MLPR Exam Project: Gender Detection

Mattia Rosso [s294711]

August 3, 2022

This project is inteded to show a binary classification task on a datased made of 12 continuous observations coming from speking embeddings. A speaker embedding represents a smal-dimensional, fixed size representation of an utterance. Features can be seen as points in the m-dimensional embedding space (and the embeddings have already been computed). This is a task where classes are balanced both in training and evaluation set

1 Dataset analysis

1.1 Training and evaluation sets

The training set contains:

- Training Set: 3000 samples belonging to Male class (Label = 0) and 3000 samples belonging to Female class (Label = 1).
- Evaluation Set: 2000 samples belonging to Male class (Label = 0) and 2000 samples belonging to Female class (Label = 1).

1.2 Training set features analysis

1.3 Features Statistics

All the features are contiguous and their main statics can be showed through a boxplot in figure 1.

1.4 Z-normalization

A useful operation that can be applied in order to avoid to deal with numerical issues and to make data more uniform is to apply Z-normalization as a preprocessing step:

$$z_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j} \forall x_i \in D$$

Where $z_{i,j}$ is the Z-normalized value correspongin to the feature j of sample i while μ_j and σ_j are, respectively, the mean and the variance computed over all values for feature j.

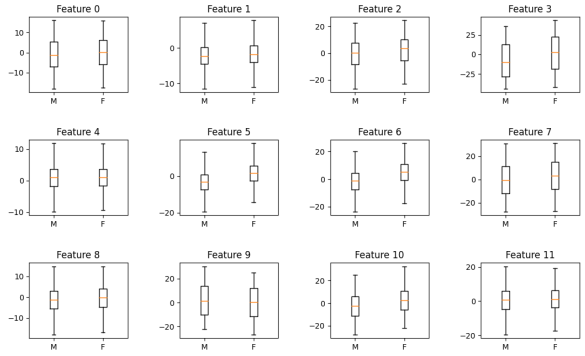

Figure 1: *Features boxplot*

1.5 Features distribution

By plotting historams for each feature, separated for male and female, it is possible to show how much each feature follows a gaussian distribution in order to understend whether a pre processing like gaussianization can be useful to go on with our classification task

We can notice from figure 2 that almost all the features are already well-distributed except for features 3, 7, 9. We can apply an additional pre-processing step in order to make all the features following a gaussian distribution.

1.5.1 Gaussianization

Gaussianization is a pre processing step that maps each fature to values whose empirical cumulative

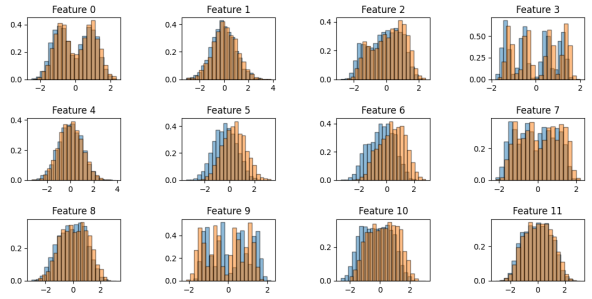

Figure 2: *Z-normalized features distribution*

distribution is well approximated by a Gaussian cumulative distribution function. For each feature x that we want to gaussianize we firstly compute the rank over the dataset:

$$r(x) = \frac{\sum_{i=1}^{N} \mathbb{I}[x_i < x] + 1}{N+2}$$

where $\mathbb{I}$ is the indicator function (1 when the condition inside [] is true, 0 otherwise). Actually, we are counting how many samples in the dataset D have a greater value with respect to the feature we are computing the rank on.
The next step is to compute the transformed feature as $y = \Phi^{-1}(r(x))$ where Φ is the inverse of the cumulative distribution function.

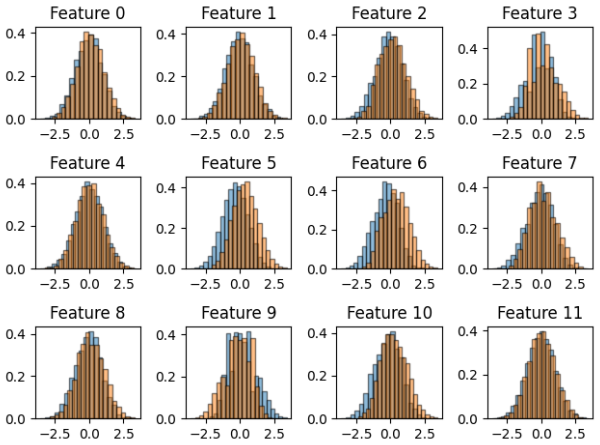

Figure 3: *Gaussianized features distribution*

1.6 Features correlation

We can show how much features are correlated by using a heatmap plot showing a darker color inside cells $[i, j]$ for which it exists an high correlation among feaure i and feature j. We are going to use the Pearson correlation coefficient to compute the correlation among feature X and feature Y:

$$\left|\frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}}\right|$$

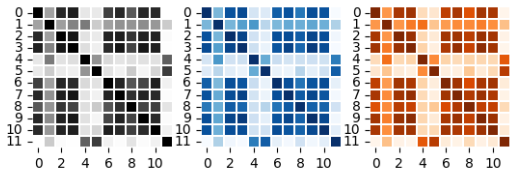

Figure 4: *Z-normalized+Gaussianized features correlation: grey the whole dataset, orange F class, blue M class*

Correlation is quite high among most of the features, we can understand that applying PCA would be meaningful for values of m not below 10 or 9 at most. For smaller values of m we would lose important information coming from high-correlated features.

2 Dimensionality reduction

Before proceeding with the classification task i would spend some words on the possible dimensioanlity reduction rechniques that can be applied: PCA, LDA.

