Pulsar Project Report

Alex Carluccio (S302373)
Luigi Federico (S295740)

Abstract

In this report we will analyze how effective are different classifiers and different preprocessing techniques applied to a binary classification task. The dataset used to train and test the models is the HTRU2 Data Set, provided by the University of Manchester, describing a sample of pulsar candidates collected during the High Time Resolution Universe Survey. The purpose of this report is not to describe the best possible application, but to make an analysis of the various techniques presented during the course.

Introduction

Pulsars are a rare type of Neutron star that produce radio emissions detectable here on Earth. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter.

Pulsar Features

Each candidate, that can be a Pulsar or not, is described by 8 continuous variables, and a single class lable. The first four are simple statistics obtained from the integrated pulse profile. The remaining four variables are similarly obtained from the DM-SNR curve . These are summarised 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 of the DM-SNR curve
8. Skewness of the DM-SNR curve
9. Class Label

Histograms of the HTRU2 dataset features (training set) are shown in the following. Features are sorted by the order used to describe them, Non-Pulsar are represented by the blue color while Pulsar by the red color.

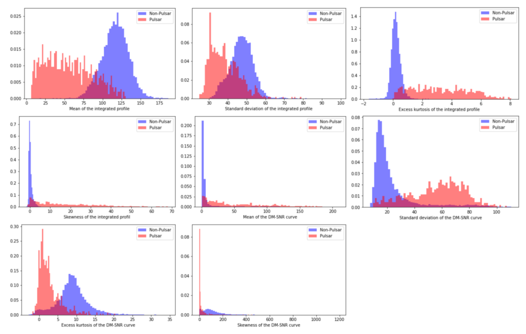

Figure 1: Histograms Plots Training Data

A preliminary analysis of the graphs shows us that, in some cases, the raw features have irregular distributions and are characterized by a presence of outliers (especially for plots of features 4, 5, 7 and 8). We expect that the classification approaches may produce a non-optimal result and for this reason it is better to pre-process our data. We therefore pre-process the data using the "Gaussianization" method. However, we also want to analyze the raw data to see how they perform compared to the normalized ones, although we expect the metric results to vary based on the amount of samples available. Assuming that our features have their empirical cumulative distribution function well approximated by a Gaussian c.d.f. , the Gaussianization procedure allows mapping the features to a uniform distribution and then trasforming the mapped features through the percent point function (the inverse of the cumulative distribution function).

The histograms of the modified HTRU2 dataset features (always for training set) after Gaussianization are shown in the following (Fig. 2). Also in this case the plots are sorted by the order used to describe the corresponding feature. Non-Pulsar are represented by the blue color and Pulsar by the red color.

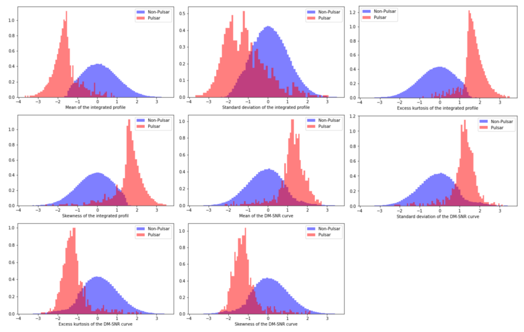

Figure 2: Histogram Plot - Gaussianized Training Data

A correlation analysis of Gaussianized features done by means of an heat map could bring us important informations. The heat map shows the absolute value of the Pearson correlation coefficient:

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

As it is possible to see from Fig.3, features group (5, 6, 7, 8) have very high correlation values for all the combinations and also the couples (3, 1) and (0, 2) are correlated.

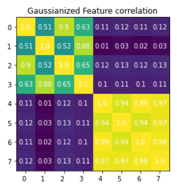

Figure 3: HeatMap of Gaussianized Features

This suggest we may have benefits using PCA to reduce the size of the dataset.

Classification

For our analysis we will train the classifiers by means of K-fold cross-validation approach (K = 5), shuffling the data before splitting them. We chose this approach because it should be more robust then the single-split approach, since it allow us to re-train the models over the whole dataset. We evaluate performances on the validation subset.

