Politecnico di Torino

Fingerprint spoofing detection

Riccardo Cardona, Nicholas Berardo

July 12, 2023

Contents

Chapter 1

Introduction

1.1 Abstract

The goal is to build the best model for the Fingerprint-Spoofing Detection task, using the models developed during the laboratory and the lessons.

1.2 Overview

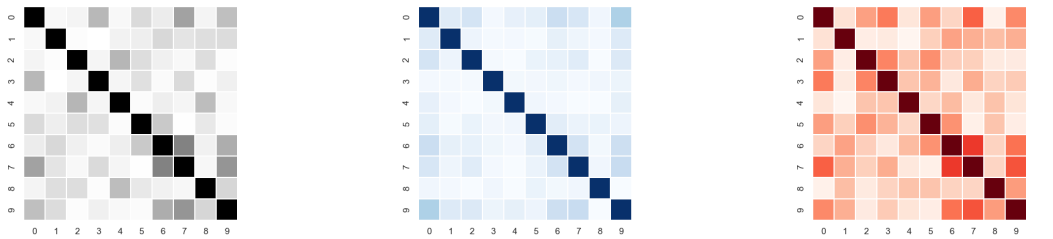
The data set and test set are composed of 10 dimensional features. The last column is the class label which is **1** for the *authentic fingerprint* and **0** for the *spoofed fingerprint*.

The datasets are imbalanced, we have more spoofed fingerprint than the authentic fingerprint. The Training set is composed of 2325 samples and the Test set is composed of 7704 samples.

The application we are going to build has π =0.5 meaning that we have the same prior probability for both classes, but we have different costs. The cost of False Negative C_{fn} is 1 but the cost of False Positive C_{fp} is 10, so classifying a spoofed fingerprint as an authentic one has an higher cost due to security vulnerability.

1.3 Features Analisys

As we said, the Fingerprint Spoofing Detection task has 10 features. Here below we are going to analyze them.

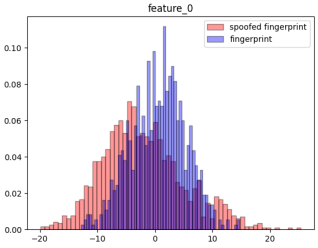


(a) All classes (b) Authentic Fingerprint (c) Spoofed Fingerprint

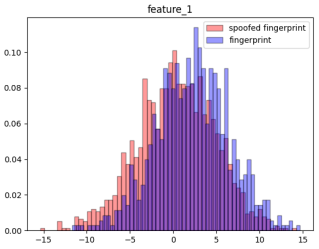
Figure 1.1: Heatmap

As we can see from the Figure 1.1, the correlation between the features is very small. This lead us to think that using PCA for dimensionality reduction is useless (it could also remove some important information!), but we will try using it.

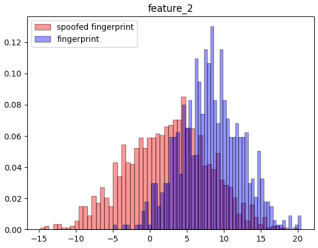
Now we are going to see the histograms for each of the 10 features. It's easy to see that each feature has a Gaussian distribution.



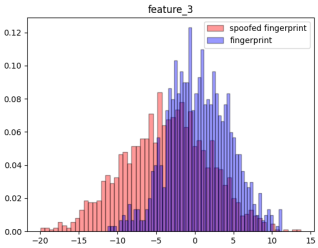
(a) RAW Feature 0



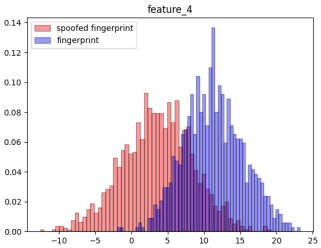
(b) RAW Feature 1



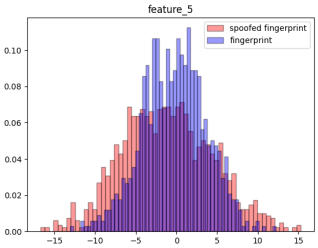
(c) RAW Feature 2



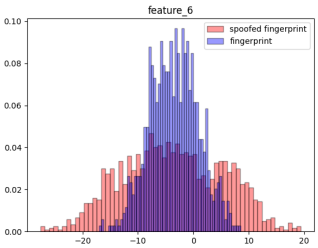
(d) RAW Feature 3



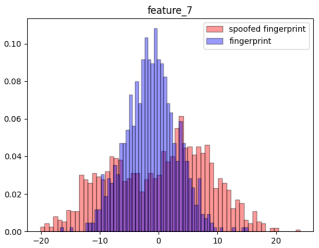
(e) RAW Feature 4



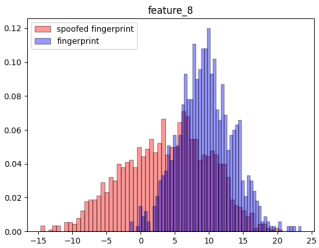
(f) RAW Feature 5



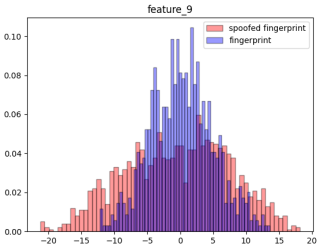
(g) RAW Feature 6



(h) RAW Feature 7



(i) RAW Feature 8



(j) RAW Feature 9

Figure 1.2: RAW Feature Histogram

1.4 PCA Usage

PCA (Principal Component Analysis) is a Dimensionality Reduction technique that can be used to remove noise, simplify classification and data visualization. It can be also used to avoid overfitting, but it may produce underfitting if too many dimensions are removed (removing too much useful information). So it is a useful technique, but sometimes it can be useless because if used could lead only to bad models.

