Project - Avila

Sandro Cumani

sandro.cumani@polito.it

Politecnico di Torino

Outline

- Introduction
- Avila features
- Building a classifier for the Avila task
- Experimental validation
- Conclusions

Introduction

The Avila data set[1] has been extracted from 800 images of the the “Avila Bible”, a giant Latin copy of the whole Bible produced during the XII century between Italy and Spain

The paleographic analysis of the manuscript has individuated the presence of 12 copyists. The pages written by each copyist are not equally numerous

The pages have been split into groups of 4 consecutive rows

Each group corresponds to a pattern (sample)

[1] C. De Stefano, M. Maniaci, F. Fontanella, A. Scotto di Freca, Reliable writer identification in medieval manuscripts through page layout features: The “Avila” Bible case, Engineering Applications of Artificial Intelligence, Volume 72, 2018, pp. 99-110.

Introduction

The original task requires identifying the copyist of each pattern

Here we consider the simpler tasks of discriminating between the two most frequent copyists (named A and F in the original dataset)

We keep the original train / evaluation split

The training set contains 4286 samples of copyist A and 1961 samples of copyist F

The evaluation set contains 4286 samples of copyist A and 1962 samples of copyist F

Avila features

Each pattern consists of 10 features, in part pre-processed by means of Z-normalization (centering and scaling to unit variance)

- intercolumnar distance
- upper margin
- lower margin
- exploitation
- row number
- modular ratio
- interlinear spacing
- weight
- peak number
- modular ratio / interlinear spacing

Avila features

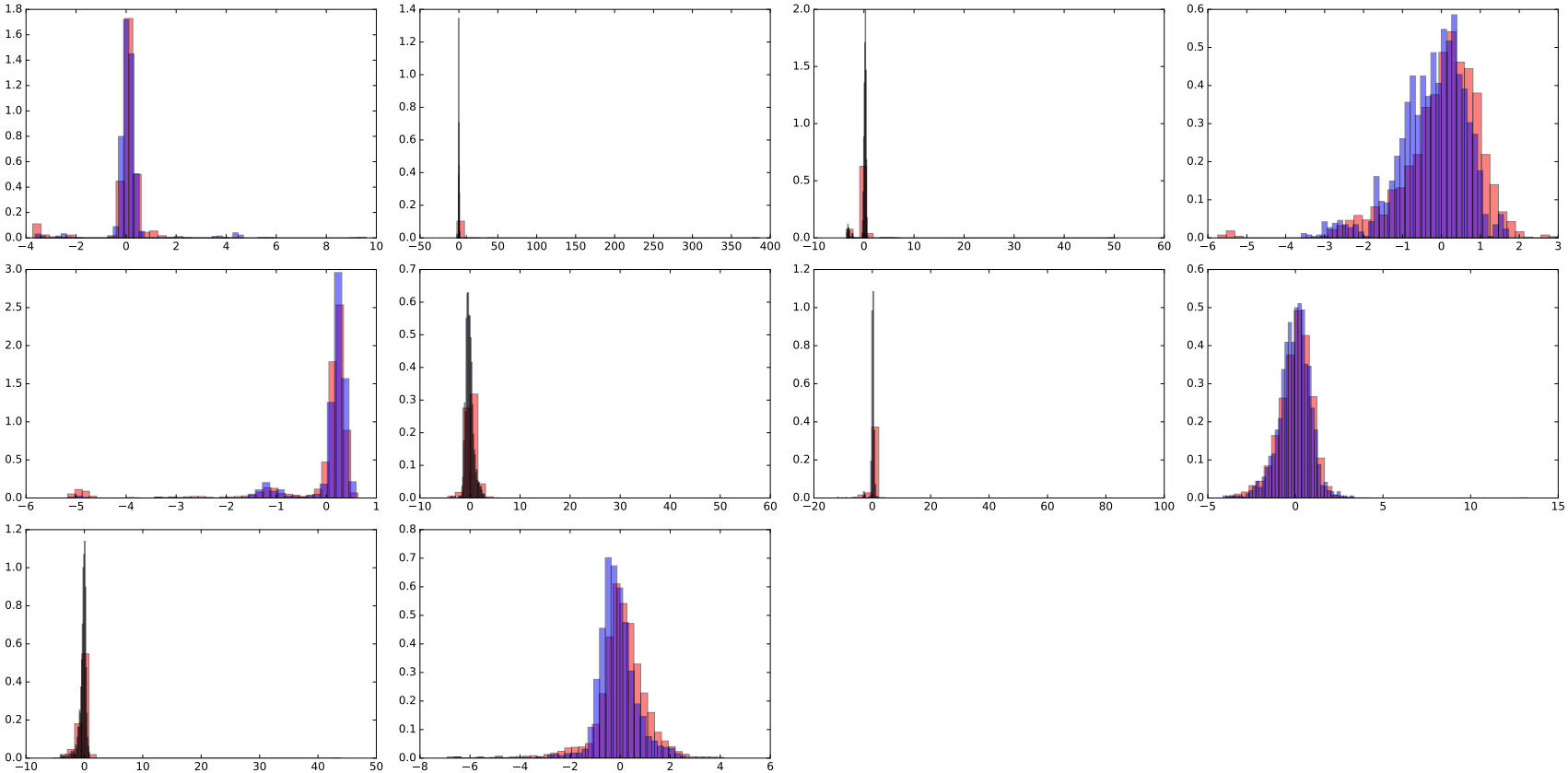
Features have been slightly altered for teaching purposes

In the original dataset, features can be considered continuous, except for row number, which consists of 48 discrete values. However, since the number of different values is large, we treat the row number as a continuous feature as well

The modified features are all continuous

Avila features

Histogram of the (modified) Avila dataset features (training set). Features are sorted by their order, from left to right, top to bottom. Red histograms refer to copyist A, blue histograms to copyist F



Avila features

A preliminary analysis of the training data shows that, in many cases, the raw features have irregular distributions, characterized by a presence of significantly large outliers

Due to the presence of outliers, we expect that classification approaches may produce sub-optimal results (especially Gaussian-based methods)

We therefore further pre-process data by “Gaussianizing” the features

Avila features

Gaussianization is a procedure that allows mapping a set of features to values whose empirical cumulative distribution function is well approximated by a Gaussian c.d.f.

The processing consists in mapping the features to a uniform distribution and then transforming the mapped features through the inverse of Gaussian cumulative distribution function

Avila features

Let x be the feature we want to transform

We first compute its rank over the training dataset:

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

x_i is the value of the considered feature for the i-th training sample. We add 1 to the numerator and 2 to the denominator to avoid numerical issues in the next stage (i.e. we assume the existance of a feature smaller than all the others and a feature larger than all the others)

We then compute the transformed feature as

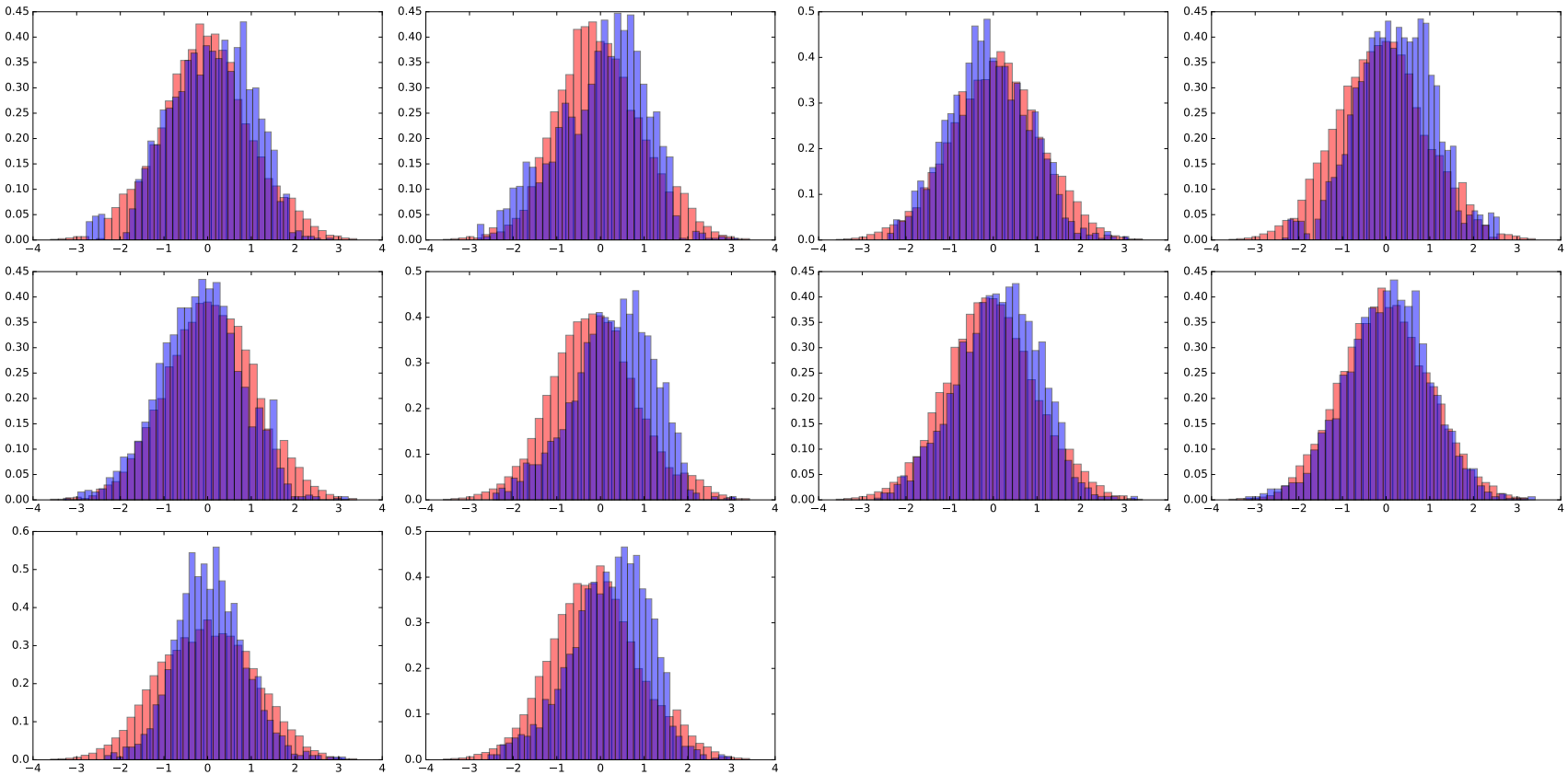
$$y = \Phi^{-1}(r(x))$$

where Φ is the inverse of the cumulative distribution function (percent point function, p.p.f) of the standard normal distribution

NOTE: we have to apply the transformation both to training and evaluation samples. The ranking should always be computed using the training data

Avila features

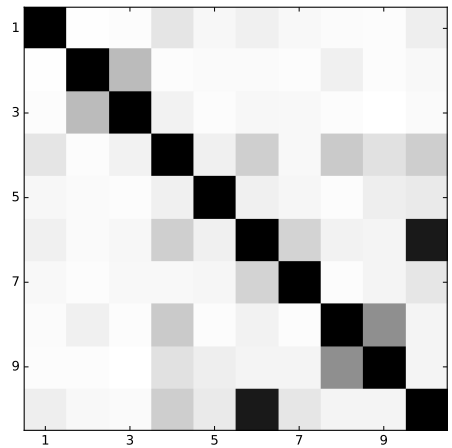
Histogram of the (modified) Avila dataset features (training set) after Gaussianization. Features are sorted by their order, from left to right, top to bottom. Red histograms refer to copyist A, blue histograms to copyist F

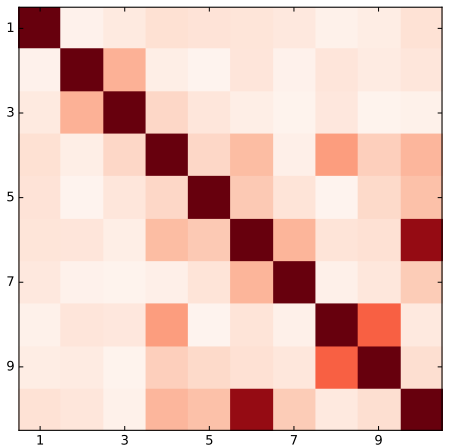


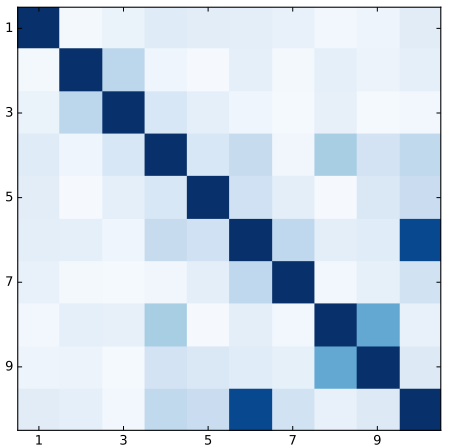
Avila features

A correlation analysis of Gaussianized features shows that feature 6 and 10 are strongly correlated.