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

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

Our target application will be the balanced one.

In order to find the most promising classifier, we will mesure performances through the normalized minimum Detection Cost Function (min DCF), which is a metric that measures the cost we would pay if we made optimal decisions using the recognizer scores.

Gaussian Classifier

We start considering the following Gaussian classifiers: MVG classifier with and without tied covariance matrices, both with and without the Naive Bayes assumption. Each of these assumes that the data has a gaussian distribution, i.e.

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

The MVG classifier with tied covariance matrices assumes that the covariance matrix Σ is the same for all classes, which have their own μ_c. The Naive Bayes assumption presuppose that features are indipendently distributed and uncorrelated with each other. The MVG classifiers with the Naive Bayes assumption have diagonal covariance matrices for each class.

As we observed from the heat map, our feature are highly correlated. Therefore, we expect to get a not-optimal result with the diagonal covariance matrices since the Naive Bayes assumption seems to be inconsistent in this dataset. Instead, considering the high number of samples at our desposal, the MVG classifiers with full covariance, tied or not, may output promising results since they are able to capture correlation. In the following table we will analyze the min DCFs by different MVG Classifiers.

	5-Fold		
	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Raw Features			
Full Cov	0.141	0.286	0.672
Diag Cov	0.193	0.315	0.747
Tied Full Cov	**0.112**	0.224	0.573
Tied Diag Cov	0.161	0.267	0.580
Raw Features + PCA m = 7			
Full Cov	0.171	0.300	0.714
Diag Cov	0.218	0.631	0.828
Tied Full Cov	0.152	0.280	0.609
Tied Diag Cov	0.171	0.301	0.637
Gaussianizated Features			
Full Cov	0.153	0.247	0.697
Diag Cov	0.153	0.277	0.606
Tied Full Cov	0.132	0.232	0.534
Tied Diag Cov	0.163	0.292	0.614
Gaussianizated Features + PCA m = 7			
Full Cov	0.153	0.246	0.690
Diag Cov	0.163	0.247	0.661
Tied Full Cov	0.134	0.244	0.533
Tied Diag Cov	0.137	0.254	0.562
Gaussianizated Features + PCA m = 6			
Full Cov	0.154	0.242	0.695
Diag Cov	0.156	0.240	0.644
Tied Full Cov	0.136	0.248	0.546
Tied Diag Cov	0.141	0.256	0.588
Gaussianizated Features + PCA m = 5			
Full Cov	0.152	0.247	0.697
Diag Cov	0.157	0.241	0.633
Tied Full Cov	0.136	0.248	0.542
Tied Diag Cov	0.141	0.257	0.594

By comparing the results in the table above, we can observe that the best min DCFs are obtained throught the gaussian classifier with tied covariance matrices applied on the raw training dataset. As we foretold, the classifiers with diagonal covariance matrices have given the worst results. We can also notice that all the models have worse performances for imbalanced tasks.

Despite our expectations, Gaussianization does not seem to improve the performance of the models. Compared to the raw dataset, PCA involves a loss of information. On the other hand, for the gaussianized dataset, dimensionality reduction can be exploited without loss of useful information, as forecasted from the correlation analysis. Indeed, it is easy to see that the difference between the gaussianized dataset and the ones reduced with PCA is minimal in performance.

Logistic Regression

We turn our attention to discriminatory patterns. Given the previous considerations about the limited effectiveness of PCA on the raw set of fetaures we will consider 3 cases: raw features, gaussianizated features and gaussianizated features with PCA (m=6). Since classes are not balanced (the majority of data belongs to the false pulsar class) we use a prior weighted version of Logistic Regression to rebalance the costs of the classes, as it is possible to see in the next formula.

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

The model parameters are w and b. The model assumes that decision rules are linear hyperplanes orthogonal to w. The hyper-parameter λ allows for a tradeoff between good separation of classes, but poor generalization on unseen data ($\lambda \simeq 0$), and poor saparation of classes, but simpler solutions with small norm of w ($\lambda \gg 0$).

