MLPR – 2023 – Fingerprint spoofing detection

Sergiu Mohamed Abed (s295149)

February 1, 2024

1 Introduction

The goal of this project is to implement several classifiers taught in this course, train them, validate them, choose the best performing ones according to our metrics and finally, evaluate the chosen models and some of the discarded ones to see how good our choices were during the validation phase.

More specifically, we must train and test several classifiers to identify whether a sample corresponding to a fingerprint image is authentic (label 1) or a spoofed one (label 0).

2 Dataset analysis and dimensionality reduction

2.1 Dataset analysis

The training dataset consists of 2325 fingerprint images, of which 800 correspond to authentic fingerprints and 1525 to spoofed fingerprints. The spoofed fingerprints have been generated by using 6 different spoofing methods.

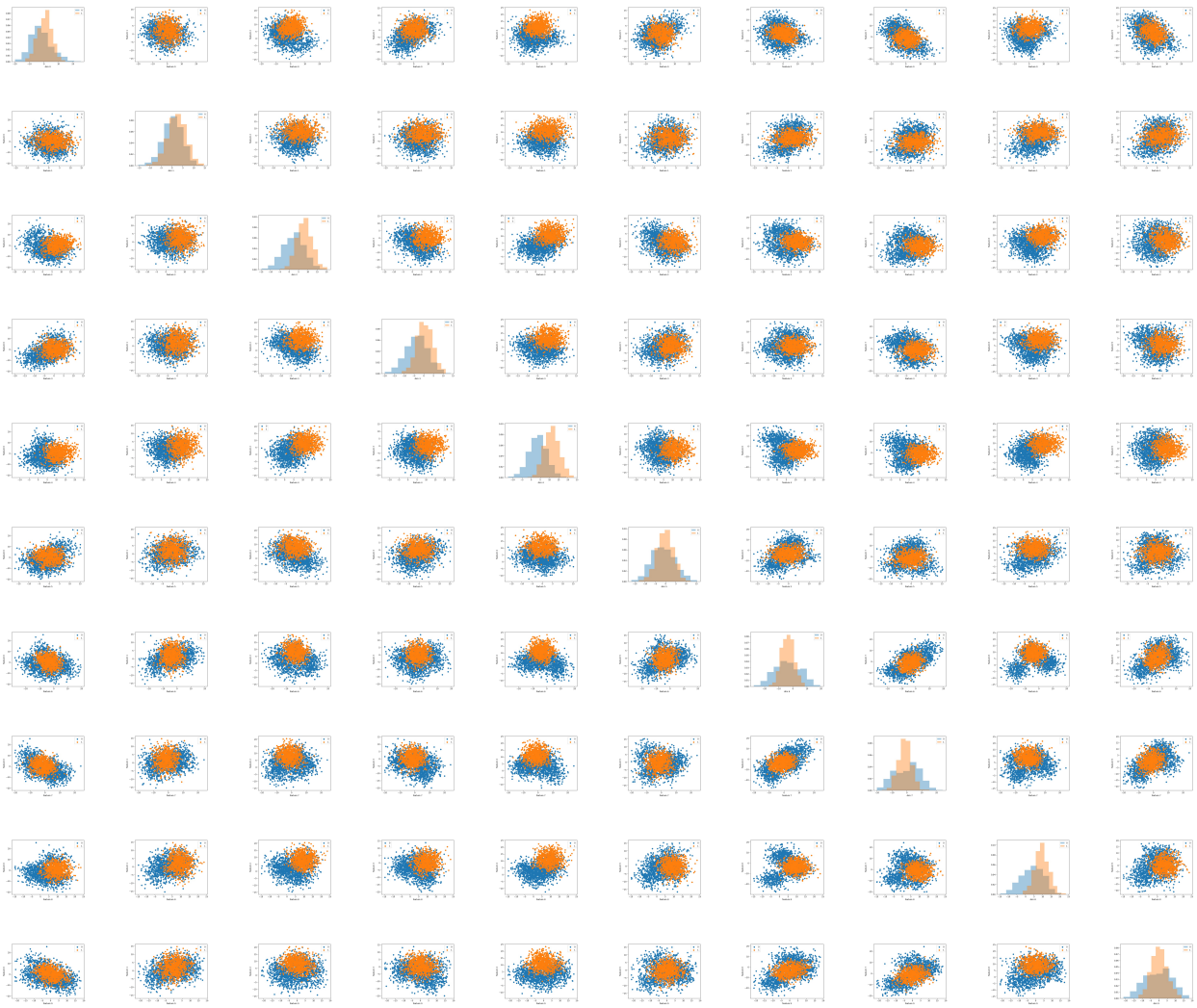
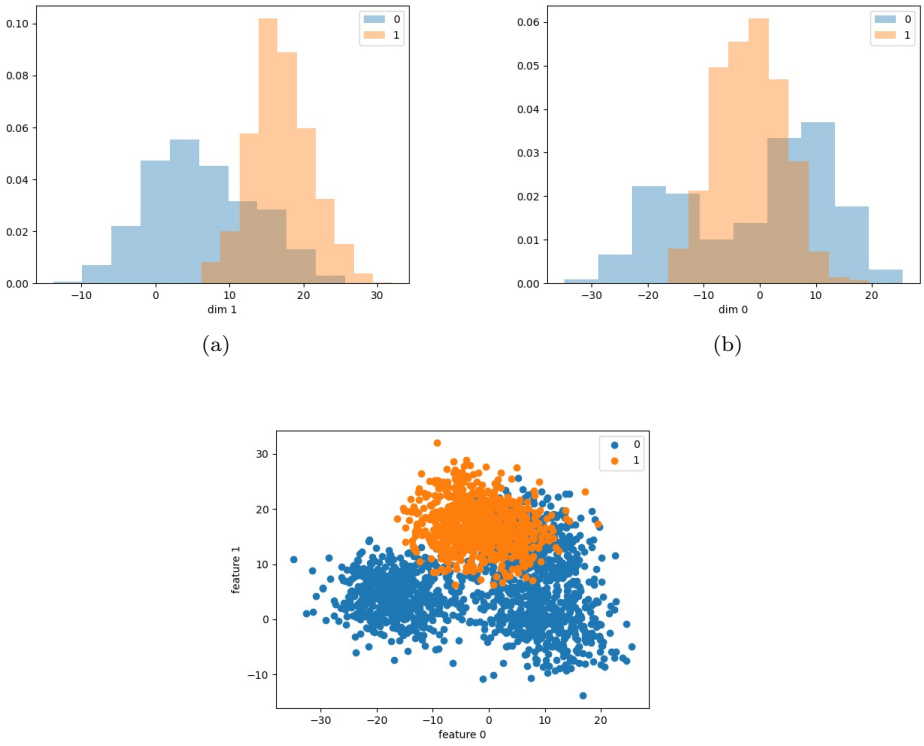


Figure 1: Histograms and Scatter plots on features of training set samples

As it can be noticed in Figures 1 and 2, the two classes are not linearly separable. Looking at the histograms we see that the distributions of the classes overlap a lot. Even after applying PCA, there is still quite a lot of overlap (see Figure 2.a and 2.b). In this case, we expect a linear model will not perform well.



(a) (b)

(c)

Figure 2: Histograms and Scatter plot on first two principal components

The histograms in Figure 2.b tell us that, while the target class may be well defined by a Gaussian distribution, the class of spoofed fingerprints cannot be decently approximated by a Gaussian distribution, since it looks like an overlap of two Gaussians shifted from the origin. This fact can be seen also in the scatter plots, especially in Figure 2.c, where we can notice three clusters describing the non-target samples. This fact could be related to the spoofing methods, i.e. samples corresponding to the same spoofing method may belong to the same cluster, but unfortunately we don't have access to this information. Nonetheless, we can expect that a Gaussian Mixture Model (GMM) would be better suited for this task then a Multivariate Gaussian (MVG) model.

Looking at Figure 3 (a), (b) and (c), we see that the features of the samples in the training set are highly correlated, especially considering just the non-target samples. The target class, on the other hand, has weaker correlation among features, so applying Naive Bayes assumption on this class could be safe. A tied covariance assumption does not seem reasonable, given how much the correlation of the features between the two classes differ.

