Francesco Blangiardi s288265

Gender Recognition Report

Introduction

This project was developed exclusively by Francesco Blangiardi (s288265) as the final assignment of the "Machine Learning and Pattern Recognition" course. The task consists in a binary classification problem on a dataset composed of synthetic speaker embeddings representing the acoustic characteristics of a spoken utterance. The aim of the classification is to separate the embeddings generated according to the gender (male or female) of the speaker. The techniques used to perform the task are all part of the program of the MLPR course, and include several classifiers (gaussian, logistic regression, GMM etc…) with several combinations of preprocessing techniques, and all of them were first evaluated on a training set and later applied to the test set, both sets being provided by the professor. No deep learning techniques were used and no ML libraries were used, as requested by the professor (some basic python libraries including numpy and scipy were allowed).

Code Structure

The code is organized in two parts:

1. The Library. The src_lib folder contains the implementation of all the techniques used to perform the task. The most important classes contained in this package are:
   - **PreProcess:** this class represents a preprocessing technique. It contains a method **learn** that accepts a dataset (usually either the whole training dataset or an instance of the k-fold training subset) and allows the class to compute the parameters for the transformation. After *learn* has been called, the transformation can be applied to a dataset through **apply.** Moreover preprocessing techniques can be concatenated through **addNext,** which accepts another PreProcess object. The relevant subclasses of Preprocess are: **PCA, Gaussianize** and **Znorm.**
   - **Model:** this superclass represent a model that can perform a binary classification task. It contains some general informations about the model as well as an instance of *PreProcess.* It contains method **train** which accepts a dataset and computes the model parameters and a method **predict** which instead computes a prediction of the input and returns scores for each pattern (interpretable as likelihood ratios) as well as some secondary informations on the prediction accuracy. Both methods call the suitable methods of he *PreProcess* object. The relevant subclasses are: **MVG_Model, LRBinary_Model, QuadLR_Model, SVML_Model, SVMNL_Model, GMMLBG_Model** (with its subclasses) and **Fusion_Model**
   - **KCV:** a wrapper class that allows performing k-fold cross validation, it contains a *Model*. Its most relevant method is **crossValidate** which takes a dataset as parameters and returns the concatenated scores of each validation fold by using k-fold cross validation on the *Model* instance.
   - **BD_Wrapper:** a wrapper class that contains some methods relevant for Bayes Decision

operations (i. e. DCF computations, Bayes error plots etc.)
- **C_Wrapper:** an ad hoc class that was used only partially to provide score calibration to a *Model,* however its features are implemented in a more general way by **Fusion_Model**

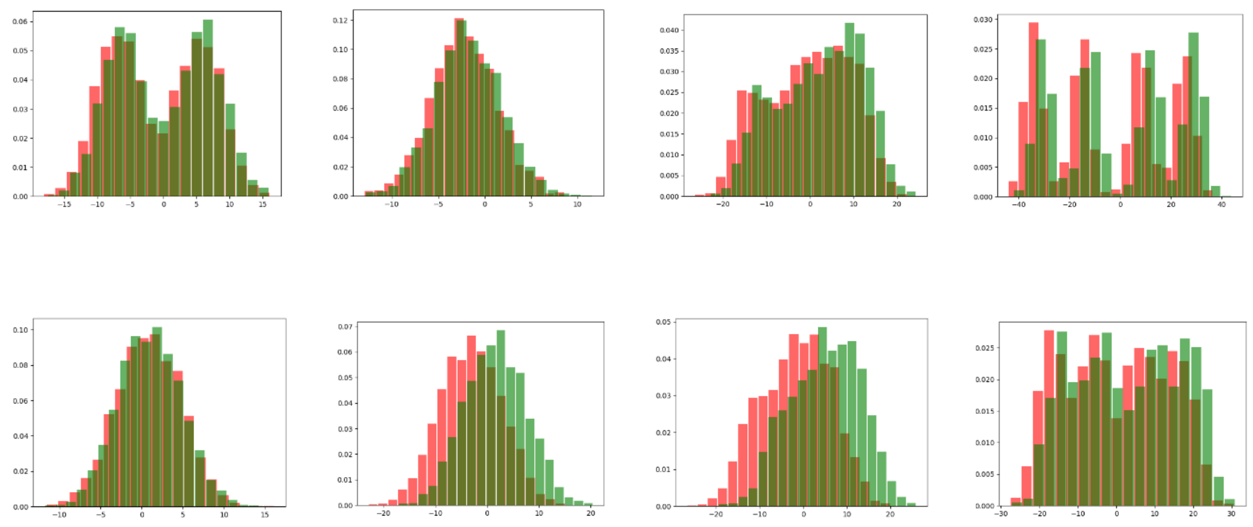
2. The Experiments. In the main folder of the repository it's possible to find several experimentsXXX.py scripts. These scripts contain the code to reproduce all the experiments that are discussed in the report. To run a specific experiment uncomment the related line in the last section of the script file and then run the script from the command line with no parameters.

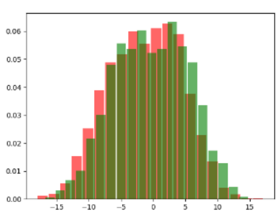
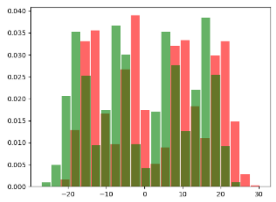
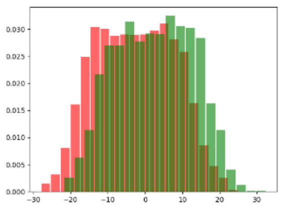
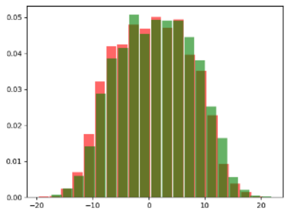
Furthermore the repository contains another both package named **data** with the dataset used as well as a folder **imgs** that contains all the images that are used in the report and more. If the reader is interested in some of the plots that are not explicitly shown in this report, it is possible to find them there.

The dataset

Both train and test set are well balanced, and consist of 3000 and 2000 patterns per class respectively.
Each pattern consists of 12 continuous features, no particular interpretation of them was given by the professor. Moreover, the speakers (both male and female) belong to 4 different age groups, altough the age for each pattern is not provided.

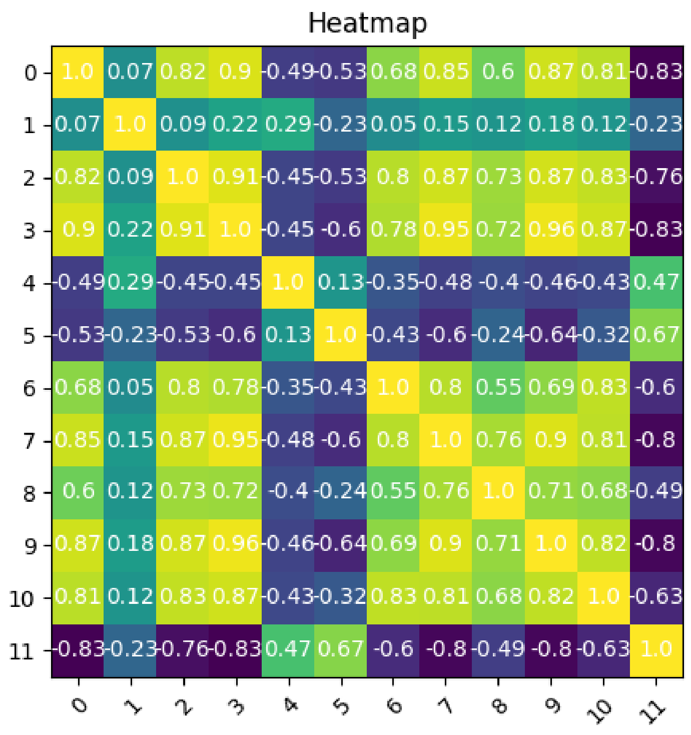
As a first step for analyzing the feature, we first show the histograms of each of the 12 features taken from the training set (ordered from top-left to bottom-right, red for male and green for female):



As we can observe most of the features have a somehow regular distribution without outliers and most of them show approximately a gaussian distribution, so we may expect the MVG classifiers to perform well on this dataset. Some of the features (namely 4 and 10) show some irregularities that may be due to the fact that the patterns include 4 different age groups, but they have the advantage of being discriminative for our task. On the other hand, all the other feature except for feature 6, 7 and 11 show a significant overlap between female and male speakers and may not be too relevant for our models.

