Classification of the HTRU2 dataset

by Tommaso Quario and Francesco Carlucci

The dataset

HTRU2 is a data set which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey.

Pulsars are a rare type of Neutron star that produce radio emission detectable here on Earth. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter. As pulsars rotate, their emission beam sweeps across the sky, and when this crosses our line of sight, produces a detectable pattern of broadband radio emission. As pulsars rotate rapidly, this pattern repeats periodically. Thus pulsar search involves looking for periodic radio signals with large radio telescopes.

Each pulsar produces a slightly different emission pattern, which varies slightly with each rotation. Thus a potential signal detection known as a 'candidate', is averaged over many rotations of the pulsar, as determined by the length of an observation. In the absence of additional info, each candidate could potentially describe a real pulsar. However in practice almost all detections are caused by radio frequency interference (RFI) and noise, making legitimate signals hard to find.

Machine learning tools are now being used to automatically label pulsar candidates to facilitate rapid analysis. Classification systems in particular are being widely adopted, which treat the candidate data sets as binary classification problems. Here the legitimate pulsar examples are a minority positive class, and spurious examples the majority negative class. At present multi-class labels are unavailable, given the costs associated with data annotation.

The dataset contains 16,259 spurious examples caused by RFI/noise, and 1,639 real pulsar examples. These examples have all been checked by human annotators. The dataset has been split into Train and Evaluation (Test) data. The training set is stored in the file ”Train.txt” and contains 8108 false pulsar signals and 821 true pulsar signals. The evaluation set is stored in the file ”Test.txt” and contains 8151 false pulsar signals and 818 true pulsar signals. Candidates are stored in both files in separate rows. Each row lists the variables first, and the class label is the final entry. The class labels used are 0 (false pulsar signal) and 1 (true pulsar signal). Our goals will be to perform classification for the proposed task by employing different models, to analyze results and to motivate our choices.

Attribute Information

Each candidate is described by 8 continuous variables, and a single class variable. The first four are simple statistics obtained from the integrated pulse profile (folded profile). This is an array of continuous variables that describe a longitude-resolved version of the signal that has been averaged in both time and frequency. The remaining four variables are similarly obtained from the DM-SNR curve. These are summarized below:

1. Mean of the integrated profile.

2. Standard deviation of the integrated profile.

3. Excess kurtosis of the integrated profile.

4. Skewness of the integrated profile.

5. Mean of the DM-SNR curve.

6. Standard deviation of the DM-SNR curve.

7. Excess kurtosis of the DM-SNR curve.

8. Skewness of the DM-SNR curve.

9. Class

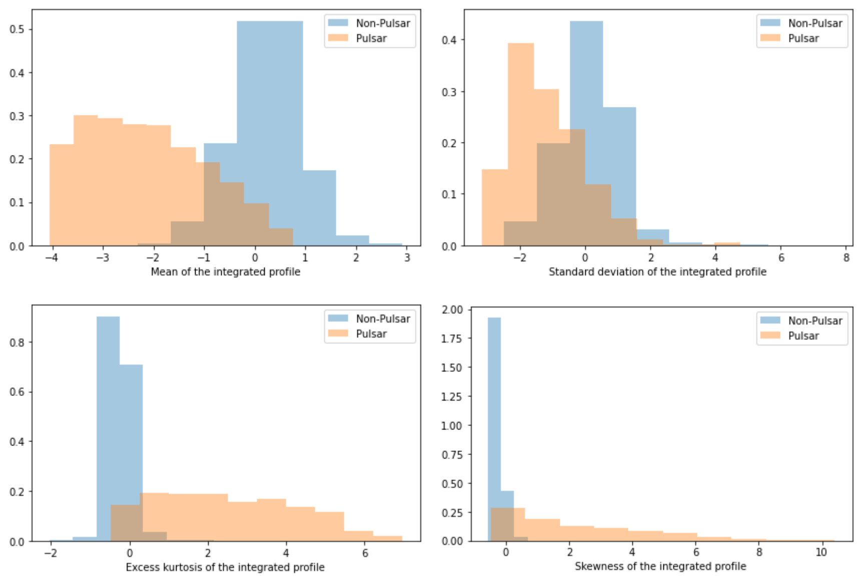
Pre processing

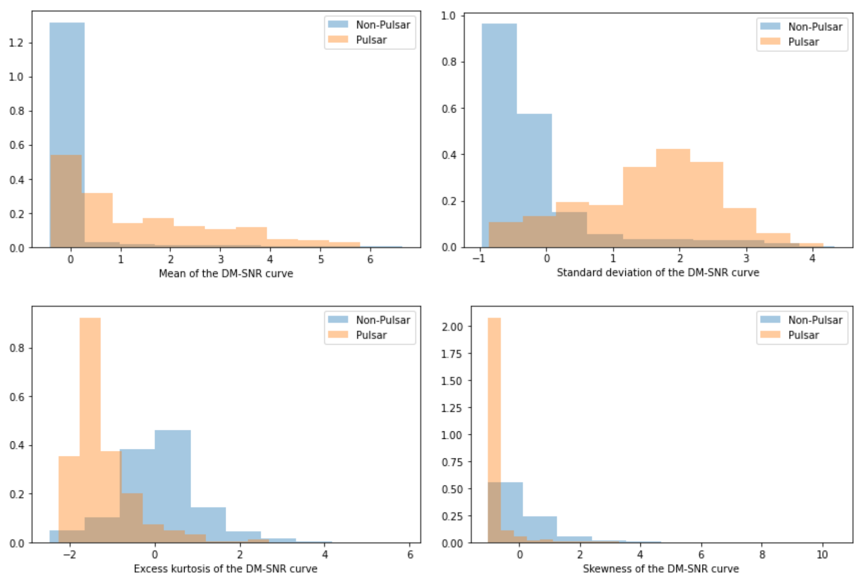
Since the values of our data are high, we decided to apply z normalization. This allows us to reduce risks of overflow and simplifies our work.

$$z_i = \frac{(x_i - \mu)}{\sigma} \qquad \forall\, i \in 1..n$$

where x is the observed values of our i-th sample, mu is our mean vector computed along the columns and sigma is our standard deviation vector computed along the columns.

