**MLPR Exam Project: Gender Detection**

**Mattia Rosso [s294711]**

September 4, 2022

### his project is inteded to show a binary clas- sification task on a datased made of 12 con- tinuous observations coming from speking

T

uniform is to apply Z-normalization as a preprocessing step. We are going to transform each sample of our dataset as:

### embeddings. A speaker embedding represents a small-dimensional, fixed size representation of an

*σj*

*zi,j*

= *xi,j* −*µj* ∀*xi* ∈ *D*

**utterance. Features can be seen as points in the m-dimensional embedding space (and the embed- dings have already been computed). This is a task where classes are balanced both in training and evaluation set.**

# Dataset analysis

## 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 anaylsis and only once we will have selected the most promis- ing models we will train these ones using the entire Training Set and final considerations will be done con- sidering their behaviour on the Evaluation Set.

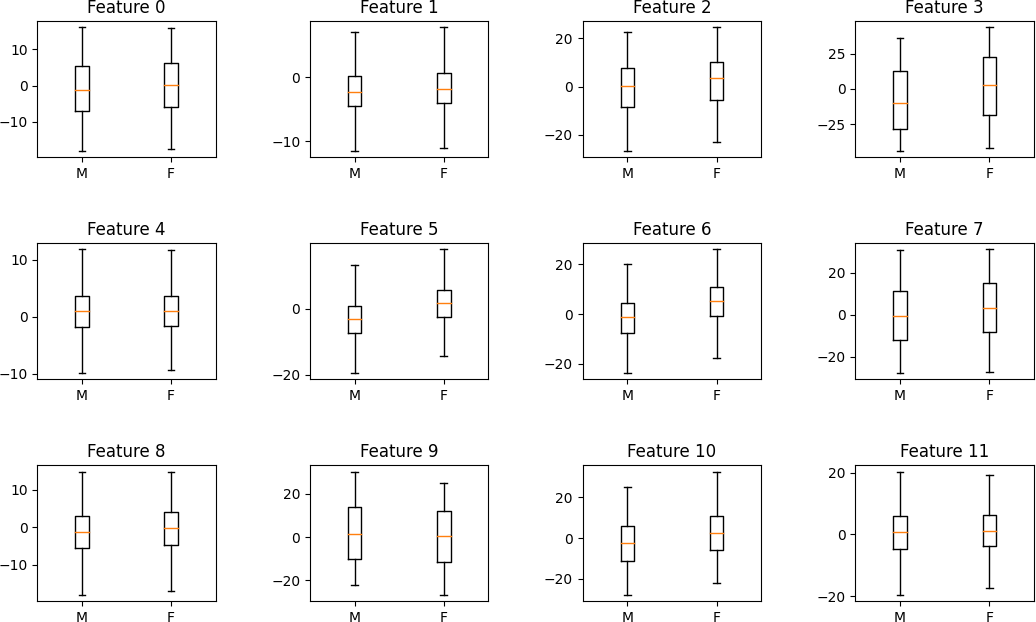
## Features Statistics

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

## Z-normalization

A useful operation that can be applied in order to avoid to deal with numerical issues and to make data more

Where *zi,j* is the Z-normalized value corresponding to the feature *j* of sample *i* while *µj* and *σj* are, respec- tively, the mean and the variance computed over all the values for feature *j*.

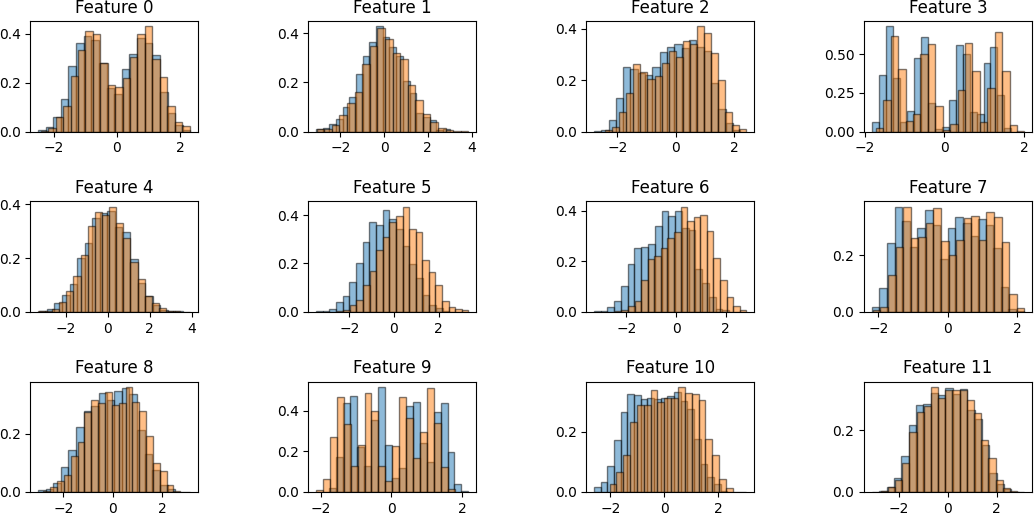


**Figure 1:** *Raw features boxplots*

## Features distribution

By plotting one histogram for each feature, separately for classes male and female, it is possible to show if the samples (separately for each feature) follow a Gaus- sian distribution and how well. This is done in order to understend whether a pre processing step like Gaus- sianization can be useful or not for our training data.

