Project-II by Group Rome

Cimen Gokcen
EPFL
gokcen.cimen@epfl.ch

Angelopoulos Vasileios
EPFL
vasileios.angelopoulos@epfl.ch

Petrescu Viviana
EPFL
viviana.petrescu@epfl.ch

Abstract

This report present

1 Music Recommendation

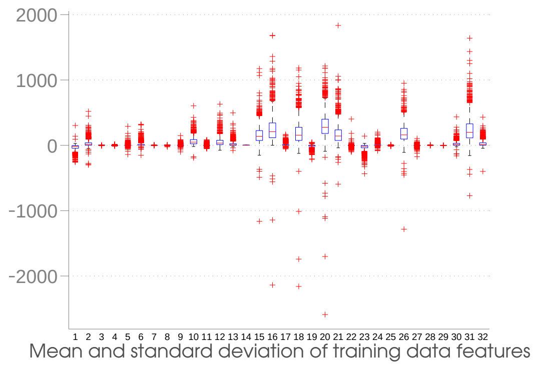
1.1 Data description

We have a two-class classification problem. The training data $Xtrain$ contains 1500 observations, each 32 dimensional. One training sample has 31 real valued features and 1 categorical feature, the 14th feature. Our task is to predict the category for unseen test data, consisting in 1500 samples. We measure the accuracy of our estimation using RMSE, $0 - 1loss$ and $log - loss$.

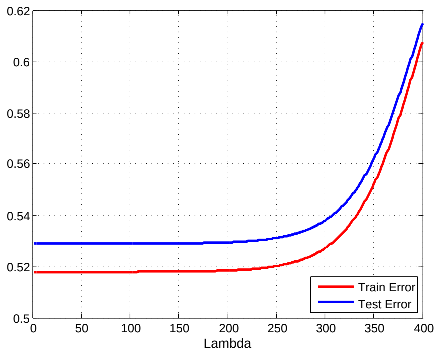
1.2 Data visualization and cleaning

We noticed the samples were not equally distributed among classes, about 35% of the samples were coming from one class and 65% from the other. We did not have time to study the implications of this on our model estimation. We changed the labelling from -1/1 to 0/1 because our cost function was better expressed using 0 and 1.

The training data were again not centered as seen in $Fig.$ **??** After changing feature 14 to a dummy variable we normalized the data to have 0 mean and standard deviation 1. We noticed more outliers in the data of classification than in the one from regression. We removed outliers in a similar way as in the regression task which left us with 1424 samples (5% of the data were removed).



(a) Mean and standard deviation for the training data features. The input is not normalised and contains outliers.



(b) Train and test error assessment with penalty factor (λ) for penalized logistic regression for 50-50 split.

1.3 Logistic Regression

We applied Logistic Regression (logRegression) and Penalized Logistic Regression (penLogRegression) using gradient descent with learning rate α. We obtained the optimal α for logRegression and penalty factor λ for penLogisticRegression using the cross validation technique with a $50\% - 50\%$ split for training and validation data.

In penLogRegression, the choice of penalty factor λ is crucial. We sampled 400 values for λs in the interval 10^{-2} to 10^{3}. As can be seen in $Fig.$ **??**, for small λ values, training error is much lower than the test error while for large λ values the test error gets as high as the training error which is a sign of under-fitting. For small λ values, the training and test error estimated in penLogisticRegression show very similar behavior with the ones for logRegression.

The same α value that is obtained for logRegression gives also the best accuracy for penLogisticRegression. Increasing the value of α decreases all error estimations until around $\alpha = 0.8$ value which gives the best prediction accuracy for our data.

$Fig.$ **??** shows the classification accuracy percentage of logRegression and penLogRegression on the test data using 5-fold cross validation. The improvements with penLogRegression is little but might perform better on unseen data since it is more robust to outliers. Our best accuracy on the test data of 96.348% is obtained using penLogRegression with $\lambda = 0.03$ and $\alpha = 0.8$.

For our best setups, we report three types of errors using RMSE, $0 - 1loss$ and $log - loss$ in table **??**. penLogisticRegression consistently performs better.

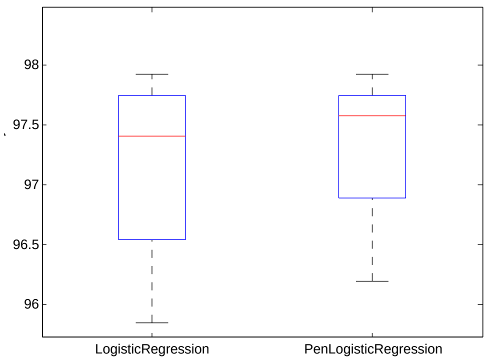


Figure 2: Comparison of logistic regression and penalized logistic regression based on validation accuracy.