It's also useful taking a look at the correlation matrix of the features:

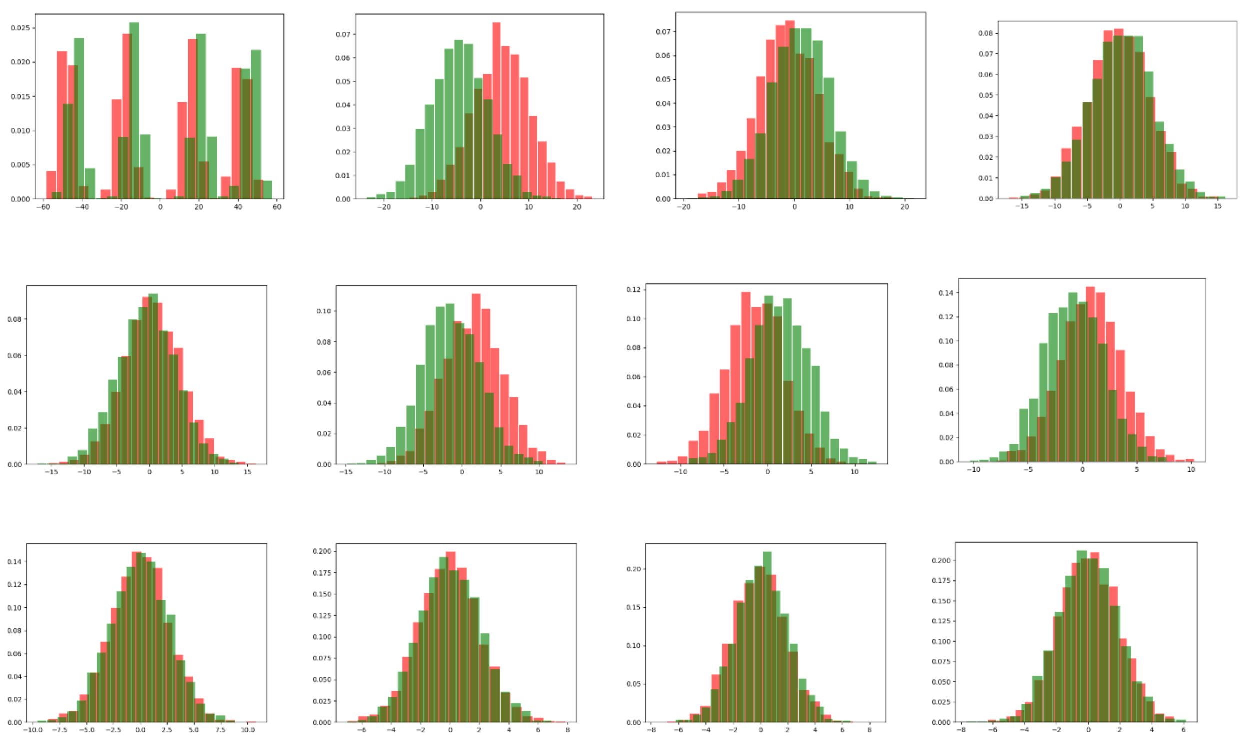


	0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	0.07	0.82	0.9	-0.49	-0.53	0.68	0.85	0.6	0.87	0.81	-0.83
1	0.07	1.0	0.09	0.22	0.29	-0.23	0.05	0.15	0.12	0.18	0.12	-0.23
2	0.82	0.09	1.0	0.91	-0.45	-0.53	0.8	0.87	0.73	0.87	0.83	-0.76
3	0.9	0.22	0.91	1.0	-0.45	-0.6	0.78	0.95	0.72	0.96	0.87	-0.83
4	-0.49	0.29	-0.45	-0.45	1.0	0.13	-0.35	-0.48	-0.4	-0.46	-0.43	0.47
5	-0.53	-0.23	-0.53	-0.6	0.13	1.0	-0.43	-0.6	-0.24	-0.64	-0.32	0.67
6	0.68	0.05	0.8	0.78	-0.35	-0.43	1.0	0.8	0.55	0.69	0.83	-0.6
7	0.85	0.15	0.87	0.95	-0.48	-0.6	0.8	1.0	0.76	0.9	0.81	-0.8
8	0.6	0.12	0.73	0.72	-0.4	-0.24	0.55	0.76	1.0	0.71	0.68	-0.49
9	0.87	0.18	0.87	0.96	-0.46	-0.64	0.69	0.9	0.71	1.0	0.82	-0.8
10	0.81	0.12	0.83	0.87	-0.43	-0.32	0.83	0.81	0.68	0.82	1.0	-0.63
11	-0.83	-0.23	-0.76	-0.83	0.47	0.67	-0.6	-0.8	-0.49	-0.8	-0.63	1.0

Fig. 1: Pearson correlation matrix of the 12 raw features.

As we can see from it there are several features that are strongly correlated (either positively and negatively), therefore it's possible that applying PCA to our dataset may simplify calculations for some models without a decrease in performance.

We can also observe the changes that PCA brings to our histograms:



Altough PCA is not aimed at directly separating the classes (after all it's an unsupervised technique), we can see that the distributions of feature 1, 2,3,6,7 and 8 are quite well separated, and except for feature 1 the distribution of each class shows a rather clear gaussian-like shape. We can conclude that the directions found by the PCA in this case are also directions that can separate well the two classes, and therefore classifiers that usually model correlations between features (for example full covariance MVG) can possibly have also an increase in performance when using PCA. For our experiments we will mainly use PCA 8. We will also see the effects of Gaussianization and Znorm on the model performances altough PCA 8 is our main preprocessing technique.

Model Decisions

The next step we're going to take consists in analyzing the performances of several types of models when solving our classification problem. Our main metric to compare the models will be at first the minDCF on three different applications ($\pi = 0.5, \pi = 0.9$ and $\pi = 0.1$) ; then, an analysis of the actual DCF will be done on the best models together with an evaluation of possible score calibration strategies; finally we will select a model as our final classifier to solve our main task, which will be followed by an overall evaluation of our decisions based on the results computed on the test set.
Model decisions and parameter optimization are going to be done exclusively on the train set and will be performed through the k-fold protocol:

1. The train set will be split into K = 5 folds. The partition is the same in all the performed experiments and should have a balanced number of patterns from both classes in all folds.

2. Each training will consist of 5 iterations: each time a different fold is selected for validation and the remaining four are used for training the model
3. At each iteration preprocessing parameters are learned only on the 4 folds used for training
4. After the model is trained, a prediction on the validation fold is performed (in a form that can be interpreted as a likelihood ratio per each validation sample). After all 5 iterations are finished, these prediction are concatenated in an array which is therefore made of a score per each training sample.
5. Finally, the array of scores can be used to make our assessments (based on minDCF or actual DCF).

We will now start with an evaluation of minDCF for several different models.

MinDCF Evaluation

MVG Classifiers

The first family of models that we will see are Gaussian classifiers. Since these models do not have hyperparameters there's no need to perform parameter optimization, however we will still analyze the four main types of MVG classifiers (Full Covariance, Diagonal Covariance, Tied Covariance and Tied Diagonal Covariance) with different preprocessing techniques. Here are the minDCF results:

Raw features:

model	Π =0.5	Π = 0.9	Π = 0.1
Full Covariance	0.0490	0.1206	0.1266
Tied Covariance	0.0470	0.1263	0.1210
Diagonal Covariance	0.5654	0.8615	0.8170
Tied Diagonal	0.5683	0.8579	0.8210

Pca 8:

model	Π =0.5	Π = 0.9	Π = 0.1
Full Covariance	0.0446	0.1226	0.1403
Tied Covariance	0.0446	0.1256	0.1320
Diagonal Covariance	0.0673	0.1623	0.1703
Tied Diagonal	0.0650	0.1610	0.1633

Gaussianization

model	Π =0.5	Π = 0.9	Π = 0.1
Full Covariance	0.0631	0.1773	0.1856
Tied Covariance	0.0593	0.1646	0.1790
Diagonal Covariance	0.5409	0.8356	0.8029

Tied Diagonal	0.5403	0.8236	0.8030

Z normalization

model	Π =0.5	Π = 0.9	Π = 0.1
Full Covariance	0.0490	0.1206	0.1266
Tied Covariance	0.0470	0.1263	0.1210
Diagonal Covariance	0.5653	0.8616	0.8170
Tied Diagonal	0.5654	0.8616	0.8170

Z normalization + PCA 8

model	Π =0.5	Π = 0.9	Π = 0.1
Full Covariance	0.1770	0.4210	0.4493
Tied Covariance	0.1740	0.4090	0.4489
Diagonal Covariance	0.1843	0.4343	0.4553
Tied Diagonal	0.1833	0.4283	0.4549

As we could expect from the analysis of features MVG classifier can achieve good performance in our task.
The Full Covariance and Tied Covariance models outperform the other two types with all preprocessing techniques. In particular, for raw features the diagonal and the tied diagonal models have relatively very low performances, which may be due to the fact that these models (differently from FC and T) should work with the assumption that the features are uncorrelated, which is far from true in our case.
Pca however solves the problem for these two model (since PCA features are uncorrelated features extracted from the raw ones with a small loss of information) and improves the performance of the FC and T models, even if just for the balanced application.
Gaussianization seems to harm performances a little; a possible explanation for this is that raw features are already well fit for our task, and that gaussianization being a non linear transformation does interfere partially with the correlations between features, as we can see from the correlation matrix of the gaussianized features:

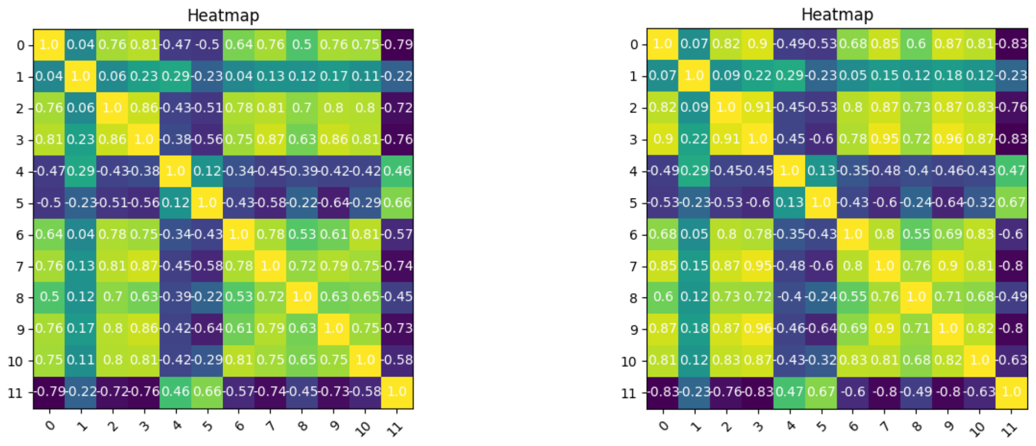
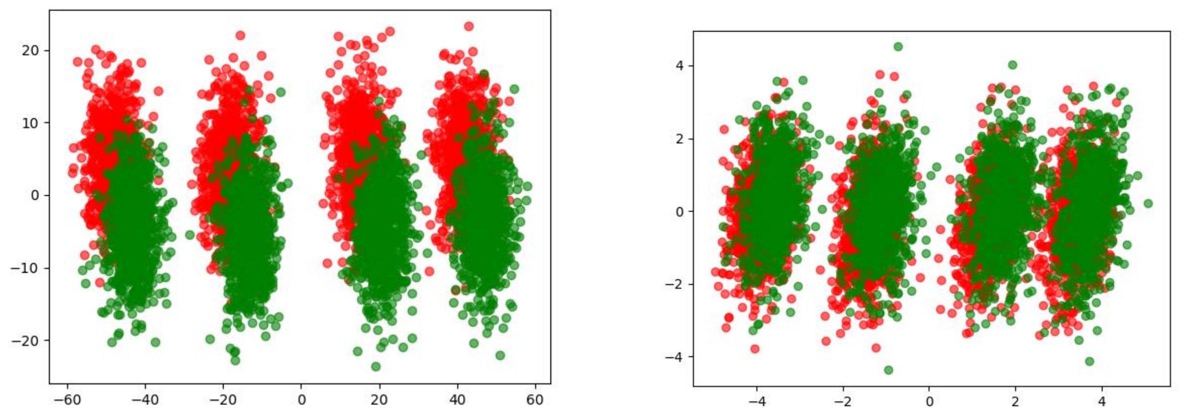