In the following we will compare the models (by minDCF) using different values of λ and different values of $\pi_T = (0.5, 0.1, 0.9)$ to estimate the best value of the hyper-parameter.

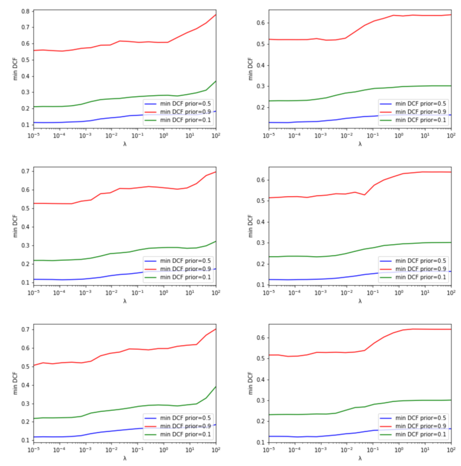

Figure 4: Linear Log-Reg with 5-Fold, Left: Train Data. Right: Gaussianized Data. Top: π_T=0.1. Middle: π_T=0.5. Bottom: π_T=0.9

We can see from the graphs in Fig. 4 that we obtain better results with small values of λ, especially with $\lambda = 10^{-5}$.

	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Raw Features			
MVG (Tied Full-Cov)	0.112	0.224	0.573
LogReg (λ=10^{-5}, $\pi_T = 0.1$)	**0.113**	0.210	0.557
LogReg (λ=10^{-5}, $\pi_T = 0.5$)	0.116	0.218	0.525
LogReg (λ=10^{-5}, $\pi_T = 0.9$)	0.119	0.219	0.505
Gaussianized Features			
LogReg (λ=10^{-5}, $\pi_T = 0.1$)	0.128	0.230	0.521
LogReg (λ=10^{-5}, $\pi_T = 0.5$)	0.124	0.233	0.513
LogReg (λ=10^{-5}, $\pi_T = 0.9$)	0.127	0.231	0.516
Gaussianized Features with PCA=6			
LogReg (λ=10^{-5}, $\pi_T = 0.1$)	0.128	0.229	0.532
LogReg (λ=10^{-5}, $\pi_T = 0.5$)	0.122	0.232	0.517
LogReg (λ=10^{-5}, $\pi_T = 0.9$)	0.124	0.234	0.531

As shown in the table above, the implementation with $\pi_T = 0.5$ seems to perform better for almost all the combinations analized. Even if the LogReg with balance $\pi_T = 0.1$ on raw feaatures seems to be the best result with $minDCF = 0.113$, we prefer to select as LogReg candidate model the one with $\pi_T = 0.5$ because in the reality we don't know if the raw data will be unbalanced or not and we want to avoid the bias since the samples are not normalized. Moreover, the results obtained with $\pi_T = 0.5$ are comparable with those of $\pi_T = 0.1$ since they differ only by 3 thousandths , i.e. a loss of 2% in the main application.

Up to now, the MVG with tied covariance and the linear Logistic Regression on raw dataset are the most promising models and they both are linear. We have seen that MVG tied full-cov performed better then MVG full-cov, which is a quadratic model, so we predict that linear model will perform better then quadratic ones on this dataset. We will test this hypotesis repeating the analysis for Quadratic Logistic Regression just for the raw dataset, using the k-fold strategy.

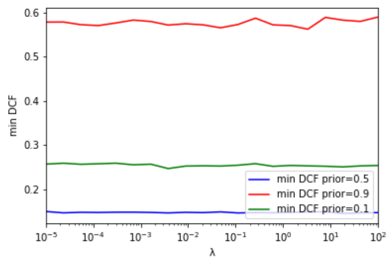

Figure 5: Quadratic Logistic Regression on Raw Features - π_T=0.5

	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Raw Features			
QuadLogReg (λ=10^{-5}, $\pi_T = 0.5$)	0.149	0.256	0.578
QuadLogReg (λ=10^{-4}, $\pi_T = 0.5$)	0.149	0.258	0.561

As we foretold, we have achieved lower performance than linear models. We will further test our assumption by employing other quadratic models.

