Project - Image detection

Sandro Cumani

sandro.cumani@polito.it

Politecnico di Torino

Outline

- Introduction
- Features
- Building a classifier for the task
- Experimental validation
- Conclusions

Introduction

Object detection task

Detect whether an image depicts a cat

Training set consisting of images of different animals (cats, dogs, rabbits, ...)

Overall, there are 300 images of cats, and 2374 images of other animals (out of 10 possible species)

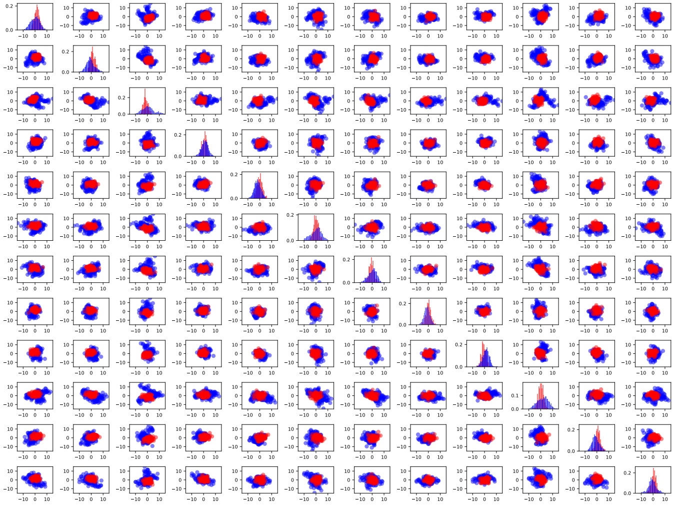
Features consist of 12-dimensional image embeddings

Two possible target applications: $(\pi = 0.5, C_{fn} = 1, C_{fp} = 1)$ and $(\pi = 0.1, C_{fn} = 1, C_{fp} = 1)$

Primary metric: average cost for the two working points

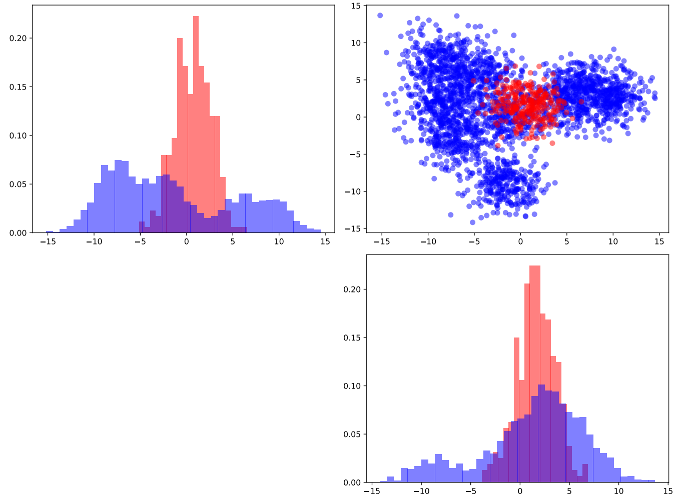
Features

Histogram and 2D scatter plots of the dataset features



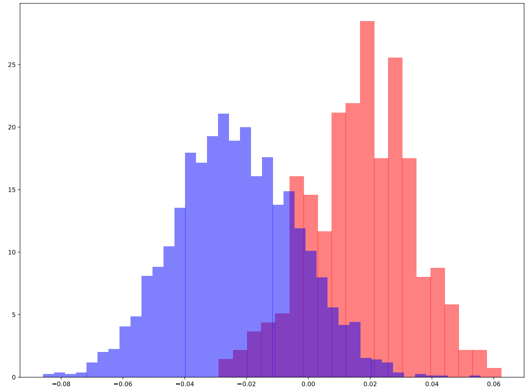
Features

Histogram and 2D scatter plots of the dataset features - principal components



Features

Histogram of dataset features - LDA direction



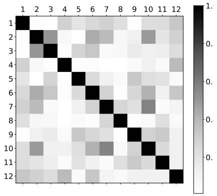
Features

- Non-target class characterized by multiple clusters
- Target class might be modeled well by a simple Gaussian distribution
- Gaussian densities may not be sufficient for non-target class
- LDA shows that a linear classifier may be able discriminate the classes to some degree, but, given the distribution of the features we observed in the scatter plots, we expect non-Gaussian and/or non-linear models (e.g. MVG) to be significantly better for the task

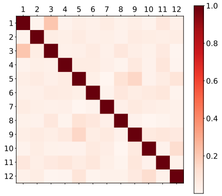
Features

Pearson correlation coefficient for the dataset features

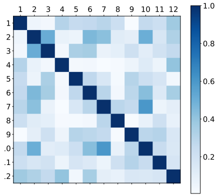
(a) Dataset



(b) Target class



(c) Non-target class

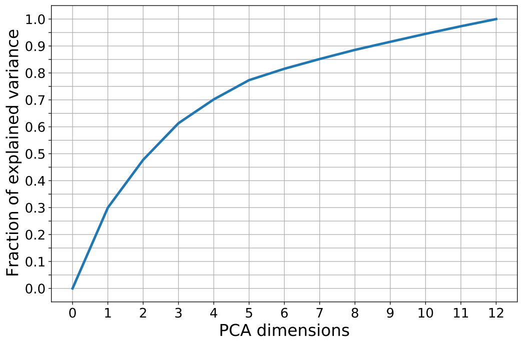


Target class features are weakly correlated, whereas non-target class shows significantly larger correlations.

While PCA may remove some correlation, the number of dimensions is small, thus it may remove useful information that characterizes the target class

Features

PCA — explained variance



With 11 directions we explain about 97% of the dataset variance. We obtain 95% with 10 directions, and 90% with 9 directions. To start, we will consider these three values for PCA.