We can notice from figure 2 that almost all the fea- tures are already well-distributed (w.r.t. the Gaussian distribution) except for features 3, 7, 9. We can thus apply an additional pre-processing step in order to



**Figure 2:** *Z-normalized features distribution*

make all the features following a Gaussian distribution: this step is called Gaussianization and it will be always applied after Z-Normalization preproccesing.

### 1.4.1 Gaussianization

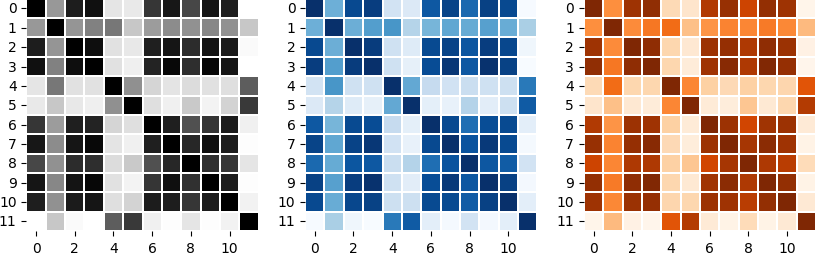
Gaussianization is a pre-processing step that maps each fature to values whose empirical cumulative distribution are well approximated by a Gaussian cumulative distribution function. For each feature *x* that we want to gaussianize we firstly compute the rank over the dataset:

cells [*i, j*] for which it exists an high correlation among feaure *i* and feature *j*. We are going to use the Pear- son correlation coefficient to compute the correlation among feature *X* and feature *Y* :

*Cov*(*X,Y* )

|√ √ |

*V ar*(*X*) *V ar*(*Y* )



**Figure 4:** *Z-normalized 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 mean- ingful for values of *m* not below 11 or 10 at most. For smaller values of *m* we would probably loose important information coming from high-correlated features.

*r*(*x*) = Σ*N*

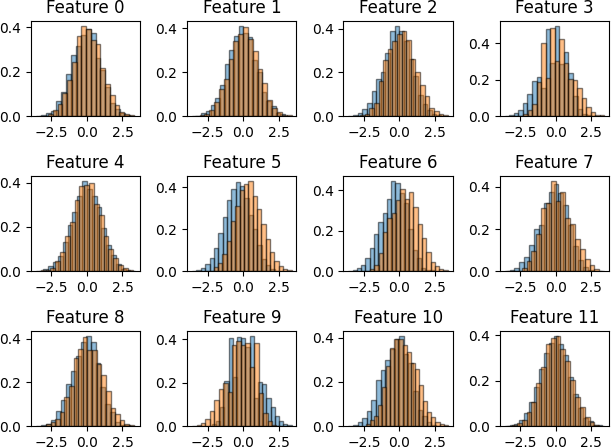
I[*xi<x*]+1

*i*=1

*N* +2

where 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* = Φ−1(*r*(*x*)) where Φ is the inverse of the cumulative distribution function.



**Figure 3:** *Gaussianized features distribution*

## Features correlation

We can show how much features are correlated by using a heatmap plot showing a darker color inside

# Dimensionality reduction

Before proceeding with the classification task we will spend some words on the possible dimensioanlity re- duction techniques that we have analyzed and that could be applied: PCA and LDA.

## 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 tech- nique that, given a centered dataset *X* = *x*1*, ..., xk* , it aims to find the subspace of R*n* that allows to pre- serve most of the information (the directions with the highest variance).

{ }

Starting from the empirical covariance matrix

Σ*C* = (*x* − *x*¯)(*x* − *x*¯)*i* *i*

1 *T*

*K i*

we compute the eigen-decomposition of *C* = *U* Σ*UT* and project the data in the subspace spanned by the *m* columns of *U* corresponding to the *m* highest eigen- values:

*yi* = *PT* (*xi* − *x*¯)

where *P* is the matrix corresponding the the *m* columns of *U* associated the to the *m* highest eigenvalues of *C*. In this preliminary analysis, in order to select the optimal *m*, we can use a cross-validation approach by inspecting how much of the total variance of the data we are able to retain by using different values for *m*.

We exploit the fact that each eigenvalue corresponds 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 *s.t* Σ*m σi* ≥ *t*

Σ

*i*

# Classification models analysis

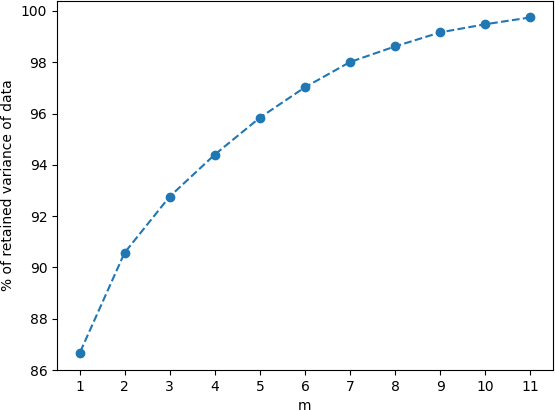
## Premises

In the next paragraphs we are going to compare differ- ent classification models. We will employ a k-fold cross

*i*

*m n σi*

validation technique (with *k* = 3) for model evaluation.



