Advanced Machine Learning - Assignment 1

Pranav Kasela
846965

Introduction

In this assignment the objective was to predict whether a customer will default or not the next month.
In the training data an imbalance in the prediction class, composed of 0 (non-defaulter) and 1 (defaulter), was detected. There were $\approx 22\%$ of defaulters. The problem is a binary classification problem.
A naive approach on the behavior of the neural network without any pre-processing was used, yielding poor results, it followed the zero rule, predicting only the non rare class.

Preprocessing

The *MARRIAGE* and *EDUCATION* were nominal variables with more than two levels, so the one-hot encoding was applied. There was a very strong correlation between the *BILL_AMT_X* for X= $1, ..., 6$ input variables, this problem was tackled taking their average *BILL_AMT* instead of all of them. Lastly a normalizing between 0 and 1 was applied to all the continuous variables (including AGE).

Models

The models were developed in Keras. The training data was splitted into training set (90%) and validation set (10%). The two equivalent approaches to address the binary classification problem were:

1. Using the Softmax units for a Multinoulli output distribution with 2 classes;
2. Using the Sigmoid units for a Bernoulli output distribution.

Both methods were implemented, even though they are mathematically equivalent, the programming approach differs a little.
For the Softmax the prediction variable was categorized transforming the class 0 in $[1, 0]$ and the class 1 into $[0, 1]$. To tackle the class imbalance problem, different models were developed using different techniques.
(During the model explanation the input and output layer will not be mentioned, since they are always there).

Nothing Special

The best configuration for the Neural Network found with the softmax output units was composed by 4 hidden layers of

$32, 64, 32, 32$ neurons respectively, each of them activated with a ReLu and with the SGD optimizer, the loss function used was the categorical crossentropy. This models yielded an weighted average f1 score of 0.81 while in the single classes it was class 0: 0.90 and class 1: 0.47.
The best hidden layers configuration for the sigmoid output was:
Dense(32) $\rightarrow$ Dropout(0.2) $\rightarrow$ Dense(64) $\rightarrow$ Dropout(0.2) $\rightarrow$ Dense(16)
each activated with a ReLu, loss function used here was the binary crossentropy. This model yielded the same average score but it did improve in the rare class obtaining a score of 0.49.

Weighted Classification

Weights for the errors in classes, errors in class 0 had a weight of 0.35 while errors in the rare class 1 had a weight of 0.65.
The configuration used is the same for each method:
Dense(32) $\rightarrow$ Dropout(0.2) $\rightarrow$ Dense(64) $\rightarrow$ Dropout(0.2) $\rightarrow$ Dense(16)
The weighted f1 scores are the same for both models:
class 0: 0.89; class 1: 0.52; weighted average: 0.81.

Oversampling using the SMOTE

The configuration used was:
Dense(64) $\rightarrow$ Dense(32) $\rightarrow$ Dense(128) $\rightarrow$ Dense(64) $\rightarrow$ Dense(16)
In this case the prediction of the non rare class was a little worse obtaining the following score - class 0: 0.85; class 1: 0.52; weighted average: 0.78.

Undersampling

The configuration used is:
Dense(64) $\rightarrow$ Dense(32) $\rightarrow$ Dense(128) $\rightarrow$ Dense(64) $\rightarrow$ Dense(16)
It didn't do much better than the SMOTE one obtaining the following score:
class 0: 0.86; class 1: 0.53; weighted average: 0.79.

Conclusion

The models were obtained using a self implemented 'Grid Search' for the number of layers between 2 and 5 and number of neurons in them were prefixed in an array. An Early Stopper was used and the number of epoch were set to a high number. While for the optimizer and batch size a trial and error was used, different optimizer with different learning rate were tested, some of them ended up overfitting the data while the other didn't improve the result, so SGD was left. Further modification of the data or the models didn't provide better results.
The best model chosen was the weighted one since it performed better in the rare class comparatively, the weights could be chosen in a better way based on the company's requirement and costs.
Finally, the validation data and train data are combined together and used for the training of the model, the test data undergoes the same transformation as the training data and validation data. The final output is written on a separate file.