Training protocol

We adopt a K-Fold protocol with $K = 5$

Our application is a mixture of two working points — we have to build a system that works well for both applications at the same time

For the moment, we measure performance in terms of minimum costs

We compute the minDCF of both working points, and their average (our primary metric min C_{prim})

We will evaluate actual DCFs (and actual C_{prim}) and score calibration once we have selected the most promising model

Gaussian classifier

The covariance matrices of the two classes are different, the target class has almost diagonal covariance but the non-target shows some correlation between features

This suggests that a MVG model should be more appropriate

However, we also have limited amount of samples for the target class

Tied and Naive Bayes models may benefit from the lower risk of overfitting, although the Tied assumptions seem quite inaccurate

We test all three approaches

We also analyze the effects of applying PCA

Gaussian classifier

MVG classifier — min DCF (K-Fold)

PCA	minDCF($\tilde{\pi} = 0.5$)	minDCF($\tilde{\pi} = 0.1$)	min C_{prim}
—	0.059	0.234	0.146
11	0.061	0.223	0.142
10	0.057	0.211	0.134
9	0.058	**0.203**	**0.131**

PCA seems effective, so we try further reducing the number of dimensions

PCA	minDCF($\tilde{\pi} = 0.5$)	minDCF($\tilde{\pi} = 0.1$)	min C_{prim}
8	**0.055**	0.209	0.132
7	0.065	0.240	0.152

9 PCA dimensions seems to be most effective for MVG (but 8 and 10 dimensions are very close)

Gaussian classifier

Tied MVG classifier — min DCF (K-Fold)

PCA	minDCF$(\tilde{\pi} = 0.5)$	minDCF$(\tilde{\pi} = 0.1)$	min C_{prim}
—	0.281	**0.633**	0.457
11	0.286	0.637	0.461
10	0.280	0.640	0.460
9	**0.269**	0.642	**0.455**
8	0.274	0.651	0.462
7	0.310	0.811	0.560

As we anticipated, the linear model can separate the classes, but has significant worse performance than MVG. In this case PCA does not bring significant improvement

Gaussian classifier

Since our PCA implementation also diagonalizes the dataset covariance matrix, we also try applying the diagonalization transformation without actually reducing the dimensionality[†]

Naive Gaussian classifier — min DCF (K-Fold)

PCA	minDCF$(\tilde{\pi} = 0.5)$	minDCF$(\tilde{\pi} = 0.1)$	min C_{prim}
—	0.072	0.296	0.184
[†]12	0.055	0.203	0.129
11	0.052	0.202	0.127
10	0.053	**0.199**	0.126
9	**0.050**	**0.199**	**0.125**
8	0.052	0.200	0.126
7	0.062	0.231	0.146

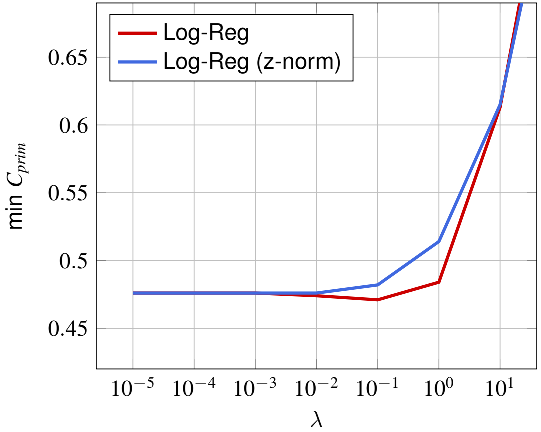
The Naive assumption proves to be effective, with slight improvements over MVG. We attribute this to the low number of samples for the target class. In this case, reducing the dimensionality does not seem particularly important.

Logistic Regression classifier

We now consider Logistic Regression models

We expect the linear model to perform poorly

In any case, we start analyzing the linear classifier, without PCA

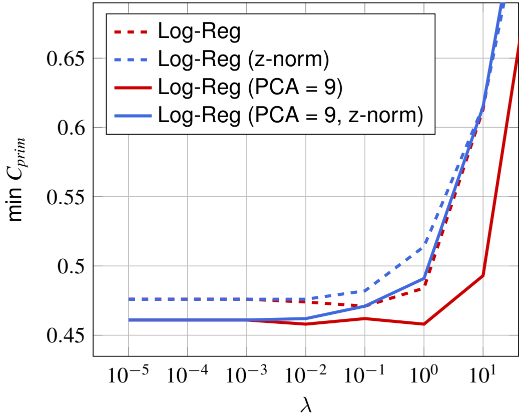


Logistic Regression classifier

As expected, the linear model performs poorly

Regularization does not seem particularly effective

Reducing dimensionality with PCA slightly improves results, but the model performs poorly

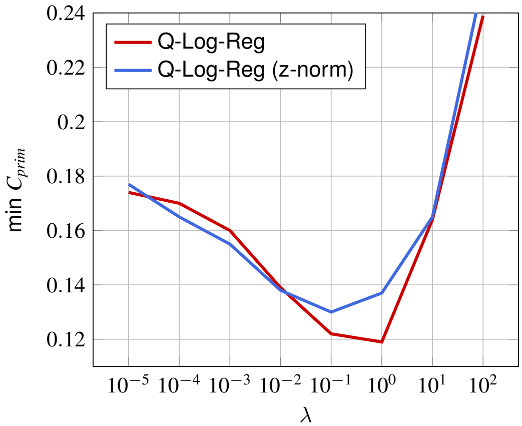


Logistic Regression classifier

Quadratic models are probably more suited for the task

Let's try quadratic feature expansion with Logistic Regression

We start analyzing the model without PCA



Logistic Regression classifier

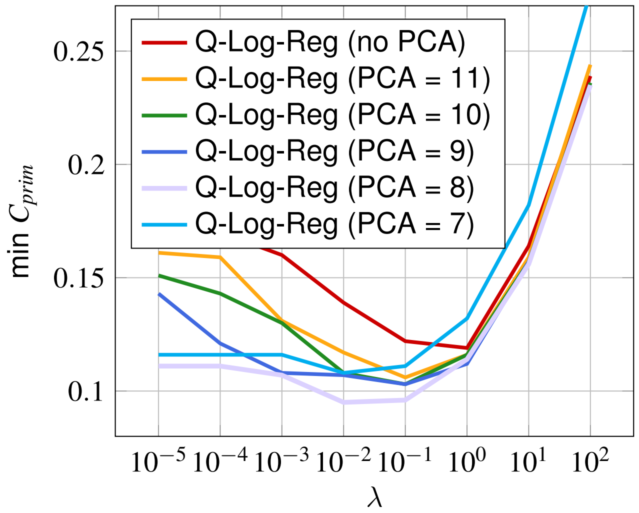
Results are significantly better, and comparable to those of Gaussian models