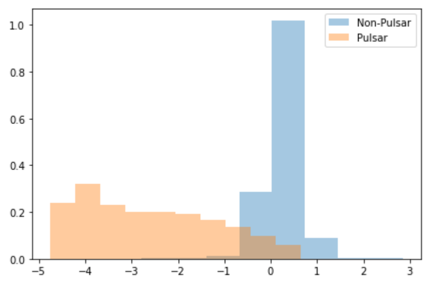
By using this procedure we center and scale our data so that our Z-normalized random variable Z will have a standard distribution with zero mean and one variance.





Histograms of the various dimensions of the dataset

If we apply Linear Discriminant Analysis and plot our dataset we notice that our data is easily separable even when reduced to one dimension, we therefore expect low error rates.



The single dimension after LDA

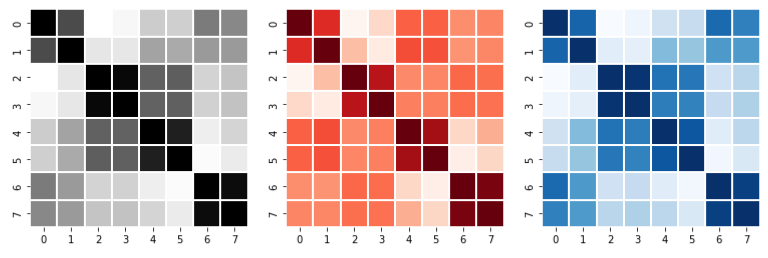
After the z normalization our features seem well distributed and lacking significant outliers.

Since this is the case we decided not to pre-process the data any further.

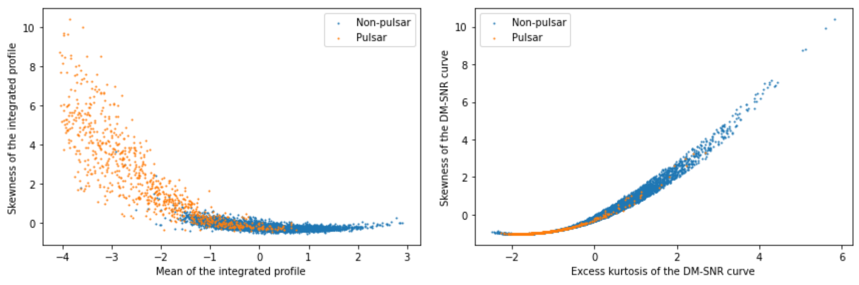
We can also analyze feature correlation by using heatmaps and scatter plots. The heatmaps show the Pearson correlation coefficient:

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

while the scatter plots simply show one dimension of our data plotted against another.



Gray: whole Z-normalized training set. Red: false pulsar signal samples. Blue: true pulsar signal samples. Darker color implies larger value for the Pearson correlation coefficient



Two scatter plots of the features of the dataset. Scatter plots for all combinations of dimensions can be found in the code folder

From the heatmaps we notice that features 2-3, 6-7 and 4-5 are correlated to a certain extent, we therefore theorize that dimensionality reduction techniques could be of use.

The method that we will try is Principal Component Analysis (PCA).

PCA is a dimensionality reduction technique that computes a linear mapping from the n-dimensional feature space to a m-dimensional space, with m n, while preserving the directions with highest variance. It is unsupervised.

We'll exploit PCA to reduce the training set dimensionality up to 5 and see how models behave.

To analyze our dataset we will employ a K-fold cross validation with K=3, this should prove more robust than simply using a subset of the dataset as a validation set. The data is shuffled before splitting.

We will also consider three different applications: a uniform prior application and two unbalanced applications where the prior is biased towards one of the two classes:

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

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

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

Our target application will be the balanced one. We want to find the most promising approach, so we can measure performances through the metric of the normalized minimum Detection Cost Function (min DCF), which measures the cost that we would pay if we made optimal decisions using the recognizer scores. We evaluate performances on the validation subset (extracted from the training set).

Gaussian classifiers

We start considering Gaussian classifiers (MVG classifier, MVG classifier with Naive Bayes assumption, MVG classifier with tied covariance and MVG classifier with Naive Bayes assumption and tied covariance). All of them assume gaussian distributed data, given the class:

$$X|C = c \simeq N(\mu_c, \Sigma_c)$$

The MVG classifier with tied covariance matrices assumes that each class has its own mean μc, but the covariance matrix Σ is the same for all classes.

The Naive Bayes Gaussian classifier corresponds to a MVG classifier with diagonal covariance matrices for the different classes. The Naive Bayes assumption supposes that features are independently distributed and, if this is the case, then the features are also uncorrelated and will have zero-covariances, which are the elements off-diagonal on the covariance matrices and this justify the diagonalization. Since some of our features are highly correlated, we expect covariances off-diagonal not to be approximately zero, so we have no guarantee that the Naive Bayes assumption will actually work out well, we may get not-optimal results.