2.1 PCA

As already anticipated, given the heatmap in figure 4 we can observe that PCA with reasonable values of m can be applied. PCA is a dimensionality reduction technique that, given a centered dataset $X = x_1, ..., x_k$, it aims to find the subspace of $\mathbb{R}^n$ that allows to preserve most of the information (the directions with the highest variance).
Starting from the sample covariance matrix

$$C = \frac{1}{K}\sum_i (x_i - \bar{x})(x_i - \bar{x})^T$$

we compute the eigen-decomposition of $C = U\Sigma U^T$ and project the data in the subspace spanned by the m columns of U corresponding to the m highest eigenvalues:

$$y_i = P^T(x_i - \bar{x})$$

In order to select the optimal m we can use a cross-validation approach by inspecting how much of the total variance of data we are able to retain by using that value of m. We exploit the fact that each eigenvalue correspond to the variance along the corresponding axis and the eigenvalues are the elements of the diagonal of the matrix Σ. We selecte m as:

$$\min_m \; s.t \frac{\sum_i^m \sigma_i}{\sum_i^n \sigma_i} >= t, \; t \geq 95\%$$

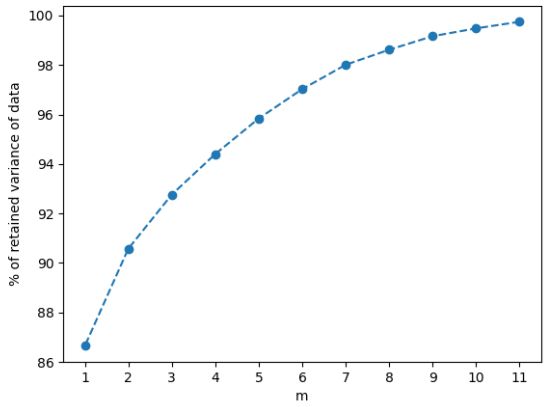

Figure 5: *3-fold cross validation for PCA impact evaluation*

We can clearly understand from figure 5 that values of $m < 9$ would rapidly decrease the amout of retained variance. We will see better later on how this would badly impact the performance of the classifiers

2.2 LDA

PCA technique is unsupervised so we have no guarantee of obtainig discriminant directions. Despite the fact that LDA allows to find at most $C-1$, where C is the number of classes, directions, so it makes no sense to apply it as a dimensionality reduction technique, it can be used as a linear classifier and we can understand it by its definition; LDA mazimizes the between-class variability over the within-class variability ratio for the transformed samples:

$$\mathcal{L}(w) = \frac{s_B}{s_W} = \max_w \frac{w^T S_B w}{w^T S_W w}$$

It can be proved that optimal solution correspond to the eigenvector of $S_W^- 1 S_B$ corresponding to the largest eigenvalue. Once that we have estimated w we can project the test samples over w and assign the class label looking at the score obtained:

$$f(x) = \begin{cases} 1, & \text{if } x < 0. \\ 0, & \text{otherwise.} \end{cases}$$

It will be showed later the result of the classifier

3 Classification models analysis

3.1 Premises

In the next paragraphs we are going to compare different classification models. We will employ a k-fold cross validation technique (with $k = 3$) for model evaluation and the best models will be chosen to train the entire training set for performing a final comparison. We will consider three types of applications:

$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.1, 1, 1)$$
$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.5, 1, 1)$$
$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.9, 1, 1)$$

and the target application will be

$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.5, 1, 1)$$

We are interested in selecting the most promising approach and we will infact perform measures in term of minimum detection cost:

$$DCF = \frac{DCF_u(\pi_T, C_{fn}, C_{fp})}{min(\pi_T, C_{fn}, (1-\pi_T C_{fp}))} = \frac{\pi_T C_{fn} P_{fn} + (1-\pi_T) C_{fp} P_{fp}}{min(\pi_T, C_{fn}, (1-\pi_T C_{fp}))}$$

and for $minDCF$ computation we will look for:

$$t' = -log(\frac{\tilde{\pi}}{1-\tilde{\pi}})$$

that allows us to obtain the lowest possible DCF (as if knew in advance this optimal threshold)

3.2 Gaussian models

The first class of models we are going to analyze are the generative gaussian models. Model assumption are that, given the dataset X we assume that the sample x_t is a realization of the R.V. X_t. A simple model consists in assuming that our data, given the class, can be described by a Gaussian distribution:

$$(X_t | C_t = c) \sim (X | C = c) \sim \mathcal{N}(x_t | \mu_c, \Sigma_c)$$

We will assign a probabilistic score to each sample in term of the class-posterior log-likelihood ratio:

$$log\ r(x_t) = log\ \frac{P(C=h_1|x_t)}{P(C=h_0|x_t)}$$

We can expand this expression by writing:

$$log\ r(x_t) = log\ \frac{f_{X|C}(x_t|h_1)}{f_{X|C}(x_t|h_0)} + log\ \frac{\pi}{1-\pi}$$

While the training phase consists in estimating the model parameters the scoring phase consists in computing the log-likelihood ratio (first term of the equation) for each sample. It will be then compared with a threshold specific for each application for computing the $minDCF$. What differentiate the different Gaussian models is the way how we estimate the model parameters.