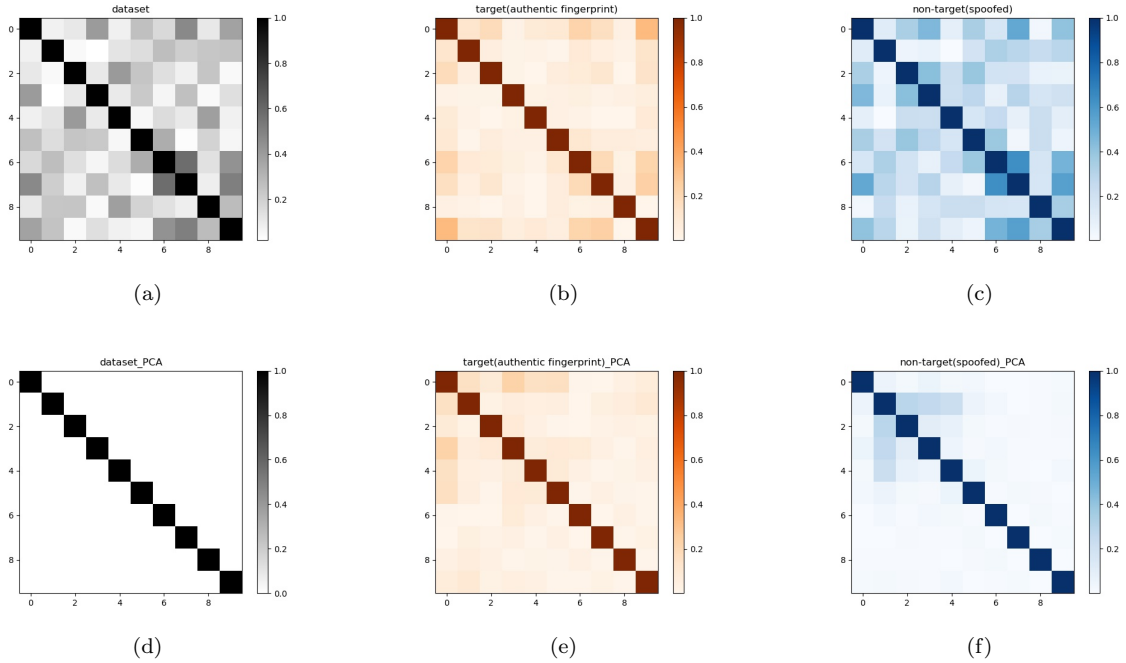
2.2 Principal Component Analysis

We want to project the dataset to a lower dimensional space while preserving as much information as possible. Principal Component Analysis (PCA) is an unsupervised way of achieving this, meaning that we don't consider the labels when looking for the basis of the new space.

We must find P that maximizes the following objective:

$$\hat{L}(P) = Tr\left(P^T \left[\frac{1}{K}\sum_{i=1}^{K}(x_i - \bar{x})(x_i - \bar{x})^T\right] P\right),$$

where K is the total number of samples, x_i is the i^{th} sample, $\bar{x} = \frac{1}{K}\sum_{i=1}^{K} x_i$ and P is a matrix whose



(a) (b) (c)

(d) (e) (f)

Figure 3: Heatmaps (a), (b) and (c) plotted before applying PCA; (d), (e), (f) after PCA

columns form the basis of the subspace.

Luckily for us, the maximization problem of $\hat{L}(P)$ has a closed form solution. For a subspace of dimension $m < n$, the optimal solution corresponds to the m eigenvectors of the empirical covariance matrix Σ corresponding to the m highest eigenvalues.

$$\mathbf{\Sigma} = \frac{1}{K} \sum_{i=1}^{K} (\mathbf{x}_i - \bar{\mathbf{x}}) (\mathbf{x}_i - \bar{\mathbf{x}})^T$$

We will consider different values for m during validation.

Z-normalization For some types of classifiers, we will apply Z-normalization, which consists of shifting the data to the center of the space and dividing each feature by its standard deviation.

3 Task setting and Training protocol

3.1 Working points and performance metric

The target application working point of our task is $(\pi_T = 0.5, C_{fn} = 1, C_{fp} = 10)$. To simplify our work, we will consider the equivalent working point $(\tilde{\pi}_T = 0.0909, C_{fn} = 1, C_{fp} = 1)$, where $\tilde{\pi}_T$ is the effective prior computed as follows:

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

We will also analyze two different working points, which are $(\pi_T = 0.5, C_{fn} = 1, C_{fp} = 1)$ and $(\pi_T = 0.9, C_{fn} = 1, C_{fp} = 1)$.

We assess the performance of the models with minDCF.

$$minDCF = \min_{\forall t \in \mathbb{R}} \frac{\pi_T C_{fn} P_{fn}(t) + (1 - \pi_T) C_{fp} P_{fp}(t)}{\min(\pi_T C_{fn}, (1 - \pi_T) C_{fp})} = \min_{\forall t \in \mathbb{R}} \frac{\tilde{\pi}_T P_{fn}(t) + (1 - \tilde{\pi}_T) P_{fp}(t)}{\min(\tilde{\pi}_T C_{fn}, (1 - \tilde{\pi}_T) C_{fp})}$$

,

where $P_{fn}(t)$ and $P_{fp}(t)$ are false negative and false positive rates, respectively, which are functions of a threshold t.

Notice that in the rightmost side of the equation, we used the effective prior, thus the costs have been set to 1.

3.2 Training protocol

For this task, we follow a k-fold cross-validation approach. We set $k = 5$, meaning that we split the training set (after shuffling) in 5 disjoint sets and for each fold we train the model on the other 4 folds and validate on the selected fold. Once each fold has been used for validation, we pool the scores and then train the model on the whole training set. $minDCF$ is calculated on the pooled scores.

4 Models validation

We now begin training and validating several classifiers. The types of classifiers we will consider are:

- MVG (Multivariate Gaussian)
- Logistic Regression
- SVM (Support Vector Machines)
- GMM (Gaussian Mixture Model)

4.1 MVG

4.1.1 Model description

MVG is a generative model. This means that we estimate a multivariate Gaussian distribution over each class, so for each class c, we estimate a probability density function $f_{\mathbf{X}|C}(\mathbf{x} \mid c) = \mathcal{N}(\mathbf{x} \mid \mu, \sigma^2)$. Plugging in the prior knowledge, we can compute the class posterior probability $\mathbb{P}(C \mid \mathbf{x})$ and for a sample $\mathbf{x}_i$ we assign the class label corresponding to the largest class posterior probability.

In the binary case, we can assign a label to a sample based on the log-likelihood ratio

$$llr(\mathbf{x}_i) = log \frac{f_{\mathbf{X}_i|C_1}(\mathbf{x}_i \mid c_1)}{f_{\mathbf{X}_i|C_0}(\mathbf{x}_i \mid c_0)}$$

being less than or greater than a threshold depending on the working point of the task.

$$llr(\mathbf{x}_i) \lessgtr -log \frac{\pi}{1-\pi}$$

If it is greater than the threshold, the sample is assigned class 1, otherwise class 0.

The optimal parameters corresponding to the two distributions describing the two classes can be found by determining their empirical mean vectors and their empirical covariance matrices.

$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{i|c_i=c} \mathbf{x}_i$$

$$\boldsymbol{\Sigma}_c = \frac{1}{N_c} \sum_{i|c_i=c} (\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^T$$

Naive Bayes and Tied Covariance assumptions may be employed to overcome over-fitting. This is because making this assumptions reduces the number of parameters of the model. Naive Bayes assumes the features of the samples are uncorrelated, making the elements of the covariance matrix above and below the main diagonal equal to 0. Tied Covariance assumes the distributions of the two classes have the same covariance matrix, so we would have to find just one covariance matrix instead of two.

One last important aspect to mention is that MVG under full-covariance or Naive Bayes is a quadratic model, i.e. $llr(\mathbf{x})$ has a quadratic form. MVG under tied-covariance assumption is a linear model, since the covariance factors in the numerator and the denominator of $llr(\mathbf{x})$ cancel out. Hence, we expect poor results assuming tied-covariance, but we will test it anyway, since MVG is not computationally expensive.

4.1.2 Results

During validation, we tested MVG on the original space of the dataset (after shifting it to the origin) and under PCA.

Mode	PCA dim	$minDCF(\tilde{\pi} = 0.0909)$	$minDCF(\tilde{\pi} = 0.5)$	$minDCF(\tilde{\pi}) = 0.9)$
Full	N/A	**0.336**	0.105	0.190
	9	0.339	0.101	0.186
	8	0.337	0.103	0.182
	7	0.338	0.104	0.184
	6	0.343	0.104	0.191
Diag	N/A	0.470	0.135	0.234
	9	0.386	0.106	0.181
	8	**0.379**	0.105	0.185
	7	0.388	0.105	0.185
	6	0.380	0.104	0.186
Tied	N/A	0.480	0.170	0.400
	9	0.484	0.172	0.392
	8	0.489	0.175	0.394
	7	**0.475**	0.173	0.397
	6	0.479	0.170	0.392

Table 1: MVG validation results under 3 different working points

"N/A" stands for the fact that PCA was not used.

Looking at Table 1, we find that the best configuration is (Full covariance, no PCA) with $minDCF = 0.336$. Compared to other quadratic models in the next sections, the results are not quite as good. As mentioned in Dataset analysis section, fitting a Multivariate Gaussian distribution over the non-target class does not seem too reasonable and this is reflected in the results obtained. As we will see in the GMM section, considering a combination of multiple Multivariate Gaussian distributions for the non-target class leads to much better results.