Since PCA improved results both for linear Log-Reg and Gaussian models, we analyze the effects of PCA

We adopt a grid search approach

Since Z-norm does not seem to be effective, for the moment we do not apply the normalization

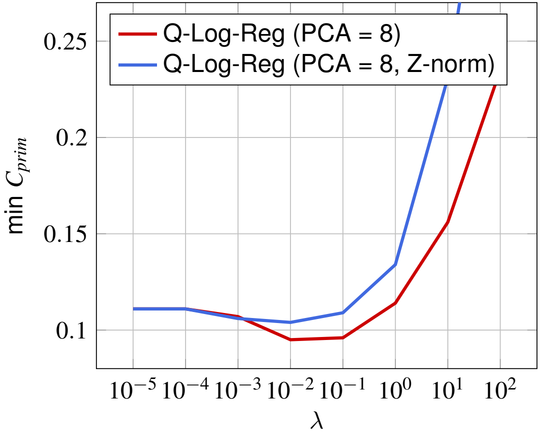
Logistic Regression classifier



PCA (applied before feature expansion) is effective also in this case, with 8 dimensions providing the best results

Logistic Regression classifier

We also analyze whether Z-norm may help for this configuration (we follow a greedy approach and we do not test all combinations again)



Also in this case Z-norm does not seem effective

Logistic Regression classifier

The Q-Log-Reg model has been trained with the empirical prior of the training set, corresponding to $\pi_T \approx 0.11$

The best results correspond to a model trained with PCA = 8 and $\lambda = 0.01$

Q-Log-Reg — min DCF (K-Fold)

minDCF($\tilde{\pi} = 0.5$)	minDCF($\tilde{\pi} = 0.1$)	min C_{prim}
0.054	0.136	0.095

Since we have two operating points, we can try to re-balance the classes for a target application in the range $\pi_T \in [0.1, 0.5]$

Notice that, in any case, we must decide for a single model (i.e. choose a single training prior for the model)

Logistic Regression classifier

Prior-weighted Quadratic Logistic Regression (we show only the results corresponding to the best value of the regularization coefficient λ^*)

Q-Log-Reg (PCA = 8) — min DCF (K-Fold)

Prior	λ^*	minDCF($\tilde{\pi} = 0.5$)	minDCF($\tilde{\pi} = 0.1$)	min C_{prim}
†‡Emp. (Fold)	0.01	0.054	0.136	0.095
‡Emp. (Dataset)	0.1	0.051	0.139	0.095
0.1	0.1	**0.049**	0.139	0.094
0.2	0.1	0.051	**0.132**	**0.092**
0.5	0.1	0.055	0.140	0.098

† May be slightly different between the different fold iterations due to random sampling

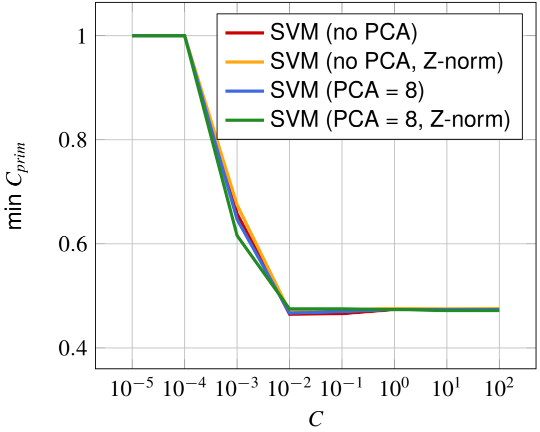
‡ The final model would be the same if trained with the same λ

Using a prior $\pi_T = 0.2$ provides slight improvement. A bit surprisingly, a matched prior does not provide always the best results for a specific working point (but results are nevertheless very close, so the differences may not be significant)

Logistic Regression classifier

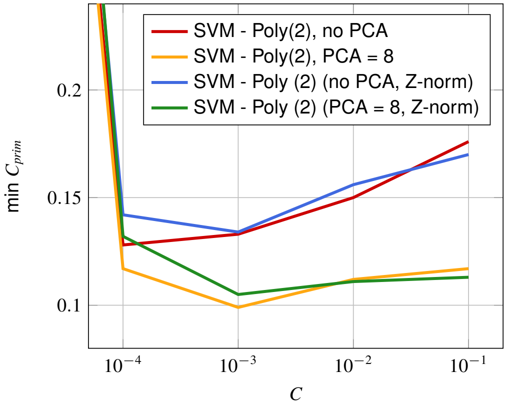
Up to now the best model is Q-Log-Reg, with PCA = 8 and prior-weighting with a prior $\pi_T = 0.2$

We now consider SVM models. We briefly analyze linear SVM, but we expect results to be poor. This is confirmed experimentally:



Support Vector Machine

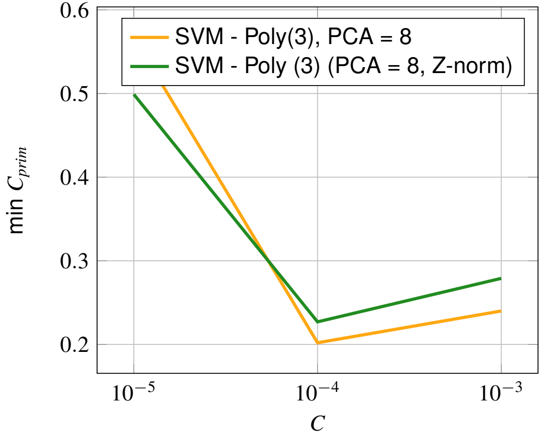
We move to kernel SVM. We start from polynomial kernels. For the moment we consider only models without PCA and with PCA = 8. Since re-balancing was not particularly effective for Log-Reg, for the moment we consider the standard model without class balancing.