Heat map showing the absolute value of the Pearson correlation coefficient $\left| \frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}} \right|$. Gray: whole dataset. Red: samples of copyist A. Blue: samples of copyist F. Darker color implies larger value







This suggests we may benefit from using PCA to map data to 9, uncorrelated features to reduce the number of parameters to estimate.

Classifying Avila features

We start considering simple Gaussian classifiers

Although we employed Gaussianization over the whole dataset, the histograms for each class show, in some cases, moderate deviations from the Gaussian assumptions.

Since within-class covariance matrices are almost diagonal, we consider both diagonal and full-covariance models

Classifying Avila features

To understand which model is most promising, and to assess the effects of using PCA, we can adopt two methodologies:

- We can split the training dataset into development (for model training) and validation subsets (single-fold in the following)
- We can employ K-Fold cross-validation

In this lecture we will analyze both options. For the project, you can choose the one that you deem more appropriate for your dataset.

Classifying Avila features

Single split:

- The final classifier will be the same that we evaluate on the validation set: model selection and hyper-parameter will be optimal at least for the validation set
- We need to train fewer models, so training is faster
- We have fewer data for validation and model training

Classifying Avila features

K-Fold cross-validation:

- More data available for training and validation
- The final classifier will be obtained by re-training over the whole training set, so it will leverage additional data
- Decisions are made over the validation set for the models trained using folds. They may not be optimal for the model learned from all training data

We shall consider both approaches for the moment. The single fold consists of 80% development (i.e. model training) data and 20% validation data, randomly sampled. The K-Fold is implemented with K=5. Data has been shuffled before splitting, so that the data of different folds are homogeneous.

Classifying Avila features

Our main application will be a uniform prior one:

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

We will also consider unbalanced applications

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

where the prior is biased towards one of the two copyists.

For the moment, we are interested in choosing the most promising approach. We therefore measure performance in terms of normalized minimum detection costs.

Min DCF measures the cost we would pay if we made optimal decisions for the test set (in our case the validation set) using the recognizer scores

We will analyze how to choose an optimal threshold in a second stage.

Classifying Avila features

MVG Classifiers — min DCF on the validation set

	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Gaussianized features — no PCA						
Full-Cov	**0.563**	0.980	**0.979**	**0.588**	1.000	**0.969**
Diag-Cov	0.649	**0.977**	0.992	0.653	**0.995**	0.981
Tied Full-Cov	0.716	0.994	0.997	0.713	0.999	1.000
Tied Diag-Cov	0.734	0.994	1.000	0.734	1.000	1.000
Gaussianized features — PCA (m = 9)						
Full-Cov	**0.563**	0.991	0.992	**0.593**	**0.989**	**0.977**
Diag-Cov	0.652	**0.971**	**0.984**	0.673	**0.989**	0.979
Tied Full-Cov	0.728	0.997	0.997	0.722	1.000	1.000
Tied Diag-Cov	0.730	0.997	0.997	0.721	1.000	1.000
Gaussianized features — PCA (m = 8)						
Full-cov	0.588	0.995	0.984	0.620	0.993	0.976
Diag-Cov	0.654	0.970	0.979	0.682	0.986	0.975
Raw features (no Gaussianization) — no PCA						
Full-Cov	0.700	0.992	0.989	0.776	0.998	0.982

Classifying Avila features

The Full-Cov model obtains the best performance in K-fold cross validation protocol both with and without PCA. With the single fold protocol we obtain similar results

PCA is not effective neither for Full nor Diagonal covariance models

In the first case, the model is able to account for correlations, and we have enough data to obtain robust estimates. We can use PCA to reduce the number of parameters, but this does not improve our estimate (however, consistently with our previous observations, PCA with $m = 9$ does not degrade performance either, whereas further reduction decreases accuracy)

In the second case, PCA diagonalizes the global covariance matrix, but this does not imply that the within-class covariance matrices of each class would be diagonal. The correlation analysis shows similarities between the two classes, but there are some differences in feature correlations for the two copyists.

Classifying Avila features

The Tied models perform in general worse. Again, the covariance matrices are fairly similar, however, since we have enough data, estimating separately the matrices for the two classes provides a better model

Gaussianization significantly improves performance over raw features

The results between single-fold and K-fold are consistent, suggesting that the amount of data is enough for validation and model training also for the single-fold set-up (this may change for other classifiers)

Overall, the best candidate is currently the MVG model with Full Covariance matrices without PCA (under both the K-fold and the single fold protocols). However, all models are (almost) useless for the imbalanced tasks

Classifying Avila features

We now turn our attention to discriminative approaches

Given the limited effectiveness of PCA for generative models we only consider using the whole set of features

In this case we compare the results with and without pre-processing (Gaussianization)

Classifying Avila features

We start considering regularized (linear) Logistic Regression

Since classes are not balanced, we re-balance the costs of the different classes, minimizing

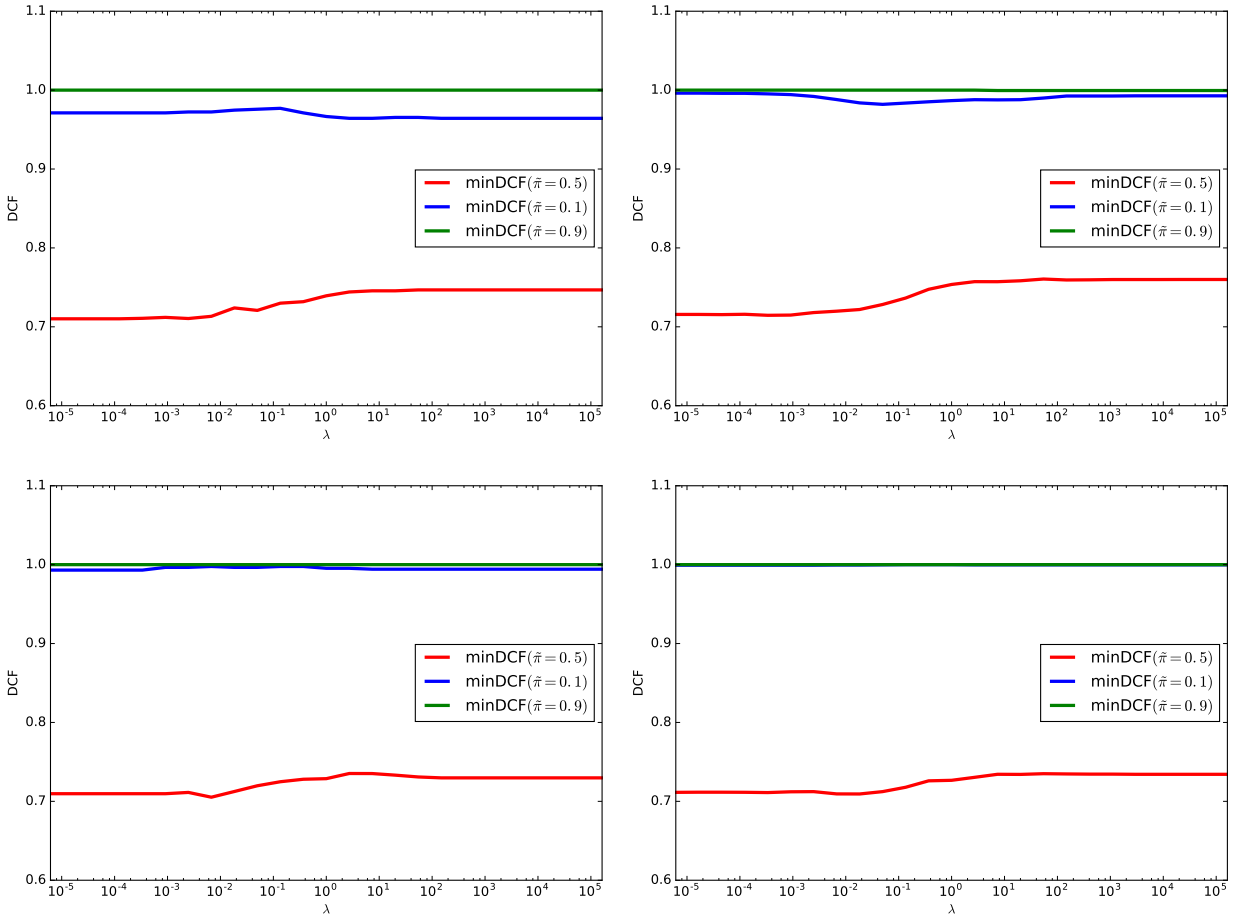
$$J(\boldsymbol{w}, b) = \frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \frac{\pi_T}{n_T} \sum_{i=1|c_i=1}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)}\right) + \frac{1 - \pi_T}{n_F} \sum_{i=1|c_i=0}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)}\right)$$

We start considering our main application, thus we set $\pi_T = \frac{1}{2}$

We can compare the models using different values of λ

Classifying Avila features

Linear Log-Reg: min DCF for different values of λ. Top: no pre-processing. Bottom: Gaussianized features. Left: single fold. Right: K-fold. Our primary metric is the red line ($\tilde{\pi} = \frac{1}{2}$)



Classifying Avila features

Results with K-fold and single fold are similar.

Regularization provides little to no benefit. Best results are obtained with small values of λ. Setting $\lambda = 0$ (not reported here) provides the same performance.

Gaussianization seems much less relevant. Indeed, logistic regression does not require specific assumptions on the data distribution. The scale normalization effects of Gaussianization seems also not necessary to improve the effectiveness of regularization.

Classifying Avila features

We can also consider training using a different prior π_T to see the effects on the other applications. We restrict the analysis to models with small regularization, and consider only the K-fold protocol.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
MVG (Full-Cov)	0.588	1.000	0.969
MVG (Tied Full-Cov)	0.713	0.999	1.000
Raw features			
Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.716	0.996	1.000
Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.706	0.993	1.000
Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.732	0.993	1.000
Gaussianized features			
Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.711	0.999	1.000
Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.705	1.000	1.000
Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.729	0.999	0.998

Classifying Avila features

Overall, the MVG model with full covariances perform better.