3.2.1 MVG Gaussian Classifier

The ML solution to the previous desribed problem is given by the empirical mean and covariance matrix for each class:

$$\mu_c^* = \frac{1}{N_c} \sum_{i=1}^{N} x_{c,i}$$
$$\Sigma_c^* = \frac{1}{N_c} \sum_{i=1}^{N} (x_{c,i} - \mu_c^*)(x_{c,i} - \mu_c^*)$$

We will then compute the log densities for each sample by using the estimated model parameters

3.2.2 Naive Bayes Classifier

The Naive Bayes assumption simplifies the MVG full covariance model stating that if we knew that for each class the componenets are approximately independent we can assume that the distribution $X|C$ can be factorized over its components. The ML solution to this problem is:

$$\mu^*_{c,[j]} = \frac{1}{N_c}\sum_{i|c_i=c} x_{i,[j]}$$
$$\sigma^2_{c,[j]} = \frac{1}{N_c}\sum_{i|c_i=c}(x_{i,[j]} - \mu_{c,[j]})^2$$

The density of a sample x can be expressed as $\mathcal{N}(x|\mu_c, \Sigma_c)$ where μ_c is an array where each element $\mu_{c,[j]}$ is the the mean for each class for each component while Σ_c is a diagonal covariance matrix. The Naive Bayes classifier corresponds to the MVG full covariance classifier with a diagonal covariance matrix

3.2.3 Tied Gaussian Classifier

This model assumes that the covariance matrices of the different class are tied (we consider only one covariance matrix common to all classes). We are assuming that:

$$f_{X|C}(x|c) = \mathcal{N}(x|\mu_c, \Sigma)$$

so each class has its own mean but the covariance matrix is the same for all the classes. The ML solution to this problem is:

$$\mu^*_c = \frac{1}{N_c}\sum_{i=1}^{N} x_{c,i}$$
$$\Sigma^* = \frac{1}{N}\sum_c \sum_{i|c_i=c}(x_i - \mu^*_c)(x_i - \mu^*_c)$$

This model is strongly related to LDA (used as a linear classfication model). By considering the binary log-likelihood ratio of the tied model we obtain a linear decision function:

$$llr(x) = log\ \frac{f_{X|C}(x|h_1)}{f_{X|C}(x|h_0)} = x^T b + c$$

where b and c are functions of class means and (tied) covariance matrix. On the other hand, projecting over the LDA subspace is, up to a scaling factor k, given by:

$$w^T x = k \cdot x^T \Lambda(\mu_1 - \mu_0)$$

where $\Lambda(\mu_1 - \mu_0) = b$. The LDA assumption that all the classes have the same within class covariance is related to the assumption done for the tied model.

3.2.4 Gaussian Models Comparison

Table 1: *Gaussian Models*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Z-normalized features - no PCA			
Full Cov	0.128	0.048	0.125
Tied Cov	0.122	0.046	0.127
Naive Bayes	0.822	0.567	0.856
Z-normalized features - PCA(m=11)			
Full Cov	0.265	0.100	0.231
Tied Cov	0.257	0.098	0.227
Naive Bayes	0.278	0.108	0.245
Z-normalized features - PCA(m=10)			
Full Cov	0.303	0.115	0.267
Tied Cov	0.293	0.112	0.264
Naive Bayes	0.306	0.121	0.283
Gaussianized features - no PCA			
Full Cov	0.218	0.078	0.191
Tied Cov	0.208	0.078	0.189
Naive Bayes	0.813	0.586	0.847
Gaussianized features - PCA(m=11)			
Full Cov	0.227	0.087	0.218
Tied Cov	0.215	0.084	0.208
Naive Bayes	0.278	0.106	0.257
Gaussianized features - PCA(m=10)			
Full Cov	0.223	0.084	0.211
Tied Cov	0.212	0.082	0.207
Naive Bayes	0.279	0.103	0.254

A graphical version of the table can be helpful in analyzing results:

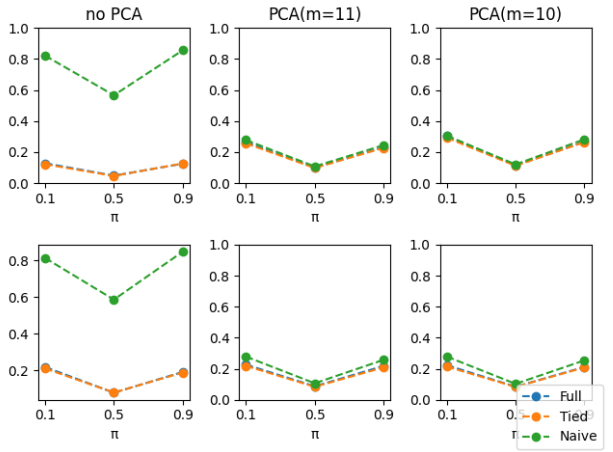

Figure 6: *Top: Z-Normalized feautres, Bottom: Gaussianized features*

Some observations:

- We can notice that Full-Cov model and Tied-Cov model achieve very good and similar results with slighlty better performances for the tied model.
- Gaussianization pre processing does't really help in achieving better results maybe because data are already well distributed according to the Gaussian assumptions

- Naive Bayes assumption doesn't hold really well, in particular if PCA is not applied. When PCA is applied it has a really good impact only on Naive Bayes model and especialliy if also combined with Gaussianization pre processing.
- Regarding Full and Tied models with PCA(m=11) there is no a high performance degradation since the minDCF obtained are still good. For lower values of m the models become less able in taking decisions and the minDCF increases
- For the MVG classifiers best performances are achieved by the Tied-Cov classifier with only Z-normalization pre processing and withouth PCA. Really good performacnes are also achieved by the Tied-Cov model trained with Gaussianized fetures and no-PCA (the PCA(m=11) version of this model achieves worse but comparable result w.r.t. to the no-PCA version and it can be selected to try to better avoid overfitting by reducing the features space and also for reducing computational effort)

Best Gaussian Models:

- Tied Cov (Z-Normalization, no PCA)
- Tied Cov (Gaussianization, PCA(m=11))

3.3 Logistic Regression Classifier

Logistic Regression is a discriminative classification model. Starting from the results obtained from the Gaussian classifiers we consider the linear decision function obtained from the expression of the posterior log-likelihood ratio:

$$l(x) = log\ \frac{P(C=h_1|x)}{P(C=h_0|x)} = log\ \frac{f_{X|C}(x|h_1)}{f_{X|C}(x|h_0)} + log\ \frac{\pi}{1-\pi} = w^T x + b$$

where b takes into account all the prior information. Given w and b we can compute the expression for the posterior class probability:

$$P(c = h_1|x, w, b) = \frac{e^{(w^T x+b)}}{1+e^{(w^T x+b)}} = \sigma(w^T x + b)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. Decision rules will be hyperplanes orthogonal to w.