Instead, the MVG classifier with full covariance matrices and the MVG classifier with tied covariance matrices are able to capture correlations and may output better results, also considering the sufficient number of samples at our disposal.

no PCA	$\pi = 0,5$	$\pi = 0,9$	$\pi = 0,1$
Full-Cov	0.141	0.661	0.285
Diag-Cov	0.191	0.732	0.312
Tied-Cov	0.111	0.572	0.223
Diag-Tied-Cov	0.160	0.580	0.265

PCA m=7	$\pi = 0,5$	$\pi = 0,9$	$\pi = 0,1$
Full-Cov	0.139	0.631	0.301
Diag-Cov	0.213	0.720	0.504
Tied-Cov	0.111	0.569	0.223
Diag-Tied-Cov	0.138	0.591	0.269

PCA m=6	$\pi = 0,5$	$\pi = 0,9$	$\pi = 0,1$
Full-Cov	0.151	0.631	0.286
Diag-Cov	0.222	0.721	0.530
Tied-Cov	0.138	0.578	0.256
Diag-Tied-Cov	0.163	0.595	0.299

Tables: the min DCFs on the 3-fold validation

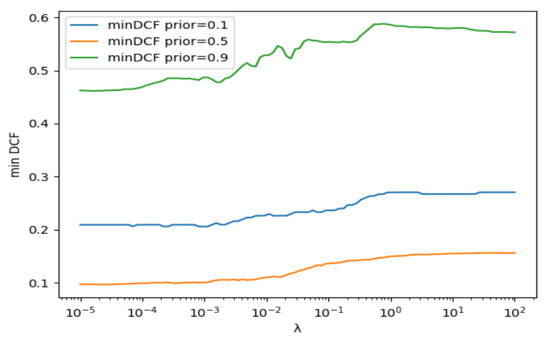
As we can see the best results are obtained through the gaussian classifier with tied covariance matrices. It is also clear that all the models perform worse for the imbalanced tasks. By applying PCA with m = 7 we obtain almost the same values for the min DCFs that we get by employing the models without PCA. This proves that dimensionality reduction can be exploited without losing useful information, as we expected from the correlation analysis. Instead, starting from m = 6, results degrade, so in the following we'll mainly refer to models with m = 7 and without PCA.

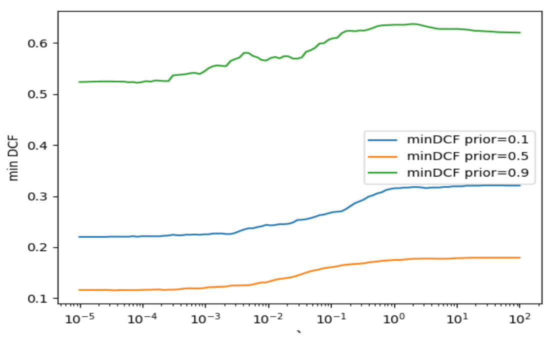
LOGISTIC REGRESSION

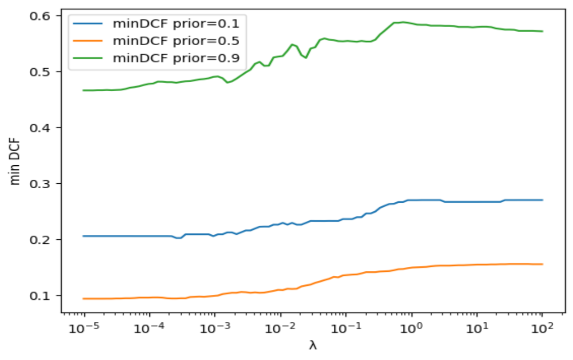
We will now consider the regularized linear logistic regression model. As our dataset is highly unbalanced we have to rebalance the costs of tìeach class modifying the objective function:

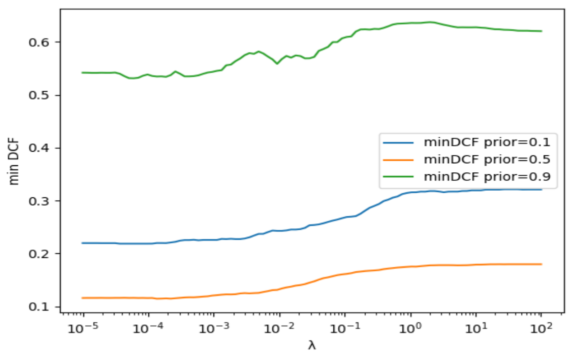
$$J(\boldsymbol{w}, b) = \frac{\lambda}{2}||\boldsymbol{w}||^2 + \frac{\pi_T}{n_T}\sum_{i=1|c_i=1}^{n} \log\left(1+e^{-z_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)}\right) + \frac{1-\pi_T}{n_F}\sum_{i=1|c_i=0}^{n} \log\left(1+e^{-z_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)}\right)$$

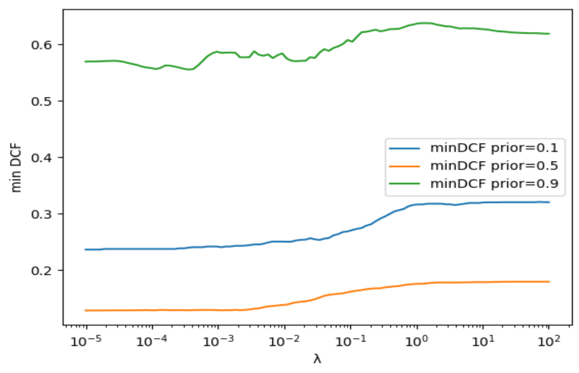
We have used a π_T=0,5. The parameters of the model w, b are optimized with the l-bfgs algorithm. The posterior log-likelihood ratio is $s_i = w^T x_i + b$. The model's decision rules $(s > t)$ are linear hyperplanes orthogonal to the vector of the weights. $\frac{\lambda}{2}||w||^2$ is the regularization term that reduces the risk of overfitting. In order to estimate the hyperparameter λ we plot the min DCF using both the kfold (left) and single split approach. The Graphs shows results in order without PCA (top) with PCA m=7 (center) and with PCA (m=6).