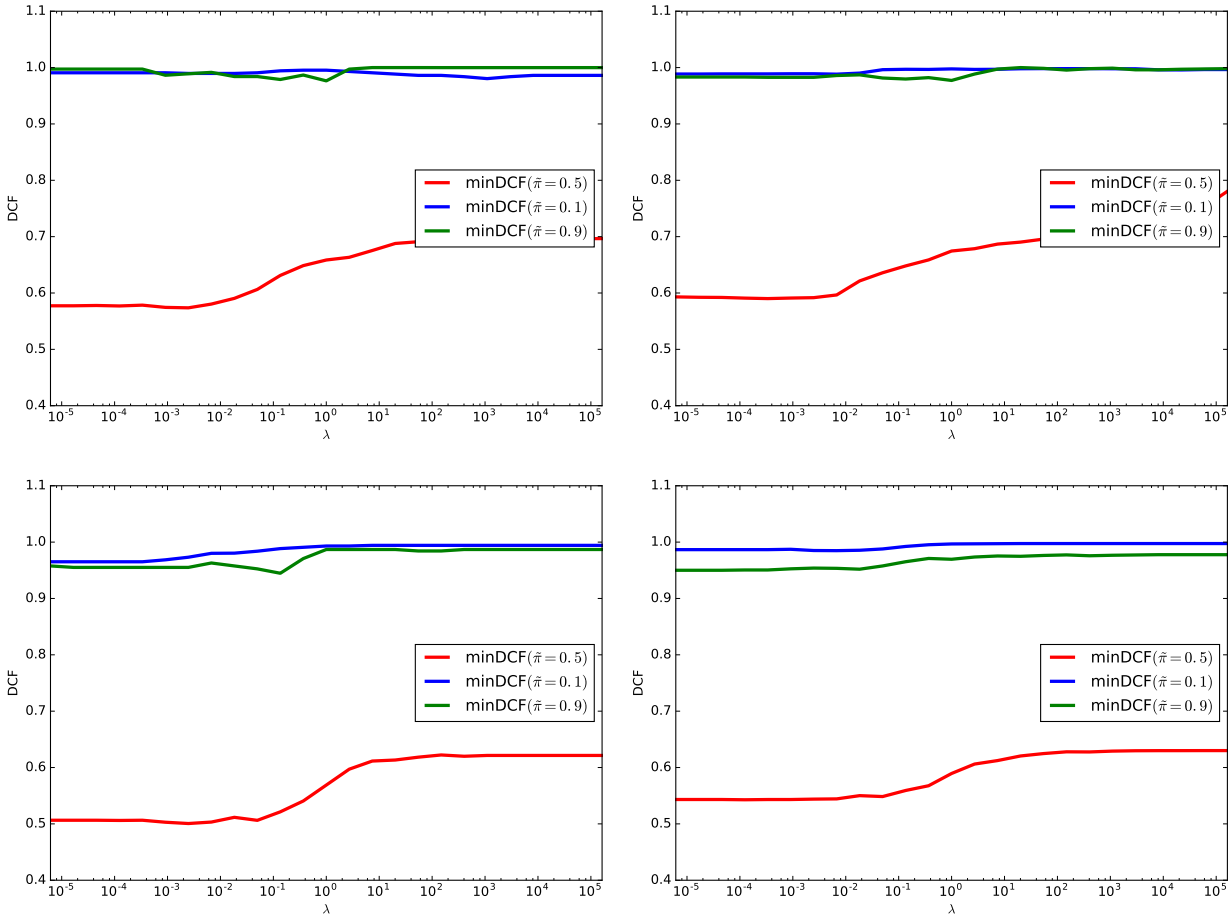
Logistic regression provides very small improvement over the MVG model with linear classification rules

Using different values for π_T does not improve the Logistic Regression models for the other two applications

Since MVG corresponds to quadratic separation rules, we repeat the analysis for Quadratic Logistic Regression

Classifying Avila features

Quadratic Log Reg: min DCF for different values of λ. Top: no pre-processing. Bottom: Gaussianized features. Left: single fold. Right: K-fold. Our primary metric is the red line ($\tilde{\pi} = \frac{1}{2}$)



Classifying Avila features

The conclusions are similar as for the linear case

Regularization is not required

Pre-processing (Gaussianization) is significantly more helpful in this case

Classifying Avila features

Again, we consider training using a different prior π_T to see the effects on the other applications. We restrict the analysis to models with small regularization, and consider only the K-fold protocol.

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Raw features			
MVG (Full-Cov)	0.776	0.998	0.982
Quad Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.593	0.989	0.983
Quad Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.607	0.976	0.992
Quad Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.578	0.991	0.966
Gaussianized features			
MVG (Full-Cov)	0.588	1.000	0.969
Quad Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	**0.543**	0.987	0.950
Quad Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.558	**0.964**	0.989
Quad Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.548	0.994	**0.939**

Classifying Avila features

With a quadratic kernel, Logistic Regression outperforms the MVG classifier

Using different values for π_T slightly improves the classification performance for the specific application: we are optimizing the loss for the application prior

If we are interested in different applications it may be worth using a different model for each of the three applications, although the benefits are small in this case

However, the results for imbalanced applications are still poor

Classifying Avila features

The previous analysis suggests that the Gaussian assumption may not be sufficiently accurate for our features

Furthermore, classes cannot be well separated by linear decision rules

We therefore turn our attention to SVMs and to Gaussian Mixture Models

We start with analyzing linear SVM, although we expect the results to be quite poor

We consider only Gaussianized features

Classifying Avila features

For linear SVM, we need to tune the hyper-parameter C. Again, we resort to both single fold and K-fold cross validation

We start with a model that does not balance the two classes

We consider the SVM formulation we developed in the lab (i.e. the bias term is simulated through feature expansion, and is thus regularized)

Classifying Avila features

To re-balance the classes we use a different value of C for the different classes:

$$\max_{\alpha} \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{H} \boldsymbol{\alpha}$$

subject to

$$0 \leq \alpha_i \leq C_i, \ i = 1, \ldots, n$$

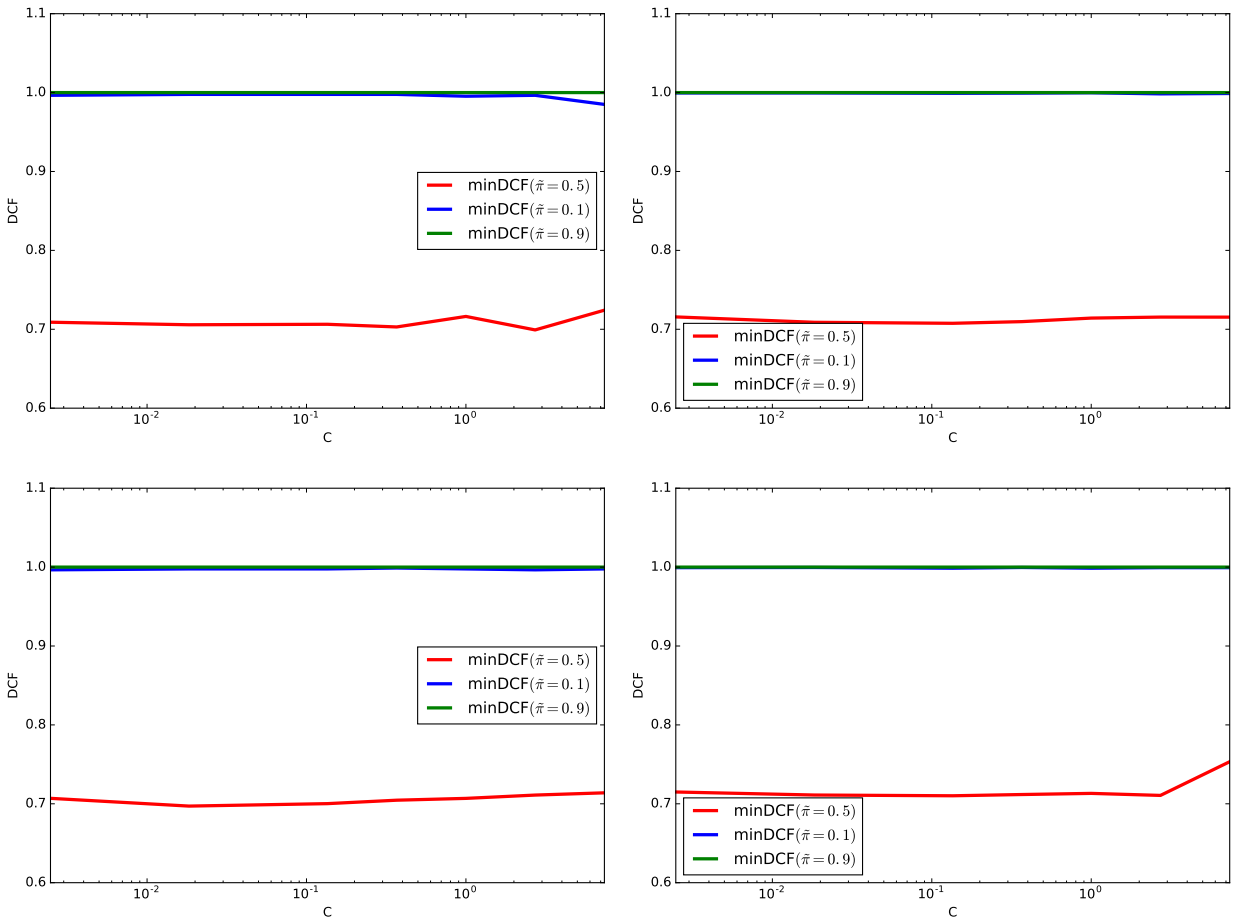
where $C_i = C_T$ for samples of class $\mathcal{H}_T$ and $C_i = C_F$ for samples of class $\mathcal{H}_F$. We omitted the constraint related to the bias, since we are not explicitly modeling the bias term

We select $C_T = C \frac{\pi_T}{\pi_T^{emp}}$ and $C_F = C \frac{\pi_F}{\pi_F^{emp}}$.

π_T^{emp} and π_F^{emp} are the empirical priors (i.e. sample proportions) for the two classes computed over the training set.

Classifying Avila features

Linear SVM (without class balancing): min DCF for different values of C. Gaussianized features. Top: standard SVM. Bottom: class balancing. Left: single fold. Right: K-fold. Our primary metric is the red line



Classifying Avila features

The choice of C does not look critical. We select $C = 0.1$

Again, the K-fold results are consistent with the single fold results, suggesting that our model is behaving correctly

We can compare linear models in terms of min DCF:

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
MVG (Tied Full-Cov)	0.713	0.999	1.000
Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.711	0.999	1.000
Linear SVM ($C = 0.1$)	0.709	0.999	1.000
Linear SVM ($C = 0.1, \pi_T = 0.5$)	0.712	0.999	1.000

Linear SVM performs similarly to other linear approaches, as expected.

Class re-balancing is not necessary, therefore we will use the default SVM formulation

Classifying Avila features

Since non-linear models perform better on this dataset, we consider two non-linear SVM formulations

The first will use a polynomial quadratic kernel (similar to the quadratic Logistic Regression model, again we expect similar results)

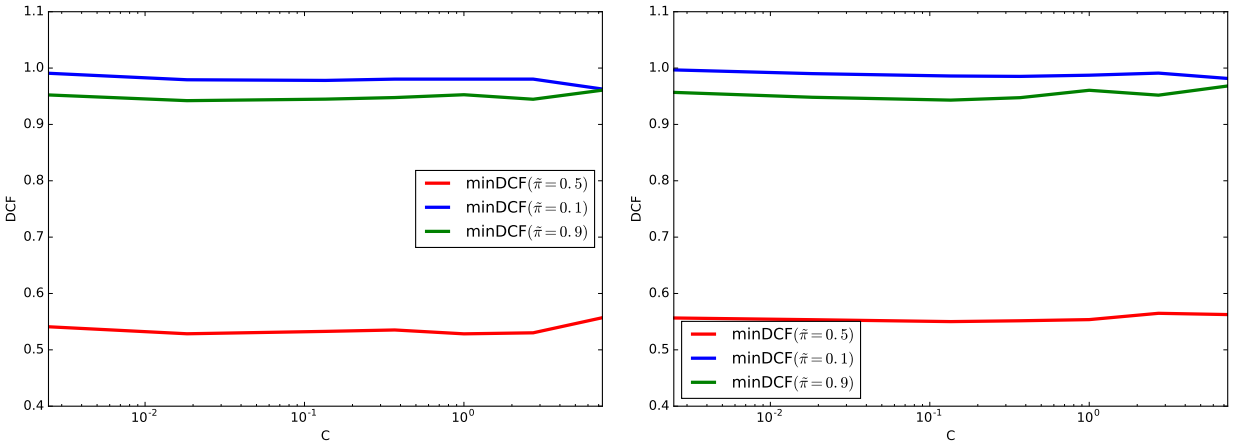
The second will employ a Radial Basis Function kernel

For the RBF kernel we also need to estimate the kernel width γ. We will use a grid search to jointly optimize C and γ

We will mainly focus on Gaussianized features

Classifying Avila features

Quadratic kernel SVM: min DCF for different values of C. Gaussianized features. Left: single fold. Right: K-fold. Our primary metric is the red line



Also in this case, the choice of C does not significantly affect the performance. We again choose $C = 0.1$, since it provides slightly more robust performance for the imbalanced applications

Classifying Avila features

We can compare quadratic models in terms of min DCF:

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
MVG (Full-Cov)	0.588	1.000	0.969
Quad Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	**0.543**	0.987	0.950
Quadratic SVM ($C = 0.1$)	0.552	0.992	0.946

Quadratic kernel SVM provides slightly worse results than Logistic Regression.

