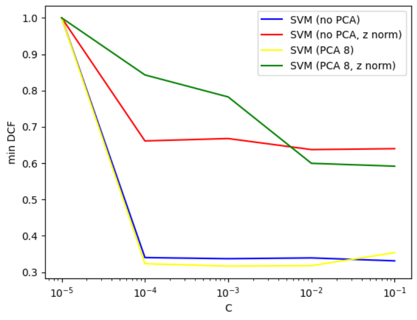


2.4.5 SVM with polynomial kernel

We can now exploit the fact that the svm with polynomial kernel behaves similarly to quadratic logistic regression, so that we can already use the most suitable preprocessing technique for the latter on the svm, and see how it performs.



K=0



K=1

We can say that models without pca and with pca=8 perform best, especially those using K=0 and C=0.1;

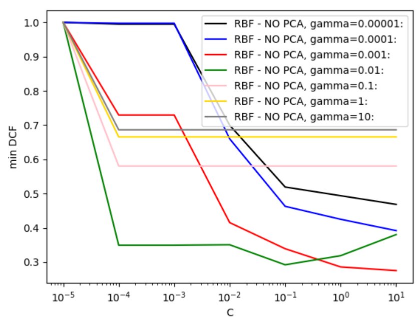
SVM with polinomial kernel (degree 2)			
K	C	RAW	Znorm
		NO PCA	
0	0.0001	0.34	<u>0.66</u>
0	0.001	<u>0.33</u>	0.70
0	0.01	0.34	0.72
0	0.1	0.34	0.76
		PCA = 8	
0	0.0001	0.32	0.84
0	0.001	0.31	0.78
0	0.01	0.31	0.74
0	0.1	<u>0.30</u>	<u>0.73</u>

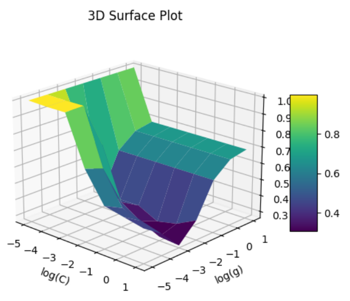
SVM with polinomial kernel (degree 2)

K	C	RAW	Znorm
NO PCA			
1	0.0001	0.34	0.66
1	0.001	0.33	0.66
1	0.01	0.33	0.63
1	0.1	0.33	0.63
PCA = 8			
1	0.0001	0.32	0.84
1	0.001	0.31	0.78
1	0.01	0.31	0.59
1	0.1	0.35	0.59

2.4.6 SVM with RBF kernel

Since K=0 performed better than K=1, we decide to use this configuration for the next measurements. In order to find a first possible configuration for the hyper parameters, i.e. C, K, and gamma, we again follow a grid approach, where we initially try for various values of gamma and C (fixing K=0)

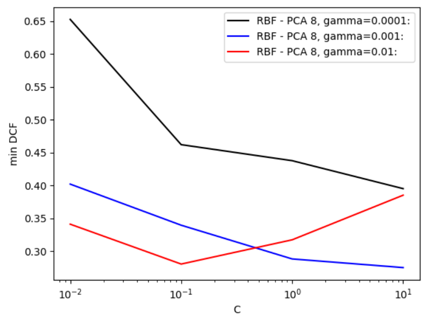


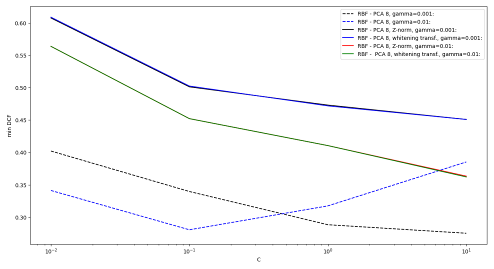


SVM with RBF kernel

	C=0.01	C=0.1	C=1	C=10
NO PCA				
γ==0.00001	0.69	0.51	0.49	0.46
γ==0.0001	0.65	0.46	0.42	0.39
γ==0.001	0.41	0.33	<u>0.28</u>	<u>0.27</u>
γ==0.01	<u>0.35</u>	<u>0.29</u>	0.31	0.38
γ==0.1	0.58	0.58	0.58	0.58
γ==1	0.66	0.66	0.66	0.66
γ==10	0.68	0.68	0.68	0.68