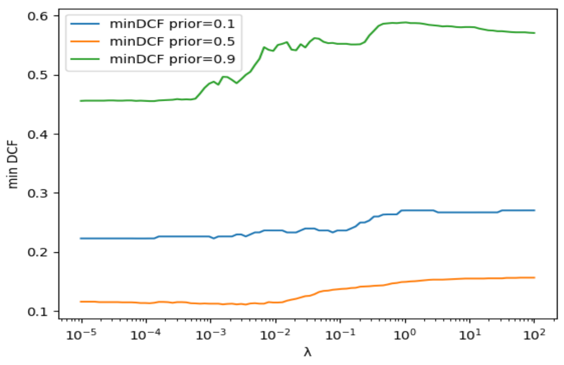












The 3 plots says us that regularization is not beneficial on the HTRU2 dataset, so we will use $\lambda=10^{-4}$ from now on. In the following tables we have computed the minDCF with regard to different prior π_T values (0,1 0,5 and 0,9) with no PCA, PCA 7 and PCA 6.

Logistic Regression $\lambda = 10^{-4}$ PCA (m=6)

Classifier	π=**0.5**	π=**0.9**	π=**0.1**
π_t=0.1	0.129	0.600	0.244
π_t=0.5	0.129	0.558	0.238
π_t=0.9	0.139	0.552	0.238

Logistic Regression $\lambda = 10^{-4}$ no PCA

Classifier	π=**0.5**	π=**0.9**	π=**0.1**
π_t=0.1	0.115	0.549	0.214
π_t=0.5	0.116	0.523	0.221
π_t=0.9	0.123	0.520	0.223

ogistic Regression $\lambda = 10^{-4}$ PCA (m=7)

Classifier	π=**0.5**	π=**0.9**	π=**0.1**
π_t=0.1	0.116	0.550	0.215
π_t=0.5	0.115	0.538	0.218
π_t=0.9	0.120	0.538	0.220

We obtain the best results for the prior we are aiming at (0,5) if we perform a dimensionality reduction with PCA=7.

Support Vector Machines

We now turn our attention to SVMs. In our analysis we will consider the linear SVM, the polynomial quadratic kernel and the Radial Basis Function kernel formulations. From the previous results we can expect linear classifiers, and therefore the linear SVM. to be more effective.

We start considering the unbalanced linear model. To solve the SVM problem, we can consider the dual formulation:

$$J^D(\alpha) = -\frac{1}{2}\alpha^T H\alpha + \alpha^T 1 \qquad 0 \le \alpha_i \le C \qquad \forall\, i \in \{1,..,n\} \qquad \Sigma^n_{i=1}\ \alpha_i z_i = 0$$

where C is a hyper-parameter, n is the number of training samples, 1 is a n-dimensional vector of ones, z_i is the class label for the i-th sample encoded as +1, if xi belongs to class 1 (true pulsar signal), or −1, if xi belongs to class 0 (false pulsar signal).

And H is a matrix whose elements are:

$$H_{i,j} = z_i z_j x_i^T x_j$$

From the dual formulation we can then derive the primal solution. Since we make use of the L-BFGS-B algorithm and it can only handle box constraints, we have to modify the formulation to include the second constraint. We have that the modified dual formulation becomes:

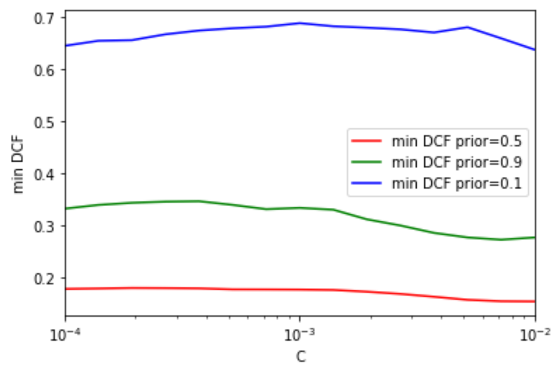
$$\widehat{J}^D(\alpha) = -\frac{1}{2}\alpha^T H\alpha + \alpha^T 1 \qquad 0 \le \alpha_i \le C \qquad \forall\, i \in \{1,..,n\}$$

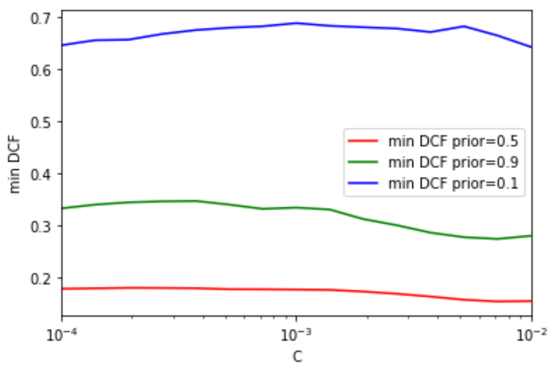
and, since we use the mapping $\widehat{x}_i = [x_i, K]$ with K=1, the matrix $\widehat{H}$ is modified accordingly:

$$\widehat{H}_{i,j} = z_i z_j (x_i^T x_j + 1)$$

We resort to 3-folds cross-validation to choose the best value for the hyper-parameter C.

