**MLPR Exam Project: Gender Identification**

**Antonio Ferrigno [s316467]**

**Simone Varriale [s]**

July 17, 2023

**This project is intended to show a binary classification task on a dataset made of 12 continuous observations coming from face images embeddings. An image embedding represents a small-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. Training and evaluation sets

The datasets provided are:

* 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).

We will use the Training Set to perform all the analysis and only once we will have selected the most promising models, we will train these ones using the entire Training Set and final considerations will be done considering their behavior on the Evaluation Set.

* 1. Features statistics
  2. Z-normalization
  3. Features distribution
     1. Gaussianization
  4. Features correlation

1. Dimensionality reduction
   1. PCA
   2. LDA
2. Classification models analysis
   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.

We will consider three types of applications:

and the target application (the one we will optimize for) will be the balanced one:

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

and for minDCF computation we will look for the threshold:

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

* 1. Gaussian models

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

Since we are dealing with a binary classification task we will assign a probabilistic score to each sample in terms of the class-posterior log-likelihood ratio:

We can expand this expression by writing:

While the training phase consists in estimating the model parameters of the Multivariate Gaussian distribution 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 to compute the minDCF. What it differentiates the different Gaussian models is the way how we estimate the model parameters of the Gaussian distribution.

* + 1. MVG Gaussian Classifier

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

* + 1. Naive Bayes Classifier

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

The density of a sample x can be expressed as N (x|,) where μc is an array where each element is the the mean for each class for each component while is a diagonal covariance matrix. The Naïve Bayes classifier corresponds to the MVG full covariance classifier with a diagonal covariance matrix.

* + 1. Tied Gaussian Classifier

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

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:

This model is strongly related to LDA (used as a linear classification model). By considering the binary log-

likelihood ratio of the tied model we obtain a linear decision function:

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:

where Λ(μ1 − μ0) = b. The LDA assumption that all the classes have the same within class covariance matrix is related to the assumption done for the tied model.

* + 1. Gaussian Models Comparison

**Table 1:** *Gaussian Models*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 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:

A group of graphs showing the number of points

Description automatically generated

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

* We can notice that Full-Cov model and Tied-Cov model achieve very good and similar results with slightly better performances for the Tied- Cov model (this means that the covariance matrices of the different classes are similar and in fact the model provides more robust estimates in this case).
* Gaussianization preprocessing doesn’t really help in achieving better results because data are already well distributed according to the Gaussian assumptions (this was also noticed previously while discussing the Gaussianization preprocessing).
* Naive Bayes assumption doesn’t hold well, if PCA is not applied. When PCA is applied it has a good impact only on Naïve Bayes model and especially if also combined with Gaussianization pre-processing (the Naive Bayes assumption that the covariance matrix of each class is diagonal holds better with lower features dimensionality and improved features distribution from a Gaussian point of view).
* Regarding Full and Tied models with PCA(m=11) there is not a high-performance degradation since the minDCFs values 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 preprocessing and without PCA. Really good performances are also achieved by the Tied-Cov model trained with Gaussianized features 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 for reducing computational effort).

Best Gaussian Models:

* Tied Cov (Z-Normalization, no PCA)
* Tied Cov (Gaussianization, PCA(m=11))
  1. Logistic Regression Classifier

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

where b considers all the prior information. Given w and b we can compute the expression for the

posterior class probability:

where σ(x) = 1 / (1+e^(−x)) is the sigmoid function. LR assumes that the decision rules will be hyperplanes orthogonal to w.

* + 1. Linear Logistic Regression (LLR)

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

where λ is an hyperparameter that represents the regularization term (needed to make the problem solvable

in case of linearly separable classes).