We can now go and test the pca that performed best previously, i.e. pca=8. We do this by exploiting a subset of the original parameters, as the graphs suggest we can benefit from having $-4 \leq \log(g) \leq -2$ and $-2 \leq \log(C) \leq 10$. We describe inside the following table only the best results we obtained globally for each model (without specifying all the C values, apart for the best model)





SVM with RBF kernel			
	RAW	Z-Norm	Whitening
PCA = 8			
γ==0.001	0.27	0.45	0.45
γ==0.01	0.28	0.36	0.36

We can see that pca does not bring substantial improvements, but it does allow us to work with fewer features for the same minDCF (the best is the one with gamma = 0.001, and C=10). As for the use of z-normalisation and whitening transformation, they bring absolutely no benefit.

2.5 Gaussian Mixture Model

The last model we consider is Gaussian Mixture Model (GMM). As we said above, we decide to use our dataset with PCA=9, in order to have a more simple model that avoids overfitting and has a good trade-off between performance and training time. By the fact that the class 'Spoofed' is composed by 6 sub-classes and 'Authentic' not, we have decided to make cross-validation between a different number of components for each class. In Particular:

- (1,2) components for class 'Authentic' (class 1).
- (2,4,8) components for class 'Spoofed' (class 0).

2.5.1 Expectations

GMM can approximate generic distribution, so we expect that these models perform better than MVG model.

2.5.2 Results

minDCF Results for GMM Experiments

Components	Full-Cov	Diag-Cov	Tied Full-Cov
(2,1)	0.28	0.32	0.33
(4,1)	0.26	0.27	0.27
(8,1)	0.26	<u>0.25</u>	<u>0.25</u>
(2,2)	0.30	0.30	0.33
(4,2)	0.26	<u>0.25</u>	0.27
(8,2)	0.27	0.26	<u>0.25</u>

Seeing that we have more than one best result, we have to choose which of these could be the best candidate. Firtsly, we decide to take the models made with components (8,1) because they are simpler and closer to the description of the dataset made in the project assignment. We decide to choose the tied full-covariance model, only because each diag-covariance model reaches 0.25 by a default approximation (0.253) instead tied-covariance model reaches 0.25 by an excess approximation (0.245).

2.6 Calibration

Summarizing, these are the best models obtained in validation:

Model	Min DCF
Quadratic Logistic Regression($\lambda = 0.001$,PCA(8)+Z-norm)	0.26
SVM with RBF Kernel($\gamma = 0.001$,K=0,C=10,PCA(8))	0.27
GMM+Tied-Cov((8,1),PCA(9))	0.25

Now we see the difference between the minDCF and actDCF based on the raw scores, and visualize the 3 error Bayes plots:

Model	Min DCF	actDCF
Quadratic Logistic Regression($\lambda = 0.001$,PCA(8)+Z-norm)	0.27	0.60
SVM with RBF Kernel($\gamma = 0.001$,K=0,C=10,PCA(8))	0.22	0.42
GMM+Tied-Cov((8,1),PCA(9))	0.22	0.25