What does PCA do?

First of all we need to compute the mean of the Dataset μ, then we need to compute the covariance matrix

$$C = \frac{1}{K}\sum_{i=1}^{K}(x_i - \mu)(x_i - \mu)^T$$

From C we need the eigen-decomposition $C = U\Sigma U^T$, where U is a matrix that contains the eigen-vectors and Σ is a diagonal matrix that contains the eigen-values in descending order. The solution are the m eigen-vectors of U corresponding to the m highest eigen-values of Σ.

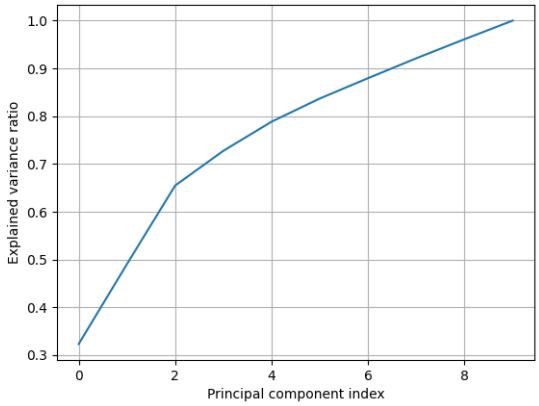


Figure 1.3: PCA variance

As we can see, if we use 9 dimensions (Number 8 on the principal component index), we are going to reach a 96% of explainance, then if we use 8 dimensions we will explain 91% of the data variance. Starting from 7 to 1 dimensions, we are lower than 90%.
We will consider only 9 and 8 as PCA parameter.

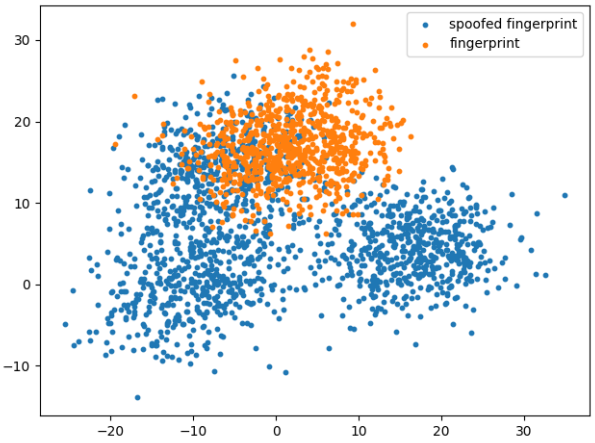


Figure 1.4: PCA

From this plot (Figure 1.4) we can see the distribution of the 2 main features.

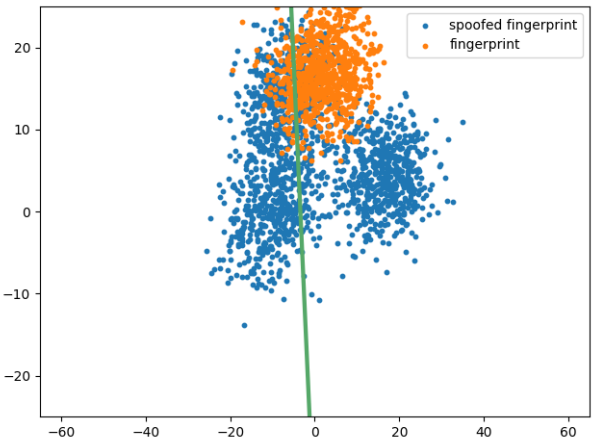


Figure 1.5: LDA

From this plot (Figure 1.5) we can see the distribution of the 2 main features using both PCA and LDA

1.5 Training

- We are going to use a K-Fold Cross Validation approach, with K = 5
- The minDCF computation is computed based on our application (π,C_{fn},C_{fp}) = (0.5,1,10), but we have computed the effective prior, which is approximated to (π,C_{fn},C_{fp}) = (0.09,1,1).
  We have also tried this two different application:
  - (π,C_{fn},C_{fp}) = (0.1,1,1)
  - (π,C_{fn},C_{fp}) = (0.9,1,1)

Chapter 2

Classification and Validation

2.1 Introduction

We are going to develop this following models:

- Generative Models
  - Multivariate Gaussian Classifier (**MVG**)
  - MVG Tied Covariance (Only one covariance matrix for everything)
  - MVG Naive (For each covariance matrix takes out only the diagonal)
  - MVG Tied Naive (Only one covariance matrix, but taking only the diagonal)
- Logistic Regression (**LR**)
  - Linear Logistic Regression
  - Quadratic Logistic Regression
- Support Vector Machine (**SVM**)
  - Linear SVM
  - Polynomial SVM
  - Radial Basis Function SVM
- Gaussian Mixture Models (**GMM**)
  - Full Covariance Matrix
  - Diagonal Covariance Matrix
  - Tied Full Covariance Matrix
  - Tied Diagonal Covariance Matrix

2.2 Multivariate Gaussian Classifiers

We are going to analyze all the Gaussian Classifiers, using the Multivariate Gaussian Classifier and all it's variants:

- Tied Covariance
- Naive Bayes
- Tied Naive Bayes

For all these 4 models we have 2 parameters: $\theta = (\mu, \Sigma)$. We can easily estimate them starting from the likelihood w.r.t. θ for the **M.V.G.**

$$\mathcal{L}(\theta) = \prod_{c}^{K} \prod_{i|c_i=c} \mathcal{N}(x_i|\mu_c, \Sigma_c)$$

Then we can build the log-likelihood because it's easier to work with.

$$l(\theta) = \sum_{c}^{K} \sum_{i|c_i=c} \log \mathcal{N}(x_i|\mu_c, \Sigma_c)$$

Now we define $l_c(\theta) = \sum_{i|c_i=c} \log \mathcal{N}(x_i|\mu_c, \Sigma_c)$ and compute the $\nabla_\mu l_c(\mu_c, \Sigma_c) = 0$ and the $\nabla_\Sigma l_c(\mu_c, \Sigma_c) = 0$ obtaining

$$\mu_c^* = \frac{1}{N_c} \sum_{i|c_i=c} x_i$$

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

We can do the same thing for the **Naive Bayes** model (that has diagonal covariance matrix)

$$l(\theta) = \sum_{c}^{K} \sum_{i|c_i=c} \sum_{j=1}^{D} \log \mathcal{N}(x_{i,[j]}|\mu_{c,[j]}, \Sigma_{c,[j]})$$

which solutions are

$$\mu_{c,[j]}^* = \frac{1}{N_c} \sum_{i|c_i=c} x_{i,[j]}$$

$$\Sigma_{c,[j]}^* = \frac{1}{N_c} \sum_{i|c_i=c} (x_{i,[j]} - \mu_{c,[j]}^*)(x_{i,[j]} - \mu_{c,[j]}^*)^T$$

Finally, the **Tied** model can be built in the same way (has a single covariance matrix for all classes)

$$l(\theta) = \sum_{c}^{K} \sum_{i|c_i=c} \log \mathcal{N}(x_i|\mu_c, \Sigma)$$

which solutions are

$$\mu_c^* = \frac{1}{N_c} \sum_{i|c_i=c} x_i$$

$$\Sigma^* = \frac{1}{N} \sum_{c}^{K} \sum_{i|c_i=c} (x_i - \mu_c^*)(x_i - \mu_c^*)^T$$

The tables below show the computation of the minDCF on the validation set (extracted using K-Fold Cross Validation with K = 5). As said before, we are going to consider the main application (0.5,1,10) and 2 other applications (0.1,1,1) and (0.9,1,1). There are also some computation of PCA with m=9 and m=8.

Classifier	pi = 0.1	pi = 0.5	pi = 0.9
	RAW Features - NO PCA		
MVG	0.616	0.331	0.111
Naive Gaussian	0.806	0.472	0.144
Tied Gaussian	0.706	0.486	0.184
Naive Tied Gaussian	0.79	0.551	0.198

Classifiers	pi = 0.1	pi = 0.5	pi = 0.9
	PCA (m=9)		
MVG	0.629	0.33	0.109
Naive Gaussian	0.74	0.369	0.113
Tied Gaussian	0.696	0.492	0.18
Naive Tied Gaussian	0.772	0.543	0.202

Classifiers	pi = 0.1	pi = 0.5	pi = 0.9
	PCA (m=8)		
MVG	0.612	0.333	0.109
Naive Gaussian	0.712	0.36	0.112
Tied Gaussian	0.686	0.485	0.181
Naive Tied Gaussian	0.774	0.544	0.201

As we can see, the MVG model, that consider all the covariance matrices, is the best either with PCA or without. PCA is useful only in the Naive Gaussian model. Tied models perform worse.

Overall the best candidate is the **MVG classifier** without PCA. However, we will see that this kind of model is also useless for our imbalanced data.

2.3 Logistic Regression

We are going to analyze both Linear Logistic Regression and Quadratic Logistic Regression.

2.3.1 Linear Logistic Regression

Logistic Regression is a discriminative model that directly computes the posterior probability $P(C = c|X = x)$. The model parameters are (w,b) that can be estimated using a frequentist approach. There's also an hyper-parameter λ that must be estimated by cross-validation. λ is needed to avoid overfitting, if it's too high then the model will underfit but if it's too low then the model will overfit.

The plots show how minDCF is affected by different values of λ. They are exploited to calibrate λ, which is the regularization coefficient.

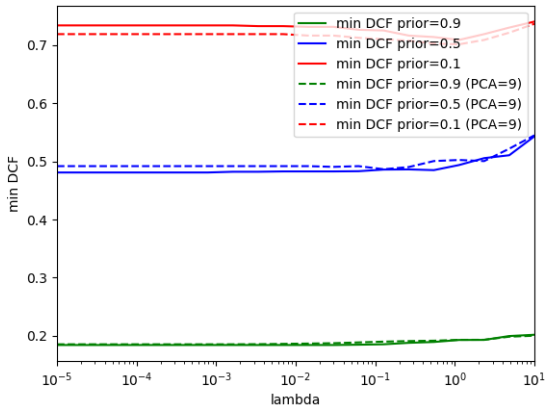


Figure 2.1: DCF - Linear LogReg

The Figure 2.1 shows that using a $\lambda = 10^{-5}$ or $\lambda = 10^{-1}$ is more or less the same, the minDCF doesn't change that much. From $\lambda = 10^{-1}$ to $+\infty$ the minDCF will be too high.

