Classification using Logistic Regression & an Artificial Neural Network

Santiago Canete
Department of Mechanical Engineering, Temple University
santiago.canete@temple.edu

Introduction: For the final project, the goal was to classify two data sets with two classes each, one was a 2D set and the other was a 5D set. We were asked to implement two types of classification techniques, one involving a non-neural net and the other involving a neural net. The classifiers were implemented in Python 3.7 using Scikit-Learn for the non-neural net approach and Keras for the neural net approach. All processes were executed using the Joblib CPU parallelization functionality. Surprisingly, using 8 CPU jobs resulted in faster training than using the GPU. Computation time was a big hurdle in the process of optimizing the classifier parameters, especially for the 5D data set.

In order to test a large number of algorithms first I performed an initial classification of the 2D data set and then I proceed to optimize the classifier using the 5D set. The parameter optimization for the classifier had to be done with the 5D set since the statistics of the 2D and 5D were completely different. For the parameter optimization the method used was an exhaustive grid search provided in the Scikit-Learn toolbox GridSearchCV(). This method could also be used with the neural net implemented in Keras by using a Scikit wrapper.

Both approaches were used within the pipeline functionality provided by Scikit. Pipelines allow to apply a set of transformations and methods to the data before performing the final classification. Throughout the multiple iterations it was noticed that using a StandardScaler resulted in a minimal improvement in error rate. Other transformations such as PCA or MinMaxScaler were also tested, the results were equal or worse than without the transformation. One method that did significantly improve performance was clustering. A K-Means clustering method was used for the non-neural net approach prior to applying the final classifier.

For all tested methods I performed a cross validation using a StratifiedKFold() with 16 splits and a constant random seed of 42 (the seed was kept as 42 for all methods used). The StratifiedKFold method ensures that there are an equal number of samples from each class in each split. It was noticed that the variation in error rate could vary in about 5% when running a single training and classification process. The metric selected for cross validation was accuracy.

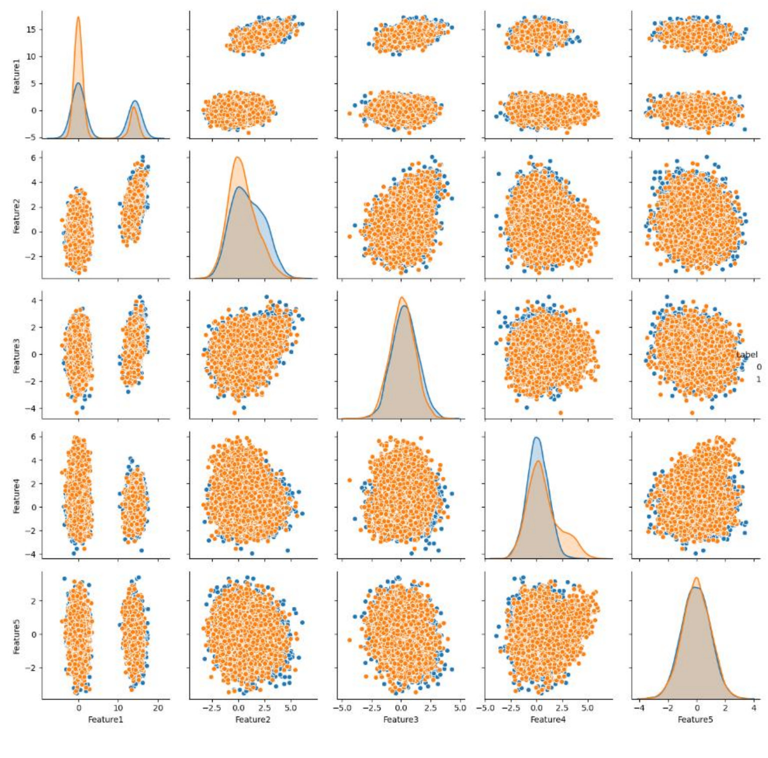

Figure 1: 5D training data set with plots of all features against each other. We can see how the overlap across features is very high, causing the error rate to also be high.

Logistic Regression with K-Means clustering and Bagging: The method was implemented in a pipeline, with the first step being a StandardScaler, then a K-Means clustering method with 450 clusters. The number of centroid seeds used was 8, the initialization method was ‘k-means++’. Then the cluster centroids were fed to a Logistic Regression (LGR) classifier with 20 estimators. The LGR used a ‘sag’ solver, an ‘l2’ penalty and an inverse regularization strength of 1.0. Finally, the LGR was wrapped into a bagging classifier with 10 splits and the default max features and samples which is 1.0. The bagging classifier used the bootstrapping technique.