**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 of the data. We will see better later on how choosing a too small value of *m* would badly impact the performances of the different classifiers.

## 2.2 LDA

PCA technique is unsupervised so we have no guar- antee of obtainig discriminant directions. Despite the fact that LDA allows to find at most *C* 1 discriminant directions (where *C* is the number of classes), so it makes no sense to apply it as a dimensionality reduc- tion technique in a bianry classification task, 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:

−

We will consider three types of applications:

(*π*˜*, Cfp, Cfn*) = (0*.*1*,* 1*,* 1)

(*π*˜*, Cfp, Cfn*) = (0*.*5*,* 1*,* 1)

(*π*˜*, Cfp, Cfn*) = (0*.*9*,* 1*,* 1)

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

(*π*˜*, Cfp, Cfn*) = (0*.*5*,* 1*,* 1)

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

*DCF* = *DCFu*(*πT ,Cfn,Cfp*) =

*min*(*πT ,Cfn,*(1−*πT Cfp*))

*πT CfnPfn*+(1−*πT* )*CfpPfp*

*min*(*πT ,Cfn,*(1−*πT Cfp*))

and for *minDCF* computation we will look for the threshold:

*t*′ = *log*(  *π*˜ )

−

1−*π*˜

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

## 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 *xt* is a realization of the

R.V. *Xt*. A simple model consists in assuming that our

data, given the class, can be described by a Gaussian distribution:

(*Xt*|*Ct* = *c*) ∼ (*X*|*C* = *c*) ∼ N (*xt*|*µc,* Σ*c*)

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:

*llr*(*xt*) = *log r*(*xt*) = *log P* (*C*=*h*1|*xt*)

*P* (*C*=*h*0 |*xt*)

L(*w*) =

*sB* = max*w*

*wT SBw*

*T*

We can expand this expression by writing:

*sW w SW w*

*fX|C* (*xt*|*h*1) *π*

It can be proved that the optimal solution corre- sponds to the eigenvector of *SW*− 1*SB* corresponding to the largest eigenvalue. Once that we have esti-

mated *w* we can project the test samples over *w*

and assign the class label looking at the score obtained:

*C*1*,* if *wT xt* ≥ *t.*

*log r*(*xt*) = *log fX|C* (*xt*|*h*0 ) + *log* 1−*π*

with *fX*|*C*(*xt*|*c*) = N (*x*|*µc,* Σ*c*)

While the training phase consists in estimating the model parameters of the Mutlivariate Gaussian dis- tribution 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 spe-

*C*(*xt*) =

*C*0*,* if *wT xt < t.*

cific for each application to compute the *minDCF* . What it differentiates the different Gaussian models is

It will be proved later how this model is related to the Tied Gaussian Generative Classifier model.

the way how we estimate the model parameters of the Gaussian distribution.

### MVG Gaussian Classifier

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

**Table 1:** *Gaussian Models*

*µ*∗ = 1 Σ*N*

*c*

*Nc*

*i*=1

*c*

*Nc*

*i*=1

*xc,i*

*c*

*c*

Σ∗ = 1 Σ*N* (*xc,i* − *µ*∗)(*xc,i* − *µ*∗)*T*

### 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 fac- torized over its components. The ML solution to this problem is:

|

*µ*∗ = 1 Σ | *xi,*[*j*]

*i c* =*c*

*c,*[*j*]

*i*

*Nc*

*i*

*c,*[*j*]

*Nc*

*i c* =*c*

*σ*2 = 1 Σ | (*xi,*[*j*] − *µc,*[*j*])2

The density of a sample *x* can be expressed as

(*x µc,* Σ*c*) where *µc* is an array where each element *µc,*[*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.

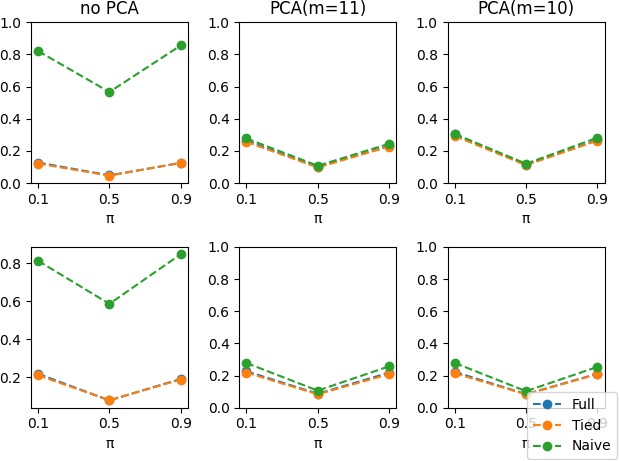
N |

### Tied Gaussian Classifier

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

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | *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 |

*fX*|*C*(*x*|*c*) = N (*x*|*µc,* Σ)

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:

*µ*∗ = 1 Σ*N*

*c*

*i*=1

*Nc*

*N*

*i*

*xc,i*

Σ∗ = 1 Σ*c* Σ*i*|*c* =*c*(*xi* − *µc*∗)(*xi* − *µc*∗)*T*

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 fX|C* (*x*|*h*1) = *xT b* + *c*

*fX|C* (*x*|*h*0)

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:

*wT x* = *k* · *xT Λ*(*µ*1 − *µ*0)

where *Λ*(*µ*1 *µ*0) = *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.