Classifying Avila features

The worse performance seems due to class imbalance. Re-training with the estimated C but with class balancing we obtain slightly better results, very close to the Logistic Regression ones

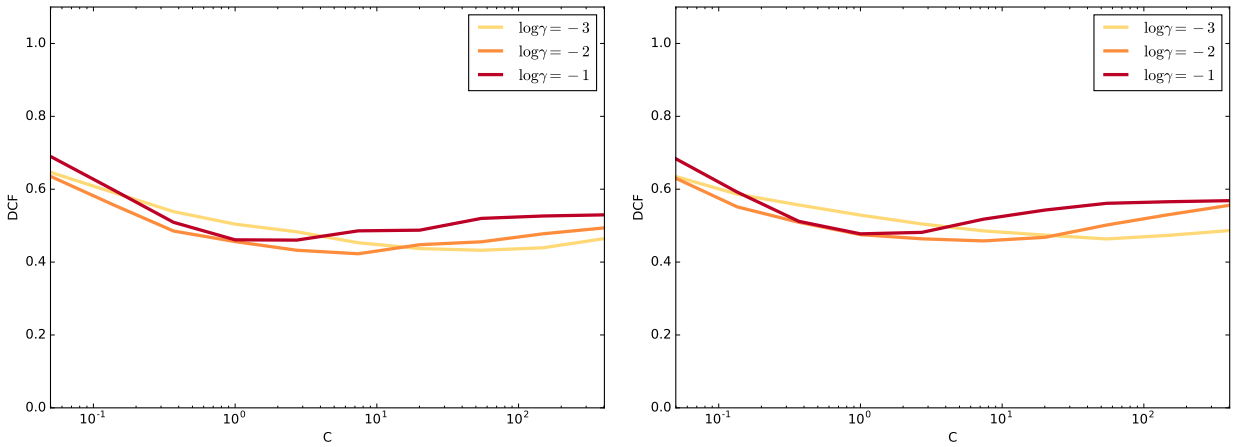
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Quad SVM ($C = 0.1$), $\pi_T = \pi_T^{emp}$	0.552	0.992	0.946
Quad SVM ($C = 0.1$), $\pi_T = 0.5$	0.545	0.986	0.956

We conclude that the two quadratic discriminative methods are equivalent on this dataset, and both perform better than the MVG classifier

We now turn our attention to the RBF kernel

Classifying Avila features

RBF kernel SVM: min DCF ($\tilde{\pi} = 0.5$) for different values of C and γ. Gaussianized features. Left: single fold. Right: K-fold.



The plot shows that both γ and C influence the results. Furthermore, both should be optimized jointly, since the optimal C depends on the chosen γ.

Single fold and K-fold results are consistent

Classifying Avila features

Since the model provides more complex separation surfaces, it may benefit more than the other approaches from additional data. In this case we may have better final results by re-train the final classifier over all data (K-fold protocol)

Best results are obtained using $\log \gamma = -2$ and $\log C = 2$ (additional values of γ were considered, but, since they provide worse results, they are not reported to avoid cluttering the figures)

Class re-balancing for the primary task (using the same values for C and γ as for the imbalanced model) provides very similar results

Classifying Avila features

The RBF kernel SVM significantly outperforms our previous models on our cross-validation set

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
MVG (Full-Cov)	0.588	1.000	0.969
Quad Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.543	0.987	0.950
Quadratic SVM ($C = 0.1$)	0.552	0.992	0.946
RBF SVM ($\log C = 2$, $\log \gamma = -2$)	**0.458**	**0.911**	**0.911**

Classifying Avila features

Class re-balancing helps for different applications:

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
RBF SVM ($\log C = 2, \log \gamma = -2$)	0.458	0.911	0.911
RBF SVM ($\log C = 2, \log \gamma = -2, \pi_T = 0.5$)	**0.457**	0.882	0.938
RBF SVM ($\log C = 2, \log \gamma = -2, \pi_T = 0.1$)	0.508	**0.863**	0.991
RBF SVM ($\log C = 2, \log \gamma = -2, \pi_T = 0.9$)	0.574	0.985	**0.869**

If we are targeting different priors it may be worth training a task-specific classifier in this case

In the following we select the model trained with balanced classes ($\pi_T = 0.5$)

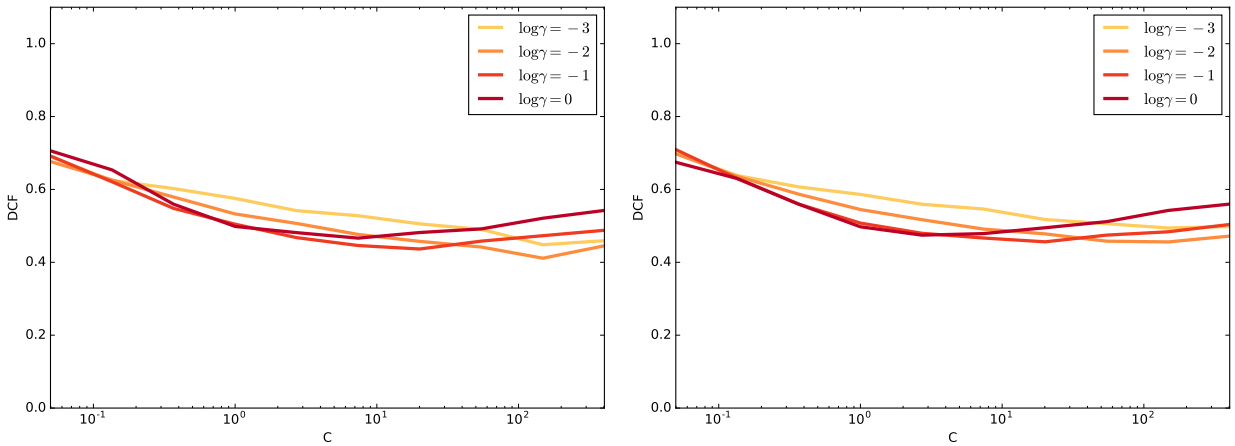
Classifying Avila features

We also repeated the analysis using the raw features. The min DCF results below show the performance of the models that best perform on the validation set ($\log C = 3, \log \gamma = -1$)

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Gaussianized features			
RBF SVM (no balancing)	0.458	0.911	0.911
RBF SVM ($\pi_T = 0.5$)	0.457	0.882	0.938
RBF SVM ($\pi_T = 0.1$)	0.508	0.863	0.991
RBF SVM ($\pi_T = 0.9$)	0.574	0.985	**0.869**
Raw features			
RBF SVM (no balancing)	0.457	0.882	0.938
RBF SVM ($\pi_T = 0.5$)	**0.427**	0.938	0.912
RBF SVM ($\pi_T = 0.1$)	0.530	**0.842**	0.988
RBF SVM ($\pi_T = 0.9$)	0.590	0.953	**0.869**

Classifying Avila features

RBF kernel SVM: min DCF ($\tilde{\pi} = 0.5$) for different values of C and γ. Raw features. Left: single fold. Right: K-fold.



The results are again consistent, although the single-fold case has slightly better performance for $\log \gamma = -2$

Since the K-fold results are probably more reliable, and show that $\log \gamma = -1$ is more robust over a larger set of values of C, we select $\log \gamma = -1$

Classifying Avila features

With the standard, non-balanced SVM formulation results are similar.

The balanced SVM model trained with the raw features achieves slightly better results than the model trained with Gaussianized features.

Currently, our best results on the validation set have been obtained with a class-balanced SVM classifier using a RBF kernel and the raw (non-normalized) features.

Classifying Avila features

The last model we consider is a generative approach based on training a GMM over the data of each class

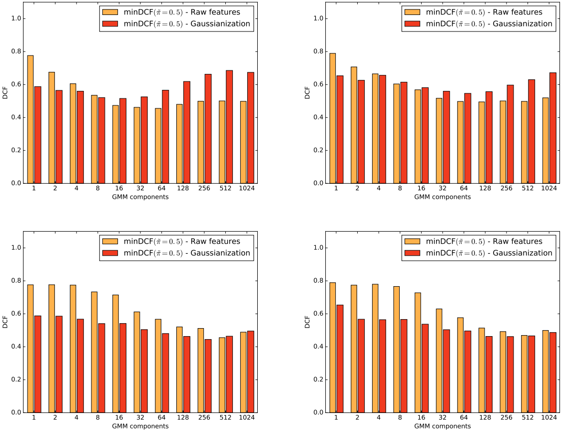
GMMs can approximate generic distributions, so we expect to obtain better results than with the Gaussian model

We consider both full covariance and diagonal models, with and without covariance tying.

For tied covariance models, tying takes place at class level (i.e. different classes have different covariance matrices)

We use the K-fold protocol to select the number of Gaussians and to compare different models

GMM: min DCF($\tilde{\pi} = 0.5$) for different models. Left: full covariance models. Right: diagonal covariance models. Top: non tied covariances. Bottom: tied covariances



Classifying Avila features

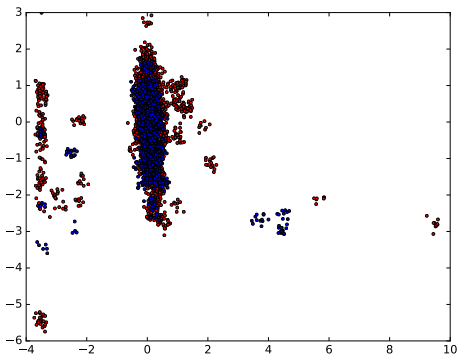
The best non tied models have similar performance, and they show some degree of over-fitting when the number of components becomes large

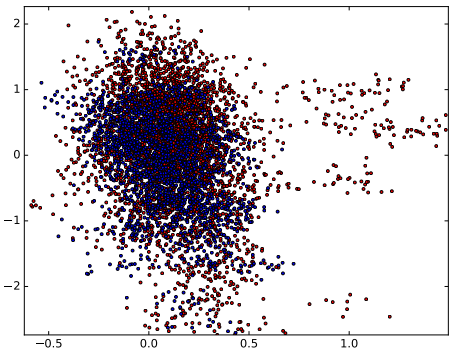
The models trained on raw features show better performance. Further analysis of the dataset shows that, indeed, the data of each copyist presents several clusters that are well separated

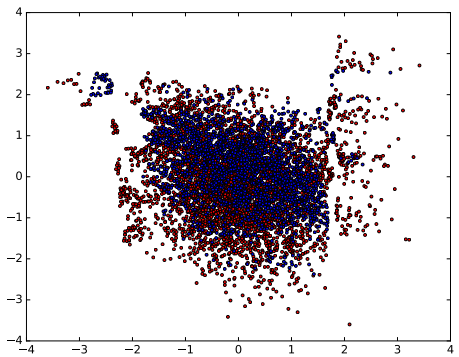
Classifying Avila features

Gaussianization reduce the dynamic range of samples far from the data mean and thus reduces the separability of some clusters. The LBG iterations also split data along different directions, thus finding different clusters. Since each component has its own Gaussian, the model ends up finding clusters that do not correspond to the natural clusters of the dataset.

2-D scatter plot of features 1 and 4. Left: raw features. Middle: raw features - zoom. Right: Gaussianized features







Classifying Avila features

Tied models perform better. This can be explained by the presence of a large number of clusters that have similar within-class variability

With limited number of components, non-tied models do not identify all the clusters. With lots of components, non tied models need to estimate one covariance matrix for each cluster from a very small number of samples

Tied models allow for more robust estimates of the within-class covariance of the clusters, and thus we can employ a larger number of components and identify most of the class clusters

In this case, Gaussianization does not have a detrimental effect for models with large number of components, and the best models are obtained using Gaussianized data.