Again, PCA = 8 without Z-norm gives the best results

Support Vector Machine

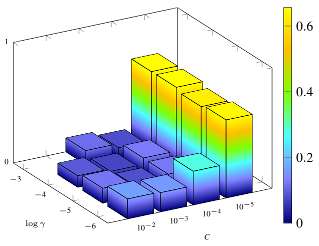
Degree 3 polynomial kernel (we try only C values around the supposed best value, based on previous results):



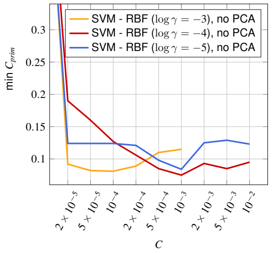
Degree 3 kernel shows more overfitting, and provides worse results, thus we drop it.

Support Vector Machine

RBF Kernel



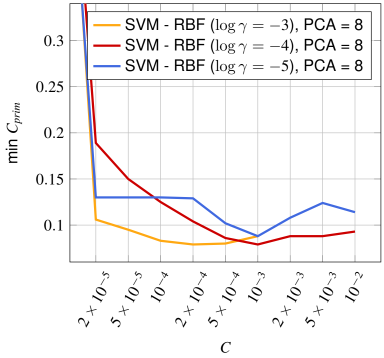
$\longrightarrow$



RBF kernel improves performance. We adopted a grid search, with refinement over the most promising points to select the hyperparameters.

Support Vector Machine

Since PCA improved the performance, we also analyze PCA models. We start the grid search around the parameters that provided the best model for raw features. Since SVM training is expensive, we restrict the analysis to 8 PCA dimensions.



PCA does not seems to be effective with the RBF kernel.

Support Vector Machine

SVM classifier — min DCF (K-Fold)

Pre-proc.	Kernel	C^*	minDCF($\tilde{\pi} = 0.5$)	minDCF($\tilde{\pi} = 0.1$)	min C_{prim}
PCA (8)	Poly ($d = 2$)	10^{-3}	0.046	0.152	0.099
PCA (8)	RBF ($\log \gamma = -3$)	2×10^{-4}	0.049	**0.109**	0.079
PCA (8)	RBF ($\log \gamma = -4$)	10^{-3}	0.039	0.118	0.079
—	RBF ($\log \gamma = -3$)	10^{-4}	0.047	0.115	0.081
—	RBF ($\log \gamma = -4$)	10^{-3}	**0.037**	0.114	**0.075**

Alternative approaches	minDCF($\tilde{\pi} = 0.5$)	minDCF($\tilde{\pi} = 0.1$)	min C_{prim}
Gaussian (best model)	0.050	0.199	0.125
Q-Log-Reg (best model)	0.051	0.132	0.092

SVM with RBF kernel, without pre-processing, is the most promising model for the moment.

Support Vector Machine

We now analyze whether re-balancing the objective function would help. We adopt a greedy approach and fix $\log\gamma = -4$ for RBF, while we perform a grid search for hyperparameter C (we report only results corresponding to the best C on the validation set)

SVM classifier — min DCF (K-Fold)

Prior	C^*	minDCF$(\tilde{\pi} = 0.5)$	minDCF$(\tilde{\pi} = 0.1)$	min C_{prim}
Polynomial (degree 2) kernel - PCA = 8				
Emp. (fold)	10^{-3}	0.046	0.152	0.099
Emp. (dataset)	10^{-3}	0.046	0.151	0.098
0.1	10^{-3}	0.047	0.149	0.098
0.2	10^{-3}	0.049	0.152	0.100
0.5	10^{-2}	0.050	0.162	0.106
RBF Kernel ($\log\gamma = -4$) - no PCA				
Emp. (fold)	10^{-3}	**0.037**	0.114	**0.075**
Emp. (dataset)	10^{-3}	0.038	**0.113**	**0.075**
0.1	10^{-3}	0.042	0.110	0.076
0.2	2×10^{-4}	0.040	0.116	0.078
0.5	2×10^{-5}	0.054	0.171	0.112

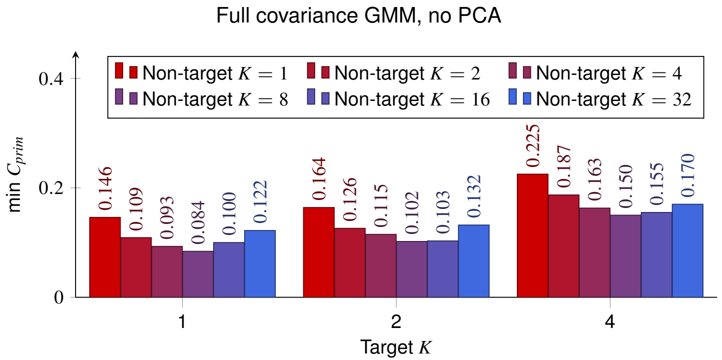
Re-balancing does not seem effective for SVM in this task

Gaussian Mixture Models

Finally, we consider GMM classifiers.

Since the target class samples seem well described by Gaussian densities, whereas the non-target class is composed is different sub-classes, we consider models with different number of components for the target and non-target classes.