Support Vector Machines

We now focus on SVMs models, we start analyzing linear SVM and balanced-linear SVM, then we consider to apply the polynomial quadratic kernel and the Radial Basis Function kernel formulations. Considering the previous results we expect linear SVMs to perform better. In the following, as for Logistic Regression models, we will only consider 5-Fold cross validation to choose a good value for the hyper-parameter C. We will also focus on Gaussianizated features with and without PCA, as well as raw data.

For linear SVM, we need to tune the hyper-parameter C. We start with a model that does not balance the two classes. To solve the SVM problem, we can consider the dual formulation:

$$J^D(\alpha) = -\frac{1}{2}\alpha^T \widehat{H} \alpha + \alpha^T 1$$

$$\widehat{x}_i = \begin{bmatrix} x_i \\ K \end{bmatrix} \qquad \widehat{H}_{ij} = z_i z_j \widehat{x}_i^T \widehat{x}_j^T$$

subject to $0 \leq \alpha_i \leq C_i, \qquad i = 1...n$, where n is the number of training samples, z_i is the class label of the i-th sample and $\mathbf{1}$ is a n-dimentional vector of ones. For our application K will be equal to 1. To rebalance the classes we will use different values of C depending on the classes. Where $C_i = C_T$ for samples of class H_T and $C_i = C_F$ for samples of class H_F. We select $C_T = \frac{\pi_T}{\pi_T^{emp}}$ and $C_F = \frac{\pi_F}{\pi_F^{emp}}$, where π_T^{emp} and π_F^{emp} are the empirical priors (sample portions) over the training set.

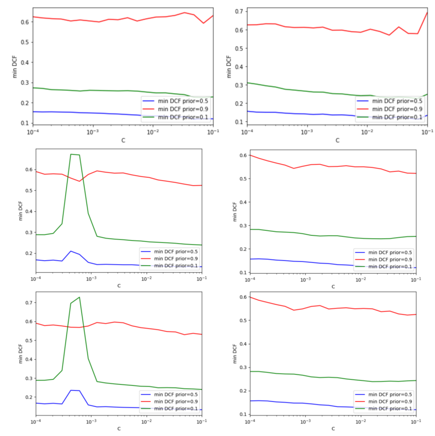

Figure 6: Left: linear SVM, Right: balaned-linear SVM with π_T=0.5; Top: raw dataset, Middle: gaussianized dataset, Bottom: gaussianized dataset + PCA m=6

Analyzing the Fig. 6, the hyper-parameter that seems to provide a lower min DCF for both balanced and unbalanced model is C=0.1, which we choose. We can compare linear models in terms of min DCF:

	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Raw Features			
MVG (Tied Full-Cov)	0.112	0.224	0.573
LogReg (λ=10^{-5}, $\pi_T = 0.5$)	0.116	0.218	0.525
Linear SVM ($C = 0.1$)	0.120	0.228	0.629
Linear SVM ($C = 0.1$, $\pi_T = 0.5$)	0.134	0.248	0.692
Gaussianized Features - no PCA			
Linear SVM ($C = 0.1$)	0.135	0.239	0.524
Linear SVM ($C = 0.1$, $\pi_T = 0.5$)	0.133	0.240	0.531
Gaussianized Features - PCA m=6			
Linear SVM ($C = 0.1$)	0.120	0.253	0.522
Linear SVM ($C = 0.1$, $\pi_T = 0.5$)	**0.119**	0.243	0.525

We can observe that they behave almost the same way and that class re-balancing is not necessery for raw features, but it is useful for the gaussianized ones with PCA m=6. Therefore we select as candidate model to represent the SVMs the one trained with the gaussianized data with PCA m = 6 because it seems to provide better results for each value of the priors. Furthermore, the result should be preferable as it is applied on normalized data, while the other approach is applied on raw data, which could greatly vary their distribution on an unseen dataset. We will see later if our assumptions are correct.