Classifying Avila features

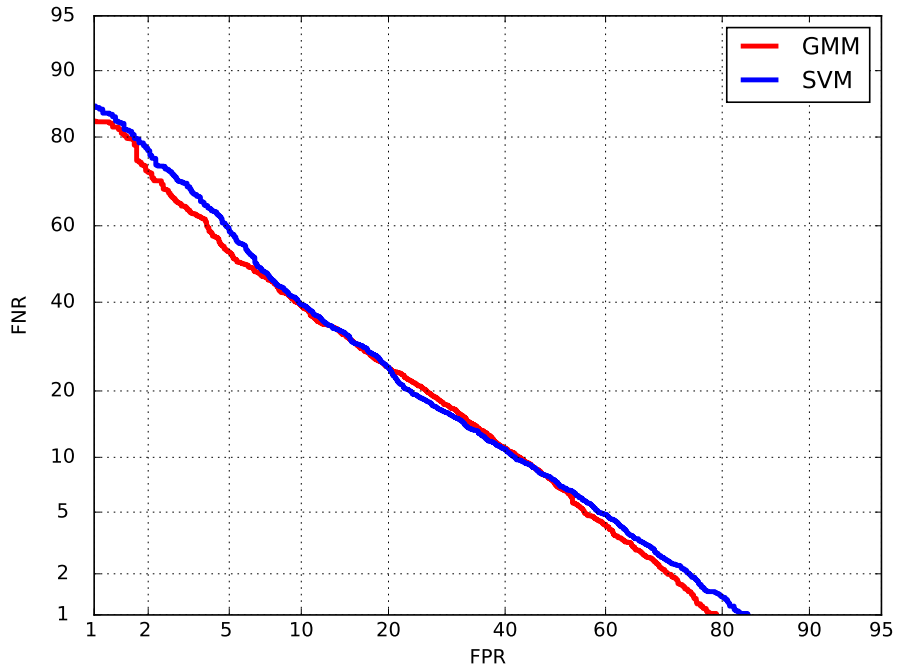
GMM — min DCF on validation set (K-fold protocol)

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Raw features			
RBF SVM ($\pi_T = 0.5$)	**0.427**	0.938	0.912
RBF SVM ($\pi_T = 0.1$)	0.530	**0.842**	0.988
RBF SVM ($\pi_T = 0.9$)	0.590	0.953	0.869
Full cov, 64 Gau	0.497	**0.866**	0.898
Diag cov, 128 Gau	0.495	0.913	0.880
Tied full cov, 512 Gau	0.456	0.891	**0.865**
Tied diag cov, 512 Gau	0.469	0.882	**0.865**
Gaussianized features			
Tied full cov, 256 Gau	**0.445**	0.902	0.872
Tied diag cov, 256 Gau	0.462	0.871	0.871

GMMs perform similarly to SVM for the primary task. For other applications, the same model proves more effective than the SVMs with RBF kernels trained with $\pi_T = 0.5$.

Classifying Avila features

A DET plot (or a ROC plot) confirms that the GMM model is superior to SVM for many operating points, but not for our main application



We select as candidate model the SVM with RBF kernel, trained with balanced classes ($\pi_T = 0.5$)

As secondary system we select the GMM with tied, full covariance matrices and 256 Gaussian components

Classifying Avila features

Up to now we have considered only minimum DCF metrics

Min DCF measures the cost we would pay if we made optimal decisions for the evaluation set using the recognizer scores

The cost that we actually pay, however, depends on the goodness of the decisions we make using those scores (in the binary case, on the goodness of the threshold we use in practice to perform class assignment)

We therefore turn our attention to actual DCFs

Classifying Avila features

We have seen that, if scores are well calibrated, the optimal threshold that optimizes the Bayes risk is $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$

To form decisions from scores we therefore have two options:

- Assume scores are calibrated (either because the recognizer gives calibrated scores or because we re-calibrated the recognizer outputs, see below) and use the theoretical threshold $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$
- Estimate, for the target application, a good threshold (e.g. through a validation set)

Classifying Avila features

We can evaluate the actual DCF (or just DCF) to assess how good the models would be if we were using the theoretical threshold for each application (i.e. we consider the scores as if they were calibrated)

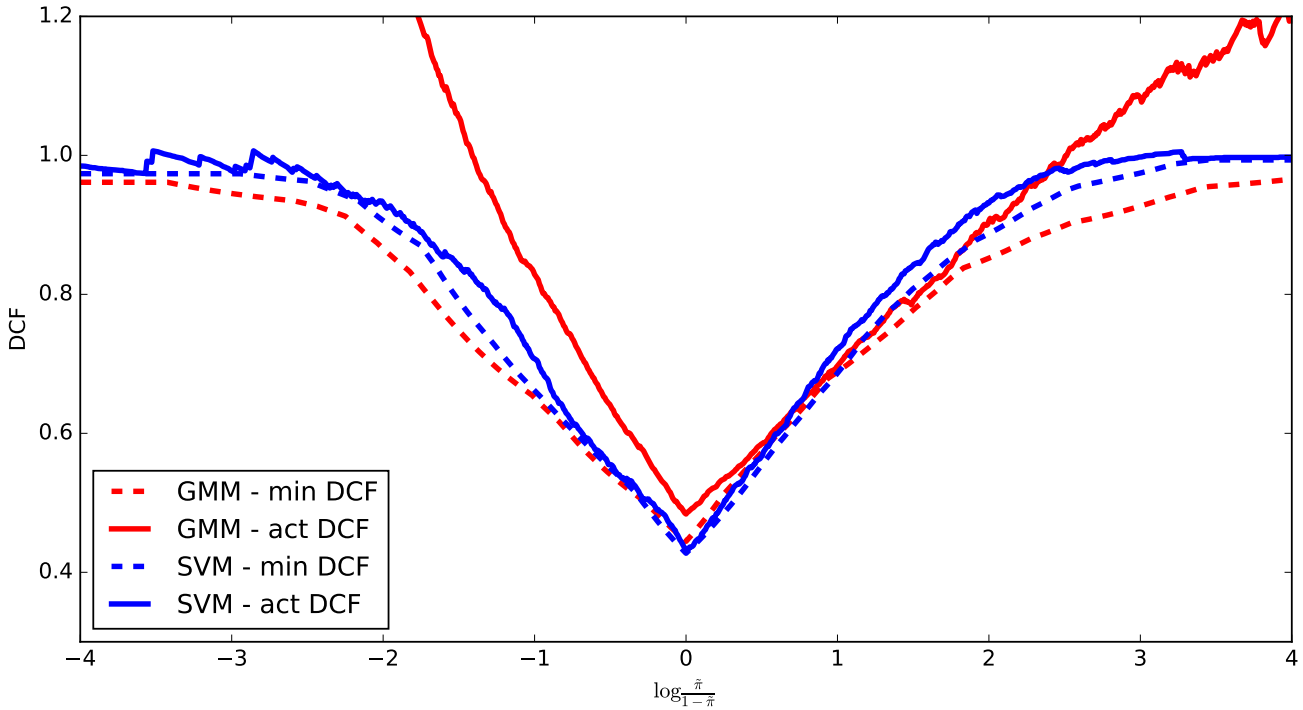
	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
SVM	0.427	0.430	0.938	0.944	0.912	0.955
GMM	0.445	0.484	0.902	1.406	0.872	0.938

We can observe that the SVM provides scores that are almost calibrated for our three applications, despite the model lacking a probabilistic interpretation

The calibration of GMM scores, on the other hand, is quite poor: for the primary application, there is a $\approx 10\%$ loss due to poor decisions, and the calibration loss becomes very large for the $\tilde{\pi} = 0.1$ application

Classifying Avila features

This is confirmed by a Bayes error plot, which shows the DCFs for different applications:



Classifying Avila features

A first solution consists in estimating a (close-to-)optimal threshold for a given application:

- Simple approach: requires taking the threshold corresponding to the minimum of the DCF on the validation set
- Does not produce scores that are well-calibrated: for different applications we have to re-estimate an optimal threshold

We can use the threshold estimated on the validation set (single-fold) or on the cross-validation set (K-Fold)

Classifying Avila features

To assess whether this approach may be effective, we can again employ cross-validation for the threshold-selection problem. Since we are estimating a single number, we simply split the validation scores in two sets. Assuming we adopted the K-fold approach for model estimation, the procedure is as follows:

- Pool the K-fold scores of the classifier (this gives a number of scores equal to the number of samples)
- Shuffle the scores and then split the shuffled scores into 2 partitions (again, we could use a second K-fold approach if needed)
- Estimate the threshold on one partition, apply the threshold on the remaining one and compare minimum and actual DCFs over the latter
- (Optional) Re-estimate the final threshold using all the original scores (this step is necessary if we use a K-fold approach also for threshold estimation, since each K-fold iteration may result in a different threshold — in the following we do not apply this step since we use the single-split approach for threshold estimation)

Classifying Avila features

Results using the estimated DCF threshold t^* for the main application. NOTE: min DCF values are different from what we computed before, since we are using a different set of validation samples (i.e. those of the split of threshold estimation protocol)

	min DCF	act DCF $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$	act DCF t^*
		$\tilde{\pi} = 0.5$	
SVM	0.432	0.433	0.442
GMM	0.457	0.484	0.462
		$\tilde{\pi} = 0.1$	
SVM	0.943	0.954	0.981
GMM	0.880	1.460	0.932
		$\tilde{\pi} = 0.9$	
SVM	0.912	0.929	0.987
GMM	0.882	0.960	0.895

Our procedure looks accurate for the main application. For the other two cases results are mixed: we have an improvement for GMM, but poor results for SVM. Furthermore, for each application we had to re-estimate the threshold

Classifying Avila features

A second approach consists in re-calibrating the scores, i.e. transforming the scores so that the theoretical threshold

$$t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$$

provides close to optimal values over a wide range of effective priors $\tilde{\pi}$

We want to compute a transformation function f that maps the classifiers scores s to well-calibrated scores $s_{cal} = f(s)$

We would like f to be monotone, since we want to preserve the fact that higher non-calibrated scores should favor class $\mathcal{H}_T$ and lower non-calibrated scores should favor class $\mathcal{H}_F$

Classifying Avila features

Several approaches are possible to estimate f:

- Isotonic regression
- Discriminative score models (e.g. logistic regression)
- Generative score models

Since it employs techniques that we have already analyzed in the course, we will focus on the second approach

Classifying Avila features

We assume that function f has a simple form: an affine function

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

Since $f(s)$ should produce well-calibrated scores, $f(s)$ can be interpreted as the log-likelihood ratio for the two class hypotheses

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

The class posterior probability for prior $\tilde{\pi}$ corresponds to

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

Classifying Avila features

If we interpret the scores as features, this has a very similar expression as the log posterior ratio of the logistic regression model

If we let $\beta' = \beta + \log \frac{\tilde{\pi}}{1-\tilde{\pi}}$, then we have exactly the same model.

We can employ the prior-weighted Logistic Regression model to learn the model parameters α, β' over our training scores (we will refer to the set of scores as calibration set in the following)

To recover the calibrated score $f(s)$ we will need to compute

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

Note that we have to specify a prior $\tilde{\pi}$, and we are still effectively optimizing the calibration for a specific application $\tilde{\pi}$. However, as we will see, the model will still provide good calibration for different applications

Classifying Avila features

Results using prior-weighted logistic regression calibration for SVM classifier. Calibration parameters have been estimated on the same split as before. Since the final scores should be calibrated, actual DCFs are computed using the theoretical threshold $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$.

	min DCF ($\tilde{\pi} = 0.5$)	min DCF ($\tilde{\pi} = 0.1$)	min DCF ($\tilde{\pi} = 0.1$)
	0.432	0.943	0.912
	act DCF ($\tilde{\pi} = 0.5$)	act DCF ($\tilde{\pi} = 0.1$)	act DCF ($\tilde{\pi} = 0.1$)
Uncalibrated	0.433	0.954	0.929
Log-Reg $\tilde{\pi} = 0.5$	0.437	0.979	0.920
Log-Reg $\tilde{\pi} = 0.1$	0.434	0.985	0.914
Log-Reg $\tilde{\pi} = 0.9$	0.440	0.978	0.929

Classifying Avila features

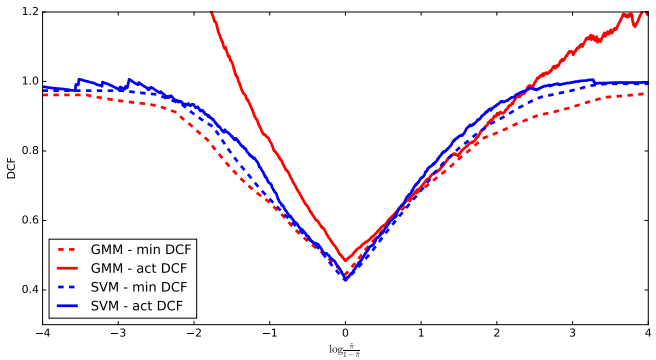
Results using prior-weighted logistic regression calibration for GMM classifier. Calibration parameters have been estimated on the same split as for SVM

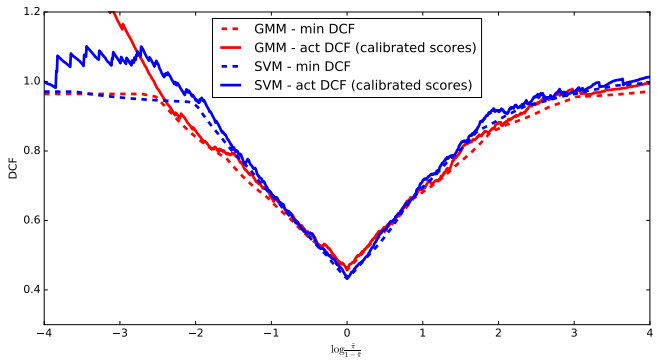
	min DCF ($\tilde{\pi} = 0.5$)	min DCF ($\tilde{\pi} = 0.1$)	min DCF ($\tilde{\pi} = 0.1$)
	0.457	0.880	0.882
	act DCF ($\tilde{\pi} = 0.5$)	act DCF ($\tilde{\pi} = 0.1$)	act DCF ($\tilde{\pi} = 0.1$)
Uncalibrated	0.484	1.460	0.960
Log-Reg $\tilde{\pi} = 0.5$	0.462	0.910	0.898
Log-Reg $\tilde{\pi} = 0.1$	0.460	0.950	0.947
Log-Reg $\tilde{\pi} = 0.9$	0.462	0.893	0.900

Classifying Avila features

We can observe that the Logistic Regression approach obtains accurate results.

