Homework 2 - Time Series Classification

Artificial Neural Networks and Deep Learning

Filippo Pagella, Luca Song, Christopher Volpi

December 23, 2022

1 Introduction

In this competition we have to work on a classification problem with multivariate time series data. We start with an exploratory analysis of the dataset, considering different strategies for data preprocessing, then we build simple models, belonging to different typologies while improving them step by step. In the end, we perform a selection of the best models and adopt ensemble learning techniques.

2 Data inspection

The dataset contains 2429 samples, where each sample is a time series of six different features having 36 time instances. For the classification task of this challenge there are 12 possible outcomes (from 0 to 11).

By inspecting the training data and grouping them in class labels, we can see that there is an unbalance between the classes on two different levels as shown in figure 1: class 9 has the most occurrences by far, then there are class 2,3 and 6 with a significant number of samples. The other eight classes have very few samples.

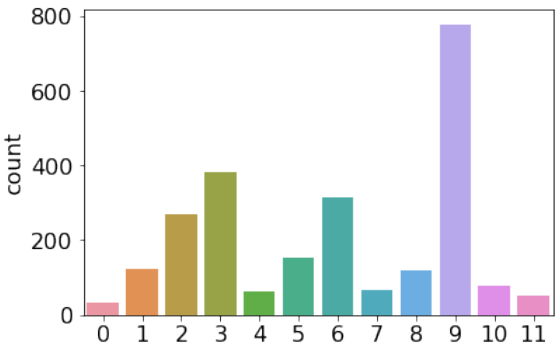

Figure 1: almost the 32% of the total samples belong to class 9. Classes 2,3 and 6 are also significant, the other together make up to almost the 30% of the total samples.

To face this problem we experiment different approaches: data augmentation, upsampling (and downsampling) and class weights.

3 Data preprocessing

As a first step we split the data, generating a stand alone test set, in order to better evaluate the model performances. The configuration we use to train the models, to submit, is 80% for the training and 20% for the validation set; considering a smaller partition for the validation set could lead to very few or even zero samples for some classes (in particular class 0).

We have more than one possible choice to preprocess data. The ones that sound the most sensible (in our opinion) in this time series problem are: normalization or standardization over all the data samples or for each singular time series (grouping and considering each of the six features

singularly). However, the results obtained through those methods where slightly worse than the prior ones.

We adopt other types of preprocessing, like breaking in half the time series sequence of each sample to see if the model could learn better, gaining more samples (since it's like doubling them from the original 2429 ones) and having shorter sequences to analyze (18 time instances instead of 36). Unfortunately, this attempt shows unsuccessful results.

We try to make a sort of feature selection to see if some of the variables are "disturbing" the training phase. The only thing, we observe, is that we don't have a relevant drop in the performances when we singularly discard some of the six features.

To manage the classes unbalance problem we try to append synthetic data to the original dataset. We adopt two different strategies. The first consists in applying singular or even multiple transformations to each time series of the original data, belonging to the less populated classes. The possible transformations we consider are: scaling, magnitude and time warping, window slice and window warping. The problem we face is that using them lead to an increased probability of misclassification. Maybe with more time we could find the optimal configuration.