−

### Gaussian Models Comparison

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

* + - * We can notice that Full-Cov model and Tied- Cov model achieve very good and similar results with slighlty better performances for the Tied- Cov model (this means that the covariance ma- trices of the different classes are similar and infact the model provides more robust estimates in this case).
      * Gaussianization pre processing does’t really help

in achieving better results because data are al-

*T*

2

*J*(*w, b*) = *λ* ||*w*||2 + 1 Σ*n*

*log*(1 + *e*−*zi*(*w xi*+*b*))

ready well distributed according to the Gaus- sian assumptions (this was also noticed previ- ously while discussing the Gaussianization pre- processing).

* + - * 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 (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 no 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 pre processing and withouth PCA. Really good performacnes are also achieved by the Tied-Cov model trained with Gaussianized fe- tures 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:

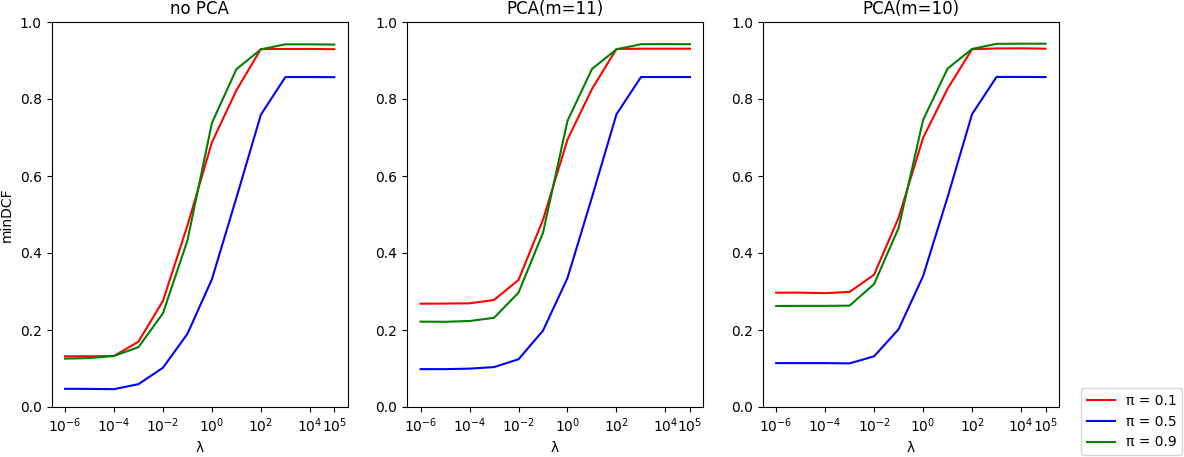
* + - * Tied Cov (Z-Normalization, no PCA)
      * Tied Cov (Gaussianization, PCA(m=11))

## 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:

*l*(*x*) = *log P* (*C*=*h*1|*x*) = *log fX|C* (*x*|*h*1) + *log*  *π* =

where *λ* is an hyperparameter that represents the regu- larization term (needed to make the problem solvable in case of linearly separable classes).



**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

where

*P* (*C*=*h*0 |*x*)

*fX|C* (*x*|*h*0)

*wT x* + *b*

1−*π*

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

Given

*b* takes into account all the prior information.

*w* and *b* we can compute the expression for the

for all the considered applications.

* PCA never helps in achieving better results.

posterior class probability:

*P* (*C* = *h*1|*x, w, b*) = *e*

1+*e*(*w*

(*wT x*+*b*) *T*

= *σ*(*wT x* + *b*)

*Comparison*: the LLR model trained with *λ* = 10−6 and

*x*+*b*)

where *σ*(*x*) =  1 is the sigmoid function. LR as- sumes that the 1d+e*e*c*−*is*x*ion rules will be hyperplanes or-

*n*

*i*=1

no-PCA achieves really similar results with the ones

thogonal to *w*.

### Linear Logistic Regression (LLR)

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

achieved by the Tied-Cov Gaussian model (Z-Norm, no PCA) for *π*˜ = 0*.*5 and *π*˜ = 0*.*9 applications but a worse performance is obtained for *π*˜ = 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

### 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:

*log P* (*C*=*h*1|*x*) = *xT Ax* + *bT x* + *c* = *s*(*x, A, b, c*)

*P* (*C*=*h*0 |*x*)

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:

**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 still remains the best choice
* Regarding the target application we are able to reach similar results comparing to the linear ver- sion of the LR while the unbalanced applications

Φ(*x*) =

*vec*(*xxT* )

*x*

*, w* =

*vec*(*A*)