A graph of a line graph

Description automatically generated

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

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 0*.*9 |
| **Z-normalized features - no PCA** | | | |
| LLR (*λ* = 10*−*3) | 0.170 | 0.059 | 0.155 |
| LLR (*λ* = 10*−*5) | 0.132 | 0.047 | 0.127 |
| LLR (*λ* = 10*−*6) | 0.132 | 0.047 | 0.126 |
| **Z-normalized features - PCA(m=11)** | | | |
| LLR (*λ* = 10*−*3) | 0.278 | 0.104 | 0.232 |
| LLR (*λ* = 10*−*5) | 0.269 | 0.098 | 0.221 |
| LLR (*λ* = 10*−*6) | 0.268 | 0.098 | 0.222 |
| **Z-normalized features - PCA(m=10)** | | | |
| LLR (*λ* = 10*−*3) | 0.299 | 0.113 | 0.263 |
| LLR (*λ* = 10*−*5) | 0.297 | 0.114 | 0.263 |
| LLR (*λ* = 10*−*6) | 0.297 | 0.114 | 0.262 |

**Table 3:** *Best models analyzed up to now.*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 0*.*9 |
| **Gaussian Models** | | | |
| (Z-Norm, no PCA) | 0.122 | 0.046 | 0.127 |
| Tied Cov (Gau, PCA(m=10)) | 0.212 |
| 0.082 | 0.207 |

* The choice of λ appear to be critical for all the applications, for the unbalanced ones. By observing figure 7 for values of λ greater than 10−3 the minDCF rapidly increases for all the considered applications.
* PCA never helps in achieving better results.

*Comparison*: the LLR model trained with λ = 10^−6 and no-PCA achieves really similar results with the ones achieved by the Tied-Cov Gaussian model (Z-Norm, no PCA) for = 0.5 and = 0.9 applications but a worse performance is obtained for = 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, λ = 10^−6, no PCA

* + 1. Quadratic Logistic Regression (QLR)

Now we are going to train a Quadratic LR model by performing features expansion. For binary linear LR the separation surfaces are linear decision functions 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:

This expression is quadratic in x but it’s 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:

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

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

A graph of a line graph

Description automatically generated

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

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 0*.*9 |
| **Z-normalized features - no PCA** | | | |
| QLR (*λ* = 10*−*3) | 0.187 | 0.063 | 0.149 |
| QLR (*λ* = 10*−*5) | 0.153 | 0.052 | 0.142 |
| QLR (*λ* = 10*−*6) | 0.150 | 0.053 | 0.141 |
| **Z-normalized features - PCA(m=11)** | | | |
| QLR (*λ* = 10*−*3) | 0.267 | 0.098 | 0.222 |
| QLR (*λ* = 10*−*5) | 0.263 | 0.098 | 0.230 |
| QLR (*λ* = 10*−*6) | 0.305 | 0.096 | 0.230 |
| **Z-normalized features - PCA(m=10)** | | | |
| QLR (*λ* = 10*−*3) | 0.299 | 0.111 | 0.241 |
| QLR (*λ* = 10*−*5) | 0.307 | 0.109 | 0.249 |
| QLR (*λ* = 10*−*6) | 0.305 | 0.109 | 0.248 |

* By rapidly looking at the results obtained we can say that quadratic version of the logistic regression performs worse w.r.t the linear version (the one without features expansion).

**Table 5:** *Best models analyzed up to now.*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 0*.*9 |
| **Gaussian Models** | | | |
| Tied Cov (Z-Norm, no PCA)  Tied Cov (Gau, PCA(m=10)) | 0.122  0.212 | 0.046  0.082 | 0.127  0.207 |
| **Logistic Regression Models** | | | |
| LLR  (Z-Norm, *λ* = 10*−*6, no PCA) | 0.132 | 0.047 | 0.126 |

* The choice of λ is still critical and λ <= 10−5 remains the best choice
* Regarding the target application we can reach similar results comparing to the linear version of the LR while the unbalanced applications are more penalized.
* When PCA is applied no effective improvements are obtained.

*Comparison*: With respect to Gaussian models, comparable performances are achieved for λ = 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, λ = 10−5, no PCA
  1. SVM Classifier
     1. Linear SVM

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

where N is the number of training samples and C is an hyperparameter.

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

where **1** is a n-dimensional vector of ones and H is the matrix, whose elements are .