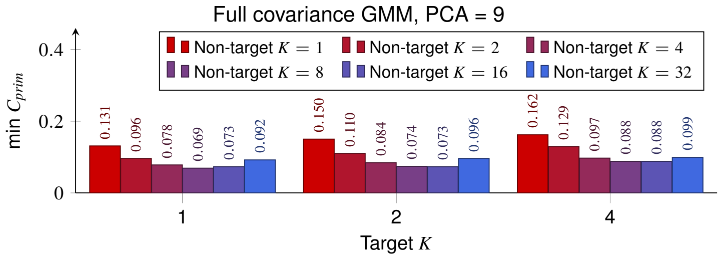
We expect fewer components for the target class to provide better models.



Gaussian Mixture Models

GMM — min DCF (K-Fold) — best configuration for different PCA dimensions

PCA	Tar K	Non Tar K	minDCF($\tilde{\pi} = 0.5$)	minDCF($\tilde{\pi} = 0.1$)	min C_{prim}
—	1	8	0.041	0.128	0.084
11	1	8	0.037	0.111	0.074
10	1	8	0.035	0.111	0.073
9	1	8	**0.036**	**0.103**	**0.069**
8	1	8	0.038	0.122	0.080



Using PCA to reduce the feature space to 9 dimensions seem to be the most effective strategy for Full Covariance GMM

Gaussian Mixture Models

Since diagonal Gaussian models performed better than MVG models (the target class has few samples), we also try different combinations for both the target and non-target densities.

For each class, we consider both Full Covariance (FC) and Diagonal (D) models

For the non-target class, we also consider Tied Full Covariance (FC-T) and Tied Diagonal models (D-T), since clusters seem to have similar distributions

Due to the time required for train all the models, we focus on using just 9 PCA dimensions (but the optimal dimensionality may depend on the model)

Gaussian Mixture Models

We report only the best results for each model

We expect few (even 1) components to be enough for the target class

Diagonal GMMs may provide better results

For Tied and Diagonal GMM it's worth trying to increase the components also for the target class, thus, for all models, we consider ranges from 1 to 8 components for the target class

We do not apply covariance thresholding, since the number of components we consider is small (but should we find that simple models would show numerical issues we may reconsider this choice)

Gaussian Mixture Models

GMM — min DCF (K-Fold) — best configuration for different density models

Tar K	Non Tar K	minDCF$(\tilde{\pi} = 0.5)$	minDCF$(\tilde{\pi} = 0.1)$	min C_{prim}
1 (FC)	8 (FC)	0.036	0.103	0.069
1 (FC)	16 (D)	0.039	0.095	0.067
1 (FC)	16 (FC-T)	**0.038**	0.102	0.070
1 (FC)	32 (D-T)	0.039	0.101	0.070
2 (D)	8 (FC)	0.042	**0.087**	**0.064**
1 (D)	16 (D)	0.039	0.090	0.065
2 (D)	16 (FC-T)	0.039	0.089	**0.064**
2 (D)	16 (D-T)	0.040	0.089	**0.064**

Gaussian Mixture Models

The results are in line with our expectations

Few (1 or 2) components are enough for the target class (diagonal models can exploit the additional components to capture some correlations between features)

Modeling the target class with diagonal covariance matrices provides better results, probably because the target class contains few samples

For the non-target class, both diagonal and tied full covariance models are effective, but require slightly more components than the full covariance model

Tied models exploit our observation that clusters have similar distribution

Overall, results for different configurations are very similar

We select the 2 (D) - 16 (FC-T) configuration as best model

Calibration and fusion

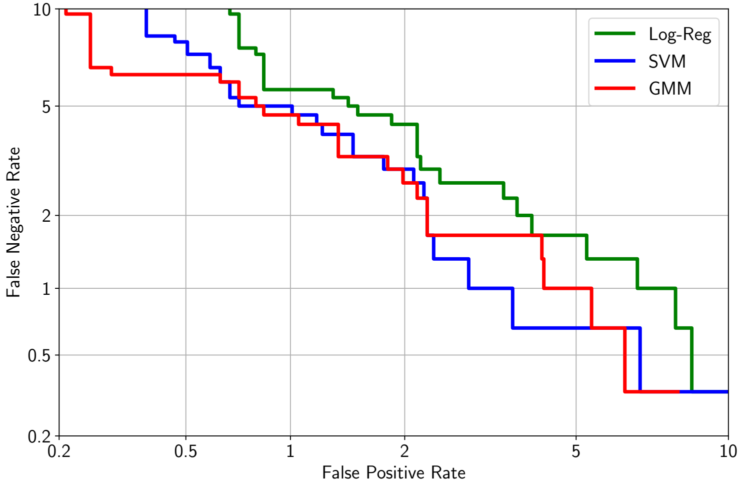
Summarizing, our best models are

	minDCF ($\tilde{\pi} = 0.5$)	minDCF ($\tilde{\pi} = 0.1$)	min C_{prim}
Quadratic Log-Reg $\pi_T = 0.2, \lambda = 0.1$, PCA = 8	0.051	0.132	0.092
RBF SVM $\log \gamma = -4, C = 10^{-3}$	**0.037**	0.114	0.075
GMM — PCA = 9 Tar: 2C Diag Non-tar: 16C Tied Full Cov	0.039	**0.089**	**0.064**

We do not consider quadratic SVM, since the model is similar to, but slightly worse than, quadratic Logistic Regression

Calibration and fusion

We can compare the performance of the three models with a DET plot:



Calibration and fusion

We observe that our GMM is overall the best approach on the validation set for our applications

The RBF SVM obtains slightly worse results. It's slightly more effective for the $\tilde{\pi} = 0.5$ working point, but significantly worse on the $\tilde{\pi} = 0.1$ working point

However, for applications that require lower FNR, the SVM models seems to be more effective than GMM (although, given the low number of errors, the differences may not be very significant)

We now consider calibration of these models, and analyze whether score-level fusion may provide additional improvement

