ML and PR Project Report – Part 9

This report is going to tackle the application of calibration on the resulting scores. In order to make best use of the few data points we have we are going to adopt a K-fold approach, so as to be able to use the entirety of our scores both for calibration training and for calibration transformation.

# Hyperparameter Tuning

The first thing to do is find how to calibrate the models well. The models taken into consideration are, as those of the last laboratory, so the Quadratic Logistic Regression, the RBF Kernel SVM, and the 8-components diagonal GMM. For details about their parameters, see 8th report.

Our calibration approach is simply finding the optimal **prior-weighted unnormalized Logistic Regression** transformation of our scores, with the aid of K-fold. This means that the parameters that matter are the number of folds, and the value of the prior for the Logistic Regression.

The choice of which parameters to explore has mainly been: for the number of folds, always keeping a value that would allow a perfect split of the datasets (i.e. a divisor of 2000, as the validation dataset size is such), while at the same time not exploring incredibly high amounts of folds, as not to make the training uselessly heavy. For the prior instead, I thought of focusing on values around the target application (), and its complementary, since Bayes Error Plots show that there is a symmetric behavior in that sense (that does not mean I completely neglected the 0.5 area. I have just taken more sparse values there).

The values taken into account are the following:

Ks = [2, 5, 10, 20, 25, 50, 100]

pi\_weight\_s = [0.05, 0.1, 0.15, 0.2, 0.4, 0.5, 0.7, 0.8, 0.85, 0.9, 0.95]

The actual DCF rankings allowed for the selection of a variety of similarly performing hyperparameters, so I took into consideration keeping the heaviness of the program limited, and priors closer to the target application. The latter choice has no theorical foundation, but since I had to choose between equally performing parameters, I thought it could be a valid choice.

The following choices were made:

* LR: K = 50, = 0.85
* SVM: K = 10, = 0.05
* GMM K = 10, = 0.1

# Calibration

The performances on the target application were the following:

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

LR witnessed a decent improvement, SVM a massive one, while GMM actually got slightly worse.

Let’s see if for generic applications we keep the same type of performance.

Immagine che contiene testo, linea, Diagramma, diagramma

Descrizione generata automaticamenteImmagine che contiene testo, linea, Diagramma, diagramma

Descrizione generata automaticamente

Quadratic Logistic Regression RBF-Kernel Support Vector Machine

Immagine che contiene testo, linea, Diagramma, schermata

Descrizione generata automaticamente

8-components diagonal GMM

LR’s calibrated version hardly ever performs worse than the non-calibrated version. We can observe that its tendency to become less and less calibrated as the log-odds increased in magnitude has also been somewhat fixed.

SVM’s calibration improves its performance under every aspect, especially, and much more than LR, when growing in log-odds magnitude.

GMM’s performance seems to not change at all, instead there are a few points where calibration is detrimental. It seems like GMM doesn’t really need calibration.

# Fusion

The fusion of the three different sets of scores, coming from the usual models, is performed using the same Logistic Regression model as before, but we will look again for hyperparameters that are good for our target application, before delving into actually analyzing the performance of the chosen calibration model.

The values of K and explored are the same as before, and the best results this time were met with a K of 20 and a of 0.05

The first observation we make are again based on the actual and minimum DCF performances of the model on our target application of , and are as follows:

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

The fusion results are better than those found for the individual calibrated models, but the drop in performance with respect to the uncalibrated GMM is still there, when looking at the actual DCF. There is an interesting result if we look at the minimum DCF instead: the Fusion model outperforms even the GMM. This could suggest that a different calibration model could breakthrough the minimum set by the GMM.

If we extended the analysis to a wider range of applications, through Bayes Plots, we get the following:

Immagine che contiene testo, Diagramma, diagramma, linea

Descrizione generata automaticamente

We are interested in looking at the “competition” between Fusion and just GMM calibrations.

The Fusion model’s minimum DCF is performing better or equal than the GMM minimum over all the considered applications, and does an incredibly better job for higher log-odds magnitudes.

In terms of actual DCF instead, the Fusion model is just slightly worse than the GMM one for low magnitude log-odds, gets much worse on the positive side of the graph, and outperforms in a zone in the negative values of log-odds.

If we compare it to the uncalibrated version of the GMM instead:

Immagine che contiene testo, linea, Diagramma, diagramma

Descrizione generata automaticamente

We can see that in terms of minimum DCF, some areas are slightly dominated by the fusion (like our application’s), while others by the GMM, but the difference is probably negligible.

In terms of actual DCF, GMM performs either as good, or better than fusion. In some areas there really isn’t any difference at all, but for others (e.g. the high positive values of log-odds), fusion fails in a notable manner.

# Conclusion

8-components diagonal GMM, is the best model of all. It already shows a good degree of calibration inherently, and actually applying more has detrimental results. Not even the Fusion model is able to outperform it.