For our main application, it's hard to see, but $\lambda = 0.4$ is the best choice.

	pi=0.1	pi=0.5	pi=0.9
		NO-PCA	
Log Reg ($\lambda = 0.4$)	0.715	0.485	0.189
		PCA m=9	
Log Reg ($\lambda = 0.4$)	0.705	0.496	0.193
		PCA m=8	
Log Reg ($\lambda = 0.4$)	0.689	0.483	0.188

Overall, the MVG without PCA has a **minDCF = 0.331** and this value is better than all the Linear Logistic Regression values (for our main application)

2.3.2 Quadratic Logistic Regression

The Figure 2.2 shows how minDCF is affected by different values of λ. Also in this case, for our main application, taking a λ=10^{-5} or λ=10^{-1} doesn't change the minDCF that much.

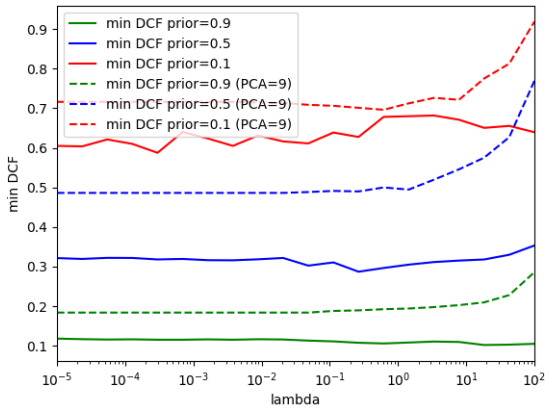


Figure 2.2: DCF - Quadratic LogReg

It is also easy to see that PCA is pretty useless for all applications. The minDCF it's always worse than the model without PCA.

	pi=0.1	pi=0.5	pi=0.9
		NO-PCA	
Log Reg ($\lambda = 0.4$, $\pi_T = 0.1$)	0.6	0.31	0.119
Log Reg ($\lambda = 0.4$, $\pi_T = 0.5$)	0.644	0.305	0.106
Log Reg ($\lambda = 0.4$, $\pi_T = 0.9$)	0.707	0.335	0.105
		PCA m=9	
Log Reg ($\lambda = 0.4$, $\pi_T = 0.1$)	0.691	0.489	0.189
Log Reg ($\lambda = 0.4$, $\pi_T = 0.5$)	0.7	0.497	0.191
Log Reg ($\lambda = 0.4$, $\pi_T = 0.9$)	0.713	0.506	0.194
		PCA m=8	
Log Reg ($\lambda = 0.4$, $\pi_T = 0.1$)	0.691	0.483	0.191
Log Reg ($\lambda = 0.4$, $\pi_T = 0.5$)	0.691	0.483	0.191
Log Reg ($\lambda = 0.4$, $\pi_T = 0.9$)	0.691	0.483	0.191

This kind of model, with a Quadratic Kernel outperforms the MVG models, for our main application.

2.4 Support Vector Machine

In this section we are going to see how Linear SVM, Polynomial SVM and RBF SVM will handle our application

SVM - Linear - Raw Feature

Linear SVM can be obtained by the minimization of the primal problem:

$$\hat{J}(\hat{w}) = \frac{1}{2}||\hat{w}||^2 + C\sum_{i=1}^{n} max_i(0, 1 - z_i(\hat{w}^T\hat{x_i}))$$

where

$$\hat{x_i} = \left[\hat{x_i}K\right], \hat{w_i} = \left[wb\right]$$

and K is a regularization term.

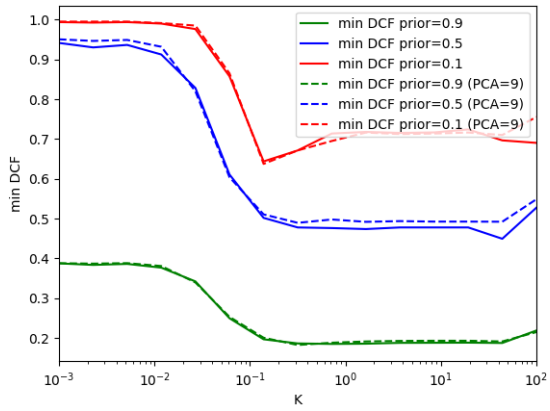


Figure 2.3: Linear SVM with C=1

From Figure 2.3 we can see how K is affected (using C = 1). It can be seen that values of K between 10^{-1} and 10^1 have a low DCF. In our cases we are going to try 3 different values of $K = [0.1, 1, 10]$.

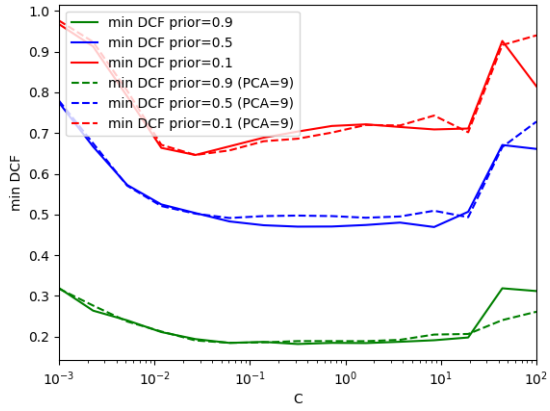


Figure 2.4: Linear SVM with K=1

Figure 2.4 shows the different values of C while setting K = 1. Again, it can be seen that the optimal values of C are the ones between 10^{-1} and 10^1. We are going to test 4 different values of $C = [0.01, 0.1, 1, 10]$. We also decided to test 0.01 even if the minDCF is "optimal" only for the $\pi = 0.1$