3.3.1 Linear Logistic Regression

We are going to look for the minimizer of the function:

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

where λ is an hyperparameter that represents the regularization term (needed to make the problem solvable in case of linearly separable classes). The choice of λ appear to be critical for all the applications, in particular for the unbalanced ones. By observing figure 7 it is clear that for values of λ greater than 10^-3 the $minDCF$ rapidly increases for all the considered applications. (WHY). Best performances are obtained for small values of λ. PCA with $m = 10$ helps in obtaining

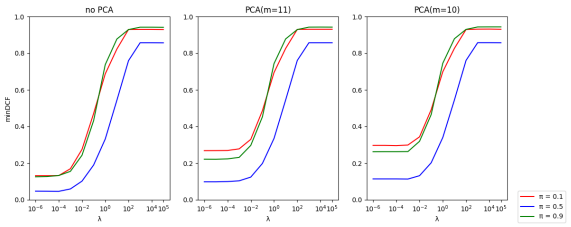

Figure 7: *minDCF for different values of λ and different priors*

Table 2: *Linear Logistic Regression - 3-fold cross validation*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Z-normalized features - no PCA			
LLR ($\lambda = 10^{-3}$)	0.170	0.059	0.155
LLR ($\lambda = 10^{-5}$)	0.132	0.047	0.127
LLR ($\lambda = 10^{-6}$)	0.132	0.047	0.126
Z-normalized features - PCA(m=11)			
LLR ($\lambda = 10^{-3}$)	0.278	0.104	0.232
LLR ($\lambda = 10^{-5}$)	0.269	0.098	0.221
LLR ($\lambda = 10^{-6}$)	0.268	0.098	0.222
Z-normalized features - PCA(m=10)			
LLR ($\lambda = 10^{-3}$)	0.299	0.113	0.263
LLR ($\lambda = 10^{-5}$)	0.297	0.114	0.263
LLR ($\lambda = 10^{-6}$)	0.297	0.114	0.262

Table 3: *Best models analyzed up to now*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Gaussian Models			
Tied Cov (Z-Norm, no PCA)	0.122	0.046	0.127
Tied Cov (Gau, PCA(m=10))	0.215	0.082	0.207

better results for the target application with $\lambda = 10^{-3}$, however, better results are achieved for $\lambda = 10^-5$ and without PCA.
Comparison: the LLR model trained with $\lambda = 10^{-5}$ and no-PCA achieves really similar results with the ones achieved by the Tied-Cov Gaussian model (Z-Norm, no PCA) for $\tilde{\pi} = 0.5$ and $\tilde{\pi} = 0.9$ applications but a worse performance is obtained for $\tilde{\pi} = 0.1$. The LLR model behaves better than the Tied-Cov Gaussian model (Gau, PCA(m=10)) in all the three applications.
Selected LLR Model:

- Z-normalized features, $\lambda = 10^{-6}$, no PCA

3.3.2 Quadratic Logistic Regression

Now we are going to train a Quadratic LR model by performing features expansion. For binary linear LR the separation surfaces are linear decision function as already discussed (and we obtain the same form as for the Tied Gaussian classifier). By looking instead at the separation surface obtained through the MVG gaussian classifier we have:

$$log\ \frac{P(C=h_1|x)}{P(C=h_0|x)} = x^T Ax + b^T x + c = s(x, A, b, c)$$

This expression is quadratic in x but linear in A and b. We could rewrite it to obtain a decision function that is linear for the expanded features space but quadratic in the original features space. Features expansion is defined as:

$$\Phi(x) = \begin{bmatrix} vec(xx^T) \\ x \end{bmatrix}, w = \begin{bmatrix} vec(A) \\ b \end{bmatrix}$$

where $vec(X)$ is the operator that stacks the columns of X. In this way the posterior log-likelihood is expressed as:

$$s(x, w, c) = s^T \phi(x) + c$$

We are now going to train the Linear Logistic Regression model using features vectors $\phi(x)$.

By rapidly looking at the results obtianed we can

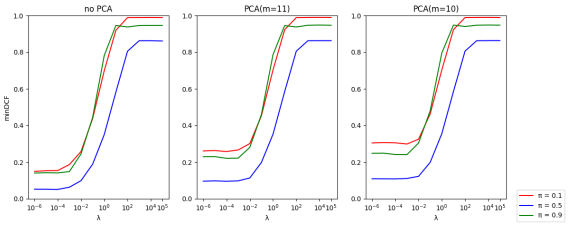

Figure 8: *minDCF for different values of λ and different priors*

Table 4: *Quadratic Logistic Regression - 3-fold cross validation*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Z-normalized features - no PCA			
QLR ($\lambda = 10^{-3}$)	0.187	0.063	0.149
QLR ($\lambda = 10^{-5}$)	0.153	0.052	0.142
QLR ($\lambda = 10^{-6}$)	0.150	0.053	0.141
Z-normalized features - PCA(m=11)			
QLR ($\lambda = 10^{-3}$)	0.267	0.098	0.222
QLR ($\lambda = 10^{-5}$)	0.263	0.098	0.230
QLR ($\lambda = 10^{-6}$)	0.305	0.096	0.230
Z-normalized features - PCA(m=10)			
QLR ($\lambda = 10^{-3}$)	0.299	0.111	0.241
QLR ($\lambda = 10^{-5}$)	0.307	0.109	0.249
QLR ($\lambda = 10^{-6}$)	0.305	0.109	0.248

