Artificial Neural Networks and Deep Learning 2022

Challenge 2

Prof. Matteo Matteucci

Lorenzo Trevisan (Personal Code: 10687901)
Giuseppe Urso (Personal Code: 10628602)
Matteo Venturelli (Personal Code: 10629913)

POLITECNICO
MILANO 1863

1 Problem Definition

The goal of this challenge consists in solving a time series classsification problem. Given a training set of 2429 samples, the team was supposed to predict the correct class label of the samples contained in an unknown test set. Each sample is a (36×6) vector which describes the temporal evolution of 6 features along 36 time instants.

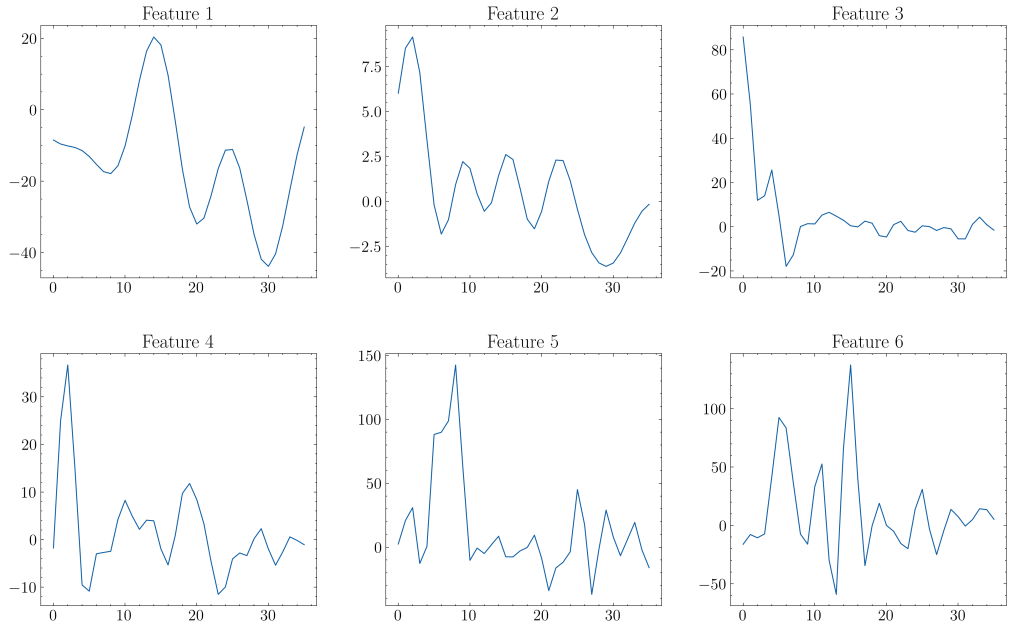

Figure 1: One of the (36×6) samples contained in the training set.

2 Proposed Solution

Time series classification is an important and challenging problem in data mining. Several deep learning architectures can be used to solve this task, and each one has its strengths and weaknesses. For this reason there's not a general model that can be applied to any scenarios; different architectures have to be tested in order to select the one that works better for the specific case study.

2.1 First Model

First of all, the 2429 samples dataset was split into training and validation subsets (with proportions 0.85 and 0.15 respectively), then the samples were organized in batches with `batch_size = 128`. In this first model no preprocessing was implemented. The actual model consisted in a basic LSTM network:

- input layer (`input_shape = (36, 6)`);
- first LSTM layer (`units = 128`, `return_sequences=True`);
- second LSTM layer (`units = 128`, `return_sequences=False`);
- droput layer (`rate = 0.5`);
- dense layer (`units = 128`, `ReLU` activation function);
- output dense layer (`units = 12`, `softmax` activation function).

The output layer had a number of neurons equal to the number of possible classes and `softmax` activation function in order to be able to interpret the output values as probabilities.

We set `SparseCategoricalCrossentropy` loss function and `Adam` optimizer with learning rate equal to 10^{-3}. The number of epochs was set to 200 as maximum value, but in practice was ruled by `EarlyStopping` callback with `patience = 25` on the validation accuracy. `ReduceLROnPlateau` callback with `patience = 10` on the validation accuracy was also implemented.

The training dataset presented high class imbalance. The least represented class contained only 34 samples, whereas the most represented one contained 777 samples. To avoid having the network perform much worse on some classes with respect to the others, we trained the model with `class_weight` argument, which allows to "pay more attention" to examples from an under-represented class, improving their accuracy.

The best results were obtained at epoch 25 and showed clear signs of overfitting: the training accuracy reached 0.9679 whereas the validation accuracy reached the much lower value of 0.6297.

2.2 Further Improvements

Generally neural networks converge faster if the input has been whitened to zero mean and unit variance. Data inspection revealed that input features were far from white: the mean of each feature was of the order of 10^2 and the standard deviation of each feature was of the order of 10^3.

Different approaches were tried out (both feature-wise and on the whole data):

- `MinMaxScaler()` was the first preprocessing method applied. Because of the high variance of the data, the team expected this typer of scaling to shrink the majority of the data to a very shallow range of values, causing learning incapability. As a matter of fact, very bad results were achieved in terms of training and validation accuracy;
- Input whitening (zero mean, unit variance) by means of `StandardScaler()` method of `scikitlearn` library showed minor improvements (approximately 1% on validation accuracy);
- `RobustScaler()` scaling method yielded the best performance since it improved validation accuracy by almost 3%. This scaler uses statistics that are robust to outliers: it removes the median and scales the data according to the range between the 1st quartile (25th quantile) and the 3rd quartile (75th quantile).

In order to be able to apply the same scaling on the test data, the scaler parameters were imported into the submission model by means of `pickle` library.

Dropout

Frammentazione dei sample

Bilstm 13.12 - Conv1d 13.12 - Transformers

2.3 Final Model

3 Plots