The following plots are focused on the values of C that proved more effective.





Linear SVM: Left: no PCA, Right: PCA m=7.

Given the previous graph, we choose C = 10e−2 since it seems to provide a lower min DCF.

Next we analyze the non-linear formulations. In SVM, non-linearity is obtained through an implicit expansion of the features in a higher dimensional space. The dual SVM formulation depends on the training samples through the dot product and it's possible to compute scores through scalar products between training and evaluation samples. For this reason, it's not required to explicitly compute the feature expansion, it's enough to be able to compute the scalar product between the expanded features, the so called kernel function k:

$$k(x_i, x_j) = \Phi(x_i^T)\Phi(x_j)$$

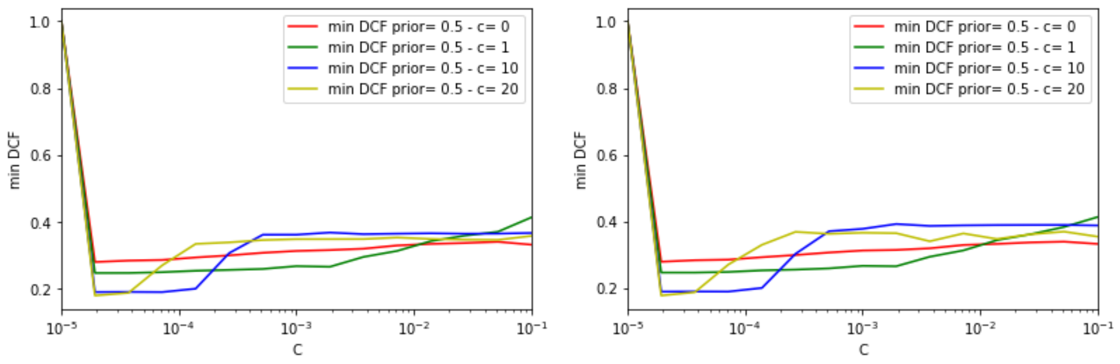
We will employ two different kernels: polynomial kernel of degree d = 2

$$k(x_i, x_j) = (x_i^T x_j + c)^d$$

and Radial Basis Function (RBF) kernel.

$$k(x_i, x_j) = e^{-\gamma||x_i - x_j||^2}$$

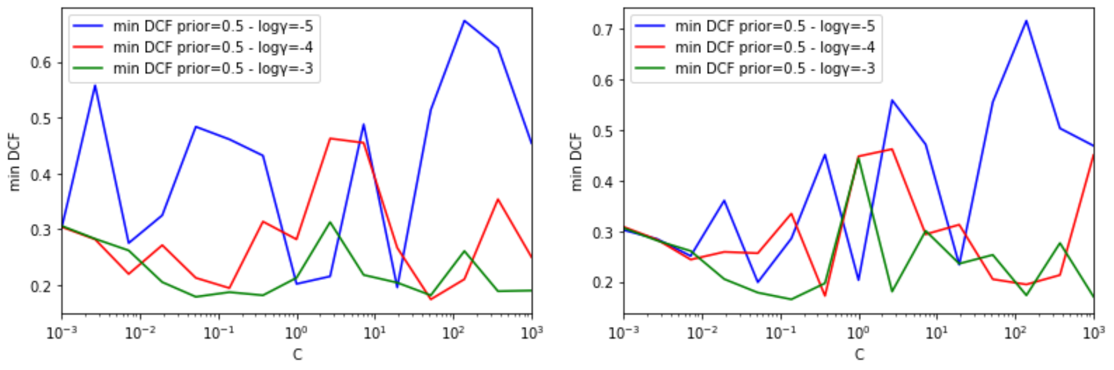
We first focus on the quadratic polynomial kernel. We have to jointly estimate the parameters C and c, so we perform a grid search with 3-folds cross-validation approach on our target application.



Polynomial SVM Quadratic Kernel: Left: no PCA, Right: PCA m=7.

As shown in the line charts, the choice of the two hyper-parameters appears to be critical. Our analysis leads us to the choice of c = 10 and C = 5 ∗ 10−5 .

We now employ RBF kernel formulation. We need to jointly estimate the two hyper-parameters γ and C. We again resort to a grid search with 3-folds cross validation approach on our main application.



SVM RBF Kernel: Left: no PCA, Right: PCA m=7.

The choice of the two hyper-parameters is again difficult, as we can see from Figure 11. The adopted values, in the end, are γ = 10−3 and C = 10−1 .

no PCA	$\pi = 0,5$	$\pi = 0,9$	$\pi = 0,1$
Linear SVM C=10^{-2}	0.154	0.659	0.273
Polynomial SVM c=10 C=$5 * 10^{-5}$	0.178	0.680	0.345
RBF SVM C=10^{-1}	0.172	0.732	0.257

$\gamma = 10^{-3}$			

PCA m=7	$\pi = 0,5$	$\pi = 0,9$	$\pi = 0,1$
Linear SVM C=10^{-2}	0.154	0.664	0.274
Polynomial SVM c=10 C=$5 * 10^{-5}$	0.187	0.669	0.340
RBF SVM C=10^{-1} $\gamma = 10^{-3}$	0.172	0.712	0.256

Table with min DCFs for linear, polynomial and RBF kernels. Calculated using estimated parameters

As we can see, in our target application with πe = 0.5 the linear model outperforms the quadratic ones. It is also interesting to notice that the PCA with m=7 does not affect the performance of the classifier at all.