Calibration and fusion

Minimum and actual costs — raw scores

	minDCF $(\tilde{\pi} = 0.5)$	minDCF $(\tilde{\pi} = 0.1)$	min C_{prim}	actDCF $(\tilde{\pi} = 0.5)$	actDCF $(\tilde{\pi} = 0.1)$	C_{prim}
Q-Log-Reg	0.051	0.132	0.092	0.067	0.260	0.164
RBF SVM	0.037	0.114	0.075	0.803	1.000	0.902
GMM	0.039	0.089	0.064	0.040	0.094	0.067

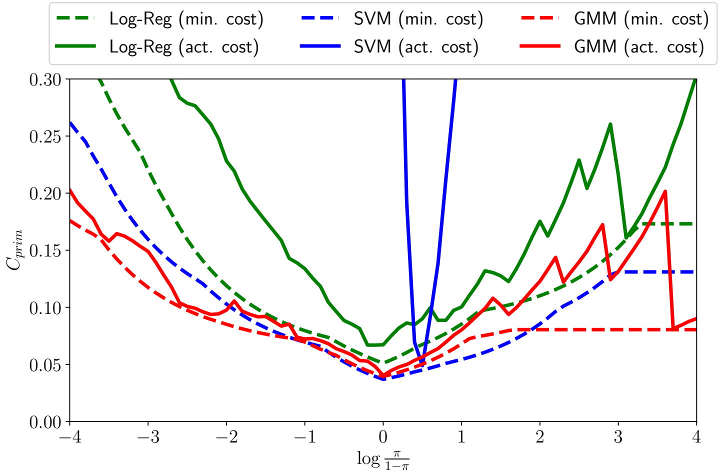
The GMM model seems well calibrated for the two working points

Log-Reg and SVM are poorly calibrated, with SVM providing almost useless decisions

We employ prior-weighted Log-Reg to re-calibrate all three models

Calibration and fusion

Bayes error plot for the three best models — raw scores



Calibration and fusion

Since our primary metric requires optimizing for two different working points, we try different target priors ($0.1, 0.2$ and 0.5)

We use a K-Fold approach again, since the absolute number of errors is quite low

NOTE: since for each fold we estimate a different transformation, the minDCF of the pooled, calibrated scores may also be slightly different

We report both minimum and actual costs for the calibrated scores

Calibration and fusion

Minimum and actual costs — Log-Reg calibration (K-Fold)

	π_T	minDCF[†] $(\tilde{\pi} = 0.5)$	minDCF[†] $(\tilde{\pi} = 0.1)$	min $C^{\dagger}_{prim}$	actDCF $(\tilde{\pi} = 0.5)$	actDCF $(\tilde{\pi} = 0.1)$	C_{prim}
Q-Log-Reg	0.1	0.052	0.136	0.094	0.057	0.148	0.102
	0.2	0.052	0.136	0.094	0.057	0.148	0.102
	0.5	0.053	0.136	0.095	0.057	0.158	0.107
RBF SVM	0.1	**0.040**	0.114	0.077	0.044	0.124	0.084
	0.2	**0.040**	0.111	0.075	0.044	0.124	0.084
	0.5	**0.040**	0.114	0.077	0.046	0.123	0.084
GMM	0.1	0.042	**0.090**	**0.066**	**0.042**	0.097	**0.070**
	0.2	0.042	**0.090**	**0.066**	0.044	0.097	0.071
	0.5	0.042	0.093	0.067	0.044	**0.095**	**0.070**

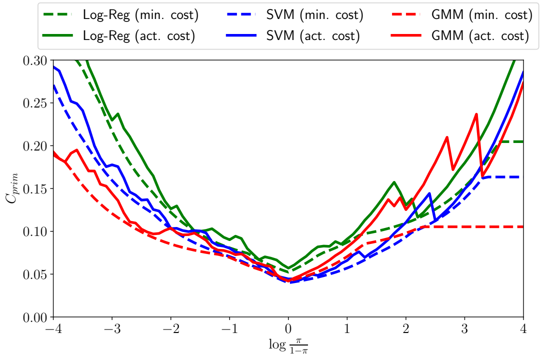
[†] evaluated on the calibrated scores

The training prior does not affect significantly the results

We choose models trained with prior $\pi_T = 0.1$

Calibration and fusion

Bayes error plot for the three best models — Log-Reg calibration $(\pi_T = 0.1)$



We still observe a relevant mis-calibration for some working points, although the calibration loss has been significantly reduced for SVM and Q-Log-Reg models over a wide range of operating points

Calibration and fusion

Finally, we consider the fusion of the best three systems

We employ again a prior-weighted Log-Reg model, with target prior set to $\pi_T = 0.1$, based on calibration results

Minimum and actual costs — Log-Reg calibration (K-Fold)

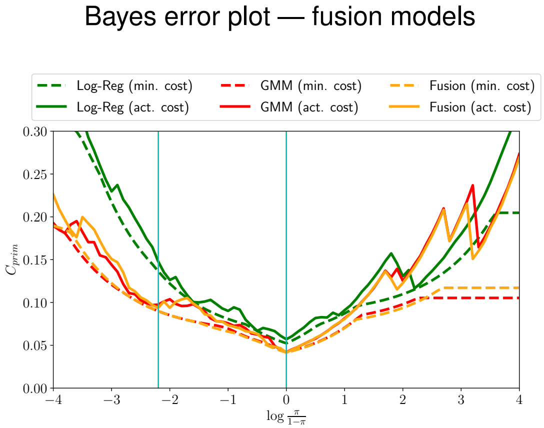
	minDCF[†] $(\tilde{\pi} = 0.5)$	minDCF[†] $(\tilde{\pi} = 0.1)$	min $C^{\dagger}_{prim}$	actDCF $(\tilde{\pi} = 0.5)$	actDCF $(\tilde{\pi} = 0.1)$	C_{prim}
[1] Q-Log-Reg	0.052	0.136	0.094	0.057	0.148	0.102
[2] RBF SVM	0.040	0.114	0.077	0.044	0.124	0.084
[3] GMM	0.042	0.090	**0.066**	0.042	0.097	0.070
[1] + [2]	0.040	0.111	0.075	0.042	0.124	0.083
[1] + [3]	0.041	0.090	**0.066**	0.042	0.097	**0.069**
[2] + [3]	0.041	0.097	0.069	0.042	0.098	0.070
[1] + [2] + [3]	0.040	0.097	0.068	0.042	0.105	0.074