*b*

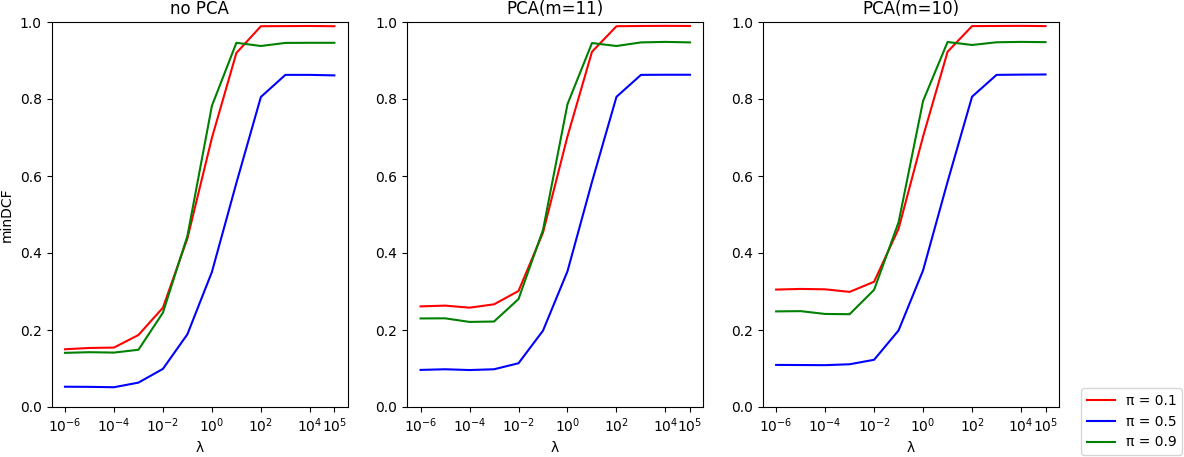
are more penalized.

* + When PCA is applied no effective improvements are obtained.

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*) = *sT φ*(*x*) + *c*

We are now going to train the Linear Logistic Regres- sion model using features vectors *φ*(*x*).



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

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

*Comparison*: With respect to Gaussian models com- parable 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

## SVM Classifier

### 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:

Σ

*J*(*w, b*) = 1 ||*w*2|| + *C N max*(0*,* 1 − *zi*(*wT xi* + *b*))

2

*i*=1

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

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

*JD*(*α*) = − 1 *αT*

Σ2

*n*

*i*=1

|  |  |  |  |
| --- | --- | --- | --- |
|  | *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 |

*Hα* + *αT*

**1** *s.t.* 0 ≤ *αi* ≤

*C and*

where **1** is a *n*-dimensional vector of ones and *H* is the matrix whose elements are *Hij* = *zizjxT xj*.

*i*

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

*w*∗ = Σ*n α*∗*z x*

*αizi* = 0 ∀*i* ∈ {1*, ..., n*}

*i*=1

*i*

*i*

*i*

* 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

and the optimal bias *b* can be computed considering a sample *xi* that lies on the margin: *zi*(*w*∗*T xi* + *b*∗) = 1. To be able to computationally solve the problem we

need to modify the primal formulation as:

without features expansion).

2

*i*=1

*J*ˆ(*w*ˆ) = 1 ||*w*ˆ||2 + *C* Σ*N max*(0*,* 1 − *zi*(*w*ˆ*T x*ˆ*i*))

where *x*ˆ*i* = *xi* and *w*ˆ = *w* .

1

*b*

results are achieved for smaller values of the hyperpa-

rameter *C* so the choice of it is crucial.

The scoring rule *w*ˆ*T x*ˆ*i* = *wT x* + *b* has the same form of the original formulation but we are also regulariz-

ing the norm of *w*ˆ : ||*w*ˆ||2 = ||*w*||2 + *b*2 and we use a

* Best performances on the target application are reached for *C* = 1 combined with Z-normalized

mapping *x*ˆ*i* = *xi* to mitigate the fact that by reg-

*K*

features and no PCA.

* PCA is not helpful in reducing the minDCF in

ularizing the bias term we could obtain sub-optimal results. According to the modification done to the pri- mal formulation we also modify the dual formulation as:

*JD*(*α*) = − 1 *αT H*ˆ*α* + *αT* **1** *s.t.* 0 ≤ *αi* ≤ *C,* ∀*i* ∈

any of the considered applications even if with PCA(m=11) there is no a high performance degra- dation for the target application.

*Comparison*: The results achieved by the model trained with Z-normalization, *C* = 1 and no PCA are closed

2

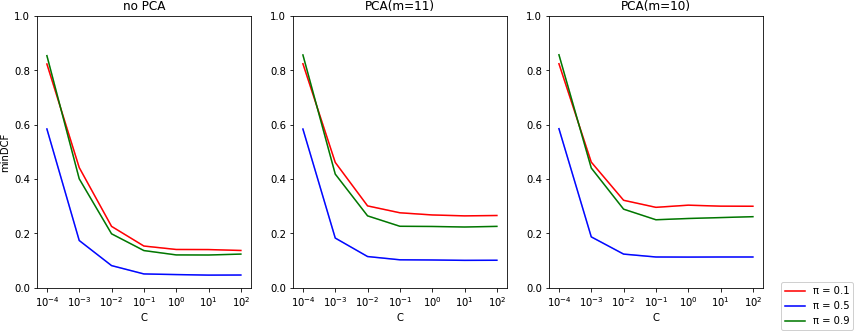
{1*, ..., n*}

where the equality constraint dissapeared (the one that L-BFGS was not able to incorporate) and the ma-

to the the results obtained by the Linear Logistic Re- gression Model but slighlty worse with respect to the results obtained by the Tied Gaussian model. Selected Linear SVM Model:

trix *H*ˆ can be computed as *H*ˆ*i,j* = *zizjx*ˆ*T x*ˆ*j* The

*i*



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

### Kernel SVM

SVMs allow for non-linear classification through an im- plicit expansion of the features in a higher dimensional space. In contrast with Quadratic Logistic Regression classifier we don’t have to compute an explicit expan- sion of the features space, it is sufficient to be able to compute the scalar product between the expanded fea-