4.2 Logistic Regression

4.2.1 Model description

Logistic regression is a discriminative model, meaning that unlike the generative case, we directly model the class posterior distribution $C \mid \mathbf{X}$. In a binary task, we work with posterior log-likelihood ratios, which look like this:

$$\log \frac{\mathbb{P}(C = c_1 \mid \mathbf{x})}{\mathbb{P}(C = c_0 \mid \mathbf{x})} = \mathbf{w}^T \mathbf{x} + b$$

The right side of the equation shows that in logistic regression we make the assumption that the separation boundaries are linear. As mentioned before, we expect linear models to give poor results for the task at hand. To turn logistic regression into a quadratic model, we can use the expanded feature space. We use a function $\boldsymbol{\phi}(\mathbf{x})$ to map the samples of the dataset to the expanded feature space and train the model exactly as before.

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

Finding the best parameter values $(\mathbf{w}^*, b^*)$ does not have a closed form solution. Instead, we must optimize an objective function, which in logistic regression corresponds to minimizing the average cross-entropy between the distribution of the actual class labels and the distribution of the predicted labels.

Given that the empirical prior ($\pi_T = 0.344$) and the effective prior of the target application working point ($\tilde{\pi}_T = 0.0909$) differ, we use a prior-weighted version of the objective function to train the model for the application target.

$$J(\mathbf{w}, b) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{\tilde{\pi}_T}{n_T}\sum_{i|z_i=1}^{n}\log(1 + e^{-z_i(\mathbf{w}^T\mathbf{x}_i+b)}) + \frac{1-\tilde{\pi}_T}{n_F}\sum_{i|z_i=-1}^{n}\log(1 + e^{-z_i(\mathbf{w}^T\mathbf{x}_i+b)})$$

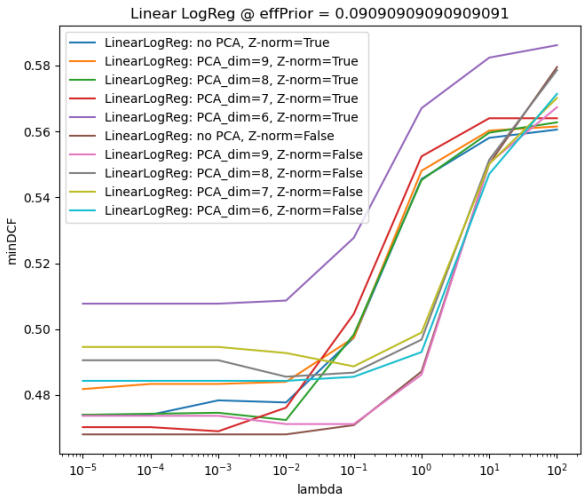
$$z_i = 2c_i - 1$$

4.2.2 Results: Linear Logistic Regression

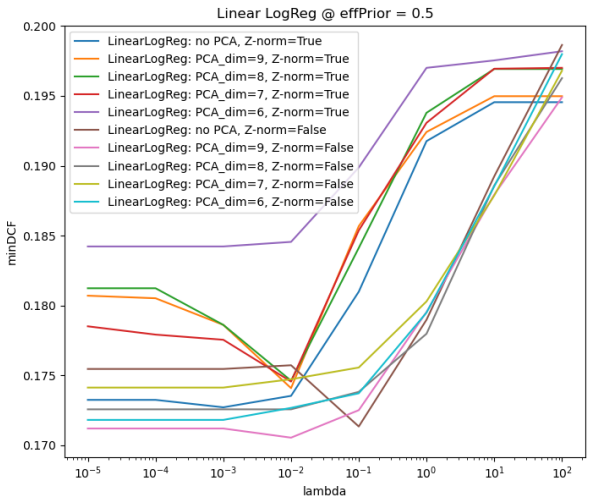
PCA dim	Z-norm	λ	$minDCF(\tilde{\pi} = 0.0909)$	$minDCF(\tilde{\pi} = 0.5)$	$minDCF(\tilde{\pi} = 0.9)$
N/A	False	10^{-2}	**0.468**	0.176	0.398
7	True	10^{-3}	0.469	0.175	0.393
9	False	10^{-1}	0.471	0.173	0.368

Table 2: **Linear LogReg** validation results under 3 different working points. For each of top 3 PCA dimensions (in terms of $minDCF(\tilde{\pi} = 0.0909)$ performance), the λ value corresponding to the lowest $minDCF(\tilde{\pi} = 0.0909)$ has been chosen

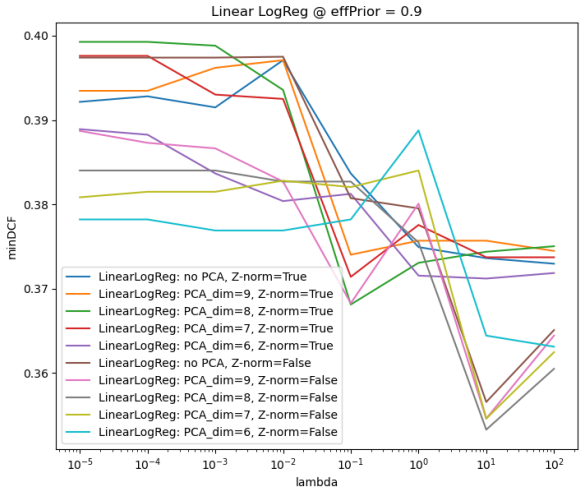
As expected, the "standard" linear version of logistic regression does not perform very well for our task. While $minDCF(\tilde{\pi} = 0.0909) < 1$ for all hyperparameters configurations we considered, we think we can do better. Table 2 sums up the best configurations and Figure 4 shows all configurations tried.



(a) $\tilde{\pi}_T = 0.0909$ (Target application)



(b) $\tilde{\pi}_T = 0.5$



(c) $\tilde{\pi}_T = 0.9$

Figure 4: Linear LogReg minDCF vs λ plots

4.2.3 Results: Quadratic Logistic Regression

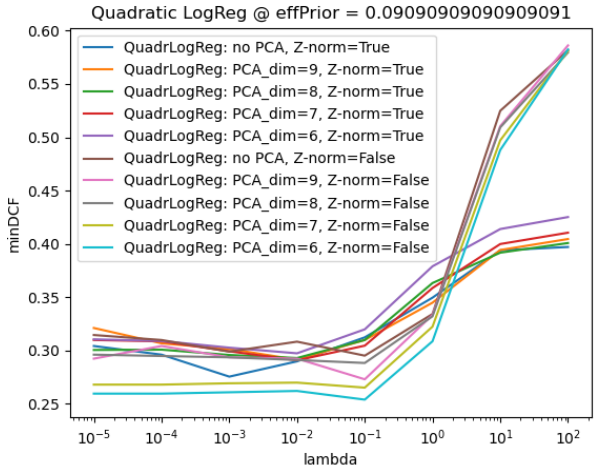
PCA dim	Z-norm	λ	$minDCF(\tilde{\pi} = 0.0909)$	$minDCF(\tilde{\pi} = 0.5)$	$minDCF(\tilde{\pi} = 0.9)$
6	False	10^{-1}	**0.254**	0.09	0.240
7	False	10^{-1}	0.265	0.09	0.235
9	False	10^{-1}	0.273	0.1	0.248

Table 3: **Quadratic LogReg** validation results under 3 different working points. For each of top 3 PCA dimensions (in terms of $minDCF(\tilde{\pi} = 0.0909)$ performance), the λ value corresponding to the lowest $minDCF(\tilde{\pi} = 0.0909)$ has been chosen

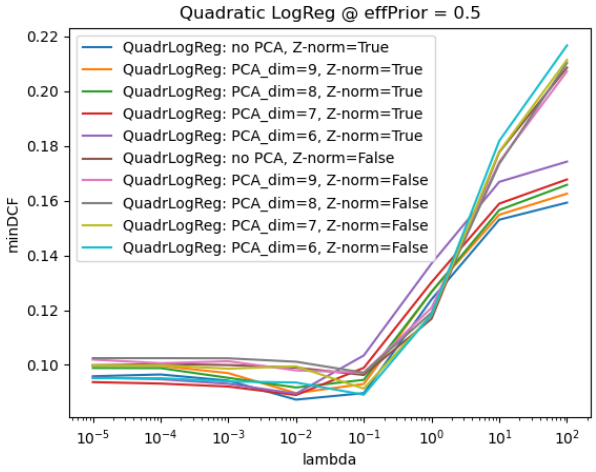
When running the experiments, I decided to reduce the PCA dimension cases to the three that provided the best results in MVG to reduce the computational time, given that the quadratic version of logistic regression is computationally expensive (increasing the dimension of the samples slows down the matrix

operations).