Although non-linear models perform worse on this dataset, we consider two formulations of non-linear SVMs to observe their performance.

We obtain non-linearity by expanding the features in a higher dimentional space. We do this by computing the kernel function $k(\mathbf{x}_i,\ \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T\,\Phi(\mathbf{x}_j)$ in two different versions:

- Second degree polynomial kernel: $k(\mathbf{x}_i,\ \mathbf{x}_j) = \left(\mathbf{x}_i^T\mathbf{x}_j + c\right)^d$
- Radial Basis Function (RBF) kernel: $k(\mathrm{x}_i, \mathrm{x}_j) = e^{-\gamma||\mathrm{x}_i - \mathrm{x}_j||^2}$

We first analyse the RBF kernel SVM formulation by jointly estimating the two hyper-parameters C and γ. We search good values by means of 5-folds cross-validation approach.

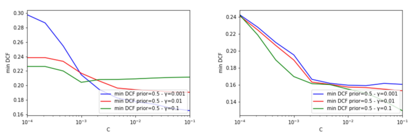

Figure 7: SVM RBF - Left: Raw features, Right: Gaussianized features + PCA m=6

As can be seen from Fig. 7, the best results are obtained using $\gamma = 0.001$ for the raw features and $\gamma = 0.1$ for the Gaussianized data, both with the hyper-parameter $C = 0.1$. We can compare the obtained values with the previous classifier to see that this model behaves as we expected.

	$\widetilde{\pi} = 0.5$
MVG Tied Full-Cov (raw)	0.112
LogReg (raw, λ=10^{-5}, $\pi_T = 0.5$)	0.116
Linear SVM ($gauss + PCA_6$, $C = 0.1$, $\pi_T = 0.5$)	0.119
	Raw Features
SVM RBF ($\gamma = 0.001$, $C = 0.1$)	0.215
	Gaussianized Features - PCA m=6
SVM RBF ($\gamma = 0.1$, $C = 0.1$)	**0.130**

Employing the quadratic kernel formunation, we evaluate its performance by keeping C fixed at 0.1 and observing the min DCF as the hyper-parameter c varies, just for our main application using the gaussianized dataset with PCA (m=6). The table below shows that the results are not good enough, as we foretold.

	$\widetilde{\pi} = 0.5$
MVG Tied Full-Cov (raw)	0.112
LogReg (raw, λ=10^{-5}, $\pi_T = 0.5$)	0.116
Linear SVM ($gauss + PCA_6$, $C = 0.1$, $\pi_T = 0.5$)	0.119
Gaussianized Features - PCA m=6	
SVM Poly ($c = 0.0$, $C = 0.1$)	0.242
SVM Poly ($c = 0.1$, $C = 0.1$)	0.214
SVM Poly ($c = 10$, $C = 0.1$)	**0.160**
SVM Poly ($c = 30$, $C = 0.1$)	0.158

Comparing the values of the min DCFs of the classifiers just analyzed, we can see that linear models are preferable to quadratic ones, as we had predicted. Despite this, all the SVMs, but the one trained with the gaussianized features with PCA m=6, do not seem to perform well enough to outperform the previously analyzed models.

The candidate models still remain linear Logistic Regression ($l = 10^{-5}$, $\pi_T = 0.5$) and MVG with tied full covariance matrices, both using raw training dataset, but we consider also the linear SVM model($C = 0.1$, $\pi_T = 0.5$) trained with gaussianized features with PCA m=6.

Gassian Mixture Models

The last model we analyze is GMM, a generative model that we will evaluate with full or diagonal and tied or not covariance matrices. We rely again to the 5-fold approach to compare different models.