From the constraints of the Lagrangian problem that allows to introduce the dual formulation we can obtain that:

and the optimal bias b can be computed considering a sample xi that lies on the margin: . To be able to computationally solve the problem we need to modify the primal formulation as:

where and .

The scoring rule has the same form of the original formulation but we are also regularizing the norm of and we use a mapping to mitigate the fact that by regularizing the bias term we could obtain sub-optimal results. According to the modification done to the primal formulation we also modify the dual formulation as:

where the equality constraint disappeared (the one that L-BFGS was not able to incorporate) and the matrix can be computed as .

A graph of different colored lines

Description automatically generated

**Figure 9:** *Linear SVM (K=0, Z-Normalized features) –*

*minDCF for different values of C an different priors*

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 0*.*9 |
| **Z-normalized features - no PCA** | | | |
| Linear SVM (*C* = 0*.*1)  Linear SVM (*C* = 1) | 0.158  0.129 | 0.052  0.047 | 0.141  0.130 |
| **Z-normalized features - PCA(m=11)** | | | |
| Linear SVM (*C* = 0*.*1)  Linear SVM (*C* = 1) | 0.275  0.269 | 0.102  0.100 | 0.233  0.226 |
| **Z-normalized features - PCA(m=10)** | | | |
| Linear SVM (*C* = 0*.*1)  Linear SVM (*C* = 1) | 0.295  0.301 | 0.114  0.113 | 0.259  0.267 |

**Table 7:** *Best models analyzed up to now.*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 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** | | | |
| (Z-Norm, *λ* = 10*−*6, no PCA) | 0.132 | 0.047 | 0.126 |
| QLR  (Z-Norm, *λ* = 10*−*5, no PCA) | 0.153 | 0.052 | 0.142 |

The 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.
* PCA is not helpful in reducing the minDCF in any of the considered applications even if with PCA(m=11) there is not 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 results obtained by the Linear Logistic Regression Model but slightly worse with respect to the results obtained by the Tied Gaussian model.

Selected Linear SVM Model:

Z-normalized features - K=0, C=1 - no PCA

* + 1. 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: where k is the kernel function. We must replace . We are going to implement two types of kernels: polynomial and rbf.

* Polynomial kernel of degree d:
* Radial Basis Function kernel:

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

**Figure 10:** *Polynomial SVM (K=1, c=1, 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 ≤ 10^−2. For this reason, we have chosen C = 10^−4 to show more detailed results.

**Table 8:** *Polynomial Kernel SVM – C = 10^−4 – 3-fold cross validation*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 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 can 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 = 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) led to worse performances in all the considered applications.

A graph of different colored lines

Description automatically generated

**Figure 11:** *RBF SVM (K = 1, γ ∈ {0.1, 0.01}) –*

*minDCF for different values of C*

The other kernel that was employed is the RBF kernel and these are the results achieved: In this case it was again used Z-normalization as pre-processing step. The hyperparameter γ must be tuned so we trained the model for different values of γ to analyze the performances and here are showed the results for γ = 0.1 and γ = 0.01. Also, the hyperparameter C is still to be chosen and needs to be estimated via cross-validation. The parameter K has been set to 1. From figure 11 we again realize that the value of the hyperparameter C is critical, and we should choose a value C ≥ 10−1. The table below shows the results obtained with C = 10 and results obtained with γ = 1 appear for completeness.

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 0*.*9 |
| **Z-Normalized features - no PCA** | | | |
| RBF SVM (*γ* = 1) | 0.291 | 0.095 | 0.281 |
| RBF SVM (*γ* = 0*.*1) | 0.163 | 0.056 | 0.137 |
| RBF SVM (*γ* = 0*.*01) | 0.153 | 0.049 | 0.133 |
| **Z-Normalized features - PCA(m=11)** | | | |
| RBF SVM (*γ* = 1) | 0.362 | 0.123 | 0.350 |
| RBF SVM (*γ* = 0*.*1) | 0.255 | 0.092 | 0.255 |
| RBF SVM (*γ* = 0*.*01) | 0.254 | 0.091 | 0.211 |
| **Z-Normalized features - PCA(m=10)** | | | |
| RBF SVM (*γ* = 1) | 0.386 | 0.136 | 0.379 |
| RBF SVM (*γ* = 0*.*1) | 0.271 | 0.106 | 0.273 |
| RBF SVM (*γ* = 0*.*01) | 0.291 | 0.107 | 0.239 |