Table 5: *Best models analyzed up to now*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Gaussian Models			
Tied Cov (Z-Norm, no PCA)	0.122	0.046	0.127
Tied Cov (Gau, PCA(m=10))	0.212	0.082	0.207
Logistic Regression Models			
LLR (Z-Norm, $\lambda = 10^{-6}$, no PCA)	0.132	0.047	0.126

say that quadratic version of the logistic regression performs worse w.r.t the linear version (the one without features expansion). The choice of λ is still critical and $\lambda <= 10^{-5}$ still remains the best choice. Regarding the target application we are able to reach similar results comparing to the linear version while the unbalanced applications are more penalized. When PCA is applied no improvements are obtained. However we can notice how the best choice for λ for unbalanced applications changes when PCA is applied (with $\lambda = 10^{-3}$ we achieve better results for $\tilde{\pi} = 0.1$ and $\tilde{\pi} = 0.9$ applications w.r.t. $\lambda = 10^{-5}$ while for target application $\lambda = 10^{-5}$ is the best choice)

Comparison: With respect to Gaussian models comparable performances are achieved for $\lambda = 10^{-6}$ and no PCA even if the model performs slightly worse. The quadratic model performs also worse than the linear model and we won't consider it in score calibration.

Selected QLR Model:

- Z-Normalized features, $\lambda = 10^{-5}$, no PCA

3.4 SVM Classifier

3.4.1 Linear SVM

Support Vector Machines are linear classifiers that look for maximum margin separation hyperplanes. The primal formulation consists in minimizing the function:

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

with N is the number of trainig samples and C is an hyperparameter.

We are also goint to take into account the dual formulation to solve the problem that consists in maximizing the function:

$$J^D(\alpha) = -\frac{1}{2}\alpha^T H \alpha + \alpha^T \mathbf{1} \; s.t. \; 0 \leq \alpha_i \leq C, \forall i \in \{1, ..., n\}, \; \sum_{i=1}^{n} \alpha_i z_i = 0$$

where $\mathbf{1}$ is a n-dimensional vector of ones and H is the matrix whose elements are $H_{ij} = z_i z_j x_i^T x_j$.

The relation between the primal and the dual formulation(between the maximizer values w^* and α^*) is:

$$w^* = \sum_{i=1}^{n} \alpha_i^* z_i x_i$$

and the optimal bias b can be computed considering a sample x_i that lies on the margin: $z_i(w^{*T} x_i + b^*) = 1$

To be able to computationally solve the problem we need to modify the primal formulation as:

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

where $\hat{x}_i = \begin{bmatrix} x_i \\ 1 \end{bmatrix}$ and $\hat{w} = \begin{bmatrix} w \\ b \end{bmatrix}$.

The scoring rule $\hat{w}^T \hat{x}_i = w^T x + b$ has the same form of the original formulation but we are also regularizing the norm of $\hat{w}$: $||\hat{w}||^2 = ||w||^2 + b^2$ and we use a mapping $\hat{x}_i = \begin{bmatrix} x_i \\ K \end{bmatrix}$ to mitigate the fact that by regularizing the bias term we could obtain sub-optimal results. The bigger is K the lower the regularization effect of b becomes weaker and the dual formulation

becomes harder to solve. According to the modification done to the primal formulation we also modify the dual formulation as:

$$J^D(\alpha) = -\frac{1}{2}\alpha^T \hat{H}\alpha + \alpha^T \mathbf{1} \; s.t. \; 0 \leq \alpha_i \leq C, \forall i \in \{1, ..., n\}$$

where the equality constraint dissapeared (the one that L-BFGS was not able to incorporate) and the matrix $\hat{H}$ can be computed as $\hat{H}_{i,j} = z_i z_j \hat{x}_i^T \hat{x}_j$ The

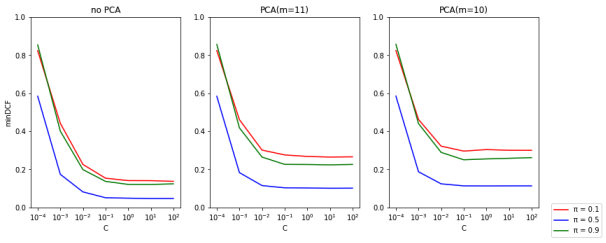

Figure 9: *Linear SVM (K=0, Z-Normalized features) - minDCF for different values of C and different priors*

Table 6: *Linear SVM - 3-fold cross validation*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Z-normalized features - no PCA			
Linear SVM ($C = 0.1$)	0.158	0.052	0.141
Linear SVM ($C = 1$)	0.129	0.047	0.130
Z-normalized features - PCA(m=11)			
Linear SVM ($C = 0.1$)	0.275	0.102	0.233
Linear SVM ($C = 1$)	0.269	0.100	0.226
Z-normalized features - PCA(m=10)			
Linear SVM ($C = 0.1$)	0.295	0.114	0.259
Linear SVM ($C = 1$)	0.301	0.113	0.267

Table 7: *Best models analyzed up to now*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Gaussian Models			
Tied Cov (Z-Norm, no PCA)	0.122	0.046	0.127
Tied Cov (Gau, PCA(m=10))	0.212	0.082	0.207
Logistic Regression Models			
LLR (Z-Norm, $\lambda = 10^{-6}$, no PCA)	0.132	0.047	0.126
QLR (Z-Norm, $\lambda = 10^{-5}$, no PCA)	0.153	0.052	0.142

only preprocessing step that has been considered is Z-normalization. As we can notice from figure 9 better results are achieved for smaller values of the hyperparameter C so the choice of it is crucial.