Here the table containing the results:

	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 0.01	0.99	0.68	0.71
C = 0.1	0.964	0.672	0.713
C = 1.0	0.67	0.722	0.715
C = 10.0	0.956	0.71	0.7
		π=0.5	
C = 0.01	0.921	0.532	0.473
C = 0.1	0.78	0.476	0.479
C = 1.0	0.532	0.466	0.473
C = 10.0	0.666	0.492	0.47
		π=0.9	
C = 0.01	0.379	0.216	0.187
C = 0.1	0.321	0.186	0.188
C = 1.0	0.216	0.184	0.188
C = 10.0	0.263	0.197	0.187

SVM - Linear - PCA with m = 9

We also tried to apply PCA with m = 9 since from Figure 2.3 and Figure 2.4 we can see that we can achieve also a good minDCF.

	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 0.01	0.991	0.704	0.701
C = 0.1	0.971	0.669	0.718
C = 1.0	0.694	0.709	0.709
C = 10.0	0.685	0.672	0.73
		π=0.5	
C = 0.01	0.973	0.538	0.496
C = 0.1	0.776	0.493	0.492
C = 1.0	0.54	0.494	0.489
C = 10.0	0.518	0.491	0.492
		π=0.9	
C = 0.01	0.382	0.216	0.187
C = 0.1	0.321	0.182	0.191
C = 1.0	0.214	0.19	0.194
C = 10.0	0.223	0.196	0.195

From these results we can conclude that using or not using PCA (with m = 9, so with almost all features) doesn't change that much.

SVM - Quadratic - Raw Feature with degree = 2

We are going to apply a non-linear separation algorithm. We have seen two kind of non-linear separation. In this section we are going to see the Quadratic one, in the next section there's the RBF.

The kernel is a function that allows the computation of a linear separation in the expanded feature space that corresponds to a non-linear separation in the original feature space. For the Quadratic kernel we have:

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

where d identifies the degree, in our case d = 2 and c is a Constant that in our case will take values 1 and 0.

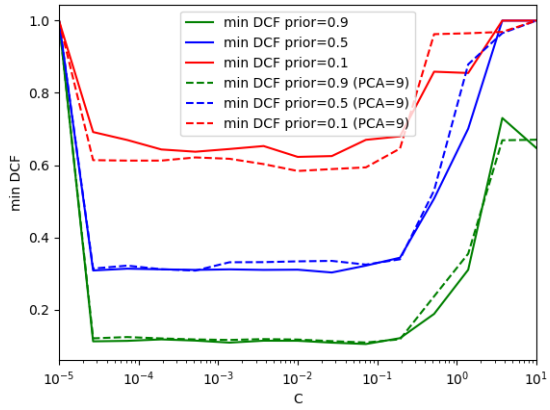


Figure 2.5: Poly K=1, c=1, d=2

Figure 2.5 shows the different values of C with K=1, c=1 e d=2. We are going to try $C = [0.01, 0.1, 1, 10]$, but from the plot it's easy to see that C=1 and C=10 won't be good.

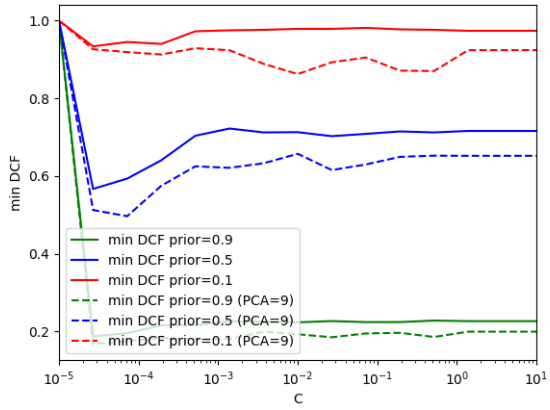


Figure 2.6: Poly K=1, c=1, d=3

Figure 2.6 shows the different values of C with K=1, c=1 but d=3. It's much worse than using d=2, so we decided to don't even show the results of this kernel.

Constant = 0	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 0.01	0.71	0.704	0.682
C = 0.1	0.713	0.674	0.678
C = 1.0	0.831	0.922	0.84
C = 10.0	0.982	0.994	0.975
		π=0.5	
C = 0.01	0.337	0.346	0.342
C = 0.1	0.324	0.344	0.346
C = 1.0	0.528	0.551	0.49
C = 10.0	0.947	0.957	0.946
		π=0.9	
C = 0.01	0.123	0.121	0.187
C = 0.1	0.125	0.11	0.115
C = 1.0	0.21	0.203	0.223
C = 10.0	0.586	0.769	0.583

Constant = 1	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 0.01	0.63	0.623	0.598
C = 0.1	0.66	0.624	0.677
C = 1.0	0.96	0.931	0.95
C = 10.0	0.996	1.0	1.0
		π=0.5	
C = 0.01	0.316	0.311	0.313
C = 0.1	0.348	0.347	0.323
C = 1.0	0.701	0.631	0.644
C = 10.0	0.966	1.0	1.0
		π=0.9	
C = 0.01	0.113	0.114	0.104
C = 0.1	0.114	0.112	0.11
C = 1.0	0.281	0.207	0.233
C = 10.0	0.656	0.647	0.569

SVM - Quadratic - PCA with m = 9 and degree = 2

We also show the results of PCA with m = 9 (always with d=2). From Figure 2.5 we can see that using PCA = 9 or don't use it doesn't change that much. Just for $\pi = 0.1$ there's a slight improvement.