**Table 10:** *Best models analyzed up to now*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 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, *λ* = 10*−*6, no PCA) | 0.132 | 0.047 | 0.126 |
| QLR  (Z-Norm, *λ* = 10*−*5, no PCA) | 0.153 | 0.052 | 0.142 |
| **SVM Models** | | | |
| LSVM  (Z-Norm, *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 γ = 0.01 because with this value we can 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, γ = 0.01, no PCA
  1. GMM Classifier

The last model we are going to consider is a generative model. The GMM problem is related to the estimation of a population distribution and can be also applied to the classification task. We resort the class-posterior probability used for the Gaussian classifiers:

More in general we can introduce a weight term instead of the prior probability that will be one of the model’s parameters of the GMM:

so, the GMM density is the sum of M Gaussians where are the model parameters. Gaussian components can be seen as clusters the samples belong to (in a hard or in a soft way) and the cluster label is a latent random variable. If we define

we can introduce a term called responsibility that represents the posterior probability that a sample belongs to a certain cluster(component):

We can assign the sample to the cluster label for which the responsibility is maximum and then re-estimate the model parameters given the cluster assignments. An evident problem is that we are forming hard clusters so we are not admitting that a sample could belong to more than one cluster(component). To handle soft-

clusters we introduce the statistics:

to obtain a re-estimate of = / N. In this way we will be able to apply the Expectation-Maximization algorithm:

* E-step: estimate (given the model parameters (, ,)):
* M-step: estimate the new model parameters by using the statistics mentioned above.

The estimation goes on starting from an initial value of the model parameters until a certain criterion is met. The EM algorithm thus require an initial estimate for the GMM parameters, so we employ the LBG algorithm to incrementally construct a GMM with 2G components from a GMM with G components. The starting point will be (1, μ, C) so we use the empirical mean and covariance matrix of the dataset. We can then build a 2-componenents model starting from one and from each of the new components we

generate other new 2 components and so on and so forth.

We are also going to introduce a Diagonal and Tied version of the GMM problem: the diagonal variant consists of a model where each component has a diagonal covariance matrix (this does not correspond with the Naive Bayes assumption did for the Gaussian models). The Tied covariance model assumes that the covariance of a single GMM = are the same but each GMM of each class has a difference covariance matrix Σc (again, this is different from the Tied MVG model). We are now 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 we 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. By considering only models trained with 8 components we are now going to show more in detail the results obtained when training with and without PCA.

A graph of different colored bars

Description automatically generated

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

* As showed in figure 12, for a relatively small number of components (8) good results are achieved by all the three types of models.
* Even if diag assumption doesn’t hold 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 well and this corresponds to the hypothesis made before according to what has already been analyzed for the Gaussian models.
* Gaussianization never helps in achieving better results. According to what we had seen with the MVG models the already good distribution of data makes this classification task not benefitting from the Gaussianization pre-processing step and worse results are achieved when it is applied.
* PCA is not helpful in achieving better results (we can see an improvement for some applications when PCA(m=10) is applied w.r.t to PCA(m=11) but all these results are worse w.r.t the no PCA versions of the trained models). With Gaussianization and PCA(m=11) instead we can notice an improvement for ˜π = 0.9 application.