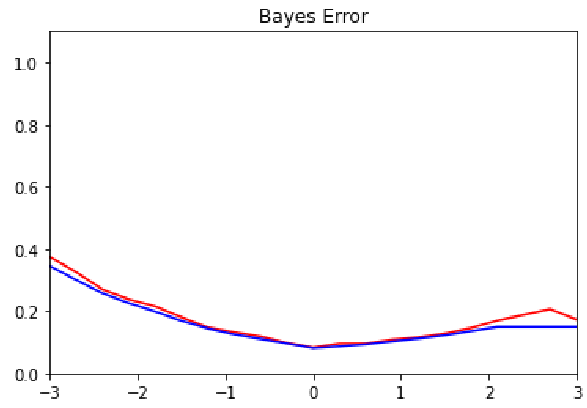


Figure 3: GMM Tied Covariance Error Bayes Plot

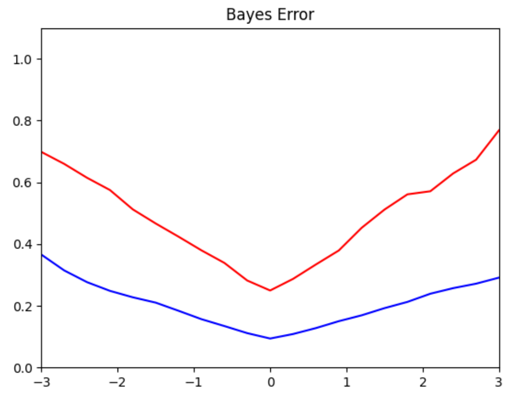


Figure 4: Quadratic Logistic Error Bayes Plot

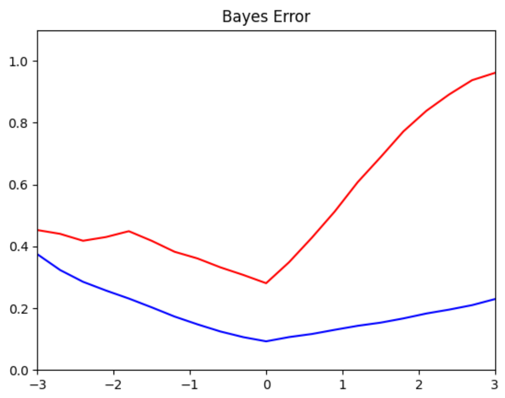


Figure 5: RBF SVM Error Bayes Plot

As we can see, GMM with Tied Covariance model is well calibrated while SVM model and Q-LR model are not well calibrated either in our point of work or in other applications. We need to calibrate them. This means that they can be improved by applying a transformation function f to previously computed scores, in order to obtain calibrated scores. To estimate f, discriminative score models such as Logistic Regression, can be used by passing the scores to the model. They would act as if they were a feature of the model. Following this path, f(s) can be interpreted as the log-likelihood ratio:

$$f(s) = \log \frac{f_{S|C}(s \mid H_T)}{f_{S|C}(s \mid H_F)} = \alpha s + \beta$$

while the class posterior probability for a given prior $\widetilde{\pi}$ would be equal to:

$$\log \frac{P(C = H_T \mid s)}{P(C = H_F \mid s)} = \alpha s + \beta + \log \frac{\tilde{\pi}}{1 - \tilde{\pi}}$$

where $\widetilde{\pi}$ is a model parameter which represents the application we are optimizing the scores for. Finally, to retrieve the calibrated scores, f(S) can thus be computed as:

$$f(s) = \alpha s + \beta = \alpha s + \beta' - \log \frac{\tilde{\pi}}{1 - \tilde{\pi}}$$

In our case, we used a single-split on the training set, dividing it into two in the following way: 1/10 for the training part of the calibration model (since we do not need too much data) and 9/10 for the calibration part. Below are the bayes error plots of the calibrated models;