**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*

tures: *k*(*x*1*, x*2) = *φ*(*x*1)*T φ*(*x*2) where *k* is the kernel

function. We have to replace *H*ˆ with *H*ˆ = *zizjk*(*z*1*, x*2). We are going to implement two types of kernel: poly- nomial and rbf.

* + - * Polynomial kernel of degree *d*:

*k*(*x*1*, x*2) = (*xT x*2 + *c*)*d*

1

* + - * Radial Basis Function kernel:

*k*(*x*1*, x*2) = *e*−*γ*||*x*1−*x*2||

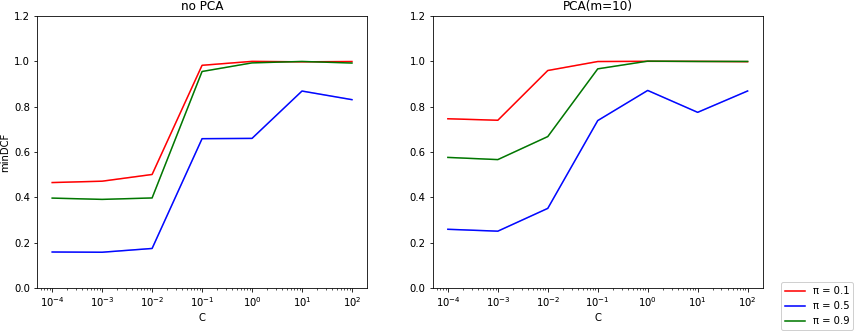
2

|  |  |  |  |
| --- | --- | --- | --- |
|  | *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**  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 |

only preprocessing step that has been considered is Z-normalization. As we can notice from figure 9 better

**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 hyperparam- eter *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 i 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 is able to achieve better re- sults 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

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

**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 |

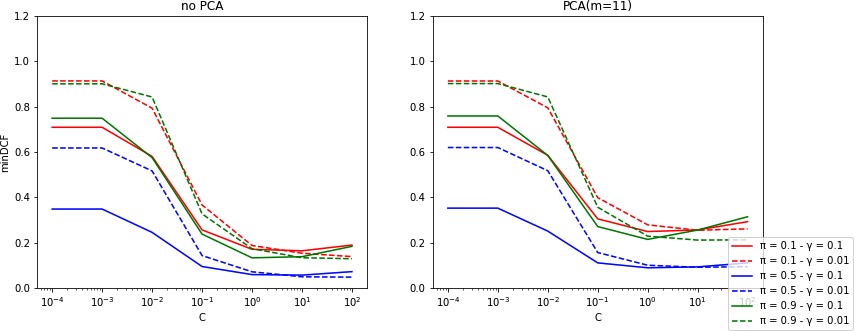
**Table 10:** *Best models analyzed up to now*

*minDCF* for the *π*˜ = 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.

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



**Figure 11:** *RBF SVM (K* = 1*, γ* 0*.*1*,* 0*.*01 *) - minDCF for different values of C*

again used Z-normalization as pre-processing step. The

∈ { }

* + - * 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 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

hyperparameter *γ* has to be tuned so we trained the model for different values of *γ* to analyze the perfor- mances and here are showed the results for *γ* = 0*.*1 and *γ* = 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. 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 also results obtained with *γ* = 1 appear for com- pleteness.

better results with respect to the two kernel ver- sions. 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

## GMM Classifier

The last model we are going to take into account 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:

*fX* (*x*) = Σ*K fX*|*C*(*x*|*c*) · *P* (*C* = *c*) =

then build a 2-componenents model starting from a single one and from each of the new componenets we generate other new 2 componenets and so on and so forth.

We are also going to introduce a Diagonal and Tied

*K*

*c*=1

Σ

*c*=1

N (*x*|*µc,* Σ*c*) · *πc*

version of the GMM problem: the diagonal variant con-

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

*X* ∼ *GMM* (*M,* S*, w*) ⇒ *fX* (*x*) =

Σ

sists 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 covari- ance of a single GMM Σ*c,g* = Σ*c* are the same but each

*M*

*g*=1

*wg* · N (*x*|*µg,* Σ*g*)

GMM of each class has a difference covariance matrix

Σ*c* (again, this is different from the Tied MVG model).