In this approach the scaling step did not change the result significantly. Several other methods were tested as well. Some of the classification methods tested were Random Forest, Support Vector Machine, Stochastic Gradient, Gaussian Mixtures and K Nearest Neighbors. Other ensemble methods were also evaluated such as Adaboost and Gradientboosting. An implementation of two stacking methods was also performed Voting classifier and a Super Learner.

ANN Pipeline with Bagging: The neural net implemented was the artificial neural net provided by the Keras toolbox. Through an exhaustive search the optimal number of hidden layers was four with the number of neurons shown in Figure 2. The search included a set of activation functions, initialization kernels, optimizers, loss functions, dropout rates, epochs, and batch sizes. The array of neurons tested went from 5 to 100 in increments of 5. A cross validation of 16 splits was performed using the training data set. This process required many hours to complete.

A StandardScaler was applied to the data prior to the training and classification. The ANN was considerably more sensitive to scaling than the non-neural net approach. Then, the ANN was wrapped into a bagging classifier with 10 splits and the default max features and samples which is 1.0. The bagging classifier used the bootstrapping technique. Bagging reduced error considerably.

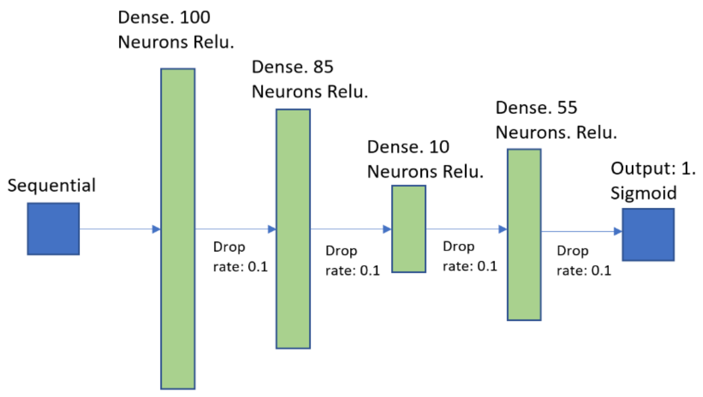

Figure 2: The diagram shows the structure of the artificial neural network (ANN) used. The optimizer used was Adadelta with a binary crossentropy loss function and with accuracy as the metric. The net was initialized using a LeCun uniform kernel. The number of epochs was 100 and the batch size was 100.

Results:

	Data Set		
Algorithm	**Train**	**Dev Test**	**Eval**
LGR 2D	8.04%	7.95%	7.70%
ANN 2D	7.90%	7.85%	7.95%
LGR 5D	36.71%	36.66%	36.66%
ANN 5D	36.31%	36.65%	36.64%

Table 1. For the LGR classifier we can see how the error for the training set is almost equivalent to the dev and eval sets. Meaning that there is no overfitting. For the ANN the error in the training set is lower than for the dev and eval sets, this shows there is overfitting.

Conclusions: For this final project I implemented and measured the performance of a large range of classifiers, ensemble and stacking methods.

For the non-neural net approach, most classifiers performed similarly with the right parameter tuning. Algorithms like KNN or RF tended to overfit the data, resulting in low error rates for the training set and no improvement for the development set. This behavior was expected for these algorithms. A large boost in performance was achieved by using ensemble techniques. The best performance happened when using a SuperLearner stacking method. This method tests different combinations of a set of classifiers to train a meta-classifier. One benefit of this method is that it guarantees that classification will be at least as good as the best of the provided classifiers. Unfortunately, the trained model was lost, and I was forced to use a previously trained model in the interest of time.

The method selected used a scaling step followed by a k-means clustering process and finally used a bagging classifier with an LGR base estimator. The bagging classifier uses a bootstrap sampling to obtain the data subsets for the given base learner. The boost in performance achieved through bagging was expected since it allows for a reduction in bias and is less sensitive to perturbations in the training set. This can be interpreted from the results since there is very little overfitting.

For the neural net method, the selected approach was an ANN. The reason for this is because it is easier to tune, and it is better suited for binary classification. The parameters obtained were a product of an exhaustive search, it was hard for me to fin a pattern in the behavior since performance is correlated to the combination of all parameters. Through the drop out rate I was able to manually tune the amount of overfitting of the model.

Overall, for the given data set there was no considerable difference from one classifier to another. The two things that improved performance were scaling and ensembling. Parameter tuning was also important, but it appeared to flatten out the error rate after a few tweaks. The key observation I made was that when training the models I would achieve slight differences in performance even when using a constant random state and the same parameters, in some cases improving upon the best achieved score in the course. In my case I selected the models that performed best in a cross validation of 16 splits which I believe to be e better comparison. Maybe, in future iterations the performance measure can be obtained through cross validation in order to compare between students and algorithms since one sample from one model is not enough to show a statistically significant difference between classifications.