† evaluated on the calibrated scores

Fusions do not improve significantly over single systems

The overall best result is obtained combining Q-Log-Reg and GMM, but is essentially the same performance of the GMM model alone

Calibration and fusion



The plot confirms that fusion is not effective

However, since fusion seems marginally improving results in the range included between our two working points (cyan lines), we select the fused model Q-Log-Reg + GMM (in yellow) as final model

Evaluation

We now analyze performance on the evaluation set

We start from the selected model, and then we will analyze the different choices we made

Note that we do not train any new model at this stage, we simply re-evaluate models that we have already trained

We only consider minimum and actual C_{prim} costs, without showing the results for each target working point

Evaluation

We start from the best three systems and their fusions

Minimum and actual costs — Evaluation set

	Validation set		Evaluation set	
	min $C_{prim}^{\dagger}$	C_{prim}	min C_{prim}	C_{prim}
[1] Q-Log-Reg	0.094	0.102	0.108	0.121
[2] RBF SVM	0.077	0.084	0.084	0.088
[3] GMM	0.066	0.070	0.075	0.081
[1] + [2]	0.075	0.083	0.081	0.088
[1] + [3]	**0.066**	**0.069**	0.076	**0.078**
[2] + [3]	0.069	0.070	**0.074**	0.082
[1] + [2] + [3]	0.068	0.074	**0.074**	0.082

† evaluated on the calibrated scores

Evaluation

Despite Log-Reg showing poor calibration, the fused model proves to be the best in terms of actual cost

We observe that costs are slightly higher than on the validation set

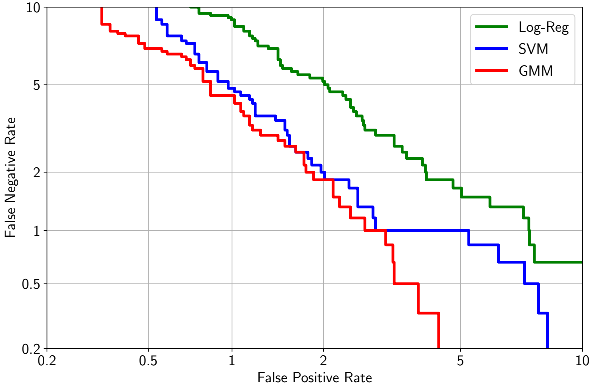
However, results are consistent, and the relative performance of the different methods is similar

The fused model is also well-calibrated for the target applications

Evaluation

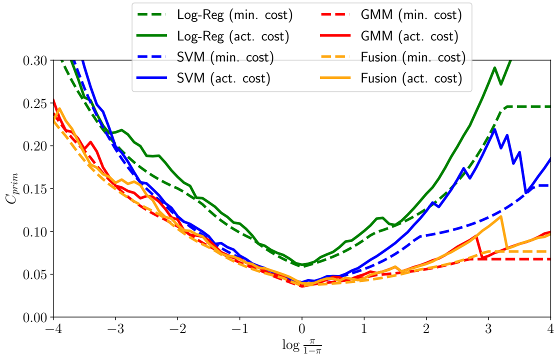
The DET plot over the evaluation results confirms that GMM is the best model.

In contrast with our previous analysis, we observe that SVM is not better in the in low FNR region



Evaluation

The Bayes error plot confirms our previous findings



Log-Reg present higher mis-calibration than the other models

The other models are well-calibrated, except in the right-most region (low FNR, high target prior)

Fusion is only marginally better than the GMM model

Evaluation

We now analyze our previous choices (selected models, hyper-parameters) to assess whether we obtained optimal or close to optimal results

Since linear classifiers are not suited to the task, we do not consider these models anymore

Since Full Covariance and Diagonal (Naive Bayes) Gaussian (but not the Tied Gaussian) models are both specific cases of our GMMs (1 (FC) - 1 (FC) and 1 (D) - 1(D) configurations)[1], we do not explicitly consider these models, but we analyze them in the context of GMMs

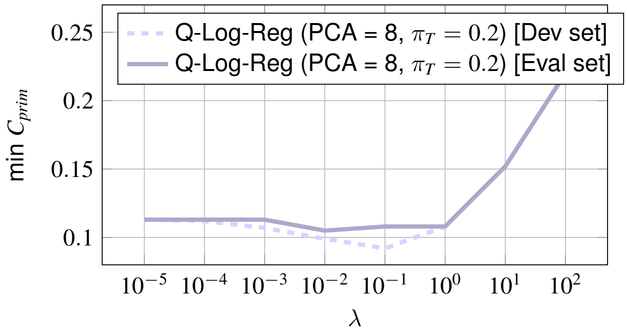
For simplicity, we consider, for the moment, only minimum costs

[1] This holds as long as we do not apply covariance thresholding

Evaluation

We start from Q-Log-Reg models, with PCA, and analyze the hyper-parameter selection

We compare the results for different hyper-parameters for both validation (Dev for development) and evaluation (Eval) sets:



The best hyperparameter would have been $\lambda = 0.01$, with C_{prim} on eval of 0.105, rather than the 0.108 given by the chosen model ($\lambda = 0.1$) — however, the difference is very small

Evaluation

We can repeat the analysis for the other considered target priors (results are on Eval set):

Q-Log-Reg (PCA = 8) — min costs (Eval set)
In blue: selected model; in red: best model(s)

