2. Feature Analysis

Each sample is described by 12 continuous-valued features and a single class variable/label. The features do not have a physical interpretation. In figure 1 we can see the distributions of the features. The orange ones refer to the Female class while the blue ones to the Male class. We noticed that the values are quite high and for this reason we decided to apply Z-normalization to the dataset. This operation transforms the dataset such that the mean of all the values is 0 and the standard deviation is 1 (centering and scaling to unit variance in technical terms). ( MAYBE ADD FORMULA and text like old report).

Then we can see that the distributions of a couple features are slightly irregular but there isn't a large presence of outliers. For this reason we decided that “Gaussianizing” the features is not necessary for this task.

( ADD IMAGES OF FEATURES )

Now let’s analyse the correlation between features. Figure 2 contains the heatmaps with the Pearson coefficient. ( MAYBE ADD FORMULA )

( ADD IMAGES OF HEATMAPS )

Darker colour (and higher values, the maximum is 1) implies a stronger correlation, while lighter colour the opposite. Firstly, we can see that the heatmaps aren’t that different and this means that the correlation between features doesn’t depend on the class. Secondly, we notice that features 5-6 are strongly correlated (coefficient = 0.87) and a few other couples have a coefficient higher than 0.7 . This means that we could apply PCA: it reduces the original space with n dimension into a m subspace with m << n, without losing many information and speeding the overall computations. We decided to look for the m value that retains at least 95% of the variance of the data. The results is m = 8. However, m = 10 retains more than 98% of the variance, so we decided to consider three configurations from this point: noPCA, PCA with m = 10 and PCA with m = 8.

3.1 Classifying the features

To perform the classification task we can use two approaches: single fold and k-fold. In the first one we basically split the original data (the training set) into validation data (33,3 %) and training data (66,6 %). This approach is faster, but the model has less data to use. K-folds instead consists of dividing the data into "k" folds (we chose k = 5), training the model on k-1 of them and evaluating it on the remaining one. This process is repeated k times, with each fold being used as the test set once. The model may not be optimal because it’s not trained over the whole training set, but it has more data available compared to the single fold approach. To tackle the first issue we’ll get the final classifier by re-training over the whole training set. We will consider three different applications: a uniform prior application and two unbalanced applications where the prior is biased towards one of the two classes.

( INSERT FORMULA )

In order to find the best approach we will measure performances through the normalized minimum Detection Cost Function (min DCF), which measures the cost that we would pay if we made optimal decisions using the recognizer scores. The evaluation is carried out on the validation subset (extracted from the training set).

3.1 Gaussian Classifier

The first classifiers that have been used are the Gaussian classifiers: multivariate gaussian (MVG) classifier, MVG classifier with Naive Bayes assumption and MVG classifier with tied covariance. All of them assume gaussian distributed data:

(ADD FORMULA and text like old report)

Now we measure the min DCF for the three of them:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Single Fold | | |  | 5-fold | | |
|  | A picture containing symbol  Description automatically generated = 0.5 | A picture containing symbol  Description automatically generated = 0.1 | A picture containing symbol  Description automatically generated = 0.9 |  | A picture containing symbol  Description automatically generated = 0.5 | A picture containing symbol  Description automatically generated = 0.1 | A picture containing symbol  Description automatically generated = 0.9 |
| Z-Normalized Features – no PCA | | | | | | | |
| Full-Cov | 0.092 | 0.240 | 0.270 |  | 0.114 | 0.297 | 0.350 |
| DIag-Cov | 0.420 | 0.729 | 0.785 |  | 0.463 | 0.771 | 0.778 |
| Tied Full-Cov | 0.082 | 0.222 | 0.221 |  | 0.109 | 0.299 | 0.342 |
| Z- Normalized Features – PCA (m = 10) | | | | | | | |
| Full-Cov | 0.172 | 0.314 | 0.477 |  | 0.187 | 0.407 | 0.538 |
| DIag-Cov | 0.173 | 0.320 | 0.464 |  | 0.184 | 0.436 | 0.546 |
| Tied Full-Cov | 0.163 | 0.319 | 0.491 |  | 0.183 | 0.428 | 0.535 |
| Z-Normalized Features – PCA (m = 8) | | | | | | | |
| Full-Cov | 0.254 | 0.481 | 0.652 |  | 0.267 | 0.549 | 0.677 |
| DIag-Cov | 0.239 | 0.503 | 0.651 |  | 0.264 | 0.520 | 0.673 |
| Tied Full-Cov | 0.247 | 0.451 | 0.661 |  | 0.261 | 0.508 | 0.667 |