Fig. 2: Comparison of the correlation for gaussianized vs raw features. Values are slightly different

Another possible problematic of the gaussian classifier is that it can be affected by the difference in terms of class distribution between train and validation folds (in fact ranking is computed only on the train set and later gets applied to the validation set), which is another reason why gaussianized feature are not supposed to be beneficial to a well balanced and regulary distributed dataset.

Another interesting behavior comes from Z normalization: altough it has no effect with respect to the classifier using raw features, it seems that applying Z normalization together (before) PCA has a negative effect on performances. This is an unexpected result, so we can try to find an empyrical explanation for this fenomenon looking at the plots of the transormed features:



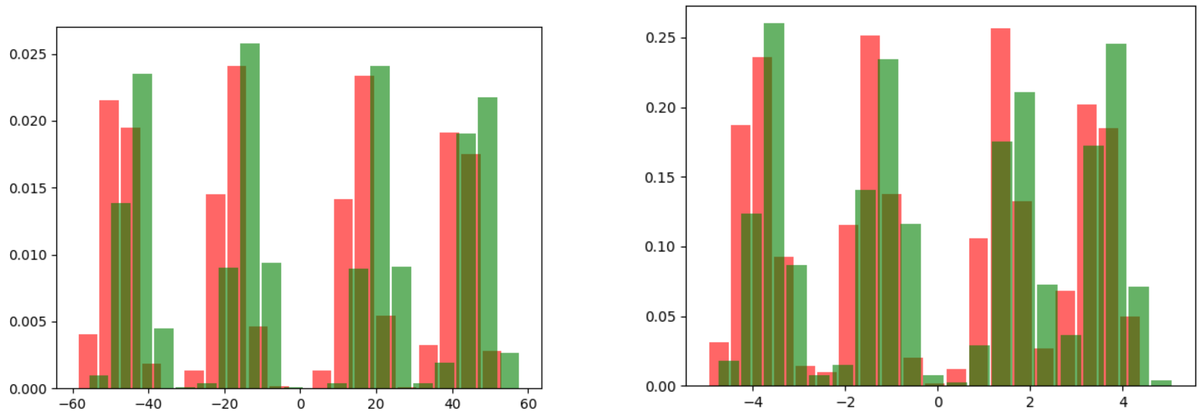


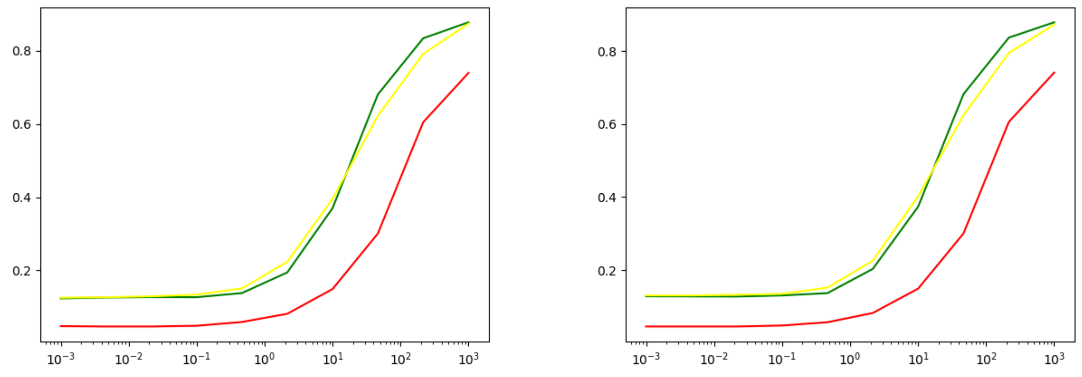
Fig. 3: Scatter of the first two PCA features and histogram of the first (left: no Znorm, right: Znorm)

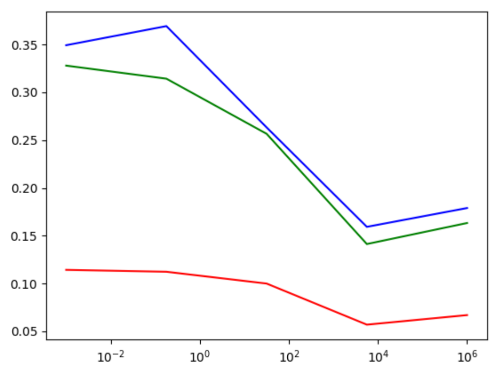
As we can clearly see from the scatter of Fig. 3 the PCA applied after Z normalization does not separate the features as well as PCA on raw features, hence why the combination of the two has lower performance compared to both of the preprocessing techniques used singularly.

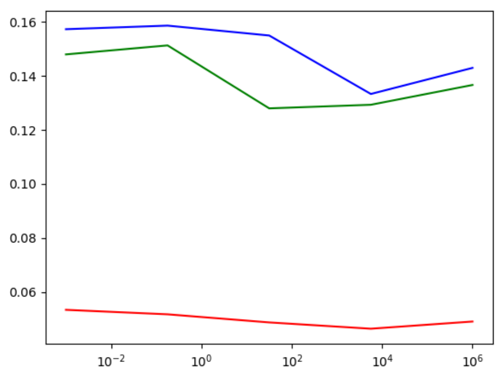
Logistic Regression

The next family we will try is that of Logistic Regression classifiers. We consider both linear LR and quadratic LR with feature expandion. Since we also need to find an optimal parameter *lambda* for our classifiers, we also perform several experiments with different values of the parameter. Also, since our classes are balanced, the prior weighted version of the logistic regression is not included in this report.

We first show the plots of the minDCF for varying values of the lambda parameter (top: linear LR, bottom: quadratic LR, left: raw features, right: pca 8) for our different applications (red: 0.5, green: 0.9, yellow/blue: 0.1):







For reference here are reported the values for the best models:

Raw:

model	Π =0.5	Π = 0.9	Π = 0.1
Linear lambda =0.001	0.0466	0.1233	0.1253
Quadratic lambda=10e4	0.0569	0.1413	0.1593

PCA:

model	Π =0.5	Π = 0.9	Π = 0.1
Linear lambda =0.001	0.0453	0.1283	0.1310
Quadratic lambda=10e4	0.0463	0.1293	0.1333

Gaussianized:

model	Π =0.5	Π = 0.9	Π = 0.1
Linear lambda =0.001	0.0580	0.1686	0.1730
Quadratic lambda=10e-2	0.0553	0.1536	0.1560

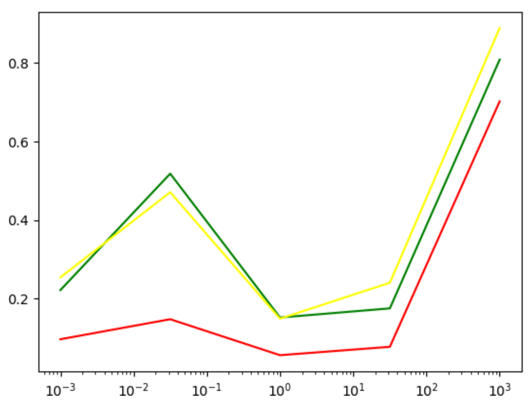
As we can see from the plots, the linear model on raw features perform well with a low value for lambda, hence they don't require regularization, and can actually match the performances for the best MVG classifiers. The same considerations can be made for linear model using pca, and just like the MVG model it does retain better performances on the main application.
Quadratic models instead seem to need some regularization to generalize better, which leads to the plots reaching their lowest minDCF on lambda = 10e4. An explanation for this is that the model may tend to overfit on the train set because of the high dimensionality of the expanded data, and therefore struggles to reach good performances on the validation set. This is supported by the fact that using pca increases performances, getting them close to the linear model, since the dimensionality of the expanded features is more than halved (12^2 vs 8^2)

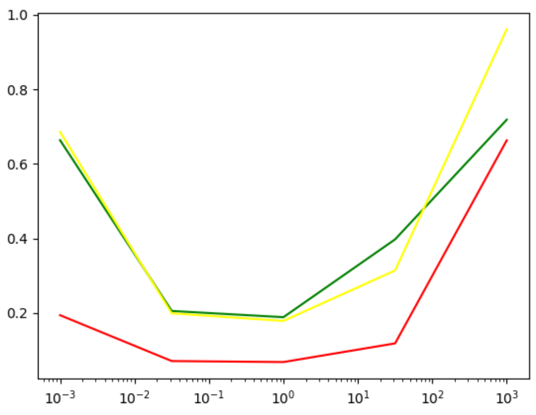
Finally some measurements of the best models with gaussianizations are reported, but similarly to MVG models the Gaussianization preprocessing doesn't bring any improvement (except for a slight improvement for quadratic lambda's balanced application with respect to the model trained on raw features).