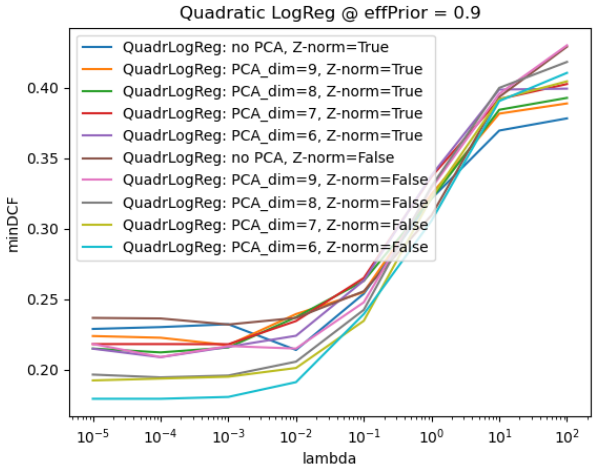
Casting the problem to the expanded space proved very effective. We see a significant drop in minDCF. The best model we have so far has $minDCF(\tilde{\pi} = 0.0909) = 0.254$. Figure 5 shows all configurations tested for quadratic logistic regression.



(a) $\tilde{\pi}_T = 0.0909$ (Target application)



(b) $\tilde{\pi}_T = 0.5$



(c) $\tilde{\pi}_T = 0.9$

Figure 5: Quadratic LogReg minDCF vs λ plots

4.3 SVM

4.3.1 Model description

The goal of SVM (Support Vector Machines) is to find a linear hyperplane separating the space so that, for a binary task, most of the points (as many as possible) belonging to the same class are on the same side of the hyperplane.

Unlike logistic regression, which is a linear model trying to maximize the class-posterior probabilities, SVM tries to maximize the distance of the sample points from the hyperplane. This means that SVM scores do not have a probabilistic interpretation.

We implement soft-margin SVM and we do this by solving the following dual problem:

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \mathbf{x}_i^T \mathbf{x}_j$$

such that

$$0 \leq \alpha_i \leq C, \forall i$$

$$\sum_{i=1}^{n} \alpha_i z_i = 0$$

where C is a hyperparameter that we tweak during validation.

By default, SVM is a linear model, so we expect it to give poor results on our dataset. Like in logistic regression, we can cast the problem to an expanded space in order to obtain non-linear separations. The advantage with respect to logistic regression is that if we work with the dual problem, we don't need to map the samples to the expanded space. Instead, we only need a function called **kernel** that defines the dot product between two vectors in the expanded space.

We leverage two kinds of kernels:

- Polynomial kernel of degree d: $k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^T \mathbf{x}_2 + const)^d$
- Gaussian Radial Basis Function kernel: $k(\mathbf{x}_1, \mathbf{x}_2) = e^{-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2}$

4.3.2 Results: Linear SVM

PCA dim	Z-norm	C	$minDCF(\tilde{\pi} = 0.0909)$	$minDCF(\tilde{\pi} = 0.5)$	$minDCF(\tilde{\pi} = 0.9)$
7	True	10^2	**0.464**	0.176	0.401
6	False	10^0	0.471	0.173	0.393

Table 4: **Linear SVM** validation results under 3 different working points. For each of top 2 PCA dimensions (in terms of $minDCF(\tilde{\pi} = 0.0909)$ performance), the C value corresponding to the lowest $minDCF(\tilde{\pi} = 0.0909)$ has been chosen

Again, the linear models don't prove to be well suited for our task, as shown in Table 4 and Figure 6.

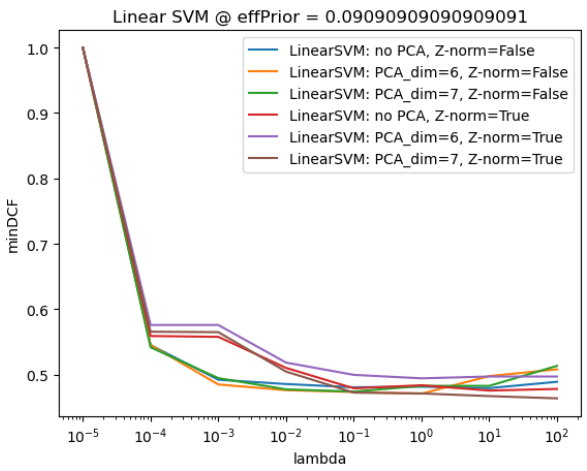
4.3.3 Results: SVM with Polynomial kernel of 2^{nd} degree

PCA dim	Z-norm	const	C	$minDCF(\tilde{\pi} = 0.0909)$	$minDCF(\tilde{\pi} = 0.5)$	$minDCF(\tilde{\pi} = 0.9)$
6	False	1	10^0	**0.262**	0.09	0.196
N/A	True	1	10^{-2}	0.283	0.08	0.191

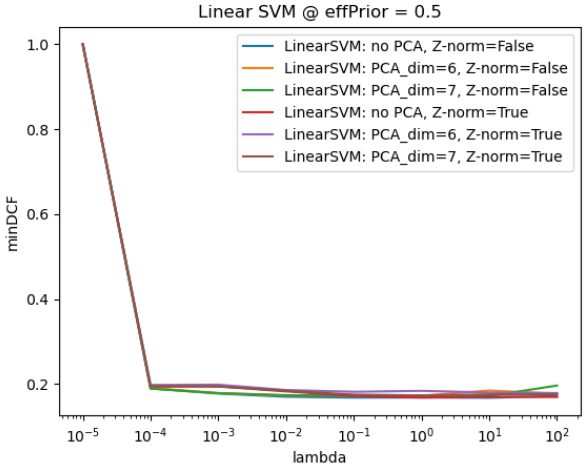
Table 5: **Poly(2) SVM** validation results under 3 different working points. For each of top 2 PCA dimensions (in terms of $minDCF(\tilde{\pi} = 0.0909)$ performance), the C value corresponding to the lowest $minDCF(\tilde{\pi} = 0.0909)$ has been chosen

Once again, working with our dataset in the expanded space (without actually performing any mapping) led to improved results. The best result obtained has $minDCF(\tilde{\pi} = 0.0909) = 0.262$, which is very close to the best one of quadratic logistic regression, although a bit worse. The advantage seen during the experiments in the case of kernel based SVM is that it runs faster during classification, due to the fact that the samples are not mapped to an expanded space.

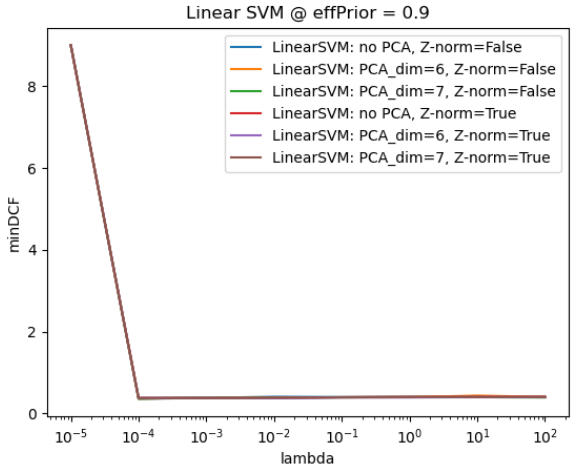
One interesting detail we can see in Figure 7 (especially in figure (a)) is that setting the hyperparameter $c_{poly} = 0$ (the constant of the polynomial kernel) leads to very poor result. All models under this case have $minDCF(\tilde{\pi} = 0.0909) > 0.7$.