so the GMM density is the sum of *M* Gaussians where

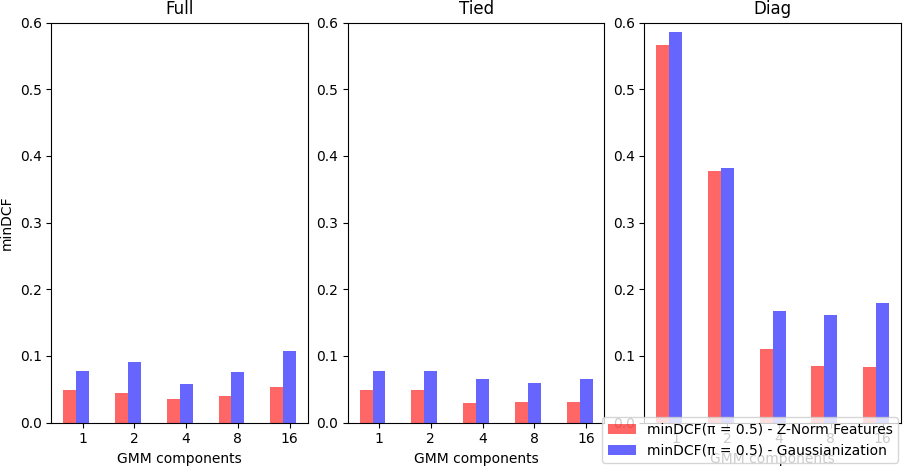
*M* = [*µ*1*, ..., µM* ]*,* = [Σ1*, ...,* Σ*M* ]*, w* = [*w*1*, ..., wM* ]

S

are the model parameters. Gaussian components can be seen as clusters the samples belog to (in a hard or in a soft way) and the cluster label is a latent random variable. If we define

*fXi,Gi* (*xi, g*) = *wg*N (*xi*|*µg,* Σ*g*) and

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 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. By considering

*fX* (*xi*) = Σ*M fX ,G* (*xi, g*)

*i*

*g*=1

*i*

*i*

we can introduce a term called *responsability* that rep- resents the posterior probability that a sample belongs to a certain cluster(component):

*γg,i*

= *P* (*Gi*

= *g*|*Xi*

= *x*) = *fXi,Gi* (*xi,g*) =

*fXi* (*xi*)

*wg* N (*xi*|*µg,*Σ*g* )

Σ

*g' wg'* N (*xi*|*µg' ,*Σ*g'* )

We can assign the sample to the cluster label for which the responsability 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:

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

only models trained with 8 compoents we are now going to show more in detail the results obtained when training with and without PCA.

*Fg* = Σ*N γg,ixi*, *Sg* = Σ *γg,ixixT* ,

*i*

*i*

*i*=1

* As showed in figure 12, for a relative small number

*i*=1

*N*

*Ng* = Σ*N γg,i*

of components (8) good results are achieved by

all the three types of models.

to obtain a re-estimate of *wg* = *Ng* . In this way we

will be able to apply the Expectation-Maximization

algorithm:

* E-step: estimate (given the model parameters

(*Mt,* S*t, wt*)):

*γg,i* = *P* (*Gi* = *g*|*Ci* = *x, Mt,* S*t, wt*)

* M-step: estimate the new model parameters by usign 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 al- gorithm to incrementally construct a GMM with 2*G* componenets 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

* Even if diag assumption doesn’t hold really well

especially when the number of components is low

it performs better with higher number of compo- nents.

* The Tied model is the one with best results even

if the Full model still performs really 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 improvements 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

**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 |

versions of the trained models). With Gaussian- ization and PCA(m=11) instead we can notice an improvement for *π*˜ = 0*.*9 application.

**Table 12:** *Best models analyzed up to now*

PCA

# Score Calibration

## 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)

## Calbrating Scores For Selected Models

We are going to transform the scores provided by each model so that the theoretical threshold *t* = *log*  *π*˜ provides close to optimal values over a wide rang1e−o*π*˜f 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:

*f* (*s*) = *αs* + *β*

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | *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** | |  |  |
| (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 |

*f* (*s*) = *log fS|C* (*s*|H*T* ) = *αs* + *β*

*fS|C* (*s*|H*F* )

and the class posterior probabilty for prior *π*˜ corre- sponds to:

*log P* (*C*=H*T* |*s*) = *αs* + *β* + *log*  *π*˜

*P* (*C*=H*F* |*s*) 1−*π*˜

The GMM Tied(8 components) trained with Z-

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 com- posed of the scores for a certain model). If we let:

*β*′ = *β* + *log*  *π*˜

1−*π*˜

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

Normalization and no PCA is the one that achieves the best results among all the models evaluated up to now.