Method	RMSE	0-1-Loss	Log-Loss
LogRegression	0.17	0.033	161.32
PenLogRegression	0.15	0.0267	132.34

Table 1: Best predicted error estimates for the test data.

1.4 Feature transformation

We experimented with polynomial and exponential transformation of the values but did not manage to significantly improve the previous results obtained using penLogRegression. Our best model used a polynomial of degree 2 transformation of every feature ($\alpha = 2$ and $\lambda = 0.1$) which had a recognition rate of approximately 95.77 (with standard deviation 1.056). All other models performed much worse. Since, in general, it is better to explain the correlation in the data using simpler models, we decided to report our best predictions using penLogisticRegression.

1.5 Summary

In this section, we experimented with a classification problem using logistic and penalised logistic regression. Both models gave very similar results, with best classification accuracy of 97% obtained using penLogisticRegression. Although logRegression performed very similarly, we expect penLogisticRegression to perform better on unseen data.

None of the feature transformations we tried improved the results. Using a polynomial of degree 2 for every variable performed similarly with the other models, but since the model just added complexity and no significant improvement in accuracy it was not considered relevant.

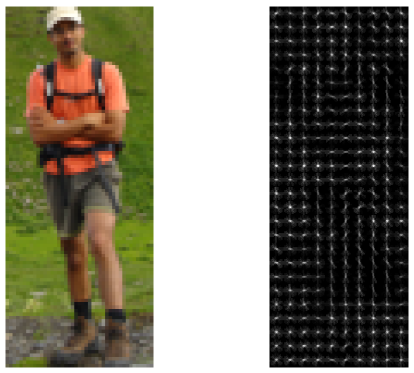
2 People Detection

2.1 Data description and visualization

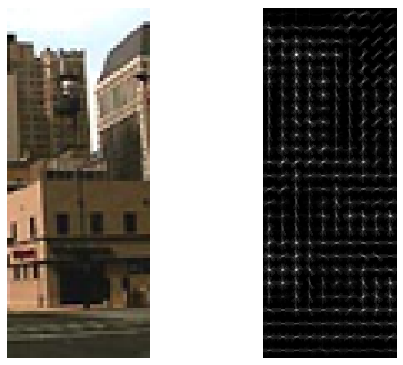
The original training data $imgs$ contains 8545 color images with size 105x43. From every image a HOG descriptor with size 26x10x36 was extracted. All of these descriptors are saved at $feats$ and form the training data that we will work on. To process this HOG features we have converted them into a vector of total length 9360.

We notice that there are 7308 images without presence of people (negative images) and 1237 with (positive images). Figure **??** presents a typical example from each category accompanied with the relevant feature descriptor.

Our task is to train various classifiers, so that we will be able to detect the presence of people in unknown cases. For these purpose we evaluate these classifiers, measuring the accuracy of their estimations using Receiver Operating Characteristics (ROC) curves.



(a) Positive image.



(b) Negative image.

Figure 3: Examples of positive and negative images accompanied with their feature descriptors.

2.2 Data preprocessing and used methods

Working on the extracted features, we split the data in order to perform our experiments. In our case we used K-fold validation with K = 5 to evaluate the performance of each classifier. Next step is to normalize the data so they will be centered with 0 mean and standard deviation 1. Here, we have to mention that we assume that there are no outliers, since the images are manually annotated and they have only two possible values for the classes.

The methods that were tested throughout this experiment are the naive Bayes classifier, the logistic regression and its penalized version, the Support Vector Machine (SVM), the K-NN classifier, and the Neural Network classifiers. All of them are compared also with the random guess of the class in the last subsection of this task.

Moreover, we have to say that at the beginning of the experiments we developed and tried PC Analysis, but, except the huge computational time that was necessary to perform it, it did not give almost any improvements to the dimensionality of our data.

2.3 Naive Bayes classifier

In this part we tried to fit a simple Bayes classifier to our data. The assumptions that we made were that we have data that follow the normal distribution and the estimates of the prior probabilities are empirical and were made from the relative frequencies of the classes in training.

The results from the K-fold validation were not very satisfying, since the average TPR was very low (0.031). The reason that may be hidden behind that are the strong assumptions that we made in order to fit this model. Naive Bayes model assumes that the data that will process every time follow a specific distribution (in our case Gaussian), but also that the value of a particular feature is unrelated to the presence or absence of any other feature, given the variable of the class. In our case, this is possibly wrong, so the classifier cannot fit a appropriate model.

2.4 Logistic and Penalized Logistic Regression

Our purpose in this part was to fit the logistic regression and its penalized version to our data. The value for the parameter α that we used was 10^{-2}, since all of the values close to this performed well enough for our experiments. Moreover, regarding the value for λ in case of the penalized logistic regression, we have to mention that all of the values that were used performed the same. This is the reason that both of the method give the same results, with average TPR equal to 0.754.