In Figure 3 we can see the min DCF computed on the validation set with different prior probabilities and with the configurations of PCA specified previously. We can notice that PCA degraded performances because the min DCF increased in all cases, except for the diagonal covariance where we the behavior was curious: minDCF decreased from noPCA to PCA with m = 10 but then increased again in the PCA with m = 8 configuration. This means that for the diag-cov, the assumption of independent components may be correct but we have to be careful, because reducing too much the number of dimensions led us to a loss of discriminant information. However, the tied full-cov classifier obtained the best results and we notice that dimensionality reduction degraded them by a significant factor, so the best configuration at this point is the noPCA. Finally, we could see that the unbalanced tasks leaded to worse results for all models with respect to the balanced one. All the considerations above are valid both for single-fold and k-fold. From this analysis we understand that linear models performed better than quadratic ones.

3.3 Support Vector Machines

We now turn our attention to SVMs. We will see linear svm (with and without class balancing) and quadratic svm (with polynomial and rbf kernel), even if from the previous results we expect the linear SVM to perform better. The main difference between support vector machines and logistic regression is that they try to find the hyperplane that separates the classes with the largest margin. (MAYBE ADD MORE THEORY FROM OLD REPORT )

For linear SVM we need to tune the hyper-parameters C and K. I tried K=1.0 and K=10.0 with both single-fold and k-fold cross-validation.

( ADD IMAGE OF LINEAR SVM UNBALANCED )

From figure (ADD NUMBER) we select as best values K = 1.0 and C = 2\*10-3. We notice that k-fold results are consistent with single-fold results, meaning that our model behaves correctly.

Now let’s look at the min DCF for linear svm with class balancing:

( ADD IMAGE OF LINEAR SVM BALANCED )

Figure (ADD NUMBER) shows that K = 1 provides a better and more stable min DCF. We decide to choose a different value of the hyper-parameter C for each prior. The table below summarizes the computed min DCF:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Single Fold | | |  | 5-fold | | |
|  | A picture containing symbol  Description automatically generated = 0.5 | A picture containing symbol  Description automatically generated = 0.1 | A picture containing symbol  Description automatically generated = 0.9 |  | A picture containing symbol  Description automatically generated = 0.5 | A picture containing symbol  Description automatically generated = 0.1 | A picture containing symbol  Description automatically generated = 0.9 |
| Z-Normalized Features – no PCA | | | | | | | |
| Linear SVM (C=2\*10-3, unbalanced) | 0.093 | 0.236 | 0.236 |  | 0.990 | 0.998 | 1.005 |
| Linear SVM (C=2\*10-3, ) | 0.112 | 0.216 | 0.350 |  | 0.126 | 0.315 | 0.455 |
| Linear SVM (C=10-2, ) | 0.112 | 0.216 | 0.350 |  | 0.126 | 0.314 | 0.457 |
| Linear SVM (C=10-3, ) | 0.127 | 0.207 | 0.395 |  | 0.127 | 0.310 | 0.419 |
| Z- Normalized Features – PCA (m = 10) | | | | | | | |
| Linear SVM (C=2\*10-3, unbalanced) | 1.000 | 1.000 | 1.015 |  | 0.873 | 0.999 | 0.990 |
| Linear SVM (C=2\*10-3, ) | 0.164 | 0.307 | 0.499 |  | 0.245 | 0.411 | 0.750 |
| Linear SVM (C=10-2, ) | 0.164 | 0.307 | 0.499 |  | 0.245 | 0.407 | 0.752 |
| Linear SVM (C=10-3, ) | 0.170 | 0.307 | 0.522 |  | 0.194 | 0.395 | 0.578 |
| Z-Normalized Features – PCA (m = 8) | | | | | | | |
| Linear SVM (C=2\*10-3, unbalanced) | 0.997 | 1.000 | 1.011 |  | 0.969 | 0.998 | 1.005 |
| Linear SVM (C=2\*10-3, ) | 0.252 | 0.440 | 0.651 |  | 0.943 | 1.000 | 1.005 |
| Linear SVM (C=10-2, ) | 0.252 | 0.440 | 0.651 |  | 0.998 | 0.998 | 1.005 |
| Linear SVM (C=10-3, ) | 0.252 | 0.427 | 0.658 |  | 0.703 | 0.854 | 0.876 |

The class re-balancing has improved the model and the best results are given by the linear model with πT = 0.9 and C = 10-3. Intuitively, this configuration had the best performance due to the fact that the training datatset is imbalanced.