(a) $\tilde{\pi}_T = 0.0909$ (Target application)



(b) $\tilde{\pi}_T = 0.5$



(c) $\tilde{\pi}_T = 0.9$

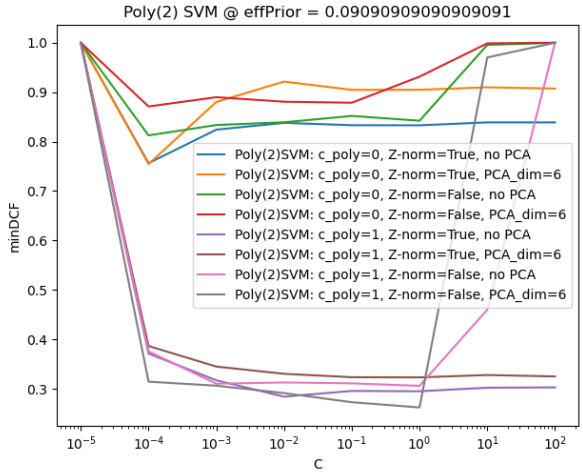
Figure 6: Linear SVM minDCF vs C plots

4.3.4 Results: SVM with Polynomial kernel of 3^{nd} degree

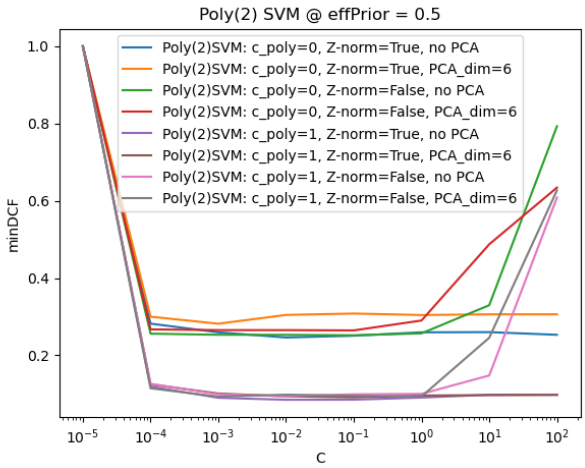
PCA dim	Z-norm	const	C	$minDCF(\tilde{\pi} = 0.0909)$	$minDCF(\tilde{\pi} = 0.5)$	$minDCF(\tilde{\pi} = 0.9)$
6	False	1	10^{-2}	0.532	0.131	0.252

Table 6: **Poly(3) SVM** validation results under 3 different working points. For PCA dimension equal to 6, the C value corresponding to the lowest $minDCF(\tilde{\pi} = 0.0909)$ has been chosen

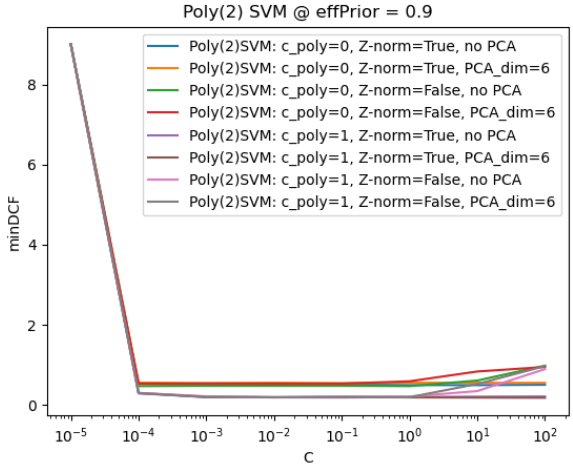
We tested the 3^{rd} degree polynomial kernel to see if we could benefit from an increase in model complexity. Given the worse results with respect to the ones obtained so far, we conclude that it is not suited for this task.



(a) $\tilde{\pi}_T = 0.0909$ (Target application)



(b) $\tilde{\pi}_T = 0.5$



(c) $\tilde{\pi}_T = 0.9$

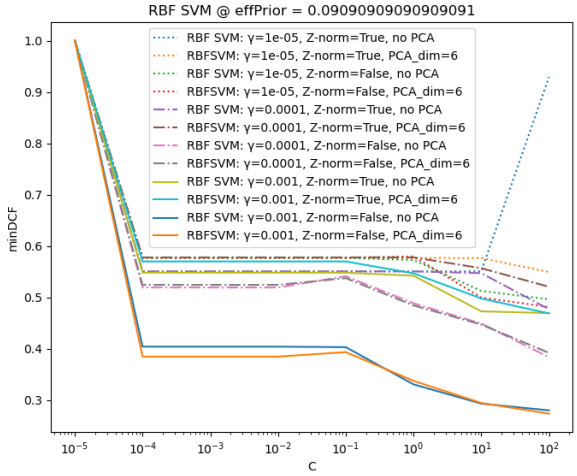
Figure 7: SVM 2^{nd} degree Polynomial minDCF vs C plots

4.3.5 Results: SVM with RBF kernel

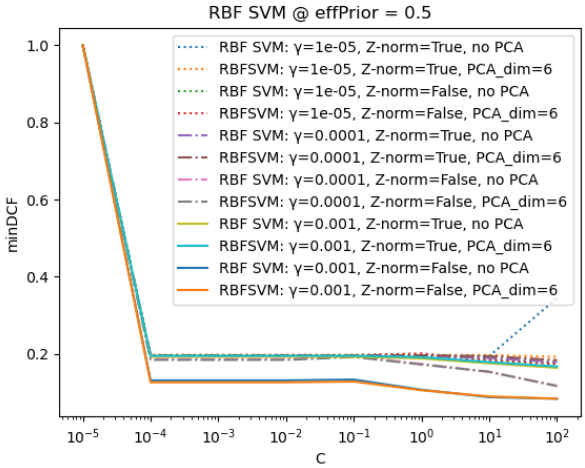
PCA dim	Z-norm	γ	C	$minDCF(\tilde{\pi} = 0.0909)$	$minDCF(\tilde{\pi} = 0.5)$	$minDCF(\tilde{\pi} = 0.9)$
6	False	0.001	10^2	**0.273**	0.08	0.170
N/A	False	0.001	10^2	0.280	0.084	0.187

Table 7: **RBF SVM** validation results under 3 different working points. For each PCA dimension tested, the C and γ value corresponding to the lowest $minDCF(\tilde{\pi} = 0.0909)$ has been chosen

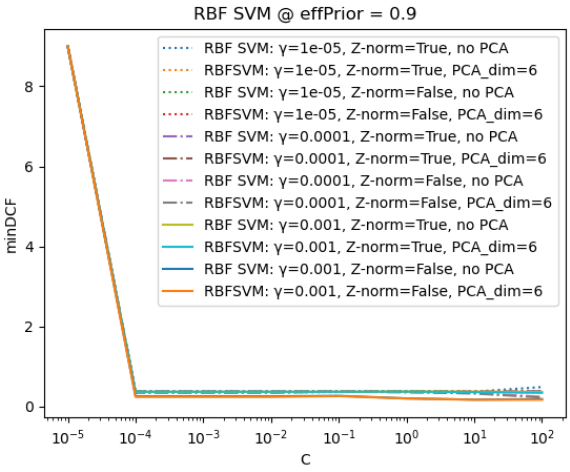
Using RBF kernel provides much better results compared to linear SVM, however it does not outperform the 2^{nd} degree polynomial kernel.



(a) $\tilde{\pi}_T = 0.0909$ (Target application)



(b) $\tilde{\pi}_T = 0.5$



(c) $\tilde{\pi}_T = 0.9$

Figure 8: SVM with RBF kernel minDCF vs C plots

4.4 GMM

4.4.1 Model description

We now go back to generative models. There are distributions that cannot be decently approximated by multivariate Gaussian distributions. **Gaussian Mixture Models (GMMs)** are a better alternative to MVGs for modeling class-conditional distributions $\mathbf{X} \mid C$. Instead of fitting a Gaussian over a distribution, it approximates it by considering a weighted combination of two or more Gaussian distributions.

$$\mathbf{X} \mid C = c \sim GMM(\mathbf{M}_c, \mathbf{S}_c, \mathbf{w}_c)$$

$$f_{\mathbf{X}_t|C_t}(\mathbf{x}_i \mid c_1) = \sum_{k=1}^{K_c} w_{c,k} \mathcal{N}(\mathbf{x}_t \mid \boldsymbol{\mu}_{c,k}, \boldsymbol{\Sigma}_{c,k})$$

where, $\mathbf{M}_c = [\boldsymbol{\mu}_{c,1}, \ldots, \boldsymbol{\mu}_{c,k}]$ contains the mean vector of each component (cluster) of the GMM fitting class c, $\boldsymbol{S}_c = [\boldsymbol{\Sigma}_{c,1}, \ldots, \boldsymbol{\Sigma}_{c,k}]$ contains the covariance matrix of each component and $\mathbf{w}_c =$

$[w_1, \dots, w_k]$ consists of the weights for each component.

As we just said, not all distributions can be well approximated by a Gaussian and our task falls in such a case. In the problem description, we are told that the spoofed fingerprints were generated by 6 different spoofing methods. From the scatter plots in Figure 1 and 2, we notice that the spoofed samples could be described by 3 components. Having this in mind, we decide to test different GMM models, in which for the non-target class (i.e. spoofed one) we try 4 clusters (i.e. power of 2 closest to 3) and 8 (i.e power of 2 closest to 6 larger than 4).

For the target class, we will try 1 and 2 components, since in the scatter plots we notice that these samples could be described by 1 component. In some of the scatter plots, the cluster corresponding to the target class seems a bit skewed, so it would be reasonable to also test 2 components for it. We will avoid testing the case (1, 1), because it corresponds to having a MVG model which we already discussed few sections ago.

We will also test Naive Bayes and Tied-Covariance approaches to see whether full-covariance method overfits the data or not. Tied-covariance is assumed among the clusters belonging to the same GMM distribution for a certain class, not across both classes.