SVM

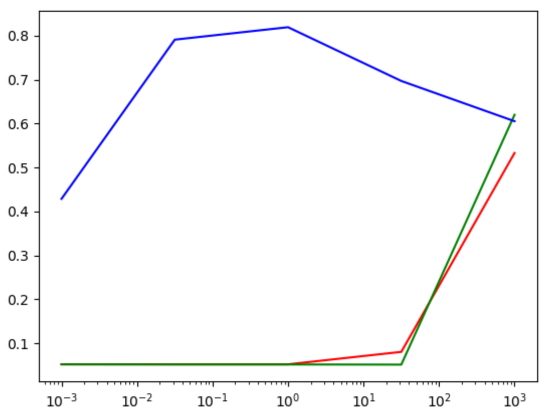
We now turn our attention to SVM models. For these models optimization on parameter C was performed, and both linear and kernel (RBF and polynomial quadratic) SVM were taken in consideration. For RBF kernel also different values of the gamma parameters were considered. As in the case of logistic regression, the class balanced version of SVM was not applied.

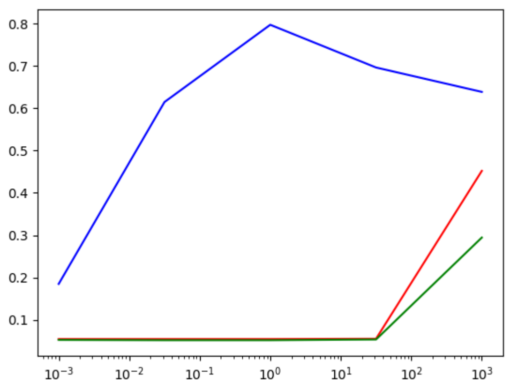
In the following the results of optimization on parameter C are shown for the linear SVM (left: raw, right: pca) for our three applications (red: 0.5, green: 0.9, yellow: 0.1):





Here are instead the plots for the RBF SVM with varying C on the main application with different values of gamma (red: 0.001, green: 0.01, blue: 0.1):





For completeness here's a summary of the measured minDCF results for the best models:

Raw:

model	Π =0.5	Π = 0.9	Π = 0.1
Linear C = 1	0.0556	0.1516	0.1486
Polynomial C = 10e-3	0.5376	-	-
RBF gamma=0.01, C = 20	0.0520	0.1783	0.1583

PCA:

model	Π =0.5	Π = 0.9	Π = 0.1
Linear C = 1	0.0683	0.1886	0.1786
Polynomial C = 10e-3	0.9786	-	-
RBF gamma=0.01, C = 20	0.0516	0.1780	0.1580

Gaussianization:

model	Π =0.5	Π = 0.9	Π = 0.1
Linear C = 1	0.0550	0.1586	0.1566
Polynomial C = 20	0.0546	0.1503	0.1676
RBF gamma=0.01, C = 20	0.0673	0.3996	0.3896

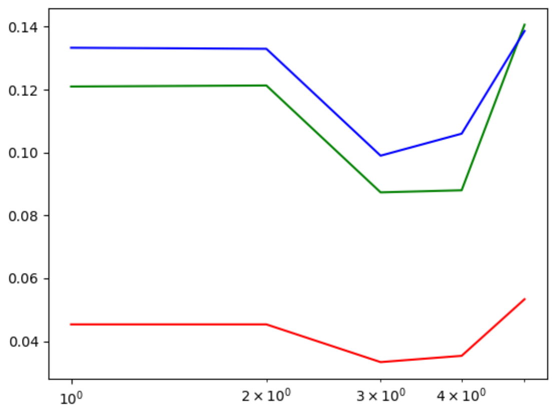
We can observe that the linear SVM does require some regularization, as the plot gives the best minDCF for C = 1. For the first time so far we also don't find an improvement when using PCA compared to the raw features, altough C= 1 still provides the best results.
As for the RBF SVM, the parameter gamma yields roughly the same plot for values of 0.01 and 0.001, while the model doesn't perform very well when gamma is 0.1. Also in this case the model requires some regularization, with C giving the best performances at value 20 (altough performances decrease for higher values). Like in the case of MVG and LR, pca gives slightly better performances for the main application when combined with the RBF kernel.
Polynomial quadratic SVM has instead relatively bad performances on the raw features and even worse performances on the PCA. This may be due to the fact that the optimization algorithm that was used to train the models struggled to find a sufficiently good solution with the given constraints on computational time. Surprisingly enough, gaussianization helps these models delivering a reasonably good result in a reasonable time.

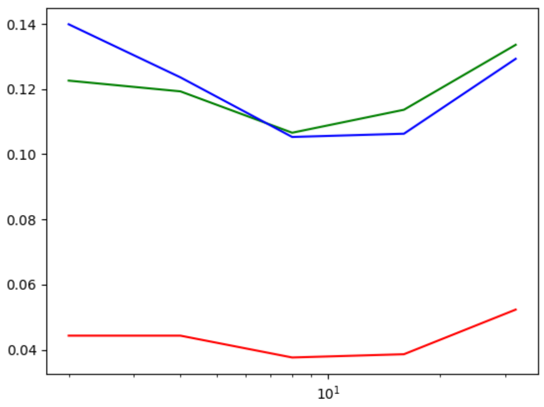
In conclusion the SVM models performed slightly worse than the MVG and LR models, but it's worth mentioning that training these kind of models takes a lot of time and can potentially reach better performances if more time is allocated for their training.

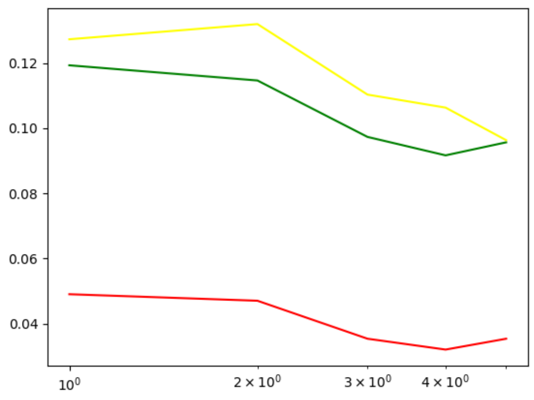
GMM

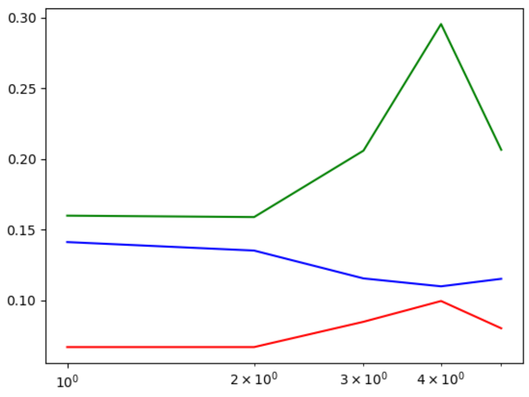
The last family of models we take as candidates for our task are Gaussian Mixture Models. We normally would expect these models to perform just as well as MVG or even better since they provide a more general and powerful way to model class distributions. To train them the LBG algorithm was used just as explained in classes and laboratories. Moreover all 4 main types of GMM will be evaluated: Full Covariance, Tied, Diagonal and Tied Diagonal. For Tied models the covariance matrixes are tied only within the same class (i.e. covariances belonging to components of different classes can be different). Parameter optimization is performed on the number of components used to model each class distribution, and values in the set [2, 4, 8, 16, 32] were considered.

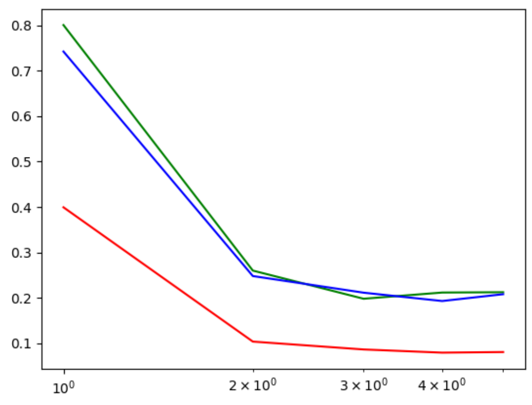
The following plots show the minDCF for varying number of gmm components (expressed as the exponent of the related power of 2). Plots include all the models (in order from top to bottom: FC, T, D, TD) with both raw features (left) and pca (right) for our three main applications (red: 0.5, green: 0.9, yellow/blue: 0.1):

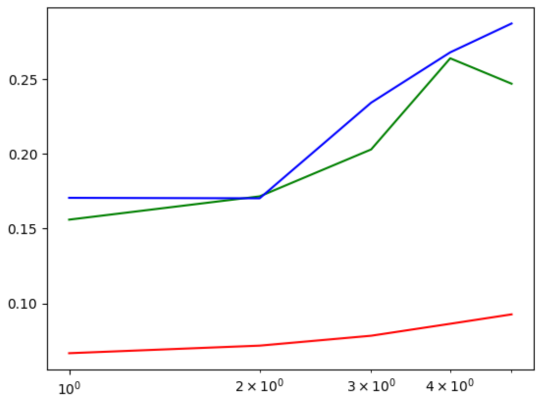


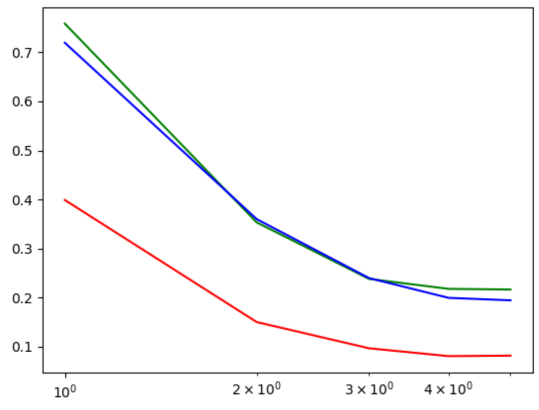


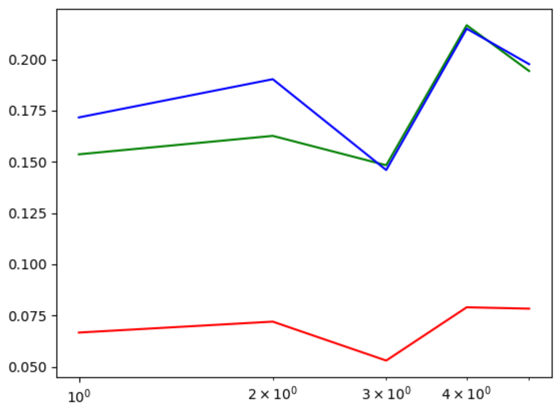












Also, as usual, values for the best models are provided:

Raw:

model	Π =0.5	Π = 0.9	Π = 0.1
FC g = 8	0.0333	0.0873	0.0989
T g = 16	0.0320	0.0916	0.1063
D g = 32	0.0793	0.2116	0.2116
TD g = 16	0.0810	0.2180	0.1996

PCA:

model	Π =0.5	Π = 0.9	Π = 0.1
FC g = 8	0.0376	0.1066	0.1053
T g = 4	0.0670	0.1590	0.1353
D g = 4	0.0666	0.1560	0.1706
TD g = 16	0.0530	0.1483	0.1460

Gaussianization:

model	Π =0.5	Π = 0.9	Π = 0.1
FC g = 8	0.0456	0.1246	0.1283
T g = 8	0.0579	0.1406	0.1669

D g = 16	0.1020	0.2740	0.2593
TD g = 8	0.1473	0.3433	0.4260

As expected GMM models generally perform better than MVG model. An easy explanation can be found looking at the plots of feature 4 and 10 as well as the scatter that was provided in Fig. 3: our patterns belong to 4 age groups per class and this is reflected in these plots showing 4 well separated clusters. For this reason it makes sense that the gmms retain better performance, since they have the possibility to properly model these 4 slightly different distributions for both classes. The best GMM models are the FC and T trained on raw features, with T being better for the main application while FC is better for the two secondary ones. D and TD models perform relatively worse (we can assume roughly the same reasons as their MVG counterpart), altough apparently with the increase of the number of components they are able to deliver overall decent performances (unlike their MVG versions).
This time pca does not bring to an improvement with respect to simply using the raw features, which is likely due to the fact that, as we can see from the plots, these models tend to overfit when the number of components increase. Only the FC version of the model is able to deliver competitive results with respect to the models trained on raw features.
As for gaussianization, even in this case it proves to be harmful to the performances

In conclusion, we can safely say that the models that perform the best in terms of minDCF among all the ones we evaluated are **GMM FC 8** and **GMM T 16** when trained on raw features. The reason can be summarized in the fact that our dataset can be very well modelled by GMM distribution with not too many components (since we have 4 different age groups forming 4 distributions) as well as the fact that these two models are fully capable of exploiting the strong correlations between the dataset features.

Actual DCF Assessments

Since we have evaluated extensively our models on the minDCF and found the two best ones, we can now assess how they perform in a real case scenario. In such case we cannot calculate the best possible separating threshold on the test set but we’re actually forced to use the application dependent theoretical threshold to separate the scores and make our predictions. As studied in class, the cost corresponding to the theoretical threshold is called actual DCF and it is the metric we’re going to use to compare model performance in a real case scenario and t their ability to provide well calibrated score predictions (i.e. scores that allow a class separation leading to an actual DCF close to the minimum DCF).
We first look at the Bayes error Plots for the two best models:

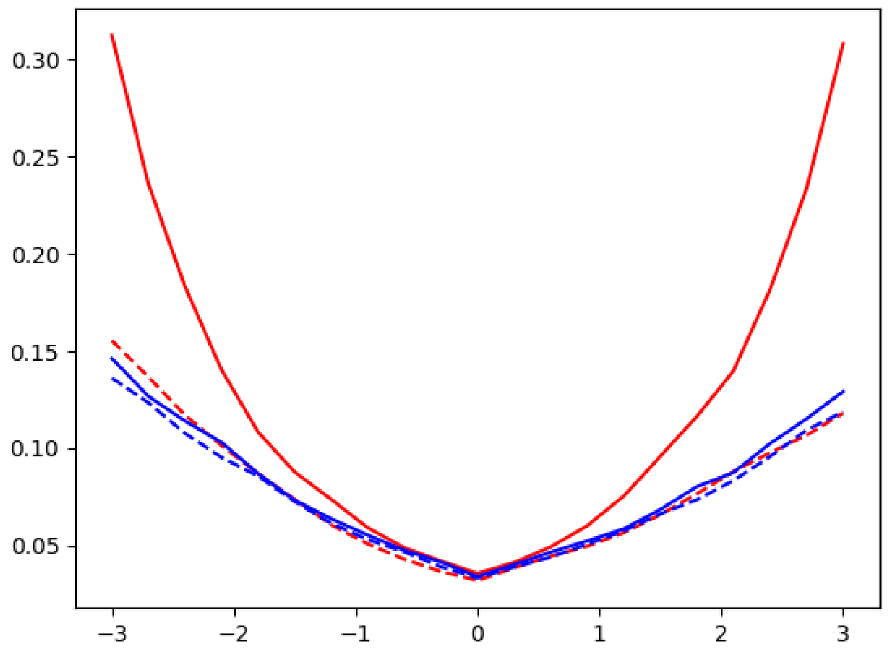


Fig 4. Bayes error plot for GMM T 16 (red) and GMM FC 8 (red). Dashed lines are minDCFs while continuous lines are actDCFs. On the x axis we have the prior log odds, a value that is linearly proportional to the application's effective prior (which determines the theoretical threshold as studied in class)

Here we also report a table with the minDCF vs actDCF values for our three main applications:

model	Π =0.5	Π = 0.9	Π = 0.1
GMM T 16	0.320 – 0.0356	0.0916 – 0.1520	0.1063 – 0.1530
GMM FC 8	0.0333 – 0.0340	0.0873 – 0.0890	0.0989 – 0.1043

It is clear that the GMM FC 8 gives well calibrated scores for a very wide variety of applications. However the same cannot be said about the GMM T 16, which gives an actDCF almost twice higher than the minDCF for applications with a very unbalanced effective prior.
This means that in those applications, GMM T 16 will not performa at its best unless we either choose a good separating threshold for the scores of the two classes or we actually calibrate the scores predicted by the model in order for them to give an accurate separation through the theoretical threshold. We briefly take both the approaches in consideration for the GMM T 16 model.

Optimal Threshold Estimation

The main idea behind this approach is to find the (optimal) threshold that leads to the minDCF in the validation set. This set can later be used on evaluation and test set to separate the predicted scores and classify their related sample. The expected result is that such threshold will provide a good spearation as a substitute of the theoretical one and that can lead to a low actual DCF. The downside of this approach is that it requires us to estimate an optimal threshold for each of the applications that need to be used/evaluated.

Since we still have to decide whether this approach is suitable for our task or not, we will only use the train set to make our evaluations. The procedure followed in the project consists of:

1. Calculating the set of scores through the cross validation procedure for each sample in the training set (just like the computation we did for the calculation of the minDCF)
2. These scores represent our validation set, but we still need an evaluation to have an unbiased idea of the effect of this approach. To do so, we follow the split protocol and use 80% of these scores as our actual validation scores and the remaining 20% of scores for evaluating the approach through act DCF computation.
3. We estimate the optimal threshold for all the relevant applications on this extracted validation set and we compute the actual DCF on the evaluation set. A comparison of this actual DCF will be done with respect to the DCF computed on this same evaluation set but by using the theoretical threshold (if it's lower and closer to the minDCF then it means that the approach should be worth being used).

Here are the results for the threshold estimation for the GMM T 16 model following the above procedure (of course the minDCF and theoretical actDCF will be different from the ones of the previous table since they are computed on different data sets):

	Best evaluation threshold (minDCF)	Theoretical threshold (act DCF)	Optimal validation threshold (optDCF)
Π =0.5	0.0265	0.0300	0.0297
Π =0.9	0.0837	0.1475	0.0920
Π =0.1	0.1028	0.1267	0.1250

As we can see the optimal threshold leads to roughly the same results on the balanced application but it can provide a great improvement for the application with 0.9 effective prior with respect to the theoretical threshold. Another insight we can give is that, as we can expect from the above table, the optimal threshold is quite similar to the theoretical one for the balanced application. Instead, for the other two applications, the optimal threshold is very different from the theoretical one (these results are not reported here but they can be found in the file measurementsTrain.md) We can therefore say that, according to our assessment, using the estimated optimal threshold on GMM T 16 does bring some benefits compared to using just the theoretical threshold during score separation.

For the sake of completeness we also report the same table for GMM FC 8:

	Best evaluation threshold (minDCF)	Theoretical threshold (act DCF)	Optimal validation threshold (optDCF)
Π =0.5	0.0265	0.0297	0.0297
Π =0.9	0.0812	0.1027	0.1009
Π =0.1	0.0585	0.0870	0.0918

As we could already tell from the bayes plot of Fig. 4, GMM FC 8 provides quite well calibrated scores for a wide range of application and the threshold estimated on the validation set are very close to the theoretical threshold. Therefore we can say that applying this approach to GMM FC 8 does not necessarily improve the actDCF with respect to the theoretical one when calculating the DCF on the evaluation set.