Constant = 0	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 0.01	0.645	0.645	0.638
C = 0.1	0.659	0.659	0.656
C = 1.0	0.92	0.871	0.916
C = 10.0	1.0	0.985	0.99
		π=0.5	
C = 0.01	0.33	0.344	0.354
C = 0.1	0.345	0.345	0.391
C = 1.0	0.449	0.575	0.574
C = 10.0	1.0	0.975	0.951
		π=0.9	
C = 0.01	0.127	0.124	0.109
C = 0.1	0.125	0.113	0.125
C = 1.0	0.181	0.203	0.172
C = 10.0	0.605	0.66	0.527

Constant = 1	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 0.01	0.576	0.584	0.59
C = 0.1	0.586	0.624	0.667
C = 1.0	0.894	0.992	0.975
C = 10.0	0.998	1.0	0.982
		π=0.5	
C = 0.01	0.335	0.334	0.306
C = 0.1	0.35	0.302	0.346
C = 1.0	0.688	0.772	0.56
C = 10.0	0.993	1.0	0.975
		π=0.9	
C = 0.01	0.118	0.117	0.105
C = 0.1	0.132	0.114	0.107
C = 1.0	0.22	0.236	0.189
C = 10.0	0.677	0.67	0.594

SVM - Quadratic - PCA with m = 8 and degree = 2

We also tried PCA with m = 8 because it's even better for some cases.

Constant = 0	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 0.01	0.624	0.654	0.666
C = 0.1	0.66	0.648	0.658
C = 1.0	0.848	0.989	0.798
C = 10.0	0.981	0.999	0.97
		π=0.5	
C = 0.01	0.324	0.331	0.333
C = 0.1	0.321	0.336	0.317
C = 1.0	0.593	0.934	0.494
C = 10.0	0.97	0.999	0.91
		π=0.9	
C = 0.01	0.127	0.12	0.113
C = 0.1	0.124	0.11	0.116
C = 1.0	0.285	0.359	0.203
C = 10.0	0.511	0.74	0.563

Constant = 1	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 0.01	0.561	0.541	0.53
C = 0.1	0.546	0.592	0.634
C = 1.0	0.985	0.865	0.98
C = 10.0	1.0	0.996	0..994
		π=0.5	
C = 0.01	0.323	0.32	0.308
C = 0.1	0.349	0.316	0.336
C = 1.0	0.657	0.649	0.7
C = 10.0	1.0	0.985	0.989
		π=0.9	
C = 0.01	0.114	0.108	0.101
C = 0.1	0.111	0.113	0.102
C = 1.0	0.262	0.217	0.211
C = 10.0	0.697	0.743	0.647

SVM - Radial Basis kernel Function - Raw Feature

As we introduced in the Quadratic SVM section, there's another kernel function we are going to see. This one is called Gaussian Radial Basis Function or RBF:

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

where $||x_1 - x_2||^2$ represent the distance between x_1 and x_2 and the closer it is, the bigger is the kernel, meanwhile γ is an indicator of the width of the kernel. If γ is low then the kernel will be large, otherwise it will be small (if $\gamma \gg 0$ then this will be equal to 1-NN).

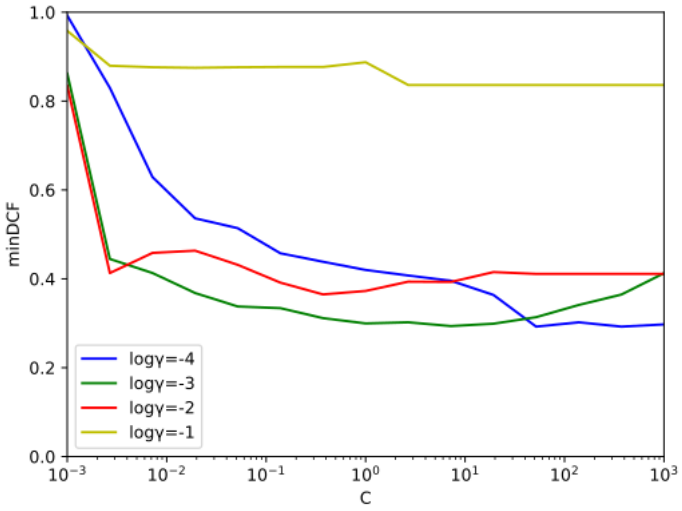


Figure 2.7: RBF

Figure 2.7 shows the different values of C with different values of γ. We are going to test $C = [1, 10, 100]$ with a $\gamma = [0.001, 0.0001]$, those are the best parameter we can get.

γ = **0.001**	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 1.0	0.592	0.575	0.578
C = 10.0	0.58	0.599	0.601
C = 100.0	0.586	0.579	0.601
		π=0.5	
C = 1.0	0.302	0.3	0.301
C = 10.0	0.293	0.298	0.302
C = 100.0	0.332	0.331	0.335
		π=0.9	
C = 1.0	0.097	0.1	0.101
C = 10.0	0.089	0.089	0.089
C = 100.0	0.101	0.103	0.102

γ = **0.0001**	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 1.0	0.671	0.662	0.659
C = 10.0	0.574	0.58	0.584
C = 100.0	0.568	0.575	0.725
		π=0.5	
C = 1.0	0.473	0.42	0.419
C = 10.0	0.406	0.381	0.354
C = 100.0	0.289	0.306	0.472
		π=0.9	
C = 1.0	0.163	0.159	0.158
C = 10.0	0.137	0.133	0.13
C = 100.0	0.097	0.103	0.276

SVM - Radial Basis kernel Function - PCA with m=9
We also tried to use PCA both with m=9 and m=8.