4.4.2 Results

PCA dim	Mode target	Mode non-target	Target: nr. clusters	Non-target: nr. clusters	$minDCF(0.0909)$	$minDCF(0.5)$	$minDCF(0.9)$
6	diag	full	2	8	**0.245**	0.076	0.151
	tied	full	2	8	0.246	0.073	0.145
	full	full	1	8	0.246	0.073	0.145
	diag	diag	2	8	0.248	0.074	0.157
	full	diag	1	8	0.249	0.068	0.153
	tied	diag	2	8	0.249	0.068	0.153
N/A	diag	diag	2	8	**0.252**	0.076	0.147
	diag	full	2	4	0.255	0.076	0.130
	full	diag	1	8	0.256	0.075	0.157
	tied	diag	2	8	0.256	0.075	0.157
	full	diag	1	4	0.259	0.076	0.171
	tied	diag	2	4	0.259	0.076	0.171

Table 8: GMM validation results under 3 different working points. For no dimensionality reduction and 6 PCA directions, here are reported the 6 best hyperparameters configurations

5 Post-validation model selection

The best models we've obtained are:

1. **Quadratic Logistic regression:** $minDCF(\tilde{\pi} = 0.0909) = 0.254$
   - PCA dim: 6
   - Z-normalization: False
   - λ: 10^{-1}
2. 2^{nd} **degree Polynomial kernel SVM:** $minDCF(\tilde{\pi} = 0.0909) = 0.262$
   - PCA dim: 6
   - Z-normalization: False
   - const: 1

- C: 10^0

3. **GMM:** $minDCF(\tilde{\pi} = 0.0909) = 0.245$
    - PCA dim: 6
    - Mode target: diagonal covariance
    - Mode non-target: full covariance
    - Target nr. clusters: 2
    - Non-target nr. clusters: 8

At this point, we conclude that GMM under the hyperparameters configuration reported above is the best performing model for our task. In the Evaluation section we will see whether this is true also on new not-previously-seen data.

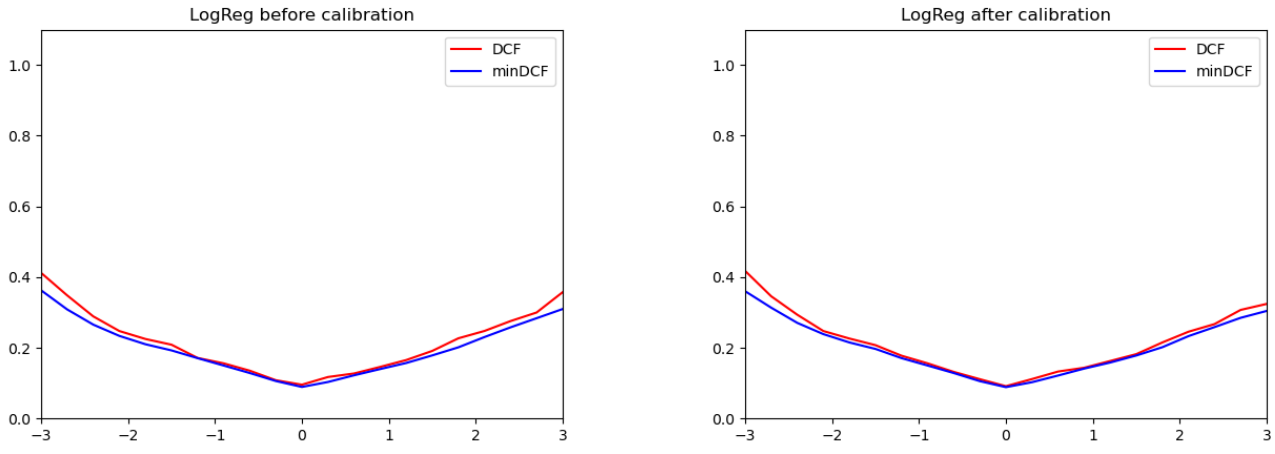
6 Calibration

Very often, the model assumptions we make are not very close to reality. For example, we could assume a Gaussian distribution over the samples of a certain class and more often than not, this assumption is not very accurate. Another issue that may arise is the mismatch between the train and evaluation sets. Because of this, in binary tasks, the log-likelihood ratios (llr) that we treat as scores may be **mis-calibrated**.

Mis-calibrated scores have the effect that the theoretical optimal threshold $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$ is not the most optimal threshold. Score calibration represents mapping the scores to a form such that the most optimal threshold is the theoretical optimal threshold.

The calibration model we use is the prior-weighted logistic regression. The model is trained like the classifiers we trained, i.e. we use k-fold cross-validation. As training set we use the mis-calibrated scores together with the true labels. The pooled outputs (as described in Subsection 3.2 Training protocol) are the calibrated scores on the validation data.

6.1 Quadratic Logistic regression



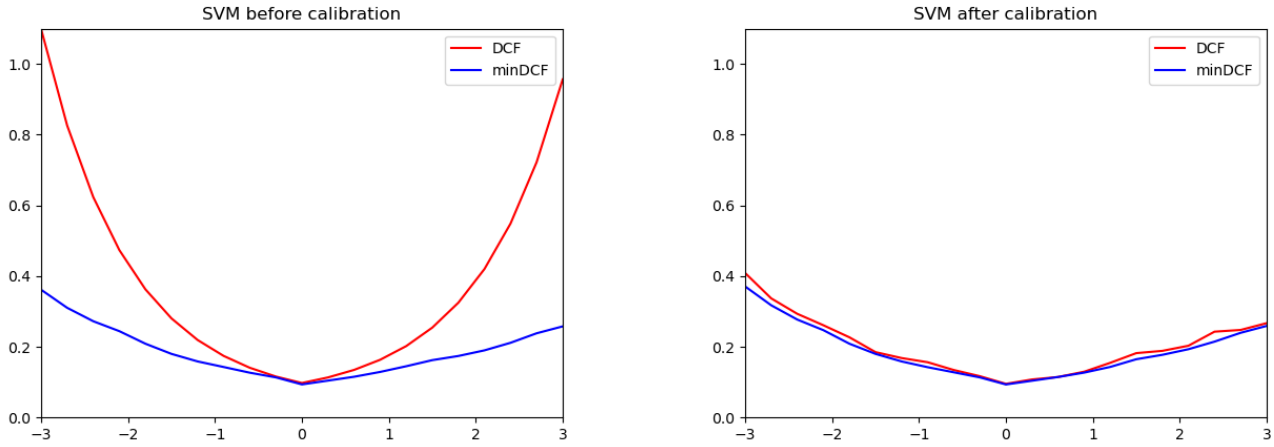
(a) LogReg before calobration

(b) LogReg after calobration

Figure 9: Bayes Error Plots for QuadrLogReg before and after calibration

As seen in Figure 9.a, the logistic regression model is decently calibrated. We see some improvements after calibration, but there is still some mis-calibration for very small and very large effective priors (see leftmost and rightmost sides of Figure 9.b).

6.2 2^{nd} order polynomial kernel SVM



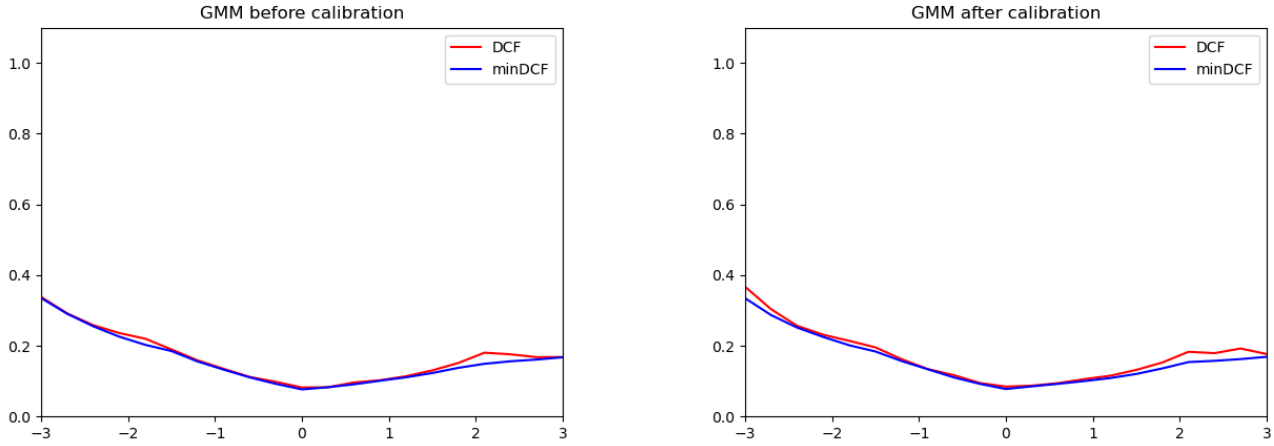
(a) SVM before calobration

(b) SVM after calobration

Figure 10: Bayes Error Plots for Poly(2)SVM before and after calibration

Figure 10.a reports extreme mis-calibration. After calibration, we obtain nearly complete calibration.