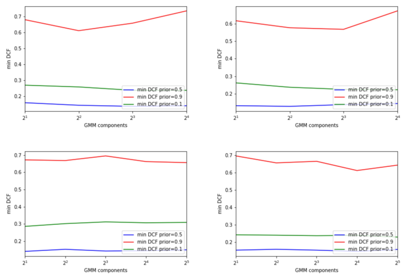

Figure 8: minDCFs as the number of GMM coponents varies - Left: Raw features, Right: Gaussianized features with PCA (m=6). Top: GMM Full-Cov, Bottom: GMM Tied Full-Cov

	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
MVG Tied Full-Cov (*Raw*)	0.112	0.224	0.573
LogReg (*Raw*, λ=10^{-5}, $\pi_T = 0.5$)	0.116	0.218	0.525
Linear SVM (*Gauss* + *PCA*$_6$, $C = 0.1$, $\pi_T = 0.5$)	0.119	0.243	0.525
Raw Features			
GMM Full-Cov (8 components, $\alpha = 0.1$)	0.136	0.237	0.658
GMM Tied Full-Cov (2 components, $\alpha = 0.1$)	0.141	0.286	0.672
Gaussianized Features - PCA m=6			
GMM Full-Cov (4 components, $\alpha = 0.1$)	**0.128**	0.237	0.576
GMM Tied Full-Cov (8 components, $\alpha = 0.1$)	0.154	0.237	0.664

The analysis shows that the tied hypothesis does not improve the model performance, unlike what happened for the MVG classifier. In any case, this model turns out to be worse than the others analyzed by means of the minDCF and for this reason we discard it.

In conclusion, the results of the most promising candidate models so far are the following:

	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Raw Features			
MVG (Tied Full-Cov)	0.112	0.224	0.573
LogReg (λ=10^{-5}, $\pi_T = 0.5$)	0.116	0.218	0.525
Gaussianized Features - PCA m=6			
Linear SVM ($C = 0.1$, $\pi_T = 0.5$)	0.119	0.243	0.525

For our target application the performances are quite similar between the three classifiers, while with $\pi_T = 0.1$ and $\pi_T = 0.9$ the Logistic Regression classifiers seems to outperform the other models.

A ROC plot confirms that classifiers perform practically equally, since the AUC (Area Under Curve) is actually the same for all models. We keep these three classifiers as candidate models and we will see below if the choice has been correct.

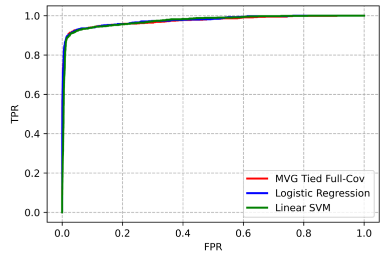

Figure 9: ROC curve

Score calibration

So far we have ascertained the quality of the models by means of the min DCF metric, which mesures the cost we would pay if we made optimal decisions for the validation set using the scores of the recognizer. However, the cost that we actually pay depends on the goodness of the threshold we use to perform class assignment, i.e. on the goodness of the decisions we make using those scores.

We therefore turn our attention to actual DCFs. If scores are well calibrated, the theoretical threshold $t = \log \frac{\widetilde{\pi}}{1-\widetilde{\pi}}$ is the optimal one that optimizes the Bayes risk. Computing the actual DCFs, assuming that the scores are already well calibrated, will show us how good the models would behave if we use the theoretical threshold for each application.

	$\widetilde{\pi} = 0.5$		$\widetilde{\pi} = 0.1$		$\widetilde{\pi} = 0.9$	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
Raw Features						
MVG (Tied Full-Cov)	0.112	0.191	0.224	0.274	0.573	1.422
LogReg (λ=10^{-5}, $\pi_T = 0.5$)	0.116	0.120	0.218	0.225	0.525	0.548
Gaussianized Features - PCA m=6						
Linear SVM ($C = 0.1$, $\pi_T = 0.5$)	0.119	0.124	0.243	0.698	0.525	0.572

From the table above we can observe that all models but Logistic Regression need recalibration as there were also very large losses.

In the case of the MVG Tied Full-Cov model, for the primary application there is $\approx 70\%$ loss that becomes very high for $\widetilde{\pi} = 0.9$ reaching a loss of $\approx 150\%$. As for the $\widetilde{\pi} = 0.1$, the loss is more contained but still existing, by $\approx 20\%$.