γ = **0.001**	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 1.0	0.588	0.585	0.579
C = 10.0	0.534	0.541	0.542
C = 100.0	0.678	0.671	0.906
		π=0.5	
C = 1.0	0.295	0.294	0.298
C = 10.0	0.297	0.301	0.299
C = 100.0	0.328	0.333	0.895
		π=0.9	
C = 1.0	0.103	0.104	0.104
C = 10.0	0.102	0.103	0.102
C = 100.0	0.107	0.108	0.285

γ = **0.0001**	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 1.0	0.672	0.672	0.662
C = 10.0	0.621	0.605	0.624
C = 100.0	0.553	0.502	0.592
		π=0.5	
C = 1.0	0.439	0.433	0.42
C = 10.0	0.407	0.391	0.376
C = 100.0	0.301	0.302	0.317
		π=0.9	
C = 1.0	0.165	0.159	0.16
C = 10.0	0.139	0.132	0.128
C = 100.0	0.103	0.106	0.12

With m=9 the results are more or less the same

SVM - Radial Basis kernel Function - PCA with m=8

γ = **0.001**	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 1.0	0.568	0.569	0.573
C = 10.0	0.528	0.54	0.545
C = 100.0	0.664	0.672	0.666
		π=0.5	
C = 1.0	0.297	0.297	0.296
C = 10.0	0.281	0.29	0.294
C = 100.0	0.318	0.315	0.305
		π=0.9	
C = 1.0	0.102	0.104	0.105
C = 10.0	0.101	0.103	0.103
C = 100.0	0.108	0.105	0.104

γ = **0.0001**	K = 0.1	K = 1.0	K = 10
		π=0.1	
C = 1.0	0.665	0.662	0.645
C = 10.0	0.617	0.607	0.679
C = 100.0	0.55	0.529	0.844
		π=0.5	
C = 1.0	0.439	0.433	0.43
C = 10.0	0.397	0.389	0.387
C = 100.0	0.306	0.29	0.686
		π=0.9	
C = 1.0	0.165	0.158	0.156
C = 10.0	0.141	0.135	0.153
C = 100.0	0.106	0.107	0.301

Also with m=8 the results are more or less the same

2.5 Gaussian Mixture Models

This is the last model we are going to consider. It's a generative model and it's called Gaussian Mixture Model. We are going to consider 4 models of GMM:

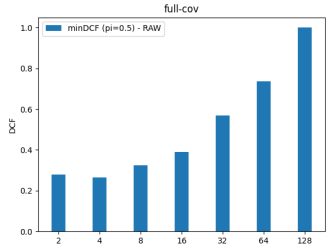
- Full covariance matrix
- Diagonal covariance matrix
- Full Tied covariance matrix
- Diagonal Tiead covariance matrix

The GMM is a distribution whose parameters are (M,S,W) where:

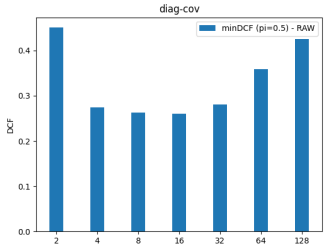
$$M = [\mu_1, \mu_2, ..., \mu_K], S = [\Sigma_1, \Sigma_2, ..., \Sigma_K], W = [w_1, w_2, ..., w_K]$$

The GMM density can be rapresented as :

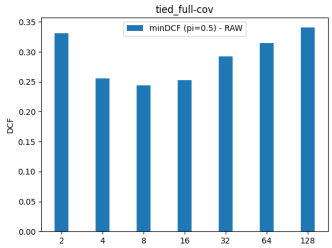
$$f_X(x) = \sum_{c=1}^{K} \pi_c \mathcal{N}(x|\mu_c, \Sigma_c)$$



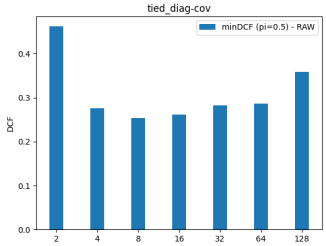
(a) GMM Full Cov



(b) GMM Diag Cov



(c) GMM Tied Full Cov



(d) GMM Tied Full Cov

Figure 2.8: GMM with π=0.5 RAW Features

Figure 2.8 show the different values of minDCF with $\pi = 0.5$ and the corresponding number of components for each type of GMM. For example we can see that for the full cov, the best number of components is 4. Here below we are going to see the results of the RAW Features and with PCA with m = 9.

GMM - RAW Feature

Components	2	4	8	16	32	64	128
				π=0.1			
Full-Cov	0.497	0.526	0.69	0.63	0.877	0.976	1.0
Diag-Cov	0.719	0.544	0.514	0.531	0.592	0.698	0.741
Full-Cov-Tied	0.616	0.611	0.558	0.542	0.588	0.563	0.71
Diag-Cov-Tied	0.805	0.558	0.501	0.56	0.601	0.673	0.765
				π=0.5			
Full-Cov	0.28	0.264	0.325	0.391	0.568	0.737	1.0
Diag-Cov	0.451	0.275	0.262	0.26	0.281	0.359	0.425
Full-Cov-Tied	0.331	0.255	0.244	0.253	0.292	0.314	0.341
Diag-Cov-Tied	0.462	0.276	0.254	0.262	0.282	0.286	0.358
				π=0.9			
Full-Cov	0.105	0.094	0.108	0.124	0.213	0.306	0.523
Diag-Cov	0.141	0.087	0.09	0.092	0.107	0.122	0.144
Full-Cov-Tied	0.111	0.098	0.083	0.087	0.104	0.108	0.134
Diag-Cov-Tied	0.144	0.086	0.089	0.088	0.093	0.11	0.131

We can see that the Tied Full Covariance is the best in almost all components. Meanwhile the worst is the Full Covariance that reach a midDCF = 1.0 with 128 components.