6.3 GMM



(a) GMM before calobration

(b) GMM after calobration

Figure 11: Bayes Error Plots for GMM before and after calibration

GMM is the least mis-calibrated model of the three, so that it doesn't actually need calibration. We apply it anyway and we see that it leads to a bit more mis-calibration.

7 Evaluation

Now we test the chosen models and some of the discarded ones on the evaluation set. The evaluation set consists of samples of the same types as the ones in the training set and they were not used for training the models.

7.1 Selected models

Model	$minDCF(\tilde{\pi} = 0.0909)$
GMM(PCA: **6**, targetMode: **diag**, nontargetMode: **full**, targetComps: **2**, nontargetComps: **8**)	**0.216**
QuadrLogReg(PCA: **6**, Z-norm: **False**, λ: $\mathbf{10^{-1}}$)	0.238
Poly(2)SVM(PCA: **6**, Z-norm: **False**, const: **1**, C: $\mathbf{10^{0}}$))	0.242

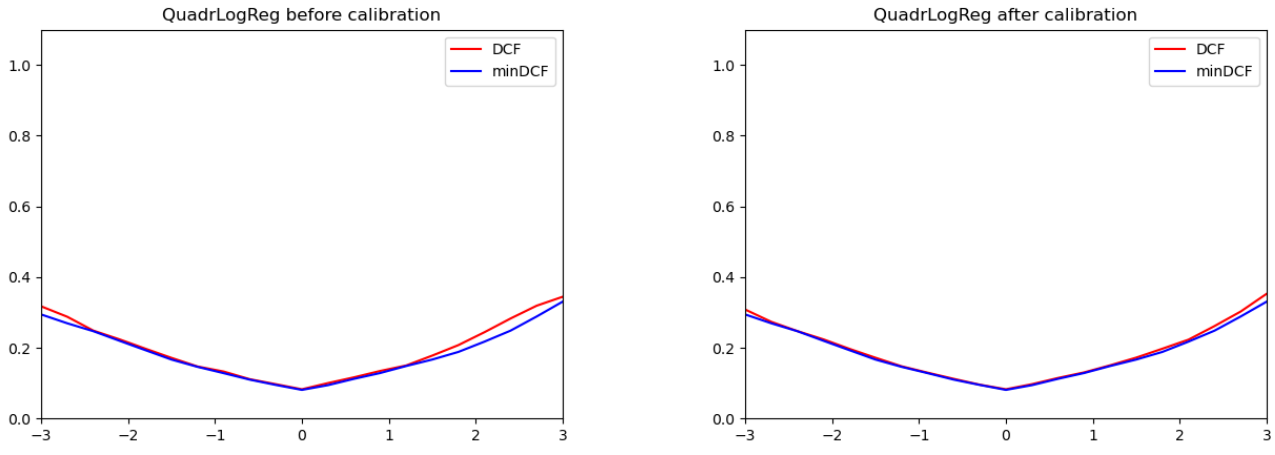
Table 9: Evaluation of the 3 best models seen during validation, one for each model category we tried.

We see that among the best three models, GMM is still the best performing one also after testing them on unseen data. Now we have to see if this particular GMM model is still the best when compared to some of the discarded models after validation.

7.1.1 Calibration on evaluation

To calibrated the scores on the evaluation data, we use the calibration model we trained on the scores on the validation data. So, for example, to calibrate the scores of GMM, we use the calibration model trained on the mis-calibrated scores of GMM of validation data to calibrate the mis-calibrated scores of GMM of evaluation data. We don't use the evaluation data to train anything.

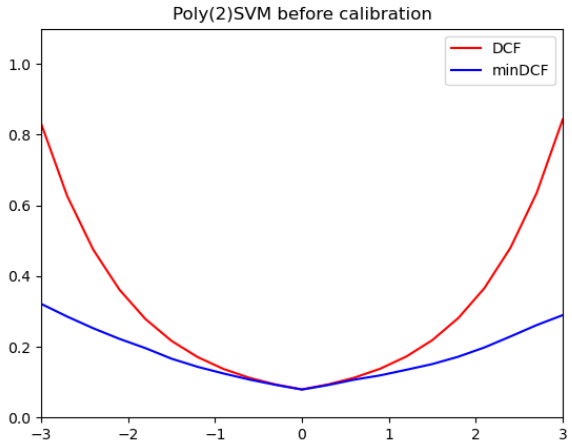
We notice the same aspects as in the calibration of the validation scores. Logistic regression and GMM models are fairly well calibrated from the beginning, whereas SVM is severely uncalibrated. In all three cases, calibration is successful, with a bit of mis-calibration corresponding to very small and very large effective prior values (i.e. at the left and right extremities of the Bayes Error Plots). However, in Figure 14.b, GMM sees more significant mis-calibration for very large effective priors (i.e. at the right extremity), but not too serious.



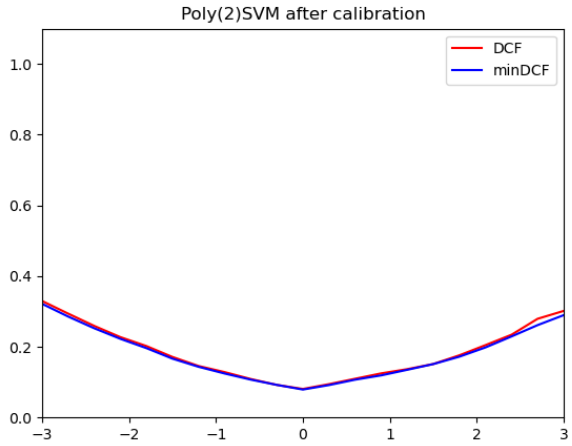
(a) LogReg before calobration

(b) LogReg after calobration

Figure 12: Bayes Error Plots for QuadrLogReg before and after calibration on evaluation data

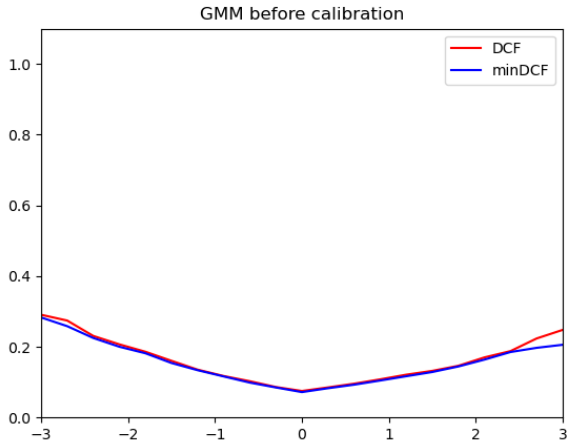


(a) SVM before calobration

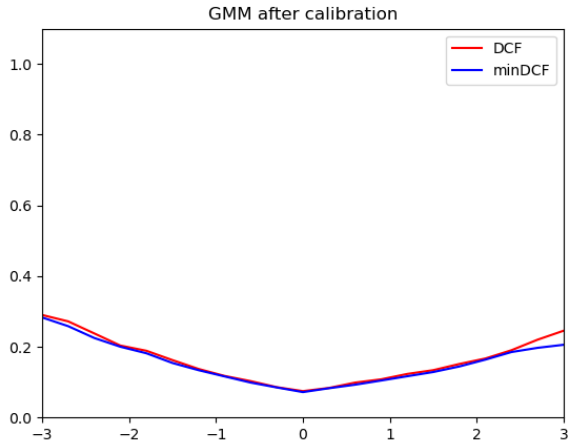


(b) SVM after calobration

Figure 13: Bayes Error Plots for Poly(2)SVM before and after calibration on evaluation data



(a) GMM before calobration



(b) GMM after calobration

Figure 14: Bayes Error Plots for GMM before and after calibration on evaluation data

7.2 Evaluation of some discarded models

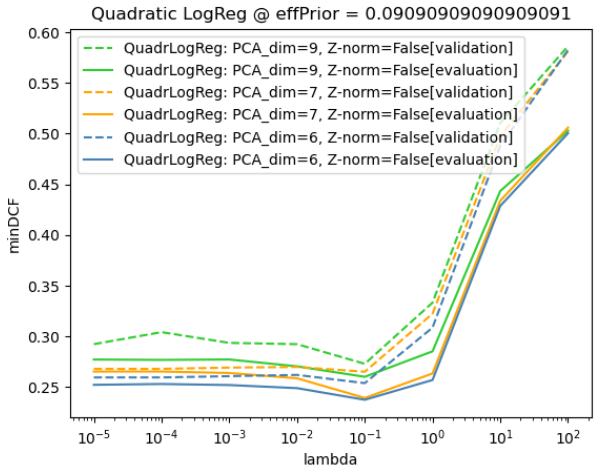
7.2.1 Quadratic Logistic Regression

PCA dim	Z-norm	λ	$minDCF(\tilde{\pi} = 0.0909)$	$minDCF(\tilde{\pi} = 0.5)$	$minDCF(\tilde{\pi} = 0.9)$
6	False	10^{-1}	**0.238**	0.081	0.226
7	False	10^{-1}	0.239	0.083	0.228
9	False	10^{-1}	0.260	0.086	0.224