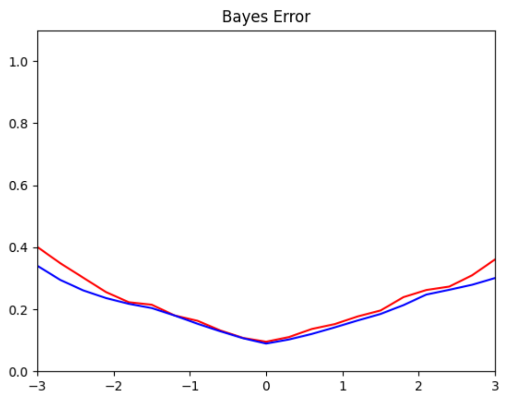


Figure 6: Calibrated Quadratic Logistic Error Bayes Plot

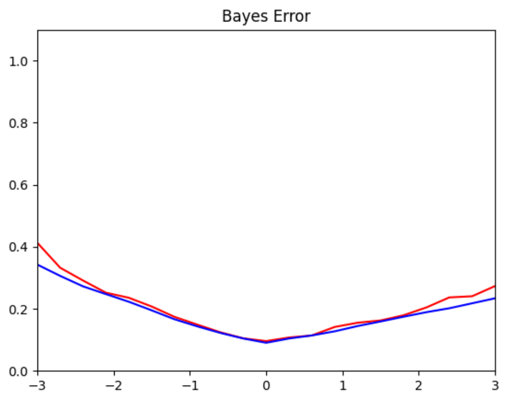


Figure 7: Calibrated RBF SVM Error Bayes Plot

As we can see, the calibration brought important benefits to both models. We will use the same approach for calibration during the evaluation phase, i.e. a single split on the training set. Anyway, **we believe that the GMM-Tied-Cov+PCA(9) will outperforms the others**.

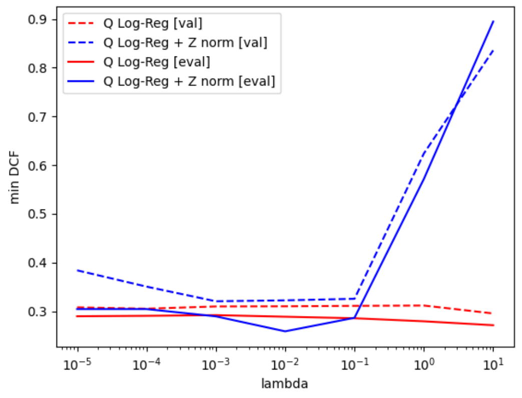
3 Evaluation Phase

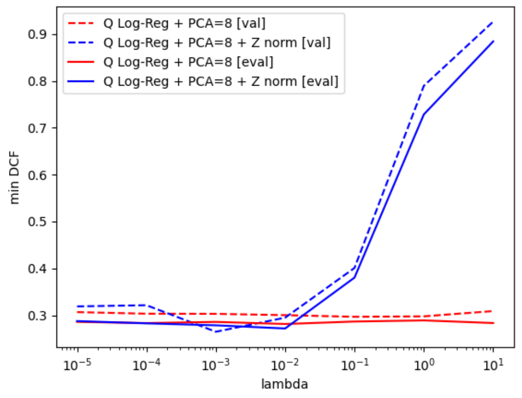
3.1 Introduction

We now turn the attention to the evaluation part. We check that the decision taken in the validation part are fine. As before, we compare them with minDCF. Since the validation has been performed with 10-fold cross-validation, the evaluation part will be executed using the entire training set. When we work on models that need calibration instead, we'll split the training set in two, in order to take into account also for the calibration set. It's worth noting that all the predictions used, were made with calibrated scores (if necessary).

3.2 Logistic Regression

Taking into account the fact that linear models perform worse than quadratic ones, we start comparing the quadratic logistic regression models by checking that our choices have been consistent with what we have done until now,.





In general, the results have been consistent with our expectations. The model, w.r.t both the validation and evaluation part, shows essentially the same trend. The graphs also show that the choice of hyper-parameters was optimal, as the mindcf related to this model is given by the parameter configurations ($\lambda = 0.001$) and the pre-processing techniques we adopted before(z-normalisation and pca=8).

Quadratic Logistic Regression (minDCF) - Evaluation		
λ	RAW	Znorm
NO PCA		
0.1	0.28	0.28
PCA(8)		
0.001	0.28	0.27

We can also observe that the model works well in other applications, by looking its error Bayes plot (Figure 8).