Taking these considerations into account, we can assume that it's not required to perform threshold optimization on our GMM FC 8 (and we take it for granted that it's also not necessary to perform score calibration on it)

Score Calibration

The second approach to obtain better actDCF consists instead of attaching a "calibrator" model right after our main model. As suggested in class, a prior weighted LR model was used for this task. The idea is to train this calibrator on the scores predicted by the model from the validation set, and since our scores can be interpreted as likelihood ratios, they will be one dimensional. This means that the weighted LR calibrator will just have to learn a "scale + shift" function (a fairly easy task) that will be applied on each score, which are therefore supposed to be mapped into well calibrated scores. The consequence of having well calibrated score is that we can use the theoretical threshold to classify our scores for a possibly wide variety of application while attaining reasonably good actual DCF compared to the minDCF
The prior weighted LR model normally requires a weight parameter Πt, but as suggested in class this should not affect particularly the quality of the calibration.
To train the calibrator and assess the quality of the improvement, the same validation and evaluation set of the threshold estimation approach were used.

Here are the results for the different applications and for different Πt used as parameter of the calibrator for the GMM T 16 model:

	minDCF Π =0.5	minDCF Π = 0.9	minDCF Π = 0.1
GMM T 16 (any version)	0.0265	0.0837	0.1028
	actDCF Π =0.5	actDCF Π =0.9	actDCF Π =0.1
No calibration	0.0300	0.1475	0.1267
PW LR Πt = 0.5	0.0300	0.0955	0.0966
PR LR Πt = 0.9	0.0283	0.0955	0.1298

PR LR $\Pi t = 0.1$	0.0336	0.0883	0.1298
Threshold estimation (for comparison)	0.0297	0.0920	0.1250

As we can observe all the calibrators provide some improvemente with respect to the model without calibration. In the case of the calibrator with $\Pi t = 0.5$ we're able to get a significant improvement also with respect to the threshold estimation technique for the 0.1 application (altough it performs slightly worse for the other two applications). In any case, the real strength of this approach is that we just need to train the calibrator to get well calibrated scores for any application, and since we assessed that this brings benefits compared to the uncalibrated model we will choose this approach as our main one over the threshold estimation.

We can also check that the calibrator benefits performances for most applications by looking at the bayes plot for the calibrated model:

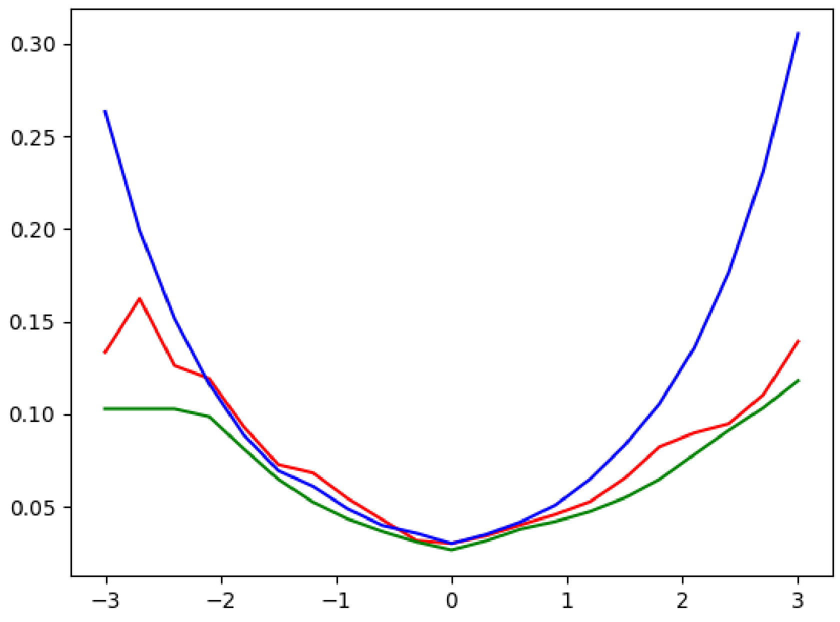


Fig. 5: Bayes plot computed on the same evaluation set that was used to train the calibrator (main model is GMM T 16). Green: minDCF, red: calibration actDCF, blue uncalibrated act DCF

As we can see the calibrated model performs worse only in a small interval of values (which are still faritly well calibrated) while it provides much better performances for most unbalanced application compared to the uncalibrated model.

Fusion Model

To further improve the performances of our two models we can also consider using a fusion of both of them. The implementation which was followed on the project is based on score fusion and uses both GMM T 16 and GMM FC 8: basically the two models are trained separately on

the same training set and produce scores independently at prediction time; these two sets of scores are then forwarded through a calibrator model (a prior weighted LR model as before) which was trained to learn a weighted combination of the two sets of scores. More in detail: our main two models will output a dataset of 2 dimensional samples (one score per model per sample); these scores will be then processed (either for training or predicting) by the calibrator (which this time works with a 2 dimensional dataset); the calibrator will output as prediction a one dimensional array of calibrated scores that represent both the prediction of the GMM T 16 and of the GMM FC 8. The expected result is that the fusion of the scores will lead to better performances than the scores taken from both the models individually. Since prior weighted LR calibrator still has to learn a rather easy low dimensional task, we can also assume that the output of our fusion model will be calibrated.
The calibrator is trained on the validation set extracted through cross validation (just like regular model calibration), so for the assessment phase we also have followed the very same split of validation and evaluation set described in the two previous approaches.

In the following it's reported a comparison in terms of minimum and actual DCF as well as a Bayes error plot between the Fusion model scores and the calibrated GMM T 16 scores computed on the extracted evaluation set:

	minDCF Π =0.5	minDCF Π = 0.9	minDCF Π = 0.1
Fusion	0.0253	0.0688	0.0807
GMM T 16 cal	0.0265	0.0836	0.1028
GMM FC 8	0.0265	0.8120	0.0585
	actDCF Π =0.5	actDCF Π =0.9	actDCF Π =0.1
Fusion	0.0281	0.0955	0.0886
GMM T 16 cal	0.0300	0.0955	0.1266
GMM FC 8	0.0297	0.1027	0.0870

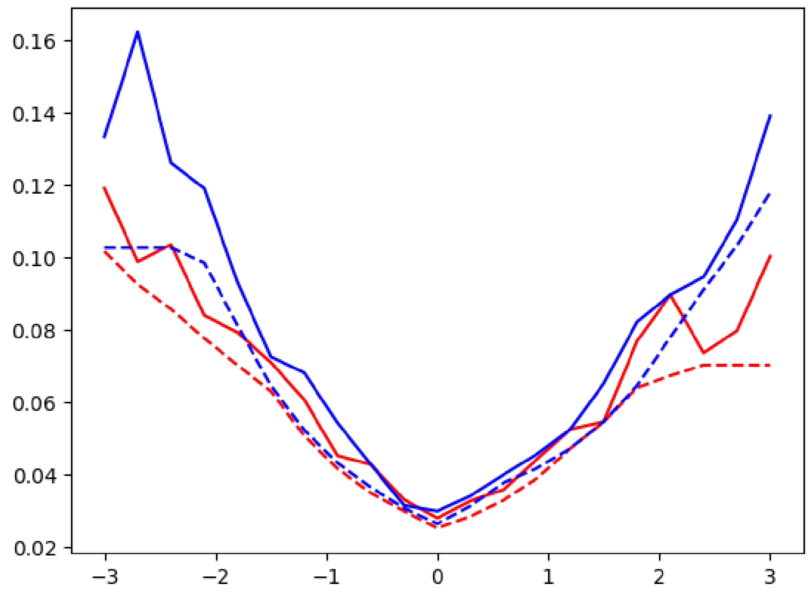


Fig 6. : the fusion model (red) produces better results than the calibrated GMM T 16 (blue) in every application. (dashed line is minDCF, continuous line is act DCF

As we can see the Fusion model takes the best of the two model and delivers scores that are well calibrated and make for a relatively low actual DCF on a very wide range of applications.

For these reasons the Fusion model is selected as the best and final model for this task.

Model Evaluations

It is now time to make use of the test set to see whether the optimizations and decision we made in the previous section actually lead to better result also on a wider dataset of unseen data. We will no longer use the kcv protocol (except for training calibrators and for threshold estimation, since they require a validation that we can extract through cross validation) and we will train our models on the entire train set. Just as before we first make some considerations on some models by comparing their related minDCFs (to see if our parameter optimizations and other kinds of excpectations are reproduced also on the test set) and we will later assess the quality of our decisions also in terms of actual DCF performances.

MinDCF Evaluation

MVG

We take a look at the best MVG classifier we found in the decision phase, namely FC, T both with raw features and PCA 8 preprocessing

Raw:

model	Π =0.5	Π = 0.9	Π = 0.1
Full Covariance	0.0530	0.1375	0.1325
Tied Covariance	0.0505	0.1350	0.1325

PCA:

model	Π =0.5	Π = 0.9	Π = 0.1
Full Covariance	0.0525	0.1390	0.1430
Tied Covariance	0.0535	0.1340	0.1425

Gaussianization:

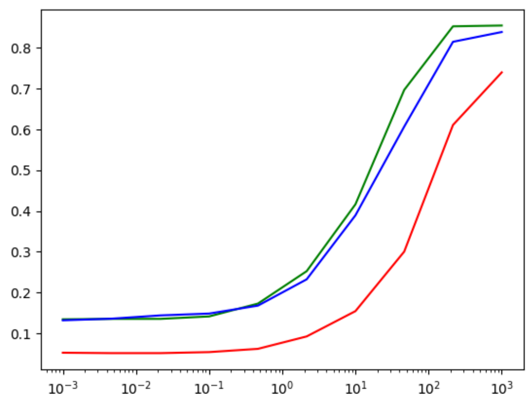
model	Π =0.5	Π = 0.9	Π = 0.1
Full Covariance	0.0725	0.1830	0.2024
Tied Covariance	0.0690	0.1770	0.1845

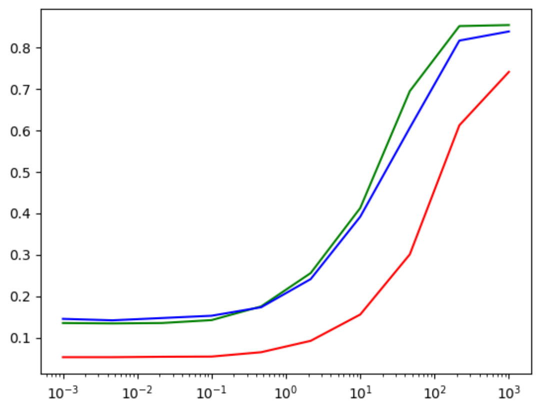
Some of the observations that we did on the decision phase are reflected by these results: FC and T have overall good performance; PCA leads to better performances on the main application and slightly worse ones on the other two (altough this is only true for FC); Gaussianization only leads to worse performances, which validates the observations that were made during the decisions phase.