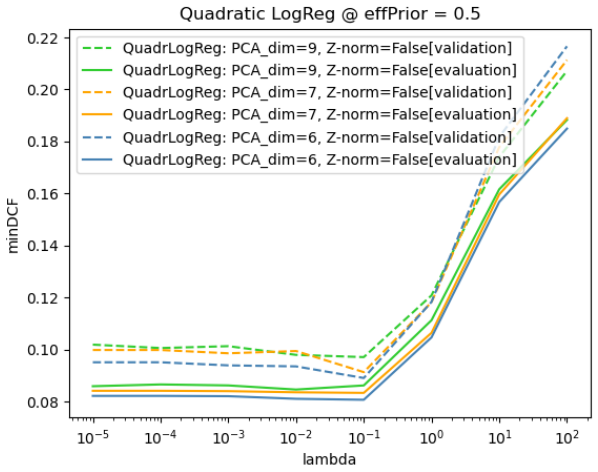
Table 10: **Quadratic LogReg** evaluation results under 3 different working points. These three configurations correspond to the best ones seen during evaluation

The best hyperparameters configurations for each PCA dimension during evaluation correspond exactly to the best ones during evaluation. So, the ($PCA_{dim} = 6$, $Z_{norm} = False$, $\lambda = 10^{-1}$) proves to be the

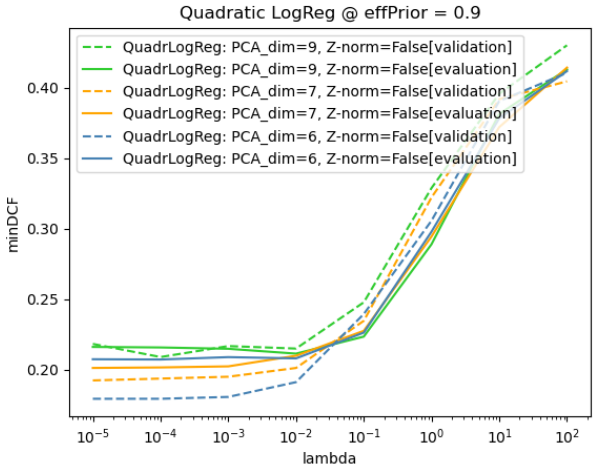
best one among quadratic logistic regression models during both validation and evaluation.



(a) $\tilde{\pi}_T = 0.0909$ (Target application)



(b) $\tilde{\pi}_T = 0.5$



(c) $\tilde{\pi}_T = 0.9$

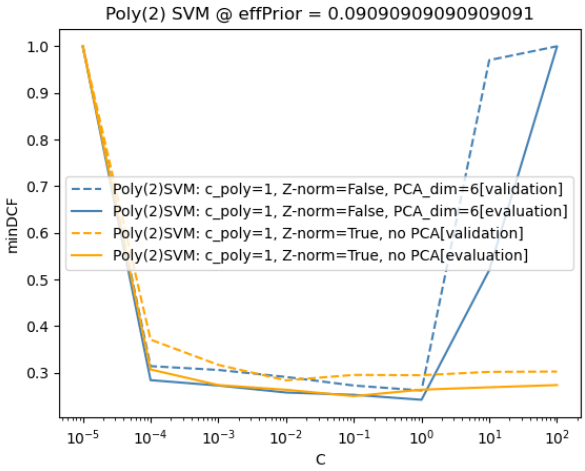
Figure 15: Quadratic LogReg minDCF vs λ plots

7.2.2 2^{nd} order polynomial kernel SVM

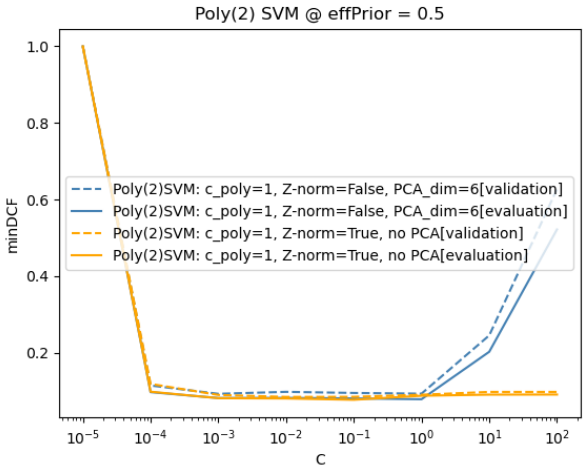
PCA dim	Z-norm	const	C	$minDCF(\tilde{\pi} = 0.0909)$	$minDCF(\tilde{\pi} = 0.5)$	$minDCF(\tilde{\pi} = 0.9)$
6	False	1	10^0	**0.242**	0.079	0.207
N/A	True	1	10^{-1}	0.250	0.081	0.185

Table 11: **Poly(2) SVM** evaluation results under 3 different working points. For each of top 2 PCA dimensions (in terms of $minDCF(\tilde{\pi} = 0.0909)$ performance), the C value corresponding to the lowest $minDCF(\tilde{\pi} = 0.0909)$ has been chosen

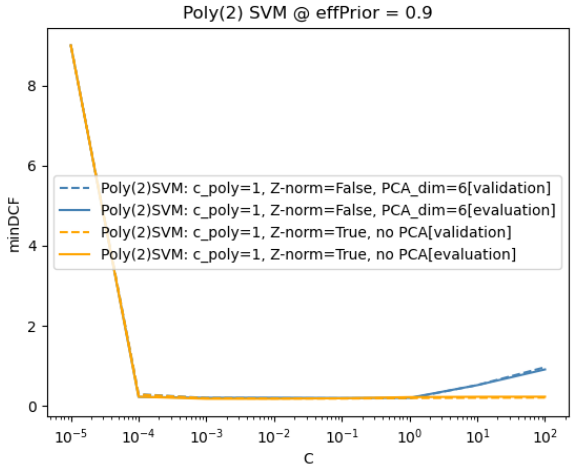
Also in the case of 2^{nd} order polynomial kernel SVM, the best model chosen during validation corresponds to the best one during evaluation. We notice, however, by comparing Table 5 and Table 11, the best configuration when not applying PCA is different between validation and evaluation. The best one during evaluation has $C = 10^{-1}$, whereas for validation has $C = 10^{-2}$.



(a) $\tilde{\pi}_T = 0.0909$ (Target application)



(b) $\tilde{\pi}_T = 0.5$



(c) $\tilde{\pi}_T = 0.9$

Figure 16: 2^{nd} order polynomial kernel SVM minDCF vs C plots

7.2.3 GMM

PCA dim	Mode target	Mode non-target	Target: nr. clusters	Non-target: nr. clusters	$minDCF(0.0909)$	$minDCF(0.5)$	$minDCF(0.9)$
6	diag	full	2	8	**0.216**	0.072	0.170
	tied	full	2	8	0.225	0.073	0.174
	full	full	1	8	0.225	0.073	0.174
	diag	diag	2	8	0.220	**0.066**	**0.160**
	full	diag	1	8	0.219	0.069	**0.160**
	tied	diag	2	8	0.219	0.069	**0.160**

Table 12: GMM **evaluation** results under 3 different working points. For no dimensionality reduction and 6 PCA directions, here are reported the 6 best hyperparameters configurations

Finally, we see that also in the case of GMM, the configuration chosen during validation performs the best also during evaluation.

8 Conclusion

Looking at the results obtain, we see that our expectations were proven to be right by our tests. We initially said that a linear model is not suited for this task and that a quadratic model will work better. This can be seen in logistic regression and SVM when comparing their "standard" linear form to their quadratic counterparts. We obtained significantly better results when casting the problem to the expanded space.

Another expectation proven to be correct was that a GMM model would perform better than a MVG, due to the fact that the non-target class (i.e. the spoofed fingerprint samples) cannot be well described by a single Gaussian distribution. Our best GMM model actually describes the target class with more than one cluster as well (2 clusters). Moreover, taking the Naive Bayes assumption on the target class for GMM lead to better results and this makes sense when looking at the heatmap at Figure 3.b. The correlation among the features is low.

Finally, the best models chosen during validation turned out to be the best ones also during evaluation. This means that we made good choices during our experiments.

We conclude that the best model for the task is GMM(PCA: **6**, targetMode: **diag**, nontargetMode: **full**, targetComps: **2**, nontargetComps: **8**).