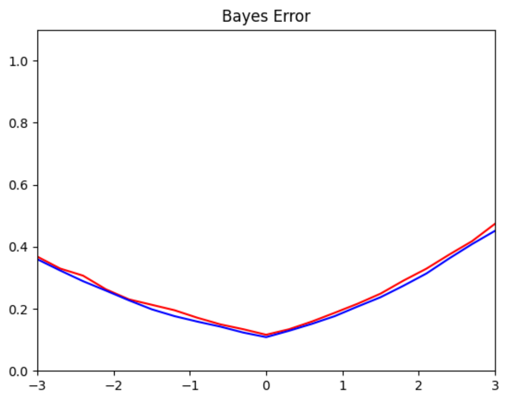


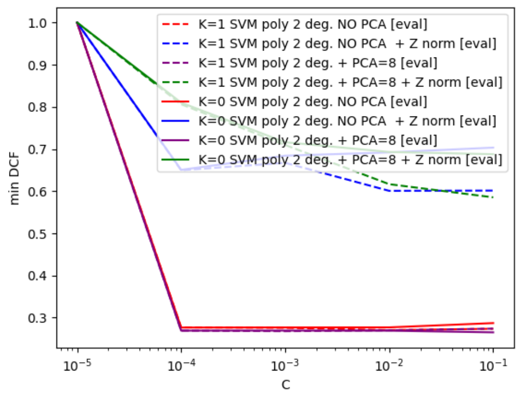
Figure 8: Quadratic Logistic Regression Error Bayes Plot with evaluation set (calibrated)

3.3 SVM

For the same reasons as before, we decide to omit the part concerning the linear svm model.

3.3.1 SVM with polinomial kernel (degree 2)

In order to verify that we have chosen the right K-value, we plot the results using the same configurations exploited during validation.



In this case, the model behaves in the same way as the one used in the previous step, not favouring K=1, albeit slightly. It shows also that pca=8 could benefit the model, as indeed is shown in the following tables:

SVM with polinomial kernel (degree 2) - Evaluation

K	C	RAW	Znorm
		NO PCA	
0	0.0001	0.27	0.65
0	0.001	0.27	0.68
0	0.01	0.27	0.69
0	0.1	0.28	0.70
		PCA = 8	
0	0.0001	0.26	0.81
0	0.001	0.26	0.71
0	0.01	0.26	0.69
0	0.1	0.26	0.68

SVM with polinomial kernel (degree 2)			
K	C	RAW	Znorm
NO PCA			
1	0.0001	0.27	0.64
1	0.001	0.27	0.66
1	0.01	0.27	0.60
1	0.1	0.27	0.60
PCA = 8			
1	0.0001	<u>0.26</u>	0.80
1	0.001	<u>0.26</u>	0.70
1	0.01	<u>0.26</u>	0.61
1	0.1	0.27	0.58

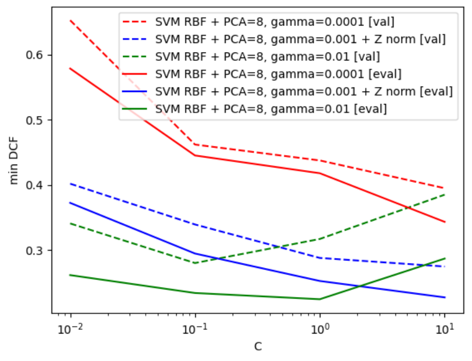
3.3.2 SVM with RBF kernel

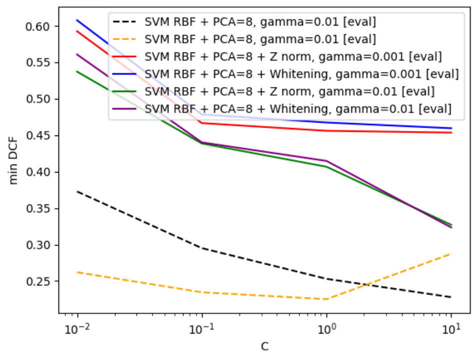
In order to understand whether the range of values taken is effective, we check, as before, that there is a zone of promising values within it.