Observing the actDCFs of the SVM model, it can be seen that for our main application the scores are well calibrated and that it is almost the same for $\widetilde{\pi} = 0.9$. The biggest loss is recorded with $\widetilde{\pi} = 0.1$ and amounts to $\approx 190\%$.

We can validate the analysis just made by means of Bayes error plots, which show the DCFs for different applications.

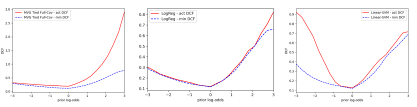

Figure 10: Bayes error plots with 5-fold approach. Left: MVG Tied Full-Cov. Middle: LogReg. Right: Linear SVM (all with their respective datasets)

The Bayes error plots shows by means of Fig. 9 that all candidate classifiers but Logistic Regression have poor calibration, as expected. For this reason we employ score calibration using a general approach which consists of computing a transformation function f that maps the scores s of each model to well-calibrated scores $s_{cal} = f(s)$. We assume that the mapping function has a linear form:

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

This function should output well-calibrates scores, so it can be interpreted as the log-likelihood ratio for the two class hypotheses:

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

so we can write the class posterior probability for prior $\widetilde{\pi}$ as:

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

Interpreting the score as features and setting up $\beta' = \beta + \log \frac{\widetilde{\pi}}{1-\widetilde{\pi}}$, we obtain an expression that corresponds to the log posterior ratio of the Logistic Regression model. We can employ the prior-weighted Logistic Regression model to learn the model parameters α, β' over our training scores. To recover the calibration scores $f(s)$ we will need to compute

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

We also apply normalization on Logistic Regression, to observe its behavior as the threshold varies. Although we have optimized the calibration for our target application, the model provides a good calibration even for different applications.

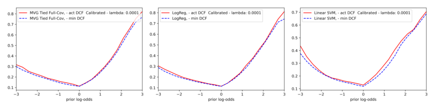

Figure 11: Bayes error plots after calibration with 5-fold approach and prior-weighted LogReg . Left: MVG Tied Full-Cov. Middle: LogReg. Right: Linear SVM (all with their respective datasets)

As shown from Fig. 10 and as expected from the previus observations, score calibration appears to have affected positivly all three classifiers. It is also clear that the choice of the prior did not affect the performances of the applications with a different effective prior.

We can again evaluate the actual DCFs after calibration, calculating them using the theoretical threshold since the final scores should be calibrated.

	$\widetilde{\pi} = 0.5$		$\widetilde{\pi} = 0.1$		$\widetilde{\pi} = 0.9$	
	minDCF	actDCF(cal)	minDCF	actDCF(cal)	minDCF	actDCF(cal)
			Raw Features			
MVG (Tied Full-Cov)	0.112	0.113	0.224	0.237	0.573	0.608
LogReg (λ=10^{-5}, $\pi_T = 0.5$)	0.116	0.120	0.218	0.227	0.525	0.539
			Gaussianized Features - PCA m=6			
Linear SVM ($C = 0.1$, $\pi_T = 0.5$)	0.119	0.129	0.243	0.257	0.525	0.549

As can be seen in the table above, all models have benefited from the recalibration. Indeed, all models provide values of actDCFs close to those of minDCFs. The loss flattened out and became acceptable for both MVG Tied Full-Cov and SVM models.

Experimental results

Now let's analyze the performance of the various classifiers on the test set, again by means of the minDCF metric.