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 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*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 0*.*9 |
| **Gaussian Models** | | | | |
| (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, *λ* = 10*−*6, no PCA) | | 0.132 | 0.047 | 0.126 |
| QLR  (Z-Norm, *λ* = 10*−*5, no PCA) | | 0.153 | 0.052 | 0.142 |
|  | **SVM Models** | |  |  |
| LSVM  (Z-Norm, no PCA)  Poly SVM  (Raw, *c* = 1, *d* = 2, PCA(m=11))  RBF SVM  (Z-Norm, *γ* = 0*.*01, no PCA) | | 0.129 | 0.047 | 0.130 |
| 0.210 | 0.062 | 0.158 |
| 0.153 | 0.049 | 0.133 |

The GMM Tied(8 components) trained with Z-Normalization and no PCA is the one that achieves the best results among all the models evaluated up to now.

Selected GMM Model:

* GMM Tied (8 componenets), Z-Normalization, no PCA.

1. Score Calibration
   1. Calibration Analysis on Selected Models

We now select one candidate for each of the previous analyzed classification models (Gaussian, Logisti Re-

gression, SVM, GMM):

* Gaussian model : Tied-Cov (Z-Normalization, no PCA)
* Logistic Regression : Linear LR (Z-Normalization, no PCA)
* SVM: Linear SVM (Z-Normalization, no PCA)
* GMM : Tied GMM with 8 components (Z-normalization, no PCA)
  1. Calibrating Scores for Selected Models

We are going to transform the scores provided by each model so that the theoretical threshold t = −log ˜π 1−˜π provides close to optimal values over a wide range of effective priors ˜π. 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:

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

and the class posterior probability for prior corresponds to:

By interpreting scores s as samples of a dataset (where 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:

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 ˜π that will be the one of our target application even if we will notice that the im- provements in terms of calibration will also involve the unbalanced applications. To obtain calibrated scores we will have to compute:

A group of graphs with numbers

Description automatically generated

**Figure 13:** *Bayesian Error Plots for each of the selected models*

*(without score calibration)*

The Tied Gaussian, Linear Logistic Regression and Tied GMM models seem to be already well calibrated for positive values of the prior log-odds. The Tied Gaussian model could be slightly improved in terms of calibration for negative values of the prior log odds. The SVM instead is not well calibrated for almost every considered value of the prior log-odds.

A graph of a number of numbers

Description automatically generated

**Figure 14:** *Bayesian Error Plots for each of the selected models*

*(With score calibration)*

**Table 13:** *Calibrated vs not calibrated scores for best selected*

*Models*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 0*.*9 |
| **Uncalibrated scores [minDCF / DCF]** | | | |
| Tied Gaussian | 0.122/0.127 | 0.046/0.050 | 0.127/0.134 |
| Lnear LR | 0.132/0.139 | 0.047/0.048 | 0.126/0.136 |
| Linear SVM | 0.132/0.289 | 0.047/0.047 | 0.129/0.285 |
| Tied GMM | 0.099/0.110 | 0.031/0.031 | 0.075/0.081 |
| **Calibrated scores [minDCF / DCF]** | | | |
| Tied Gaussian | 0.122/0.128 | 0.046/0.049 | 0.127/0.135 |
| Lnear LR | 0.132/0.141 | 0.047/0.048 | 0.126/0.132 |
| Linear SVM | 0.132/0.135 | 0.047/0.047 | 0.129/0.137 |
| Tied GMM | 0.099/0.105 | 0.031/0.031 | 0.075/0.076 |

We can easily notice that for models where scores where already well calibrated the score calibration didn’t provide effective benefits (in some cases we also obtained slightly worse results). For SVM model great benefits are obtained for the unbalanced applications.

1. Experimental Results

Now we are going to train all the previous models on the entire training set and the evaluation set will be used for the first time to assess if the results obtained so far are consistent and the best model selected is confirmed (we could also realize that other choices would have been better for this evaluation set).