Prior	λ^*_{DEV}	min C_{prim}	λ^*_{EVAL}	min C_{prim}
†Emp. (fold)	0.1	0.108	0.01	**0.102**
†Emp. (dataset)	0.01	0.102		
0.1	0.1	0.108	0.01	0.104
0.2	0.1	**0.108**	0.01	0.105
0.5	0.1	0.107	0.01	0.107

† Depending on whether we would use the fold empirical prior or the dataset empirical prior. On the evaluation set both models would be the same if trained with the same λ

Overall, our choice of hyperparameters, although not optimal, provided close-to-optimal results for Q-Log-Reg with PCA = 8 (minC_{prim} of 0.108 vs 0.102)

Evaluation

We can extend the analysis to different PCA dimensions (we consider only the non-prior-weighted version of the model)

Q-Log-Reg — min costs (Eval set)

PCA Dims	PCA only min C_{prim}	PCA + Z-norm min C_{prim}
—	0.112	0.115
11	0.117	0.118
10	0.109	0.107
9	0.104	0.111
8	0.102	0.107
7	0.107	0.111

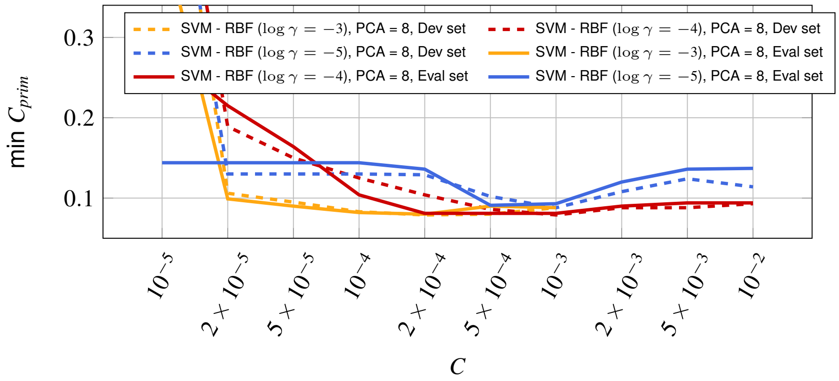
Z-norm is not effective, and 8 PCA dimensions prove to be optimal

The best model for the eval set would have been a Q-Log-Reg model trained with the dataset empirical prior and $\lambda = 0.01$

Evaluation

We now analyze the RBF SVM (we skip the quadratic SVM, since it's similar to the Q-Log-Reg model, and the analysis is more expensive)

We start from the models that employ PCA, and analyze the chosen hyperparameters



Evaluation

We can observe some differences between the validation and evaluation sets

Optimal hyperparameters are not necessarily the same, although results are very close (chosen model: first row - eval optimum: second row)

	min C_{prim} (Dev)	min C_{prim} (Eval)
PCA (8), $\log \gamma = -4$, $C = 10^{-3}$	**0.079**	0.081
PCA (8), $\log \gamma = -3$, $C = 2 \times 10^{-4}$	0.079	**0.080**
PCA (10), $\log \gamma = -4$, $C = 5 \times 10^{-4}$	0.081	**0.079**

Evaluation

Further analysis (grid search) on PCA dimensionality and Z-norm (not shown here) allows concluding that the above configuration is also close to optimal among the considered ones (PCA 10, $\log\gamma = -4$, $C = 5 \times 10^{-4}$, with a C_{prim} of 0.081 on Dev and 0.079 Eval sets)

The chosen model is therefore very close to the optimal configuration, showing that our procedure was indeed effective

Furthermore, the model is not very sensitive to PCA (as long as we keep at least 8 components)

Evaluation

Finally, we analyze our choices for GMMs

We directly report the best configuration for each model combination

	PCA	min C_{prim} (Dev)	min C_{prim} (Eval)
1 (FC) - 16 (FC)	8	0.094	0.079
1 (FC) - 32 (D)	8	0.070	0.071
1 (FC) - 64 (FC-T)	8	0.083	0.066
1 (FC) - 32 (D-T)	8	0.079	0.070
1 (D) - 8 (FC)	8	0.078	0.076
2 (D) - 16 (D)	†12	0.069	**0.067**
2 (D) - 64 (FC-T)	8	0.079	0.068
2 (D) - 16 (D-T)	†12	0.068	0.067
Chosen configuration			
2 (D) - 16 (FC-T)	9	0.064	0.078

† diagonalization only

Evaluation

In this case, our choice was sub-optimal

Full covariance models tend to overfit more both the target and the non-target distribution

While Tied FC models seemed better on the Dev set, the chosen number of components was too small (although the results with Diagonal models seem to suggest that 16 components should be enough — we therefore may have performed poor splits which led to a sub-optimal local maximum of the log-likelihood)

Had we chosen the diagonal version of the model, our results would have significantly improved

We observe that the diagonal 2 (D) - 16 (D-T) and 2 (D) - 16 (D) were already very close to the optimum on the development set

Evaluation

Summarizing, the chosen configurations led to slightly worse result

We can analyze also how the fusion would have been affected

Minimum and actual cost of chosen and best configuration for each system, and corresponding fusions — calibrated scores — evaluation set

	Chosen conf.		Optimal conf.	
	min C_{prim}	C_{prim}	min C_{prim}	C_{prim}
[1] Q-Log-Reg	0.108	0.121	0.102	0.112
[2] RBF SVM	0.084	0.088	0.079	0.081
[3] GMM	0.075	0.081	0.067	0.078
[1] + [2]	0.081	0.088	0.071	0.081
[1] + [3]	0.076	**0.078**	**0.067**	0.080
[2] + [3]	**0.074**	0.082	0.068	**0.076**
[1] + [2] + [3]	**0.074**	0.082	0.068	0.080

Evaluation

Fusion is again only slightly useful

We can observe that the delivered system achieves very close performance with respect to what we would have obtained using better hyper-parameters and model configurations

Calibration of the best model, however, is not optimal: with the best configuration we would loose around 10% relative to poor calibration

Overall, our approach was effective