Selected GMM Model:

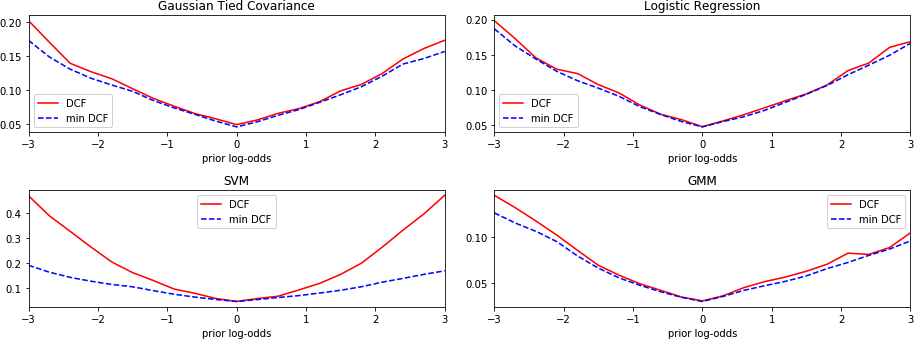
* GMM Tied (8 componenets), Z-Normalization, no

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:

*f* (*s*) = *αs* + *β* = *αs* + *β*′ *log*  *π*˜

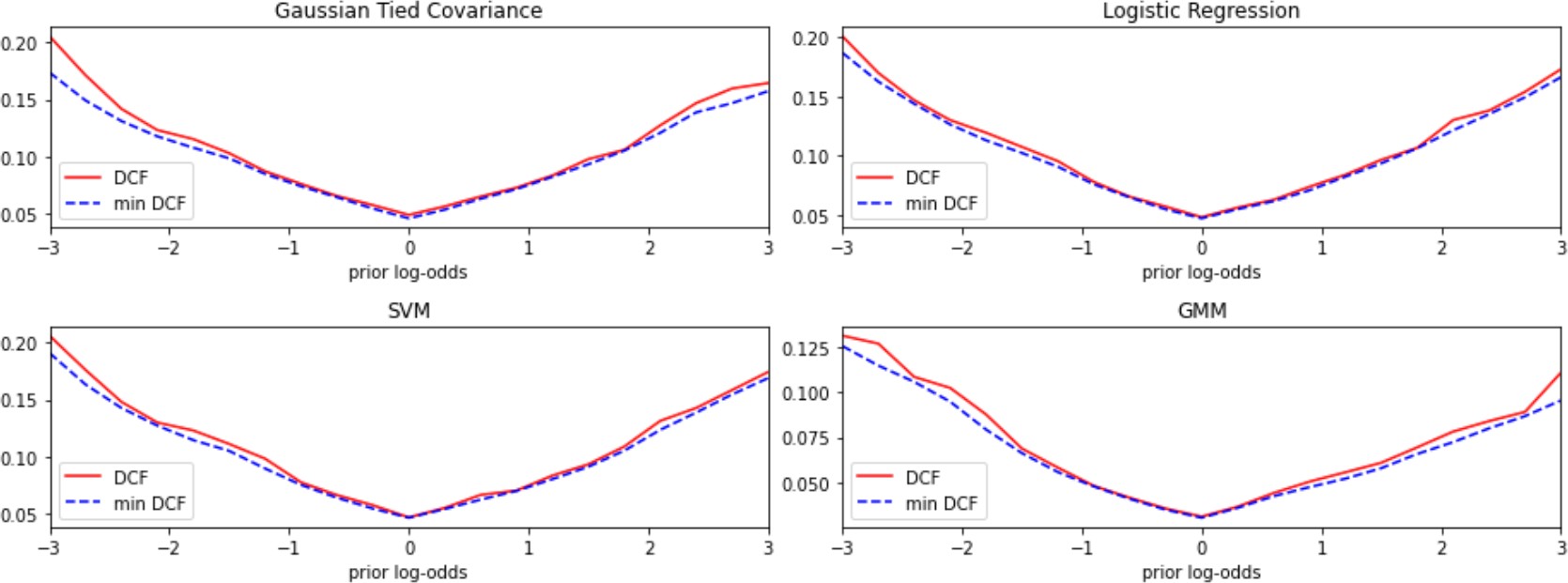
−

1−*π*˜



**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 in particular 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 values of the prior log-odds.



**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.

# 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 assest 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)*

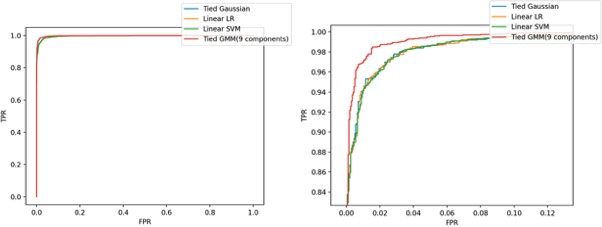
|  |  |  |  |
| --- | --- | --- | --- |
|  | *π*˜ = 0*.*1 | *π*˜ = 0*.*5 | *π*˜ = 0*.*9 |
| **Gaussian Models (Z-normalized features, no PCA)** | | | |
| Full Cov | 0.134 | 0.053 | 0.138 |
| Tied Cov | 0.133 | 0.051 | 0.135 |
| Naive Bayes | 0.810 | 0.570 | 0.882 |
| **LR Models (Z-Normalized features, 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 en- tire evaluation set are coherent with the results previ- ously 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 com- ponenets. The other three models also perform really well but slightly worse than this.



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

# 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 (despite the fact that with PCA(m=11) we have obtained not a big performance degratation in most of the models) and Naive hypothesis didn’t worked well in Gaussian models. We have been able to reach a *minDCF* 0*.*03 for the target application(*π*˜ = 0*.*5) but also for the unbalanced applications we were able to achieve good results (*minDCF* 0*.*1 for *π*˜ = 0*.*1 and *minDCF* 0*.*1 for *π*˜ = 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.

≈

≈

≈