The choice of the training prior does not significantly influence the performance, even for applications with a different effective prior. We can check the Bayes error plots for the calibrated models (left: non-calibrated scores; right: calibrated scores, model trained with $\tilde{\pi} = 0.5$)





The model provides scores that are well-calibrated over a wide range of applications (effective priors)

Classifying Avila features

Furthermore, the method allows easily combining the outputs of different classifiers

We can expect that different classifiers agree on some, but not all decisions, especially when classifiers use different approaches and / or different features

Combining the decisions of two classifiers in this case may lead to improvement

Issue: how can we combine the decisions? A simple voting scheme may look enough

However, if one classifier is almost certain about class $\mathcal{H}_T$ and two other classifiers are only slightly in favor of class $\mathcal{H}_F$, which one should we favor? And how do we break ties?

Classifying Avila features

An effective fusion approach consists in performing score-level fusion, rather than decision-level voting

We assume that the fused score is a function of the scores of the different classifiers

If $s_{t,A}$ and $s_{t,B}$ are the scores of classifiers A and B for sample $\boldsymbol{x}_t$, then the fused score for sample $\boldsymbol{x}_t$ will be

$$s_t = f(s_{t,A}, s_{t,B})$$

We will then use the fused score s_t to perform the decision (i.e. to label the sample $\boldsymbol{x}_t$)

Classifying Avila features

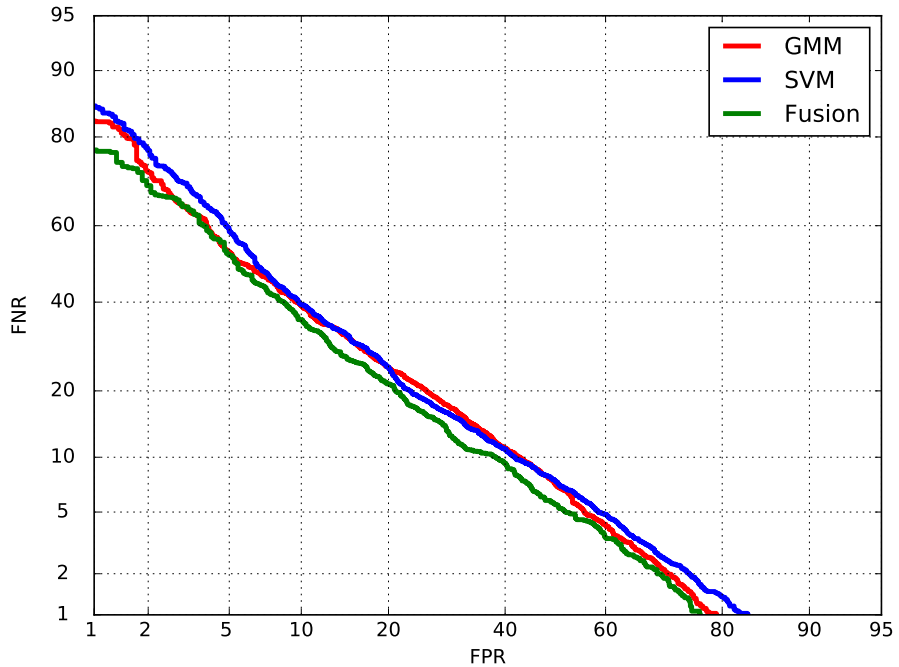
As for calibration, we can assume a simple linear form for f:

$$f(s_{t,A}, s_{t,B}, s_{t,C}, \dots) = \alpha_A s_{t,A} + \alpha_B s_{t,B} + \alpha_C s_{t,C} + \cdots + \beta$$

Again, we can use prior-weighted logistic regression to train the model parameters. The resulting scores will, hopefully, have higher accuracy and benefit from re-calibrating effects of the model

Classifying Avila features

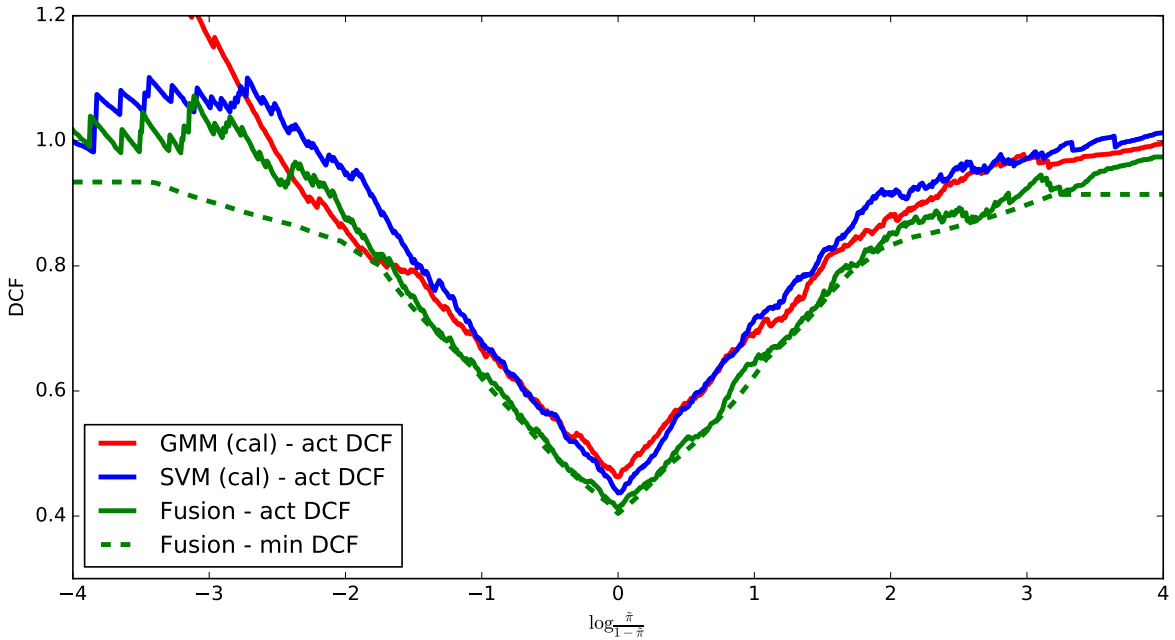
Fusing GMM and SVM scores: DET plot. Training was done using one of two partitions of the scores. The results refer to the other partition. The plot differs from the previous DET plot since we are evaluating on different subset of samples (the validation portion of score calibration training protocol)



Classifying Avila features

DCFs and Bayes error plot for the fused system:

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
SVM	0.432	0.433	0.943	0.954	0.912	0.944
GMM	0.457	0.462	0.880	0.910	0.882	0.898
Fusion	**0.404**	**0.412**	**0.851**	**0.930**	**0.848**	**0.879**



Classifying Avila features

Fusion is effective in reducing minimum costs

The fused system outperforms the individual classifiers on most operating points

The fused scores are also (almost) well-calibrated

Our final model consists thus of a fusion of an RBF-kernel SVM trained over raw features and a GMM trained over Gaussianized features

We now need to assess the quality of our model on held-out data (the evaluation set)

We will verify the performance and analyze (some) of the choices we made, to see how they affected performance for unseen data

Experimental results

We start again from the Gaussian classifiers, and we again evaluate systems in terms of minimum DCFs.

The minimum DCF provides an optimistic estimate of the actual DCF: it's the cost we would have if we were able to select the optimal threshold for the evaluation set

This allows verifying whether the proposed solution is indeed the one that can achieve the best accuracy

We will then assess how good our actual decisions are for the model, i.e. evaluate the actual DCF (we will both consider the case where we estimated an optimal threshold per application on the validation set, and the case where scores were calibrated and the theoretical threshold was then used for different applications)

Experimental results

Below are the results for the Gaussian classifiers using either the single-fold protocol (i.e. the model trained over the 80% data selected for model training) or the 5-fold cross validation protocol (i.e. the final model that was re-trained using the whole dataset)

	80% data			All data		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
NO PCA						
Full-Cov	0.596	1.000	0.981	**0.594**	1.000	0.976
Diag-Cov	0.665	0.996	0.997	0.663	0.995	**0.973**
Tied Full-Cov	0.716	0.997	0.999	0.712	0.997	0.999
Tied Diag-Cov	0.725	0.998	0.998	0.725	0.998	0.998
PCA (m = 9)						
Full-Cov	0.604	0.990	0.990	0.598	0.987	0.986
Diag-Cov	0.666	0.981	0.985	0.670	**0.979**	0.983
Tied Full-Cov	0.724	0.998	0.998	0.721	0.998	0.999
Tied Diag-Cov	0.723	0.998	0.998	0.720	0.998	0.999
PCA (m = 8)						
Full-cov	0.622	0.997	0.980	0.621	0.995	0.981
Diag-Cov	0.677	0.983	0.987	0.677	0.982	0.988
Original features (no Gaussianization)						
Full-Cov	0.699	0.998	0.992	0.729	0.999	0.996

Experimental results

The results are consistent with those obtained on the validation set

We can observe that using 80% of the training data (single-fold validation protocol) or 100% of the training data (K-fold validation protocol) does not significantly change the results: 80% of the data are enough, for these models, to obtain good estimates of the model parameters

The MVG model with full covariance over Gaussianized features is confirmed to be the best option for our primary application

PCA is not effective (although with $m = 9$ it's not detrimental)

Experimental results

The results show that our choice of relying on K-fold protocol was probably a good decision:

- The cross-validation accuracy is close to the evaluation accuracy for the K-fold protocol, which suggests that the training and evaluation populations, in this specific use case, have similar properties
- The single-fold approach provides validation results that are more different than those obtained on the evaluation set. Combined with the previous observation, this seems to suggest that results on the single-fold validation set are less reliable, probably due to the significantly smaller size of the validation set compared to the K-fold approach

Experimental results

We now consider linear logistic regression models with the estimated $\lambda = 10^{-5}$ value.

	80% data			All data		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Raw features						
LR ($\pi_T = 0.5$)	0.689	0.988	0.999	0.693	0.993	0.999
LR ($\pi_T = 0.1$)	**0.688**	0.985	0.999	**0.688**	0.993	0.999
LR ($\pi_T = 0.9$)	0.712	0.990	0.997	0.707	0.993	0.998
Gaussianized features						
LR ($\pi_T = 0.5$)	0.715	0.997	0.999	0.713	0.998	0.999
LR ($\pi_T = 0.1$)	0.706	0.998	1.000	0.704	0.998	1.000
LR ($\pi_T = 0.9$)	0.724	0.998	0.998	0.719	0.999	0.999

Experimental results

Again, results are consistent with our expectations

The linear models have similar performance as the MVG linear classifiers

The raw features, however, perform slightly better than we expected, providing small improvements with respect to the other linear models.