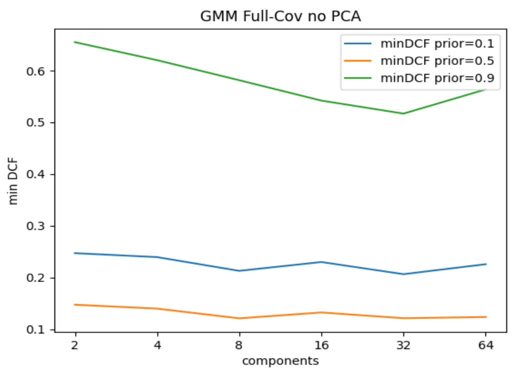
The hyper-parameter C works as a penalty for each misclassified sample. The smaller C is, the smaller the penalty will be, so a decision boundary with a large margin is chosen at the cost of a greater number of misclassifications. If instead C is chosen large, the SVM tries to minimize the number of misclassified samples and this results in a decision boundary with a smaller margin. For the linear SVM, C = 10^{-2} and this allows a larger margin for the separation rule. For the quadratic SVM and RBF SVM, the chosen values of C are in the lower range of the values tested. This suggests that performance improves with a larger margin for the separation rule. For quadratic SVM and RBF SVM, larger margins correspond to less complex separation rules. Instead for high values of C separation rules tend to have more complex shapes. Since linear models seem to perform better on this dataset, this explains why we obtain better results with small values of C.

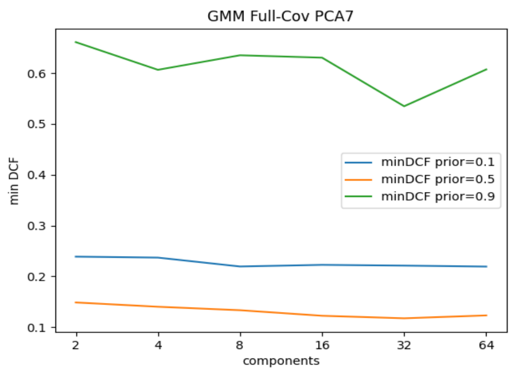
GMM

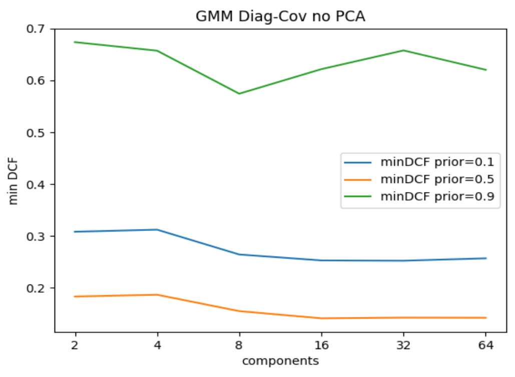
We are now considering the Gaussian Mixture Model. GMM approximate data distribution through a weighted sum of gaussian distributions, the density of the random variable of the samples is:

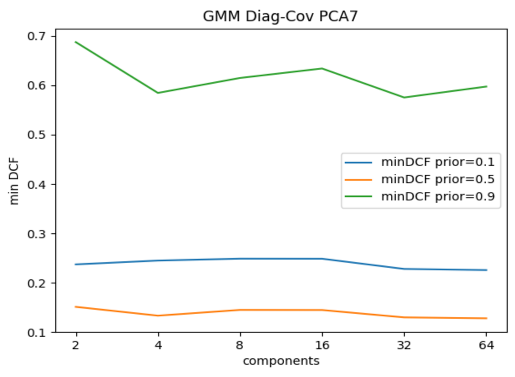
$$X \sim GMM(M, S, w) \Rightarrow f_X(x) = \sum_{g=1}^{M} w_g N(x|\mu_g, \Sigma_g)$$

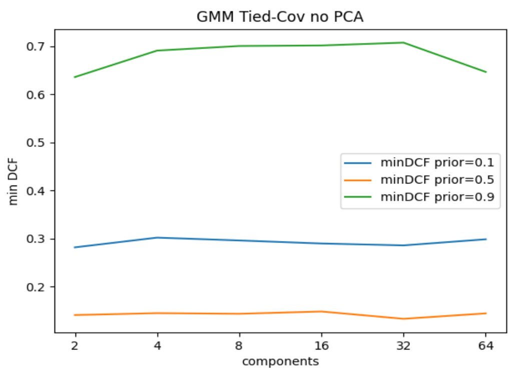
where $f_X(x)$ is the marginal distribution, the sum performs the marginalization of the joint densities of each gaussian. We can build a classifier by training a GMM for each class.
The number of gaussians of the GMM is a hyperparameter, we estimate the best one in the plot below, considering full covariance model, diagonal covariance model and tied covariance model. Tied covariance model GMMs are trained independently on the 2 classes, so tying is performed only at the class level, unlike tied MVG. We have again used the 3-fold approach and estimated both with or without PCA.