- Best performances on the target application are reached for $C = 1$ combined with Z-normalized features and no PCA is applied
- PCA is not helpful in reducing the minDCF in any of the considered applications even if with PCA(m=11) there is no a high performance degradation for the target application

Comparison: The results achieved by the model trained with Z-normalization, $C = 1$ and no PCA are closed to the the results obtained by the Linear Logistic Regression Model but slighlty worse with respect to the results obtained by the Tied Gaussian model. Selected Linear SVM Model: C=1 - Z-normalized features - no PCA

3.4.2 Kernel SVM

SVMs allow for non-linear classification through an implicit expansion of the features in a higher dimensional space. In contrast with Quadratic Logistic Regression classifier we don't have to compute an explicit expansion of the features space, it is sufficient to be able to compute the scalar product between the expanded features $k(x_1, x_2) = \phi(x_1)^T \phi(x_2)$ where k is the kernel function. We have to replace $\hat{H}$ with $\hat{H} = z_i z_j k(z_1, x_2)$. We are going to implement two types of kernel: polynomial and rbf.

- Polynomial kernel of degree d:

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

- Radial Basis Function kernel:

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

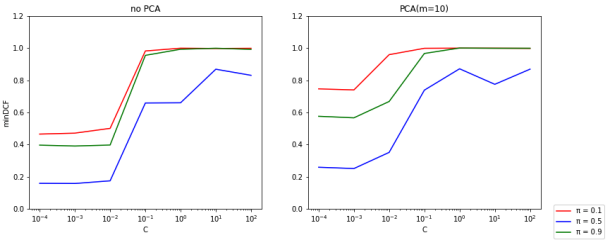

Figure 10: *Polynomial SVM (K=1, c=0, d=2, raw features) - minDCF for different values of C*

The Polynomial Kernel SVM model was trained with raw features since the Z-Normalization step led to very poor results for almost all the values of the hyperparameter C. By looking at figure 10 we can realize that the choice of C is again crucial and best performances are obtained for $C \leq 10^{-2}$. For this reason i have chosen $C = 10^{-4}$ to show more detailed results.

Table 8: *Polynomial Kernel SVM - $C = 10^{-4}$ - 3-fold cross validation*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Raw features - no PCA			
Poly SVM (c=0, d=2)	0.465	0.158	0.396
Poly SVM (c=1, d=2)	0.187	0.060	0.164
Poly SVM (c=1, d=3)	0.408	0.157	0.563
Raw features - PCA(m=11)			
Poly SVM (c=0, d=2)	0.553	0.199	0.487
Poly SVM (c=1, d=2)	0.210	0.062	0.158
Poly SVM (c=1, d=3)	0.452	0.183	0.537
Raw features - PCA(m=10)			
Poly SVM (c=0, d=2)	0.746	0.258	0.576
Poly SVM (c=1, d=2)	0.465	0.158	0.396
Poly SVM (c=1, d=3)	0.784	0.280	0.815

- The Polynomial SVM is able to achieve better results with respect to some versions of the Linear SVM. Best performances for the target application are achieved by the model trained with $c = 1$ and $d = 2$.
- The use of PCA(m=11) helps in reducing the minDCF for the $\tilde{\pi} = 0.9$ application for the model trained with $c = 1$ and $d = 2$ and doesn't affect that much the target application
- PCA(m=10) brings worse performances in for all the considered applications

The other kernel that was employed is the RBF kernel and these are the results achieved: In this case it

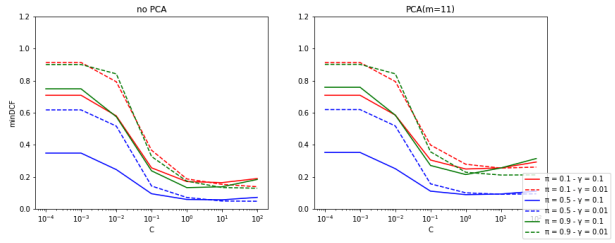

Figure 11: *RBF SVM ($K = 1$, $\gamma \in \{0.1, 0.01\}$) - minDCF for different values of C*

was again used Z-normalization as pre-processing step. The hyperparameter γ has to be tuned so i trained the model for different values of γ to analyze the performances and here are showed the results for $\gamma = 0.1$ and $\gamma = 0.01$. Also the hyperparameter C is still to be chosed and needs to be estimated via cross-validation. The parameter K has been set to 1. The value of the hyperparameter C is critical and we should choose a value $C \geq 10^{-1}$. The table below shows the results obtained with $C = 10$ and also results obtained with $\gamma = 1$ appear for completeness.

Table 9: *RBF Kernel SVM - C=10*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Z-Normalized features - no PCA			
RBF SVM ($\gamma = 1$)	0.291	0.095	0.281
RBF SVM ($\gamma = 0.1$)	0.163	0.056	0.137
RBF SVM ($\gamma = 0.01$)	0.153	0.049	0.133
Z-Normalized features - PCA(m=11)			
RBF SVM ($\gamma = 1$)	0.362	0.123	0.350
RBF SVM ($\gamma = 0.1$)	0.255	0.092	0.255
RBF SVM ($\gamma = 0.01$)	0.254	0.091	0.211
Z-Normalized features - PCA(m=10)			
RBF SVM ($\gamma = 1$)	0.386	0.136	0.379
RBF SVM ($\gamma = 0.1$)	0.271	0.106	0.273
RBF SVM ($\gamma = 0.01$)	0.291	0.107	0.239