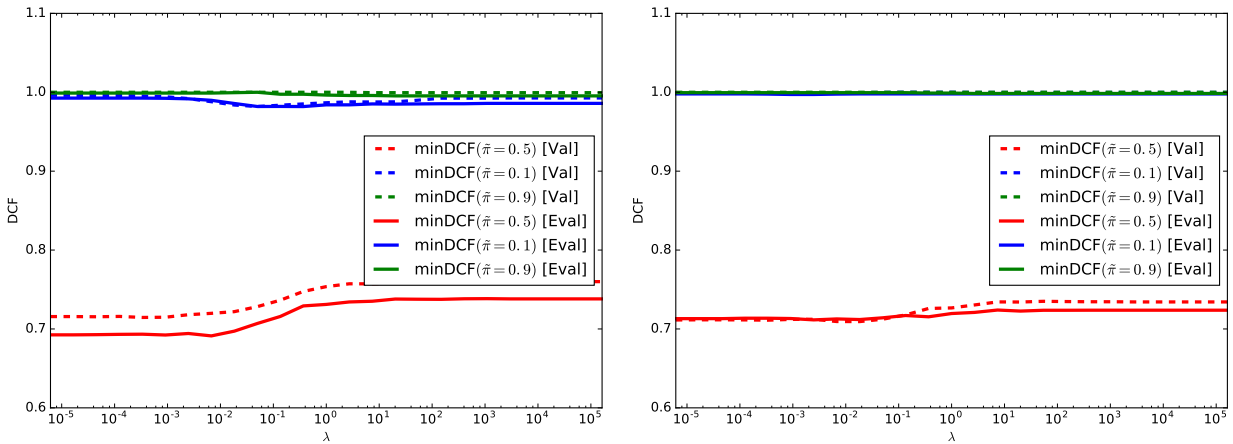
Changing the target prior is not helpful for the imbalanced applications. The best results for our primary application are obtained with $\pi_T = 0.1$, however the difference with respect to models using $\pi_T = 0.5$ is not significant

Both the K-fold and the single-fold models provide the same accuracy. Again, we have enough data to reliably estimate a linear separation rule

Experimental results

Although we do not expect significant deviations, we can also verify whether the chosen λ provides close to optimal results (we consider K-fold models only)

min DCF for different values of λ on the evaluation set. Left: no pre-processing. Right: Gaussianization. Primary metric is the red line



The curves for the validation and evaluation set have the same trend, showing that our choice was indeed effective, although the linear model is, in general, quite poor.

Experimental results

We repeat the analysis for quadratic logistic regression, with the estimated value $\lambda = 10^{-5}$

	80% data			All data		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Raw features						
QLR ($\pi_T = 0.5$)	0.586	0.990	0.983	0.594	0.981	0.989
QLR ($\pi_T = 0.1$)	0.586	0.990	0.989	0.594	0.981	0.989
QLR ($\pi_T = 0.9$)	0.575	0.991	0.959	0.584	0.986	0.970
Gaussianized features						
QLR ($\pi_T = 0.5$)	0.557	0.992	0.972	0.560	0.979	0.978
QLR ($\pi_T = 0.1$)	0.572	0.946	0.989	0.577	**0.945**	0.990
QLR ($\pi_T = 0.9$)	0.553	0.993	0.965	**0.547**	0.996	**0.960**

Experimental results

As before, results on the evaluation set are consistent with those on the validation set

Also in this case, using just 80% of the data would have been sufficient, providing very similar results to the model we chose

As we expected, in this case Gaussianized features perform better

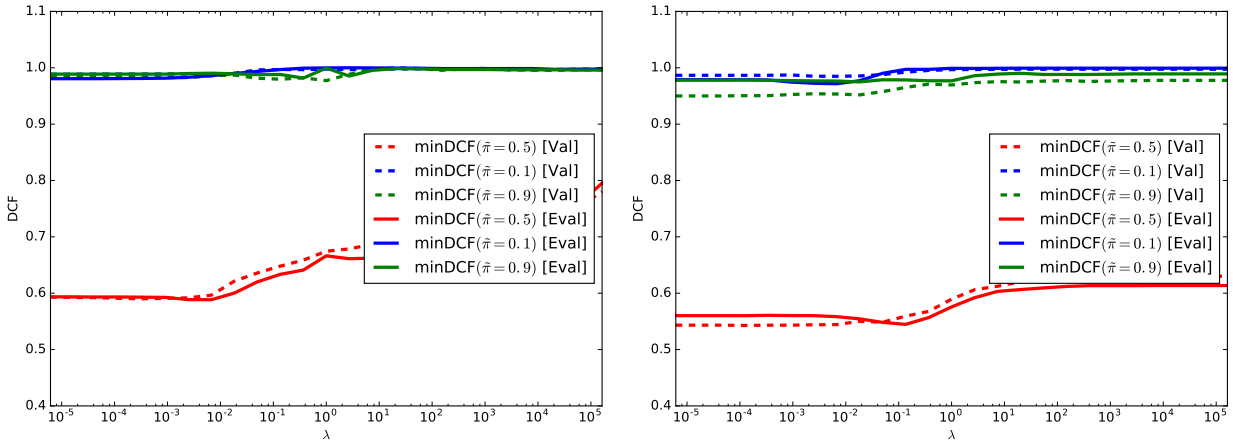
Using different priors may be useful for different applications (but the results are still very poor and very close)

A bit surprisingly, the best results for the primary application are obtained with a different training prior — however, the results with $\pi_T = 0.5$ are still very close

Experimental results

Again, we can verify that our strategy for the estimation of λ was effective:

min DCF for different values of λ on the evaluation set. Left: no preprocessing. Right: Gaussianization. Primary metric is the red line



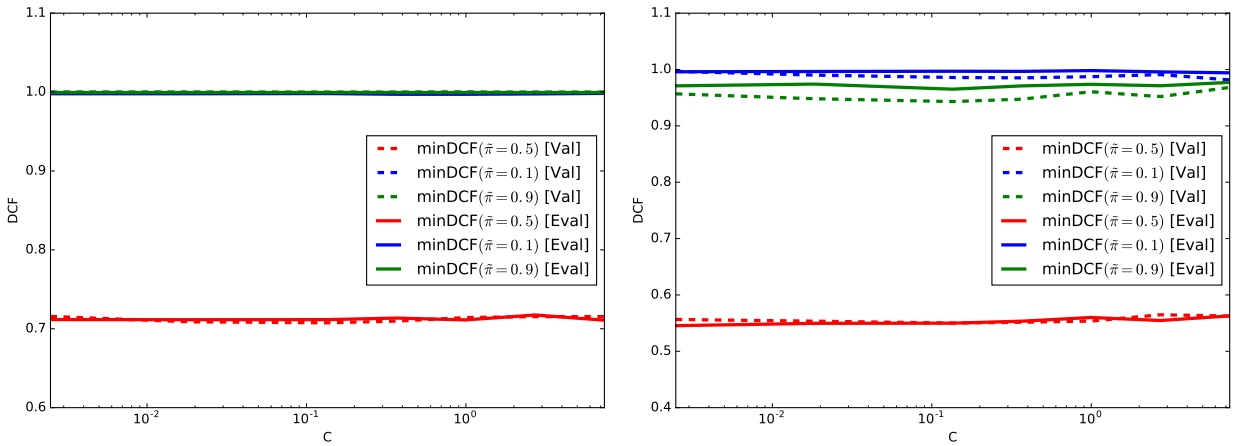
For the Gaussianized features, slightly better results could have been obtained with a larger value of λ. However, the difference is small (min DCF of 0.552 rather than 0.560). We can conclude that our choice was effective also in this case.

Experimental results

We report for completeness also the results of linear and quadratic kernel SVM with Gaussianized features ($C = 0.1$):

	80% data			All data		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Lin. SVM	0.713	0.998	0.999	0.713	0.997	0.999
Lin. SVM ($\pi_T = 0.5$)	0.712	0.997	0.999	0.710	0.997	0.999
Quad. SVM	0.554	0.996	0.972	0.549	0.996	0.976
Quad. SVM ($\pi_T = 0.5$)	0.554	0.995	0.975	0.554	0.996	0.971

min DCF for different values of C. Left: linear kernel; right: quadratic kernel (the y-axis scale differs in the two plots)



Experimental results

The results are in line with expectations

Models perform similarly to logistic regression, the latter being slightly better for our target application

Class re-balancing is not necessary in this case

The choice of hyper-parameter C was effective both for linear and quadratic models

Experimental results

We now turn our attention to the models that are part of our primary system (fusion of RBF kernel SVM and GMM). We start with the RBF kernel SVM. Again, we consider models trained with the values of C and γ estimated on the K-fold validation set.

	80% data			All data		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Gaussianized features						
RBF SVM	0.464	0.957	0.901	0.445	0.940	0.876
RBF SVM ($\pi_T = 0.5$)	0.468	0.918	0.914	0.448	0.870	0.911
RBF SVM ($\pi_T = 0.1$)	0.553	0.878	0.997	0.531	0.858	0.984
RBF SVM ($\pi_T = 0.9$)	0.560	0.995	0.901	0.544	0.993	0.878
Raw features						
RBF SVM (no balancing)	0.470	0.931	0.895	0.449	0.916	**0.871**
RBF SVM ($\pi_T = 0.5$)	0.458	0.912	0.931	**0.438**	0.869	0.912
RBF SVM ($\pi_T = 0.1$)	0.559	0.868	0.986	0.542	**0.844**	0.983
RBF SVM ($\pi_T = 0.9$)	0.598	0.980	0.902	0.561	0.967	**0.871**

Experimental results

We can observe that, in this case, the models trained using all the data perform better than the models trained with just 80% of the date (single-split validation protocol), as we anticipated.

The best results on the evaluation set are again obtained by the class-balanced model with raw features

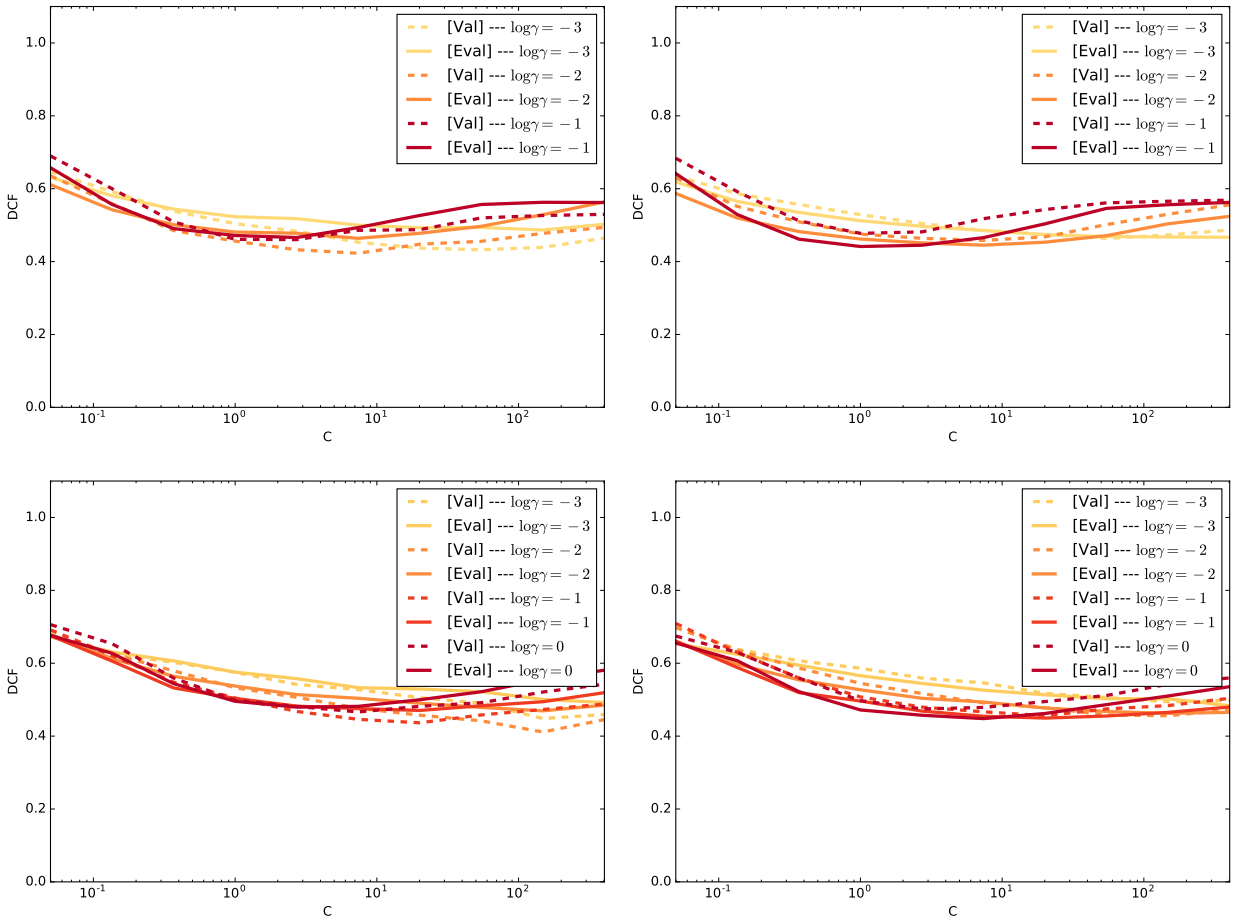
As we have observed on the validation set, using different class-balancing factors is useful for imbalanced applications

The model significantly outperforms the linear and quadratic approaches

Note: the results in the table use the hyper-parameters obtained through the K-fold protocol also for the 80% training data models.