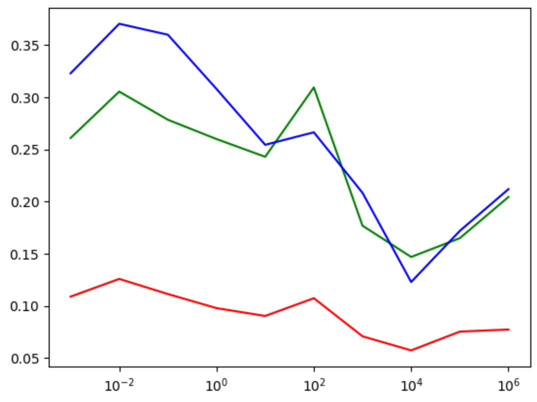
MinDCF values are slightly higher for all the models compared to the one found in training but this should be normal considering that we're now making computations on a different dataset: while before we were extracting train and evaluation sets from different fold partitions of the same training set, now the evaluation data is taken from a completely other dataset, which means that the difference in terms of class distribution between train and test might be higher with respect to the decision phase.

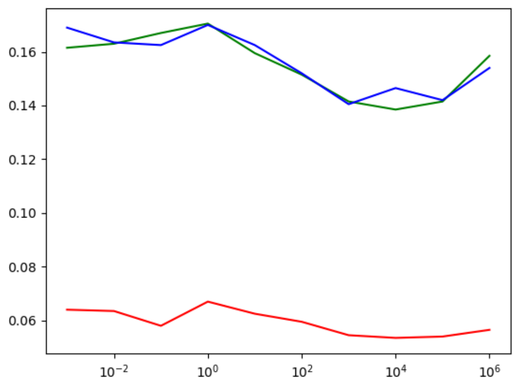
Logistic Regression

We now evaluate our decisions made on LR models. We first see wether the parameters we chose as optimal are actually the best ones also in the test set (top: linear LR, bottom: quadratic LR, left: raw features, right: pca, usual coloring for the three main applications):









Raw:

model	$\Pi = 0.5$	$\Pi = 0.9$	$\Pi = 0.1$

Linear lamda = 10e-3	0.0525	0.1345	0.1319
Quad l = 10e4	0.0755	0.1770	0.1845

PCA:

model	Π =0.5	Π = 0.9	Π = 0.1
Linear lamda = 10e-3	0.0530	0.1355	0.1455
Quad l = 10e4	0.0540	0.1415	0.1420

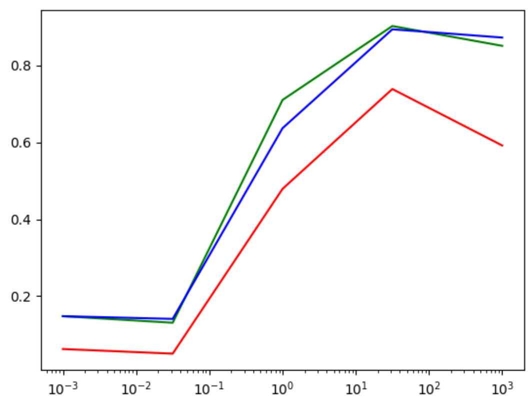
Gaussianization

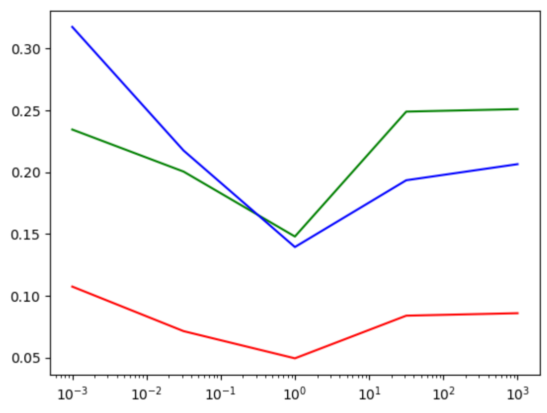
model	Π =0.5	Π = 0.9	Π = 0.1
Linear lamda = 10e-3	0.0655	0.1770	0.1709
Quad l = 10e-2	0.0615	0.1770	0.1845

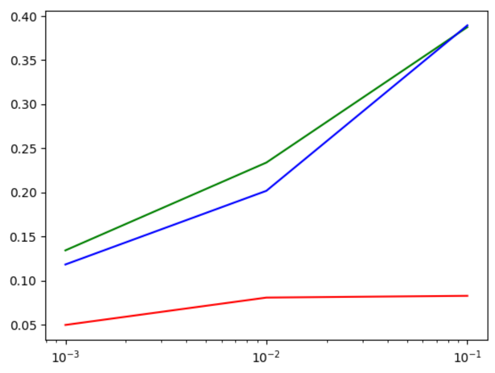
From the plot we can see that our estimated optimal parameters in the train set were optimal also in the test set both for linear LR and for qudratic LR.
Regarding raw features, the best performances come from the linear model, while the quadratic model gives relatively lower results just like we saw during the decision phase.
Once again our expectations on PCA are met only partially: there's no improvement for linear LR, but PCA still manages to fix the problem of high dimensionality of the expanded features for the quadratic model, leading to results that are close to the best ones provided by LR as a whole.
Finally the behavior of gaussianised features is in line with what we saw in the decision phase: it only improves the quadratic model for the main application but only if compared to the already low performance of the quadratic model trained on raw features.

SVM

We briefly take a look at the two SVM that had the best performances during the decision phase, namely linear SVM trained on raw features and RBF SVM with PCA 8 preprocessing. We check the quality of our parameter optimization on C (top) for both models (left: linear, right: RBF) as well as our choice of the gamma parameter for the RBF kernel:







As usual a table with some numerical results:

model	Π =0.5	Π = 0.9	Π = 0.1
Linear C = 1 raw features	0.4790	0.7105	0.6370
RBF C = 20 gamma = 0.01 PCA	0.0810	0.2340	0.2019

This time our decision were somehow far from the optimal ones.

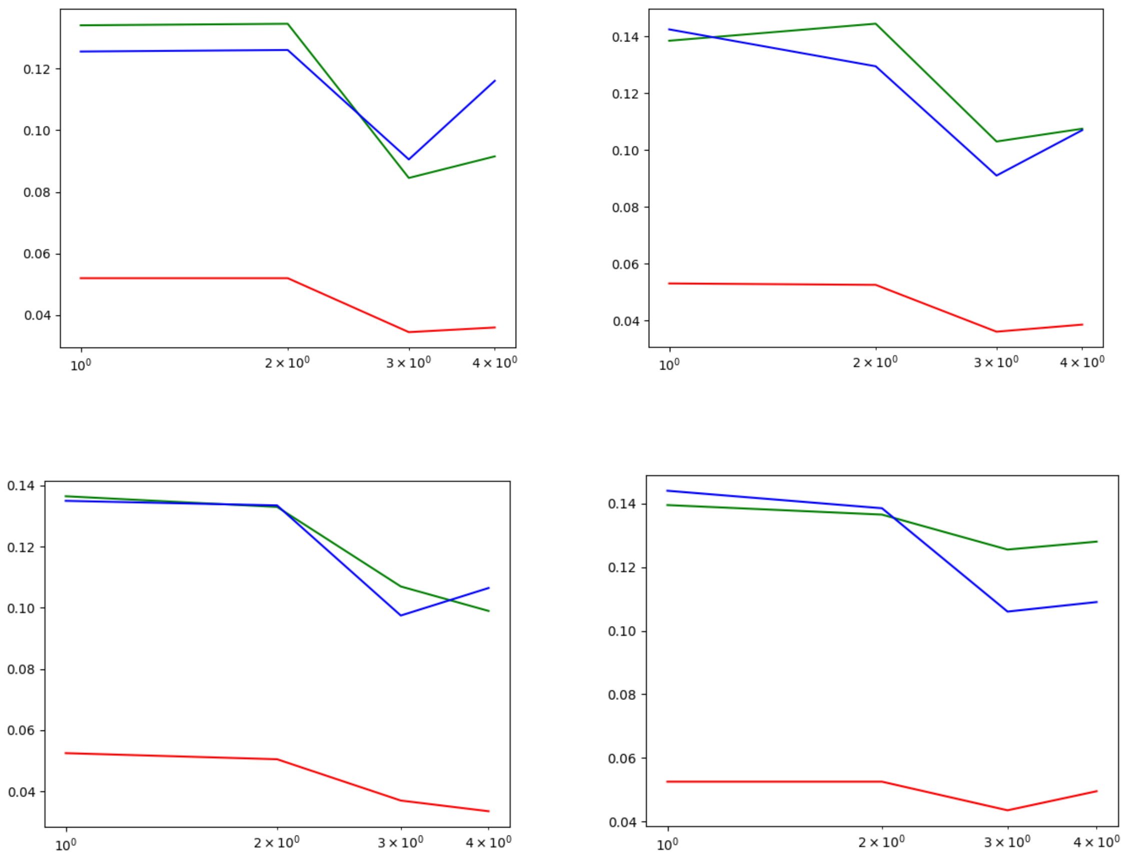
For linear SVM our optimization analysis revealed that C = 1 would lead to the best minDCF, however the assessments on the test set actually give relatively bad performances for such value. The results we were actually expecting from the model are instead shifted to lower values of parameter C (i.e. < 0.05). Maybe this erroneous estimation of the optimal parameter could have been avoided by giving less strict restrictions to the optimization function used for training during the decision phase.