GMM - PCA with m = 9

Components	2	4	8	16	32	64	128
				π=0.1			
Full-Cov	0.455	0.599	0.599	0.684	0.837	0.999	1.0
Diag-Cov	0.742	0.599	0.553	0.546	0.513	0.638	0.729
Full-Cov-Tied	0.629	0.608	0.56	0.56	0.494	0.522	0.589
Diag-Cov-Tied	0.741	0.554	0.528	0.538	0.554	0.58	0.624
				π=0.5			
Full-Cov	0.299	0.293	0.322	0.371	0.467	0.704	1.0
Diag-Cov	0.367	0.272	0.253	0.281	0.287	0.34	0.374
Full-Cov-Tied	0.33	0.261	0.269	0.254	0.285	0.324	0.322
Diag-Cov-Tied	0.368	0.268	0.248	0.276	0.306	0.328	0.359
				π=0.9			
Full-Cov	0.099	0.091	0.113	0.125	0.158	0.218	0.376
Diag-Cov	0.114	0.096	0.094	0.093	0.103	0.109	0.135
Full-Cov-Tied	0.109	0.103	0.087	0.095	0.109	0.119	0.138
Diag-Cov-Tied	0.114	0.087	0.095	0.09	0.093	0.101	0.12

Chapter 3

Evaluation

We are now going to evaluate our models using the evaluation set (or test set). The evaluation will be done thanks to the minDCF. In the validation part we used the k-fold with k = 5, but in the evaluation part we won't use k-fold because we don't need the validation test, since there's the test set. So we are going to train each model on the full training set then test this model on the test set.

3.1 Multivariate Gaussian Model

In this section we are going to show the result on the **Test set** for the Gaussian part, exaclty as we did for the validation part (that was done on the **Validation set**). We are going to test on Raw features, PCA with m = 8 and m = 9.

	pi = 0.1	pi = 0.5	pi = 0.9
	RAW Features		
MVG	0.545	0.276	0.083
Naive Gaussian	0.718	0.352	0.111
Tied Gaussian	0.789	0.455	0.176
Naive Tied Gaussian	0.829	0.48	0.177

	pi = 0.1	pi = 0.5	pi = 0.9
	PCA m = 9		
MVG	0.569	0.272	0.082
Naive Gaussian	0.635	0.31	0.087
Tied Gaussian	0.783	0.461	0.177
Naive Tied Gaussian	0.864	0.483	0.175

	pi = 0.1	pi = 0.5	pi = 0.9
	PCA m = 8		
MVG	0.571	0.271	0.083
Naive Gaussian	0.638	0.315	0.087
Tied Gaussian	0.773	0.456	0.178
Naive Tied Gaussian	0.865	0.484	0.176

We can see that for all 3 models (Raw,m=8,m=9) the MVG is the best. Only for the Naive assumption the model with PCA are better than the Raw model.
Comparing these results with the one obtained from the Validation part, they seems to be a little bit better, but not that much, so the results are consistent.

3.2 Logistic Regression

Again, in this section there're the results for the Logistic Regression part.

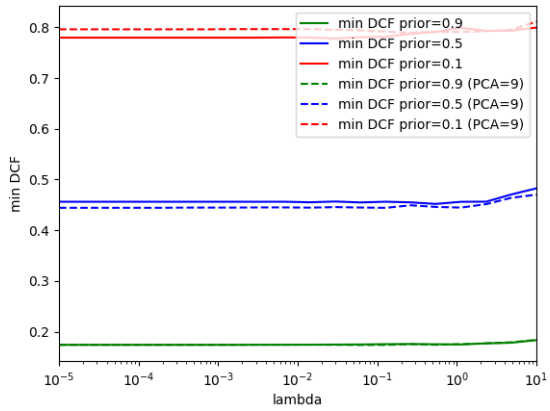


Figure 3.1: LR - compare different λ

Figure 3.1 shows the midDCF for each λ. The optimal $\lambda = 0.4$. We can also notice that for our application ($\pi = 0.5$) the model with PCA with m = 9 is slightly better. Now let's see the results.

	pi=0.1	pi=0.5	pi=0.9
	RAW Features		
Log Reg	0.789	0.454	0.175
	PCA m=9		
Log Reg	0.79	0.45	0.175
	PCA m=8		
Log Reg	0.789	0.45	0.174

We can see that using the Raw or PCA models doesn't change that much, so all the three models can be considered the "best" choise for LR, but for our application, from the Gaussian part, the MVG model has midDCF = **0.271**, meanwhile the best I can obtain using LR is minDCF = **0.45**. So the LR model isn't the best choice overall. This is telling us that the linear separation isn't the one we need, so we expect that the RBF and Poly SVM will perform better!!! (In fact those result are similar to the ones given by the Tied Gaussian, that is also linear) By the way the results are also consistent with the Validation part.

3.3 Support Vector Machine

3.4 Gaussian Mixture Model