**Table 14:** *minDCF over evaluation set for all models trained over the whole training set (Z-Normalized features, no PCA)*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 0*.*9 |
| **Gaussian Models (Z-normalized features, no PCA)** | | | |
| Full Cov | 0.134 | 0.053 | 0.138 |
| Tied Cov | 0.133 | 0.051 | 0.135 |
| Naive Bayes | 0.810 | 0.570 | 0.882 |
| **LR Models (Z-Normalized features, no PCA)** | | | |
| LLR (*λ* = 10*−*6) | 0.135 | 0.052 | 0.133 |
| QLR (*λ* = 10*−*5) | 0.148 | 0.053 | 0.141 |
| **SVM Models (Z-Normalized features, no PCA)** | | | |
| LSVM | 0.142  0.194  0.138 | 0.052  0.062  0.054 | 0.137  0.163  0.133 |
| (*K* = 0*, C* = 1) |
| Poly SVM |
| (*K* = 1*, C* = 10*−*4*, c* = 1*, d* = 2) |
| RBF SVM |
| (*K* = 1*, C* = 10*, γ* = 0*.*01) |
| **GMM Models (Z-Normalized features, no PCA)** | | | |
| GMM Full (8 comp.) | 0.092 | 0.034 | 0.086 |
| GMM Full (16 comp.) | 0.103 | 0.043 | 0.117 |
| GMM Tied (8 comp.) | 0.088 | 0.030 | 0.087 |
| GMM Tied (16 comp.) | 0.089 | 0.033 | 0.086 |
| GMM Diag (8 comp.) | 0.236 | 0.090 | 0.234 |
| GMM Diag (16 comp.) | 0.244 | 0.098 | 0.232 |

**Table 15:** *minDCF over evaluation set*

*for all models trained over the whole training set*

*(Z-Normalized features, PCA(m=11))*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 0*.*9 |
| **Gaussian Models (Z-normalized features, PCA(m=11))** | | | |
| Full Cov | 0.280 | 0.113 | 0.255 |
| Tied Cov | 0.278 | 0.111 | 0.256 |
| Naive Bayes | 0.293 | 0.116 | 0.263 |
| **LR Models (Z-Normalized features, PCA(m=11))** | | | |
| LLR (*λ* = 10*−*6) | 0.281 | 0.110 | 0.259 |
| QLR (*λ* = 10*−*5) | 0.253 | 0.108 | 0.258 |
| **SVM Models (Z-Normalized features, PCA(m=11))** | | | |
| LSVM | 0.277  0.189  0.243 | 0.110  0.068  0.109 | 0.262  0.151  0.246 |
| (*K* = 0*, C* = 1) |
| Poly SVM |
| (*K* = 1*, C* = 10*−*4*, c* = 1*, d* = 2) |
| RBF SVM |
| (*K* = 1*, C* = 10*, γ* = 0*.*01) |
| **GMM Models (Z-Normalized features, PCA(m=11))** | | | |
| GMM Full (8 comp.) | 0.179 | 0.080 | 0.196 |
| GMM Full (16 comp.) | 0.219 | 0.086 | 0.211 |
| GMM Tied (8 comp.) | 0.173 | 0.072 | 0.195 |
| GMM Tied (16 comp.) | 0.185 | 0.078 | 0.194 |
| GMM Diag (8 comp.) | 0.329 | 0.124 | 0.312 |
| GMM Diag (16 comp.) | 0.311 | 0.141 | 0.327 |

The results obtained by training the models on the entire training set and by evaluating them on the entire evaluation set are coherent with the results previously obtained and with the observations made. The ROC curve in figure 15 shows (and confirms) that the best performing model is the Tied GMM with 8 components. The other three models also perform well but slightly worse than this.

A graph of a graph of a person

Description automatically generated

**Figure 15:** *ROC curves of selected models (on the right a zoomed version)*

1. Conclusions

We can conclude by saying that in general linear models have better performances on these data with respect to the non-linear ones. Due to the high level of correlation among features the PCA didn’t help in achieving better results (even though with PCA(m=11) we have obtained not a big performance degradation in most of the models), and Naive hypothesis didn’t work well in Gaussian models. We have been able to reach a minDCF ≈ 0.03 for the target application ( = 0.5) but also for the unbalanced applications we were able to achieve good results (minDCF ≈ 0.1 for = 0.1 and minDCF ≈ 0.1 for = 0.9). The choices made on the training set via k-fold cross-validation proved to be coherent with the results obtained on the entire training set.