Report on Neural Networks Kaggle Challenge

Neil Kachappilly, Gabriel Talih

May 13, 2024

Problem Description

Our task revolves around classifying time series data into five distinct categories. Despite efforts to standardize the data through zero-padding, the variability in time series lengths presents a significant challenge for classification algorithms. Additionally, the dataset suffers from class imbalance and sporadic missing data, adding complexity to the task. These challenges necessitate tailored strategies for data imputation and class balance handling during model training. The evaluation metric, macro-F1 score, emphasizes the importance of achieving balanced classification performance across all categories, setting the stage for different approaches to neural networks that can address these complexities.

Data Preprocessing

Considering the significant missing data, the first step was to impute these values for the data to be usable. For this, I considered two approaches:

1. Fill in the missing value with the median of the existing column values.
2. Fill in the missing value with the median of the existing column for the rows corresponding to the label of the row of the missing value. This method seemed best to maintain data integrity and yielded very promising results in my personal testing, but then I realized it is less suited to the actual predictions task since I wouldn't be able to impute the missing values in the test set in that manner. Thus, my final model used the first data processing.

I also considered several different methodologies to work on balancing the distribution of labels in the data. Printing out the imbalance showed the 0 label to have far and away the most samples while 1 and 3 had significantly less. Thus, I made use of SMOTE to oversample the 1 and 3 datapoints to 4500 – around the same as labels 2 and 4. I then proceeded to undersample the 0 label to 5500 to put everything in similar ranges. This was a key step for testing my Logistic Regression and Neural Network models but much less significant when I tested my ensemble models later. In fact, for the ensemble methods, since they were so much more robust against label imbalance, I generally tried to maintain data integrity as much as possible and thus didn't oversample the minority classes and even when undersampling the 0 class, I would still keep half or more of the samples.

Finally, I tried feature engineering by reducing the records to a statistical summary (mean, median, standard deviation, etc) but the perforance on these features proved worse so I scrapped this idea.

Models and Results

First, I trained a simple Logistic Regression model on the data as a baseline before moving to implement an LSTM model, a neural network that is often recommended for Time Series classification tasks. I used two LSTM layers of 64 units each with a Dropout of 0.5 between the two followed by a Dense layer of 32 units and final Dense output layer with softmax activation so as to yield the 5 probabilities. Despite many different architectures tried, the results on the test split of the train data remained poor with the best f1 score observed being around 0.49.

Then, I did some more research as to the best models considering the unique nature of our dataset (heavy data imbalance) and ensemble methods seemed to be the way to go. I tried RandomForest, GradientBoost (XGBoost), and AdaBoost with generally positive results and the best for RandomForest and XGBoost. Though similar, XGBoost clearly had better performance, so my final task was to finetune the hyperparameters to yield the best performance on the test set. I set the objective to softmax and eval-metric to logloss after some reasearch and experimented with the number of estimators in order to cover all the xcomplexity in the data, settling on 500. I also briefly tried PCA for feature selection with a Random Forest but observed very poor results so I scrapped it. The final results from the test split of the Train set are in the table below:

Model	Self-tested F1 score
Logistic Regression	0.48
LSTM	0.49
Random Forest	0.88
Random Forest with PCA	0.24
AdaBoost	0.49
XGBoost	0.91

Table 1: Results of different models tested

In the end, my XGBoost model with 500 estimators yielded my best results and placed me where I am on the leaderboard now.