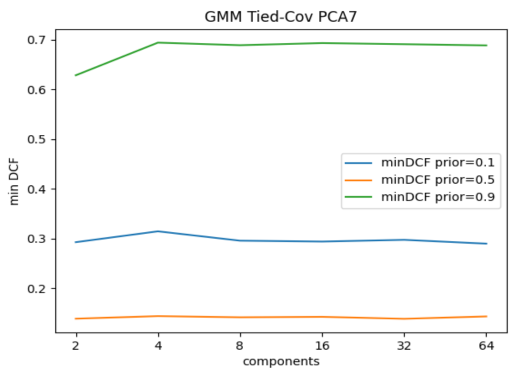












The application with prior=0.9 shows strong oscillations and the minimum DCF tends to increase with the increasing of the number of components, this is a sign of overfitting. The other 2 applications oscillate less. For our main application (prior=0.5) we choose the Full covariance 8 components model without PCA.

Table of min DCF GMM 8 Components 3-folds NO PCA

	0,1	0,5	0,9
GMM Full Cov	0,213	0,121	0,582
GMM Diag Cov	0,264	0,155	0,574
GMM Tied Cov	0,296	0,144	0,700

Table of min DCF GMM 8 Components 3-folds PCA 7

	0,1	0,5	0,9
GMM Full Cov	0,220	0,134	0,635
GMM Diag Cov	0,249	0,145	0,615
GMM Tied Cov	0,296	0,144	0,688

Below we summarize the minimum DCF obtained for the selected models

Best Models min DCF

	0,1	0,5	0,9
MVG Tied Cov	0,224	0,112	0,569
Logistic Regression (λ=10^{-4},π_T=0.5)	0,218	0,115	0,538
GMM Full Cov 8 components NO PCA	0,213	0,121	0,582

The three models perform similarly over the 3 applications, so we will evaluate all the three from now on.

Calibration

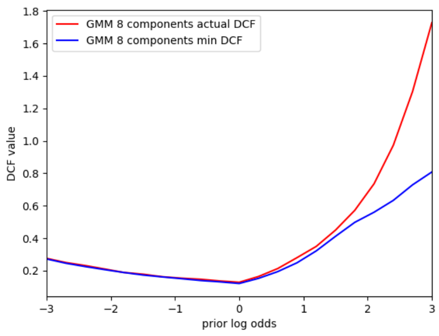
We will now evaluate if the scores of our best models are well calibrated. To do so we'll compute their actual DCF, that is the cost we would actually pay if we used the theoretical threshold $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$, and compare it with the minimum DCF previously calculated.

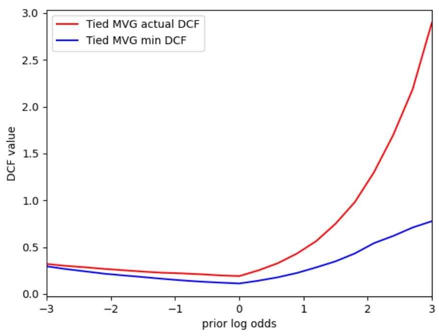
min-DCF and actual DCF comparison for the best models, 3-folds

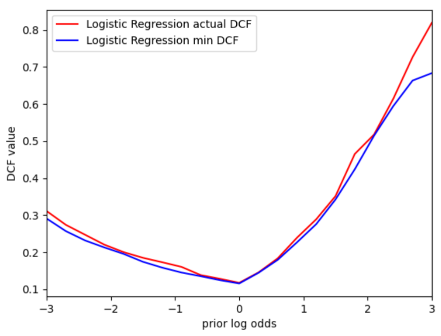
	0,1		0,5		0,9	
	min-DCF	act-DCF	min-DCF	act-DCF	min-DCF	act-DCF
Tied-Cov MVG	0,224	0,274	0,112	0,191	0,569	1,422
Logistic Regression (λ=10^{-4},π_T=0.5)	0,218	0,227	0,115	0,117	0,538	0,545
GMM Full Cov (8 components, no PCA)	0,213	0,215	0,121	0,128	0,582	0,809

We can see that the logistic regression model is already quite well calibrated, especially for the 0,5 application, while the other models need to be calibrated. We can see this also in the Bayes Error Plots.

Bayes Error plot, 3-folds approach, first two with PCA (7 dimensions).







Logistic Regression is well calibrated for almost all the applications, while the tied cov MVG needs calibration the most. To calibrate the models we could choose an optimal threshold, like the one that results in the min DCF, but then we would have to define a different threshold for each needed application. We can instead recalibrate the scores by finding a function that maps the scores to calibrated scores. If we assume the function to be linear

$f = \alpha s + \beta$

the problem becomes equivalent to training a logistic regression model that uses previous models scores as features.

This model will give the class posterior probabilities,

$$log\frac{P(C=Ht|s)}{P(C=Hf|s)} = \alpha s + \beta'$$

where α e β' are the weight and biases of the logistic regression. We will then retrieve log-likelihoods by subtracting our application prior π :

$$f(s) = log\frac{f_{S|C}(s|Ht)}{f_{S|!C}(s|Hf)} = \alpha s + \beta' - log\frac{\pi}{1-\pi}$$

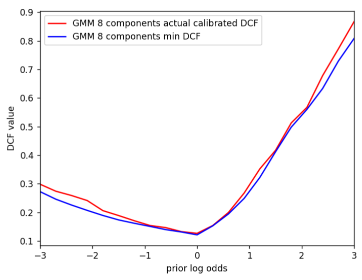
We have applied a logistic regression model with $\lambda=10^{-4}$.