The first thing that stands out is that the unbalanced SVM performs really well in the noPCA and single-fold case, however its results get worse in all other configurations. There could be many reasons and we assumed it’s just randomness: its performance happened to align favorably with the particular data in that fold.

In addition to this, we can notice that PCA degraded again the performance especially when focusing on the m = 8 configuration meaning that, even if it retains 95% of the variance, we lost discriminant information.

Regarding the non-linear formulation in SVMs, the non- linearity is obtained through an implicit expansion of the features in a higher dimensional space. The dual svm formulation depends on the training samples only through dot-products, and we can compute a classification score through scalar products between training and evaluation samples.

For this reason, it’s not required to explicitly compute the feature expansion, it’s enough to be able to compute the scalar product between the expanded features, the so-called kernel function k:

( INSERT FORMULA )

We will test two different kernels:

1. Polynomial kernel of d = 2;

k(xi ,xj ) = (xTi xj + c)d

1. Radial Basis Function (RBF) kernel:

k(xi,xj)= e-γ∥xi−xj∥^2

For the quadratic polynomial svm we need to estimate two hyper-parameters c and C through a cross-validation.

( ADD IMAGE OF POLYNOMIAL SVM )

From the above figure we see that the minimum min DCF is found with c=15 and C=3∗10-5. Now let’s estimate the two hyper-parameters γ and C for the RBF SVM:

( ADD IMAGE OF RBF SVM )

Looking at figure ( ADD NUMBER ) we selected γ = 10-3 and C = 1. Now it’s time to look at the performances:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Single Fold | | |  | 5-fold | | |
|  | A picture containing symbol  Description automatically generated = 0.5 | A picture containing symbol  Description automatically generated = 0.1 | A picture containing symbol  Description automatically generated = 0.9 |  | A picture containing symbol  Description automatically generated = 0.5 | A picture containing symbol  Description automatically generated = 0.1 | A picture containing symbol  Description automatically generated = 0.9 |
| Z-Normalized Features – no PCA | | | | | | | |
| Linear SVM (C =10-3, ) | 0.127 | 0.207 | 0.395 |  | 0.127 | 0.310 | 0.419 |
| Polynomial SVM (C = 3\*10-5, c = 15, d = 2) | 0.852 | 1.000 | 0.976 |  | 0.794 | 0.999 | 0.994 |
| RBF SVM (C = 1, ) | 0.640 | 0.885 | 0.961 |  | 0.677 | 0.957 | 0.985 |
| Z- Normalized Features – PCA (m = 10) | | | | | | | |
| Linear SVM (C=10-3, ) | 0.170 | 0.307 | 0.522 |  | 0.194 | 0.395 | 0.578 |
| Polynomial SVM (C = 3\*10-5, c = 15, d = 2) | 0.891 | 1.000 | 0.989 |  | 0.851 | 0.999 | 0.997 |
| RBF SVM (C = 1, ) | 0.679 | 0.894 | 0.978 |  | 0.674 | 0.944 | 0.981 |
| Z-Normalized Features – PCA (m = 8) | | | | | | | |
| Linear SVM (C=10-3, ) | 0.252 | 0.427 | 0.658 |  | 0.703 | 0.854 | 0.876 |
| Polynomial SVM (C = 3\*10-5, c = 15, d = 2) | 0.922 | 1.000 | 1.011 |  | 0.884 | 0.999 | 0.998 |
| RBF SVM (C = 1, ) | 0.624 | 0.926 | 0.965 |  | 0.656 | 0.980 | 0.977 |

The RBF SVM performed slightly better that the polynomial sums but anyway we can conclude that the linear approaches fit better the task. This time PCA didn’t degrade the results but it didn’t improve them either, so we cans say that it hasn’t been effective in improving the results quality. However, it helped reducing the complexity and speeding computations.

Anyway, it is important to notice that SVMs performed worse than both gaussian classifiers and linear logistic regression.