Table 10: *Best models analyzed up to now*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Gaussian Models			
Tied Cov (Z-Norm, no PCA)	0.122	0.046	0.127
Tied Cov (Gau, PCA(m=10))	0.212	0.082	0.207
Logistic Regression Models			
LLR (Z-Norm, $\lambda = 10^{-6}$, no PCA)	0.132	0.047	0.126
QLR (Z-Norm, $\lambda = 10^{-5}$, no PCA)	0.153	0.052	0.142
SVM Models			
LSVM (Z-Norm, $K = 0$, $C = 1$, no PCA)	0.129	0.047	0.130

- RBF kernel version of the SVM achieves results that are near to the ones obtained with the linear version and behaves better w.r.t. the Polynomial version
- The most promising value of γ seems to be $\gamma = 0.01$ because with this value we are able to obtain the best minDCF for the target application (when no PCA is applied)
- PCA is not able to provide any better results in terms of minDCF

Comparison: The linear version of the SVM achieves better results with respect to the two kernel versions. This is aligned with the fact that also in Logistic Regression linear model worked better. Selected Kernel SVM Models:

- Polynomial SVM: Raw features, $K = 1$, $C = 10^{-4}$, $c = 1$, $d = 2$, PCA(m=11)
- RBF SVM: Z-Normalized features, $K = 1$, $C = 10$, $\gamma = 0.01$, no PCA

3.5 GMM Classifier

The last model we take into account is a generative model. We are going to train a GMM over the samples of each class. Given the fact that almost all the features

are already well distributed according to the Gaussian hypothesis i suppose that we won't need many gaussian components for the model. Moreover, Tied GMM model is supposed to outperform the other two models as already seen in the Gaussian classifiers. As showed in

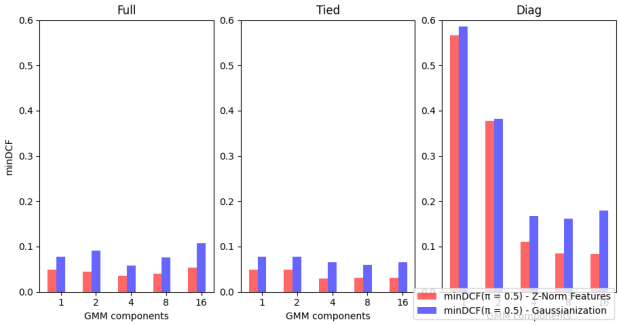

Figure 12: *GMM - minDCF for different number of components*

figure 12 for a relative small number of components (8) good results are achieved by all the three types of models. Even if diag assumption doesn't hold really well especially when the number of components is low it performs better with higher number of components. The Tied model is the one with best results even if the Full model still performs really well. Gaussianization never helps in achieving better results.
By considering only models trained with 8 compoents we are now going to show the results obtained when training with and without PCA.

Table 11: *GMM - 3-fold cross validation*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Z-normalized features - no PCA			
GMM Full (8 comp.)	0.109	0.040	0.102
GMM Tied (8 comp.)	0.099	0.031	0.075
GMM Diag (8 comp.)	0.214	0.085	0.215
Z-normalized features - PCA(m=11)			
GMM Full (8 comp.)	0.229	0.074	0.175
GMM Tied (8 comp.)	0.198	0.067	0.166
GMM Diag (8 comp.)	0.311	0.111	0.275
Z-normalized features - PCA(m=10)			
GMM Full (8 comp.)	0.225	0.082	0.212
GMM Tied (8 comp.)	0.204	0.072	0.186
GMM Diag (8 comp.)	0.310	0.108	0.260
Gaussianized features - no PCA			
GMM Full (8 comp.)	0.221	0.076	0.210
GMM Tied (8 comp.)	0.158	0.059	0.151
GMM Diag (8 comp.)	0.419	0.162	0.397
Gaussianized features - PCA(m=11)			
GMM Full (8 comp.)	0.246	0.091	0.199
GMM Tied (8 comp.)	0.175	0.066	0.169
GMM Diag (8 comp.)	0.477	0.174	0.415
Gaussianized features - PCA(m=10)			
GMM Full (8 comp.)	0.274	0.096	0.257
GMM Tied (8 comp.)	0.185	0.069	0.185
GMM Diag (8 comp.)	0.406	0.148	0.356

Table 12: *Best models analyzed up to now*

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Gaussian Models			
Tied Cov (Z-Norm, no PCA)	0.122	0.046	0.127
Tied Cov (Gau, PCA(m=10))	0.212	0.082	0.207
Logistic Regression Models			
LLR (Z-Norm, $\lambda = 10^{-6}$, no PCA)	0.132	0.047	0.126
QLR (Z-Norm, $\lambda = 10^{-5}$, no PCA)	0.153	0.052	0.142
SVM Models			
LSVM (Z-Norm, no PCA)	0.129	0.047	0.130
Poly SVM (Raw, $c = 1$, $d = 2$, PCA(m=11))	0.210	0.062	0.158
RBF SVM (Z-Norm, $\gamma = 0.01$, no PCA)	0.153	0.049	0.133

As already discussed the Tied model is the one that achieves best results (especially for our target application). PCA with $m = 10$ doesn't affect too much the results obtained while it is helpful in obtaing lower values for $minDCF$ in case of GMM-Diag with gaussianized features. As we can notice here the results are worse with respect to the ones obtained without features gaussainization and PCA but they are better w.r.t. to the GMM-Diag with gaussianized features and without PCA. WHen features gaussianization is applied the GMM-Diag model works better with a lower number of componenets (a similar behaviour was also noticed in the firstly Gaussian models)

4 Score Calibration

4.1 Calibration Analysis On Selected Models

We now select one candidate for each of the previous analyzed classification models (Gaussian, Logisti Regression, SVM, GMM):

- Gaussian model: Tied-Cov (Z-Normalization, no PCA)
- Logistic Regression: Linear LR (Z-Normalization, no PCA)
- SVM:
- GMM:

4.2 Calbrating Scores For Selected Models

We are going to transform the scores so that the theoretical threshold $t = -log\ \frac{\tilde{\pi}}{1-\tilde{\pi}}$ provides close to optimal values over a wild range of effective priors $\tilde{\pi}$. What we want to find is a monotonic function f that maps not-calibrated scores in calibrated scores.
We assume that the function f has the form:

$$f(s) = \alpha s + \beta$$

and $f(s)$ can be interpreted as log-likelihood ratio for the two class hypotheses:

$$f(s) = log\ \frac{f_{S|C}(s|\mathcal{H}_T)}{f_{S|C}(s|\mathcal{H}_F)} = \alpha s + \beta$$

and the class posterior probabilty for prior $\tilde{\pi}$ corresponds to:

$$log\ \frac{P(C=\mathcal{H}_T|s)}{P(C=\mathcal{H}_F|s)} = \alpha s + \beta + log\ \frac{\tilde{\pi}}{1-\tilde{\pi}}$$

By interpreting scores s as samples of a dataset (each sample has 1 feature) we can employ a prior weighted logistic regression model to learn the model parameters over our calibration set (the dataset composed of the scores for a certain model). If we let:

$$\beta' = \beta + log\ \frac{\tilde{\pi}}{1-\tilde{\pi}}$$

we have exactly a Logistic Regression model where α and β' are the model parameters we have to learn. We have still to specify a prior $\tilde{\pi}$ that will be the one of our target application even if we will notice that the improvements in term of calibration will also involve the unbalanced applications. To obtain calibrated scores we will have to compute:

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

4.3 Subsection

Nam ante risus, tempor nec lacus ac, congue pretium dui. Donec a nisl est. Integer accumsan mauris eu ex venenatis mollis. Aliquam sit amet ipsum laoreet, mollis sem sit amet, pellentesque quam. Aenean auctor diam eget erat venenatis laoreet. In ipsum felis, tristique eu efficitur at, maximus ac urna. Aenean pulvinar eu lorem eget suscipit. Aliquam et lorem erat. Nam fringilla ante risus, eget convallis nunc pellentesque non. Donec ipsum nisl, consectetur in magna eu, hendrerit pulvinar orci. Mauris porta convallis neque, non viverra urna pulvinar ac. Cras non condimentum lectus. Aliquam odio leo, aliquet vitae tellus nec, imperdiet lacinia turpis. Nam ac lectus imperdiet, luctus nibh a, feugiat urna.

- First item in a list
- Second item in a list
- Third item in a list

Nunc egestas quis leo sed efficitur. Donec placerat, dui vel bibendum bibendum, tortor ligula auctor elit, aliquet pulvinar leo ante nec tellus. Praesent at vulputate libero, sit amet elementum magna. Pellentesque sodales odio eu ex interdum molestie. Suspendisse lacinia, augue quis interdum posuere, dolor ipsum euismod turpis, sed viverra nibh velit eget dolor. Curabitur consectetur tempus lacus, sit amet luctus mauris interdum vel. Curabitur vehicula convallis felis, eget mattis justo rhoncus eget. Pellentesque et semper lectus.

First This is the first item
Last This is the last item

Donec nec nibh sagittis, finibus mauris quis, laoreet augue. Maecenas aliquam sem nunc, vel semper urna hendrerit nec. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Maecenas pellentesque dolor lacus, sit amet pretium felis vestibulum finibus. Duis tincidunt sapien faucibus nisi vehicula tincidunt. Donec euismod suscipit ligula a tempor. Aenean a nulla sit amet magna ullamcorper condimentum. Fusce eu velit vitae libero varius condimentum at sed dui.

4.4 Subsection

In hac habitasse platea dictumst. Etiam ac tortor fermentum, ultrices libero gravida, blandit metus. Vivamus sed convallis felis. Cras vel tortor sollicitudin, vestibulum nisi at, pretium justo. Curabitur placerat elit nunc, sed luctus ipsum auctor a. Nulla feugiat quam venenatis nulla imperdiet vulputate non faucibus lorem. Curabitur mollis diam non leo ullamcorper lacinia.

Morbi iaculis posuere arcu, ut scelerisque sem. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Mauris placerat urna id enim aliquet, non consequat leo imperdiet. Phasellus at nibh ut tortor hendrerit accumsan. Phasellus sollicitudin luctus sapien, feugiat facilisis risus consectetur eleifend. In quis luctus turpis. Nulla sed tellus libero. Pellentesque metus tortor, convallis at tellus quis, accumsan faucibus nulla. Fusce auctor eleifend volutpat. Maecenas vel faucibus enim. Donec venenatis congue congue. Integer sit amet quam ac est aliquam aliquet. Ut commodo justo sit amet convallis scelerisque.

1. First numbered item in a list
2. Second numbered item in a list
3. Third numbered item in a list

Aliquam elementum nulla at arcu finibus aliquet. Praesent congue ultrices nisl pretium posuere. Nunc vel nulla hendrerit, ultrices justo ut, ultrices sapien. Duis ut arcu at nunc pellentesque consectetur. Vestibulum eget nisl porta, ultricies orci eget, efficitur tellus. Maecenas rhoncus purus vel mauris tincidunt, et euismod nibh viverra. Mauris ultrices tellus quis ante lobortis gravida. Duis vulputate viverra erat, eu sollicitudin dui. Proin a iaculis massa. Nam at turpis in sem malesuada rhoncus. Aenean tempor risus dui, et ultrices nulla rutrum ut. Nam commodo fermentum purus, eget mattis odio fringilla at. Etiam congue et ipsum sed feugiat. Morbi euismod ut purus et tempus. Etiam est ligula, aliquam eget porttitor ut, auctor in risus. Curabitur at urna id dui lobortis pellentesque.

5 Section

Figure 13: *A majestic grizzly bear*