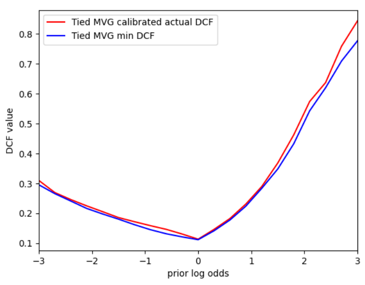
min-DCF and actual DCF comparison on calibrated scores, 3-folds

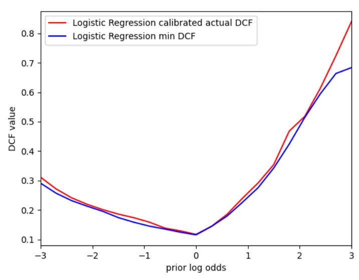
	0,1		0,5		0,9	
	min-DCF	act-DCF	min-DCF	act-DCF	min-DCF	act-DCF
Tied-Cov MVG	0,224	0,229	0,112	0,114	0,569	0,600
Logistic Regression ($\lambda=10^{-4}$,π_T=0.5)	0,218	0,226	0,115	0,117	0,538	0,545
GMM Full Cov (8 components, no PCA)	0,213	0,247	0,121	0,126	0,582	0,596

Score calibration had no practical effects on logistic regression, as it was already well calibrated, but improved the other two models, most of all for the 0,9 application. We can see that also in the calibrated bayes error plot.

Calibrated Bayes error Plot, 3-folds, first two with PCA (7 dimensions)







Now all the three models are well calibrated, even for applications with different priors, although we can notice from both the table and the graph that the calibration of GMM actually worsened for low prior application, like 0,1.

Evaluation

Finally, we can evaluate all the considered models on the HTRU2 evaluation set,in order to check if the assumptions we made, based on the training set, holds for real data. All the selected models are trained using k-fold approach, so we can now evaluate models trained on the whole training set. We evaluate all the models with the min DCF metric.

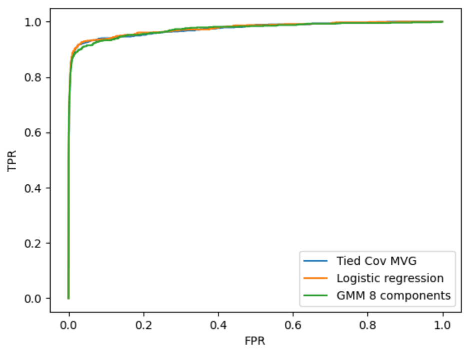
Evaluation of all the models on test set, min DCF values,no PCA

Classifier	π=**0.5**	π=**0.9**	π=**0.1**
Full Cov MVG:	0.140	0.646	0.283
Diag Cov MVG:	0.185	0.621	0.330
Tied Cov MVG:	0.110	0.591	0.207
Tied Diag Cov MVG:	0.152	0.544	0.262
Logistic Regression:	0.110	0.534	0.199
GMM Full 2 components:	0.156	0.764	0.256
GMM Full 4 components:	0.142	0.642	0.254
GMM Full 8 components:	0.122	0.598	0.218
GMM Full 16 components:	0.128	0.523	0.220
GMM Diag 2 components:	0.173	0.640	0.317
GMM Diag 4 components:	0.192	0.589	0.341
GMM Diag 8 components:	0.134	0.561	0.254
GMM Diag 16 components:	0.140	0.589	0.257
GMM Tied 2 components:	0.139	0.580	0.276
GMM Tied 4 components:	0.141	0.584	0.314
GMM Tied 8 components:	0.147	0.641	0.300
GMM Tied 16 components:	0.140	0.625	0.307
Linear SVM:	0.148	0.603	0.292
Quadratic kernel SVM:	0.183	0.613	0.355
RBF kernel SVM:	0.739	1.015	0.987

Evaluation of all the models on test set, min DCF values, with PCA (m=7)

Classifier	π=**0.5**	π=**0.9**	π=**0.1**
Full Cov MVG:	0.139	0.574	0.293
Diag Cov MVG:	0.201	0.755	0.514
Tied Cov MVG:	0.110	0.587	0.208
Tied Diag Cov MVG:	0.141	0.556	0.257
Logistic Regression:	0.109	0.538	0.202
GMM Full 2 components:	0.155	0.729	0.261
GMM Full 4 components:	0.141	0.613	0.250
GMM Full 8 components:	0.117	0.496	0.226
GMM Full 16 components:	0.118	0.495	0.215
GMM Diag 2 components:	0.159	0.662	0.249
GMM Diag 4 components:	0.122	0.571	0.252
GMM Diag 8 components:	0.127	0.563	0.247
GMM Diag 16 components:	0.123	0.532	0.234
GMM Tied 2 components:	0.138	0.567	0.278
GMM Tied 4 components:	0.143	0.645	0.310
GMM Tied 8 components:	0.141	0.622	0.301
GMM Tied 16 components:	0.137	0.595	0.326
Linear SVM:	0.148	0.603	0.295
Quadratic kernel SVM:	0.184	0.613	0.356
RBF kernel SVM:	0.753	1.013	0.989

Results in the table are consistent with our expectations, the best model are the three we have selected before, with logistic regression and Tied Cov MVG slightly better than GMM. This means that the test population has indeed the same distribution of our training set.
We can now compare our models after calibration. We can do that with the ROC plot below.



From the plot we can see that all the models have the same AUC (Area under the curve), the high slope for low value of false positive rate means that our classifiers detect a good amount of true samples doing little false positive errors.

Conclusions

Our analysis shows that linear classifiers perform better than any quadratic ones on this dataset. We obtained 0.110 min DCF with two different models, Logistic Regression and tied MVG, for the application we aimed at, prior=0,5. PCA can be applied with m=7 without losing any quality, and is therefore viable.
Using priors of 0.1 or 0.9 we still obtain acceptable results of 0.2 and 0.6 DCF respectively.