SVM with RBF kernel - Evaluation

	C=0.01	C=0.1	C=1	C=10
NO PCA				
γ=0.00001	0.63	0.48	0.47	0.45
γ=0.0001	0.59	0.45	0.42	0.34
γ=0.001	0.37	0.29	0.25	0.24
γ=0.01	0.28	0.24	0.24	0.29
γ=0.1	0.47	0.47	0.47	0.47
γ=1	0.96	0.96	0.94	0.94

This confirms the validity of the interval taken, seeing how RBF SVM performs pretty well between $-4 \leq \log(g) \leq -2$ and $-2 \leq \log(C) \leq 1$. In particular, let's see if the configuration we considered the best, is actually consistent for the evaluation.





As predicted, the model did not improve using techniques such as z-norm and whitening, but only using pca=8 and gamma=0.001. However, the choice of parameters was optimal.

SVM with RBF kernel - Evaluation			
	RAW	Z-Norm	Whitening
	PCA = 8		
γ==0.001	0.22	0.45	0.45
γ==0.01	0.22	0.32	0.32

We can also observe that the chosen model works well in other applications, watching its error Bayes plot (Figure9).

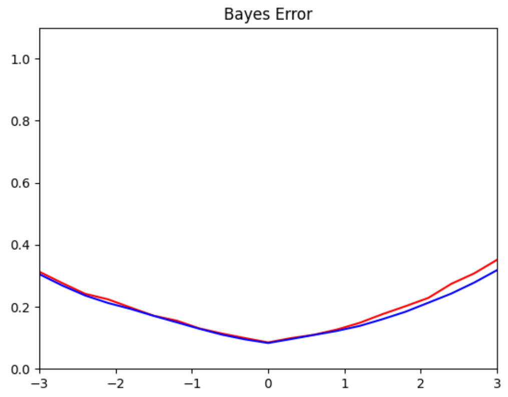


Figure 9: SVM with RBF kernel Error Bayes Plot with evaluation set (calibrated)

3.3.3 GMM

In terms of minDCF, we can confirm that our model is one of the bests, with $minDCF_{eval}$=0.22. We can also observe that the model works well in other applications, by watching its error Bayes plot (Figure 10).

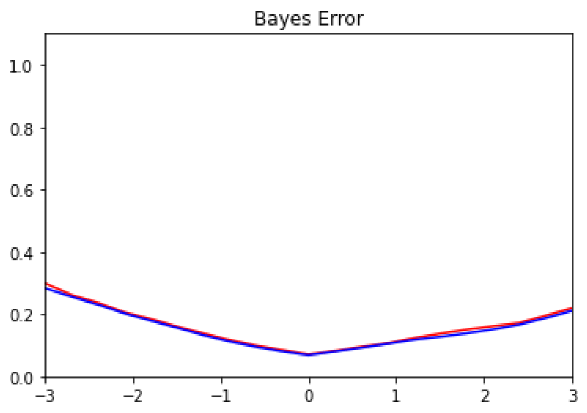


Figure 10: GMM Tied Covariance Error Bayes Plot with evaluation set

3.4 Conclusion

Model	Min DCF	actDCF
Quadratic Logistic Regression($\lambda = 0.001$,PCA(8)+Z-norm)	0.27	0.29
SVM with RBF Kernel($\gamma = 0.001$,K=0,C=10,PCA(8))	0.22	<u>0.23</u>
GMM+Tied-Cov((8,1),PCA(9))	0.22	<u>0.23</u>

Since in terms of actDCF, SVM with RBF kernel and GMM+Tied-Covariance are equivalent, we have to choose one of them. We choose the SVM model because being a discriminative approach, could potentially outperform the GMM model in contexts where there are more data.