During penalization the idea is to avoid the overfitting by imposing penalty on large fluctuations of the model parameters and reduce the influence of the outliers to our model. This is possibly the reason that both models perform equivalently. The data, as we mentioned before, do not have outliers and the fluctuations at the model parameters are avoided by the nature of our data.

2.5 Support Vector Machines

For the certain part of the experiment we used the LIBSVM 3.20 toolbox that includes SVM using various kernels. In our case, we performed analysis and K-fold validation on linear-kernel SVM, polynomial-kernel SVM with degrees 2 and 3, real-basis-kernel SVM and sigmoid-kernel SVM. In all of the cases we used the scores that were given by the program to compute the TPR and the FPR. A comparison of the various cases is given in Figure **??**.

As we can observe radial-basis-kernel SVM have significantly good performance. A fact that can be seen not only by the average TPR, which is equal to 0.893, but also observing the whole range of the FPR, where the classifier gives better results than all of the other SVM cases. The second best in performance is the polynomial SVM with degree 2, which has very similar results with the RBF case for FPR greater than $210e{-}3$. The rest of the classifier perform well for FPR greater than $210e{-}2$, with the exception of the linear case, which has quite good results for lower rates, too.

Moreover, in Table **??** we present the average percentage of the successful classifications that were made during the K-fold validation, as they are given by the program.

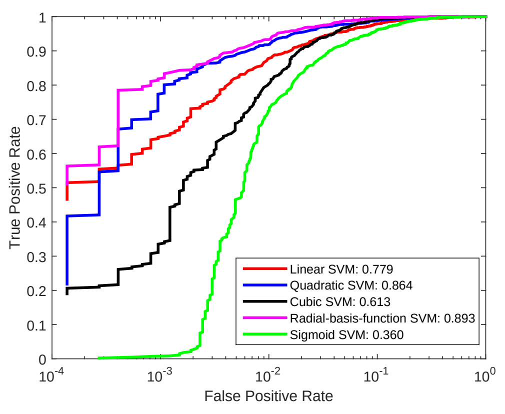
SVM kernel	Linear	Quadratic	Cubic	RBF	Sigmoid
Average success rate (%)	94.81	97.22	91.45	98.43	84.68

Table 2: Average percentage for the success rate for the various kernels of the SVM classifier, as they are given by the program of the LIBSVM 3.20 toolbox.

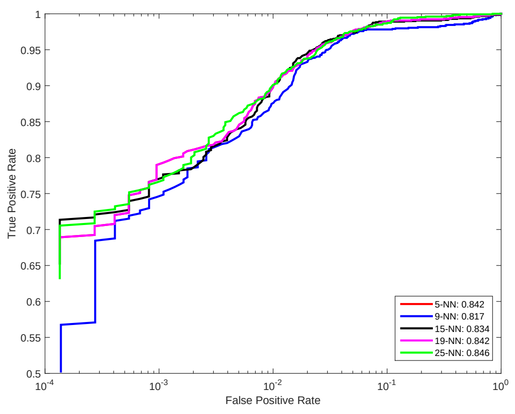
2.6 K-NN classifier

Using the K-NN classifier the most important parameter that we have to specify is the number of the neighbouring samples that will be used as a reference for the method so it will decide for any given sample the class that it belongs to. For our experiments we used a ready function, which we modified a lot in order to compute and give in output the scores of the prediction, except of the class labels. We varied the number of neighbours used among 5, 9, 15, 19 and 25. In addition, the euclidean distances were used as the way to specify the distances among the neighbouring samples.

Figure **??** presents the resulting ROC curves that were created using the scores got from the K-fold validation process. There we observe that all of the classifiers show very good performance, especially, for FPR greater than 10^{-3}. Only the 9-NN classifier seems to perform a bit worse for FPR between 10^{-4} and 10^{-3}. These observations can be explained by the distribution of the data, where most probably they are distributed in the N-dimensional (N = 9360) space with such a way that their neighbours can specify, with high probability, the class that they belong to.



(a)



(b)

Figure 4: ROC curves created using: (a) SVM with various kernel types (linear, quadratic, cubic, real-basis-function, sigmoid), (b) K-NN classifier for various number of neighbours for the computations (5, 9, 15, 19, 25).

2.7 Neural Network

2.8 Convolutional Neural Network

2.9 Comparison

Acknowledgments

We would like to thank the course teaching assistants that helped us a lot during this project preparation, not only during the exercise session, but also at their office hours. Furthermore, the help that our teacher Mohammad Emtiyaz Khan offered us was very important, and we would want to thank him, too. All of the code used and submitted along with this report was written by equally by all of our group members.