	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
	Raw			Gaussianized + PCA m=6		
MVG Full	0.140	0.283	0.646	0.157	0.309	0.626
MVG Diag	0.185	0.330	0.621	0.215	0.648	0.687
MVG Tied Full	**0.110**	0.207	0.591	0.147	0.268	0.607
MVG Tied Diag	0.152	0.262	0.544	0.166	0.309	0.602
LogReg (λ=10^{-5}, $\pi_T = 0.1$)	0.110	0.199	0.532	0.139	0.245	0.582
LogReg (λ=10^{-5}, $\pi_T = 0.5$)	**0.107**	0.199	0.542	0.139	0.245	0.605
LogReg (λ=10^{-5}, $\pi_T = 0.9$)	0.115	0.202	0.509	0.144	0.246	0.586
SVM ($C = 0.1$)	0.120	0.213	0.574	0.140	0.251	0.599
SVM ($C = 0.1$, $\pi_T = 0.1$)	0.124	0.212	0.572	0.139	0.248	0.600
SVM ($C = 0.1$, $\pi_T = 0.5$)	**0.109**	0.206	0.574	0.146	0.262	0.625
SVM ($C = 0.1$, $\pi_T = 0.9$)	0.117	0.219	0.529	0.150	0.253	0.584
GMM Full (2 components)	0.156	0.289	0.628	0.189	0.316	0.628
GMM Full (4 components)	0.151	0.280	0.652	0.163	0.292	0.602
GMM Full (8 components)	0.143	0.264	0.606	0.148	0.272	0.634
GMM Full (16 components)	0.240	0.240	0.639	0.146	0.244	0.683
GMM Tied Full (2 components)	0.140	0.283	0.646	0.157	0.309	0.626
GMM Tied Full (4 components)	0.149	0.321	0.628	0.169	0.334	0.624
GMM Tied Full (8 components)	0.139	0.313	0.676	0.165	0.327	0.703
GMM Tied Full (16 components)	0.144	0.315	0.644	0.158	0.335	0.676

As regards the LogReg and the tied-full MVG, the results are consistent with the analyzes made previously. The same cannot be said for the SVM classifier since the model we had selected was SVM with $\pi_T = 0.5$ on the gaussianized features, while the model trained on raw features shows much better results in this phase. The discrepancy can be explained by the fact that, as we explained earlier, the raw data would need a normalization to avoid that very different distributions occur as the number of samples varies. Indeed, the data normalized by means of Gaussianization show us that the best models are consistent with those chosen. This may mean that the raw data has been distributed in such a way as to enhance SVMs with $\pi_T = 0.5$ on raw features. This demonstrates, as already mentioned, that the raw data should be normalized to obtain consistent classifications as the size of the dataset varies. However, the analysis made on the gaussianized data shows us that the previous analyzes seem to make sense, i.e. that the model with $\pi_T = 0.5$ seems to be the best for SVMs.

To conclude our analysis, we can proceed with the score calibration by applying the same procedure applied previously, proceeding on the candidate models.

	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
	MVG Tied Full-Cov (*Raw*)		
minDCF	0.112	0.224	0.573
actDCF	0.202	0.278	1.338
calibrated actDCF	0.115	0.224	0.607
	LogReg (λ=10^{-5}, $\pi_T = 0.5$) (*Raw*)		
minDCF	0.116	0.218	0.525
actDCF	0.114	0.217	0.555
calibrated actDCF	0.115	0.215	0.571
	Linear SVM ($C = 0.1$, $\pi_T = 0.5$) ($Gauss + PCA_6$)		
minDCF	0.143	0.253	0.589
actDCF	0.148	0.331	0.937
calibrated actDCF	0.149	0.270	0.624

We can observe that the calibration worked just as well as it did on the training set.

Conclusions

About the analyzed dataset, the SVMs are very dependent on the distribution of the data and on the pre-processing. Therefore the most reliable models remain:

- MVG with tied full covariance matrix on raw features
- Logistic Regression with $\lambda = 10^{-5}$ e $\pi_T = 0.5$ on raw features

which both in the train and in the test phases have proved to be consistent according to the chosen hyper-parameters. A preprocessing of the data with a normalization would have standardized the results also for SVM.

For all models we have observed that linear models appear to perform better than quadratic ones.

In conclusion, we obtained low DCF values for the target application of ≈ 0.1, but also for $\widetilde{\pi} = 0.1$ and $\widetilde{\pi} = 0.9$ the results seem to be acceptable overall, being about ≈ 0.2 and ≈ 0.5 respectively.