For RBF kernel the situation is slightly different, as our decisions are not sub optimal by a large margin. We correctly acquired the combination of values that led to the lowest minDCF according to our evaluations; hower the reading we got from the plot was too optimistic: the best value for C was found to be 20, but values higher than 20 were leading to worse performance as opposed to lower values (which led to nearly the same DCF); likewise we did find gamma = 0.01 and gamma =

0.001 to have nearly the same performances, but we eventually took gamma as our optimal parameter despite the fact that gamma = 0.1 was leading to much worse results. In summary, picking a slightly lower value for both C and gamma would have been a wiser choice than the one we chose even with just the informations we acquired during the decision phase, and those values would have led to optimal performances here on the test set.

GMM

We finally take a look at the best Gaussian Mixture Models we found during decision phase. As usual we also check if our decision on the hyperparameter (the number of gmm components for each class in this case) was actually optimal also when predicting test data. Below it's possible to see the results for the GMM FC 8 (top) and GMM T 16 (bottom) models trained both on raw (left) and PCA (right) features (usual coloring for the different applications):



The numerical values:

Raw:

model	Π =0.5	Π = 0.9	Π = 0.1
GMM FC 8	0.0345	0.0845	0.0905
Quad l = 10e-2	0.0335	0.0990	0.1065

PCA:

model	Π =0.5	Π = 0.9	Π = 0.1
GMM FC 8	0.0360	0.1030	0.0910
Quad l = 10e-2	0.0494	0.1280	0.1090

In this case our observations on the train set were mostly coherent with the ones on the test set.

As we could tell from the decision phase, GMM model largely outperformed all other kind of models, with GMM T 16 being the best for the balanced application. The choice of the number of component salso reflects what we saw before, with the model using PCA (excluding FC) overfitting as the number of components grows. In any case, also in the test set PCA does not bring any benefit to the models compared to the use of raw features.

Fusion Model

Altough we didn't perform any parameter optimization on the Fusion model itself, we can still compute its minimum DCF over the whole test set. The training procedure is slightly different from the one adopted during the decision phase: first the usual cross validation procedure is performed on the whole training set; then we train the calibrator on the whole extracted validation set (previously we would train only on the 80% of it); finally, the two main models are trained on the whole train set and the whole Fusion model is ready to perform (calibrated) score predictions. Since the calibrator is the last part of the fusion model, the following minDCF values were computed on the output scores of the calibrator by predicting the test set:

model	Π =0.5	Π = 0.9	Π = 0.1
Fusion	0.0335	0.0890	0.0800

As we could tell from the decision phase, the Fusion model can be considered as an improvement over both the best GMM models since it provides the best minDCF for both the balanced application and the one with effective prior qual to 0.1.

Actual DCF Assessments

The last thing we need to do is assessing whether our decisions on model calibration and on the use of the Fusion model were optimal also in terms of actual DCF. As we specified before, this is the actual metric that shows how well our model would perform in a real case scenario where no information on the best separating threshold on the evaluation set can be obtained. We're going to see both the results provided by threshold estimation, calibration of GMM T 16 and finally the performance of our fusion model

Threshold Estimation

we first evaluate the actual DCF provided through threshold estimation for the two best GMM models on our three main application. Of course the treshold is estimated on the whole validation set extracted from the training set through cross validation, and it is later applied on the test set to acquire the optimised actual DCF metric:

model	Π =0.5	Π = 0.9	Π = 0.1
FC 8 min DCF	0.0345	0.0845	0.0905
FC 8 theory DCF	0.0375	0.0890	0.0990
FC 8 opt DCF	0.0375	0.0875	0.1069
T 16 minDCF	0.0335	0.0990	0.1065
T 16 theory DCF	0.0350	0.1635	0.1580
T 16 optDCF	0.0335	0.1035	0.1119

Once again our results are coherent with the ones we got during the decision phase.

Performing threshold estimation on GMM FC 8 is not very meaningful as the model already provides well calibrated scores, while the estimation for the GMM T 16 brings great improvement (up to 40% less actual DCF) to the imbalanced applications. In this particulare case, the optimal threshold allows the GMM T 16 model to reach the minimum DCF for the balanced application.

Score Calibration

We're now going to check the effect of applying score calibration to the GMM 16 T model.The training procedure is the same as the one we followed during decision phase, the only difference is that we're training in roughly the same way that was explained for the Fusion model: first validation set is extracted from the whole train set through cross validation; then the calibrator is trained on the validation set; finally the model itself is trained on the whole train set. Since we are still using a prior weighted LR model as calibrator, we select once again $\Pi t = 0.5$ as its main parameter

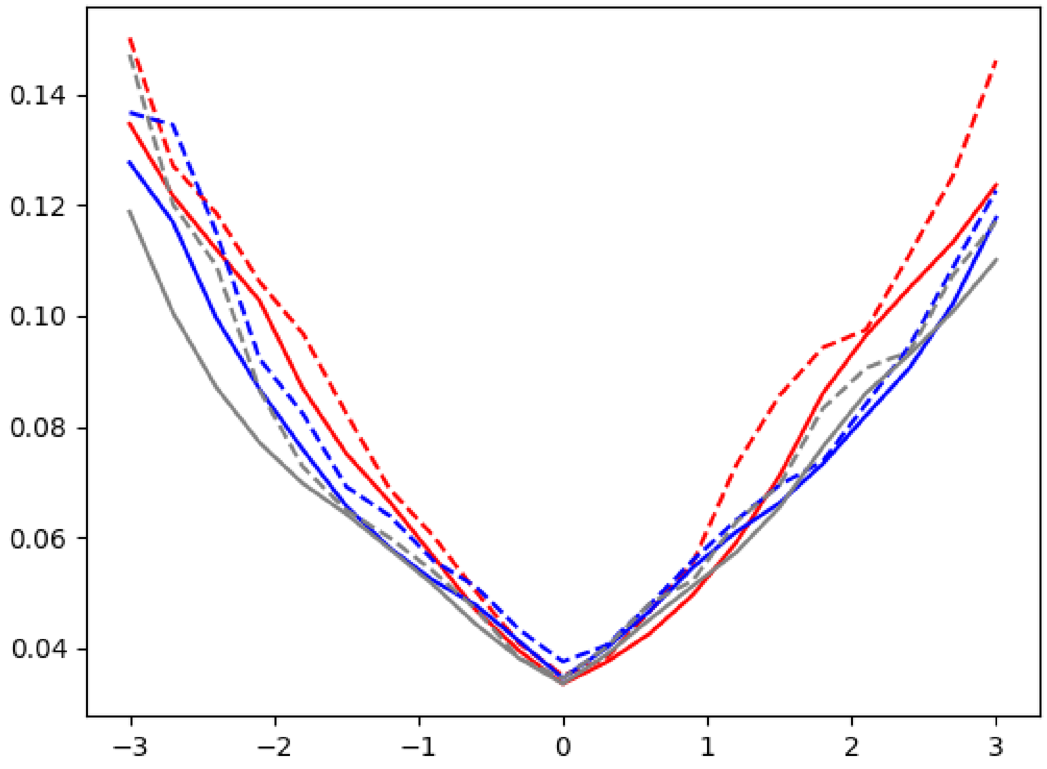
Here are reported the result for the three main applications:

model	Π =0.5	Π = 0.9	Π = 0.1
T 16 minDCF	0.0335	0.0990	0.1065
T 16 theory DCF	0.0350	0.1635	0.1580
T 16 calibrated theory DCF	0.0335	0.1040	0.1110

Also in this case we're left with results that are very close to the threshold estimation ones, with the difference that this calibrated model can be used effectively for any application without having to estimate any new parameter

Model Comparison

The last thing we need to do for this project is checking if the Fusion model does achieve better performances compared to the calibrated GMM T 16 and the GMM FC 8. We can quickly do this by plotting a Bayes error plot for the three models, since it uses the theoretical actual DCF:



Fif. 7: Bayes plot comparing the calibrated GMM T 16 (red), the GMM FC 8 (red) and the Fusion model (gray). Dashed lines are theoretical actual DCFs while continuous lines represent minDCF for each model

Also on this last occasion our result are coherent with expected ones. Altough the Fusion model is outperformed in a small range of applications by the GMM FC 8 (the fusion with GMM T 16 actually proved to be harmful in this range), it still provides the best actual DCF on most of the unbalanced application while also giving the best performance on the balanced application together with the calibrated GMM T 16

Conclusions

After this final phase of the project, we can now make some closing observations:

1. The Fusion model was arguably the best model we could find among all the techniques that were explored for the given classification task.
2. The GMM models were able to outperform other kinds of models on this task due to both the regular distribution of the data together with the presence of different, well separated clusters inside each class distribution representing the different age groups of the speakers.
3. We were able to reach an actual DCF of 0.0335 on the test on the balanced application together with relatively low DCF (<0.09) for each unbalanced application
4. We could also confirm during the evaluation phase that our decisions made using only the train set mostly proved to be effective also on the test set. This suggests also that the population of patterns belonging to the test set is sufficiently similar to the population of the train set

Thank you very much for your attention,

Francesco Blangiardi (s288265)