Experimental results

RBF kernel SVM: min DCF ($\tilde{\pi} = 0.5$) for different values of C and γ on the evaluation set. Top: Gaussianized features. Bottom: raw features. Left: single fold. Right: K-fold.



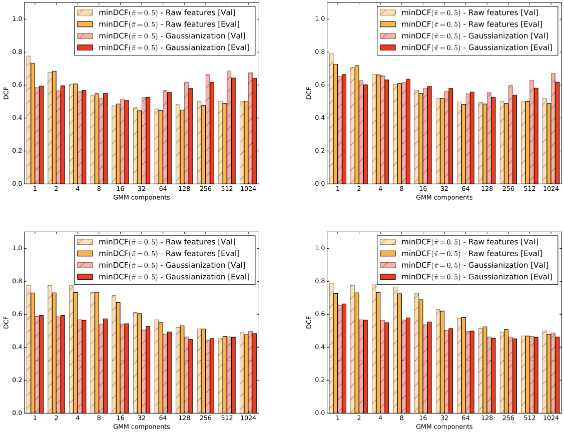
Experimental results

The effects of using different values of C and γ are consistent with what we observed on the validation set (K-fold model). Our choice of γ and C was thus effective.

For the single-fold protocol, we can observe that the values of C and γ are less consistent, and we would have chosen slightly worse values for the hyper-parameters (although the results would still be close to the optimal ones)

Experimental results

We now consider the GMM classifiers. Again, we consider min DCF on the evaluation set of models trained with different number of Gaussians. Left: full covariance models. Right: diagonal covariance models. Top: non tied covariances. Bottom: tied covariances



Experimental results

The validation results are again consistent with the evaluation results

Gaussianization is harmful for non-tied models, but tied models with Gaussianized features can obtain better results

The optimal number of components is similar between evaluation and validation set:

- For diagonal models, the 256-components version achieves the best results
- For full-covariance models, the 128-components model is slightly better, but the results of the 256-components are very close (min DCF of 0.453 vs 0.447)

Experimental results

Below the results using different proportions of training data for the tied models with Gaussianized features (based on validation results we selected the full-covariance model with 256 components)

	80% data			All data		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
256-G Diag	0.486	0.900	0.899	0.452	**0.881**	**0.871**
256-G Full	0.485	0.914	0.893	0.453	0.907	0.892
128-G Full	0.489	0.947	0.926	**0.447**	0.941	0.892

We can observe that using the whole dataset improves significantly the accuracy

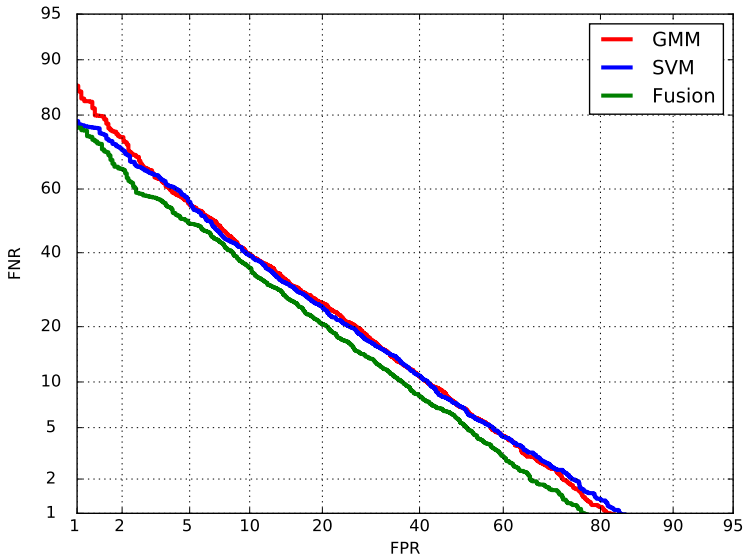
The diagonal models perform similarly for the primary task, but would have probably been a better choice, considering the slightly better results for imbalanced classes (however, the difference with respect to the 256-G model we chose is again not very significant)

Experimental results

Finally, we can evaluate the effectiveness of our model fusion

min DCF on the evaluation set for the fused system

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
SVM	0.438	0.869	0.912
GMM	0.453	0.907	0.892
Fusion	**0.403**	**0.808**	**0.838**



Fusion is effective also on the evaluation set, and allows improving results on a wide range of operating points (applications)

Experimental results

Up to now we have analyzed the systems on the evaluation set in terms of minimum DCFs

Minimum DCFs, as we said, measure the potential capabilities of our systems to produce good decisions

However, in practice we have to make decisions without knowing what is the optimal rule for the evaluation data — in our case, we need to map our scores to class labels, but we do not know the optimal score threshold

We want therefore to assess how good are the decisions that we are actually able to make using the recognizer scores

To evaluate the decision making capabilities of our systems, we consider actual DCFs (or just DCFs), which represent the (normalized) Bayes cost we would pay for our actual decisions

Classifying Avila features

As we mentioned, actual DCFs are computed using the actual decisions (i.e. the actual class predictions) we make for evaluation data

Class predictions require that we compare scores with a threshold. If scores are calibrated, then we have shown that the optimal threshold depends only on the cost of errors and the class priors through $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$, where $\tilde{\pi}$ is the effective prior for the $\mathcal{H}_T$ class

If scores are not calibrated, then we need to use a threshold t^*, different for each application, estimated with some criterion (e.g. taking the threshold that gives the minimum DCF for an application on the validation set)

Experimental results

In the previous part of the lecture we have considered both threshold optimization on validation sets and a score calibration approach

We have seen that the first approach is already effective on the validation data, but the second approach provides better results and produces scores that are calibrated over a wide range of applications

We can compare the results of the two methods on the evaluation set (t^* is the threshold estimated on the validation set as discussed before)

For score calibration, we select the models trained with target application $\tilde{\pi} = 0.5$

Experimental results

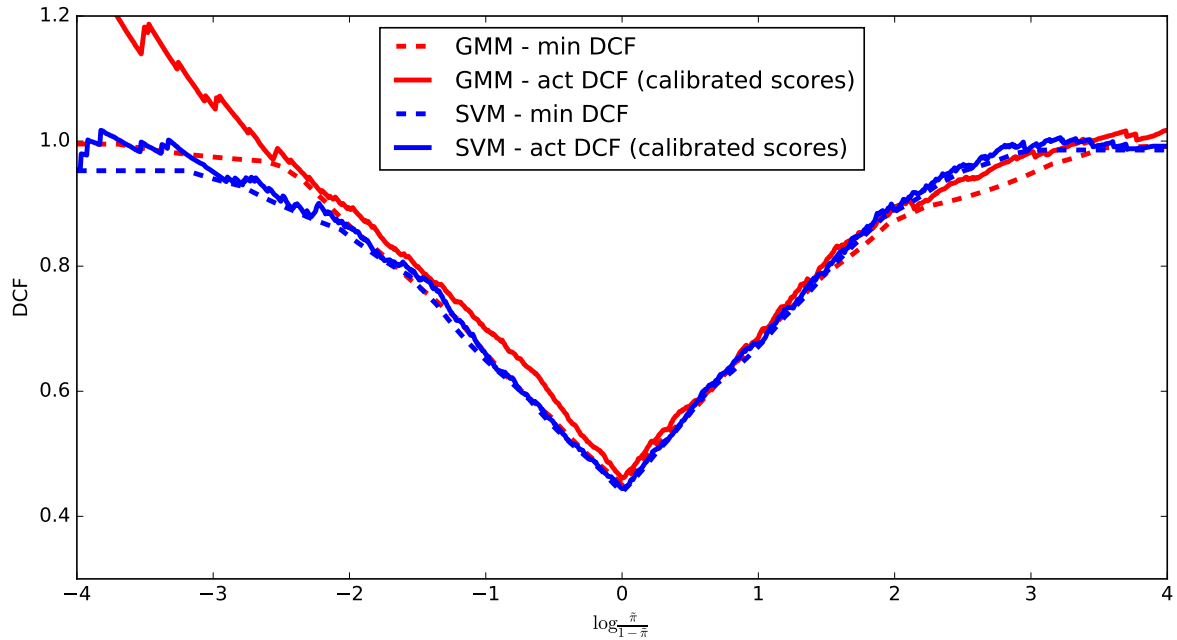
Minimum DCFs and actual DCFs obtained using different approaches to map scores to decisions

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
minimum DCF			
SVM	0.438	0.869	0.912
GMM	0.453	0.907	0.892
raw scores - actual DCF $t = t^*$ (re-estimated for each application)			
SVM	0.443	0.904	0.922
GMM	0.455	0.924	0.898
calibrated scores - actual DCF $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$			
SVM	0.444	0.880	0.917
GMM	0.461	0.942	0.902

Experimental results

Both approaches are quite effective in this case. The actual DCFs are almost always very close to the optimal min DCFs

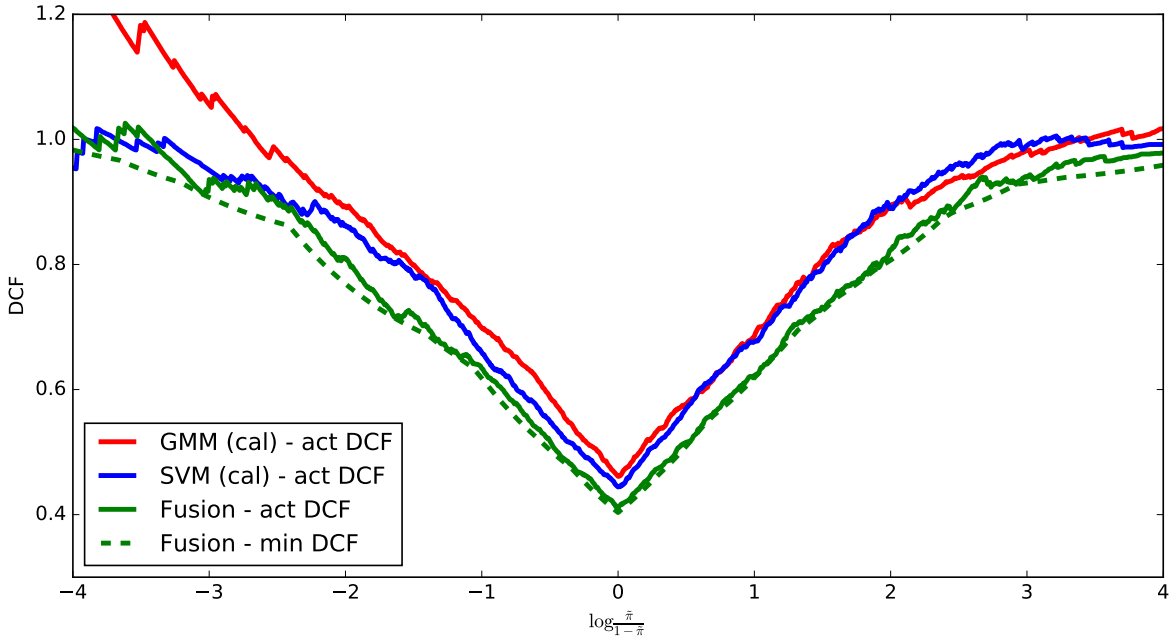
Score calibration, however, does not require re-estimation of a different threshold for other applications, as scores are well-calibrated over a wide range of applications, as can be seen from the Bayes error plot:



Experimental results

Since our fusion approach is supposed to produce well-calibrated scores, we only have to assess whether the actual DCFs computed using the theoretical thresholds are close to the minimum ones. Indeed:

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
SVM (cal.)	0.438	0.444	0.869	0.880	0.912	0.917
GMM (cal.)	0.453	0.461	0.907	0.942	0.892	0.902
Fusion	**0.403**	**0.411**	**0.808**	**0.845**	**0.838**	**0.859**



Experimental results

Concluding, our approach, consisting of the fusion of two sub-systems, is effective, and produces well-calibrated scores for a wide range of applications.

We are able to achieve a DCF cost of ≈ 0.4 for the primary application $\tilde{\pi} = 0.5$

However, the models are relatively ineffective for applications with imbalanced costs and prior proportions

Overall, the similarity between validation and evaluation results suggests that the evaluation population is sufficiently similar to the training population

The choices we made on our training / validation sets proved effective also for the evaluation data