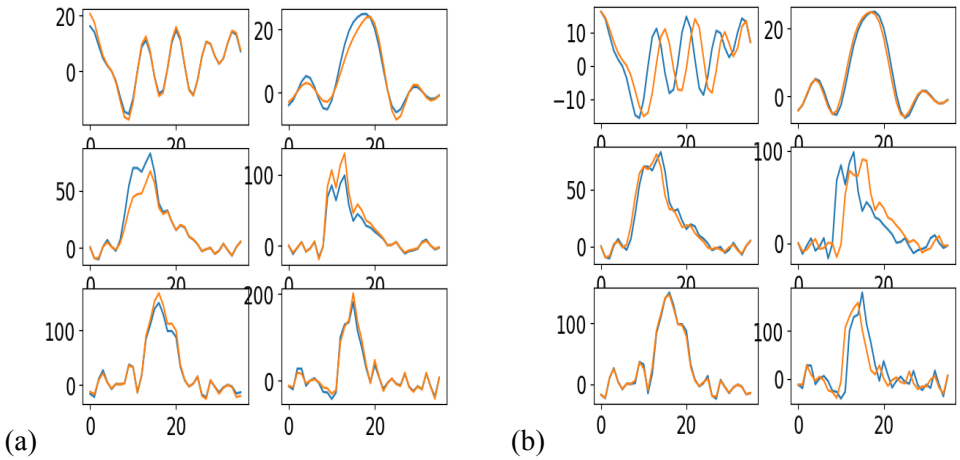

Figure 2: in (a) magnitude warp, in (b) time warp.

The second strategy can be seen as a sort of jittering and consists in computing the noise, from each time series and creating new samples, by adding to the original ones a factor, sampled from a normal distribution with zero mean and standard deviation equal to the noise divided by a constant:

$$X_i^{new} = X_i + 0.2 * \mathcal{N}(0, E[\epsilon])$$

where $\epsilon = X_i - \mu_i - \sigma_i$ is the noise, μ_i is the mean and σ_i the standard deviation of the i^{th} time series. This new synthetic dataset guarantees more training samples but leads a bit to overfitting, since the new data are very similar to the original ones and we have a poor validation set for the majority of the classes.

A few models show a slightly better accuracy in some classes using class weights, but with a trade off with the overall accuracy (same when downsampling class 9): our belief is that the hidden test set is also unbalanced and that the scarcity of data in some classes is a too hard obstacle.

4 Models development

The two main typologies of models we consider are: reccurent neural networks (RNN) and one dimension convolutional neural networks (CNN). We choose RNN because we have sequences of data, in particular we focus on Long short-term memory (LSTM) networks. We start by developing shallow architectures, with a low number of parameters, then we increase the complexity of the networks step by step. When experimenting with LSTM models, we consider, as well as the classical layers, also bidirectional LSTM layers (BiLSTM), which can interpret input information from both the past and the future, relative to the current state. We give some tries also to gated recurrent units, without much success. We observe that just one LSTM (or BiLSTM) layer isn't enough to capture, satisfying, "temporal" dependences: we get better performances while stacking

two layers and then adding two dense layers, before the Softmax classifier, with some dropouts. Stacking more than two LSTM layers or increasing too much the number of units leads the model to quickly overfit the training set.

Generally, we achieve greater results with Convolutional Neural Networks, for this reason we experiment more architectures belonging to this family. We develop three slightly different models: one lighter network composed by three convolutional layers to extract "spatial correlated" features (with less than 60.000 parameters) and two other networks with more parameters but just two convolutional layers. In particular the second model is composed by a dilated convolution (which skips some data in the sequences), instead the last model has classic convolution operations with an average pooling layer between. All these three architectures are followed by a global average pooling operation and a classifier block with two dense layers and a final Softmax dense classifier. The last model we develop is composed both by convolution operations and LSTM units. In particular it has three first convolutional layers followed by two BiLSTM layers and by a dense classifier block (as shown in table 1).

Table 1: Conv + LSTM

Layers	**Filters/Units**
Input	
Conv1	32
Conv2	64
Conv3	128
Bidirectional(LSTM)	128
Bidirectional(LSTM)	128
Dropout	0.2
Dense	64
Dropout	0.2
Dense	64
Output	12

5 Models evaluation and ensemble architecture

Before selecting the models to use in the ensemble model, we evaluate their performances by computing, not only the accuracy, but also precision, recall index and in particular F1 score, to take into account the unbalance of the dataset. The covolutional neural networks bring us to better results, between 66-70% on the test set, while LSTM architectures seem to overfit in some cases. We develop two ensemble learning strategies: bagging (bootstrap aggregating) five similar convolutional networks, trained on five different (but with possible intersection) portion of data and assembling five different models, taking their average predictions. The best ensemble model, we have developed, is composed by three CNNs, an LSTM architecture and an hybrid Convolutional-LSTM network. These architectures are the ones described in the chapter before. With this model we get 75% of accuracy in the final test set (boosting by far the scores of the single models). With another version of this model, with the same architectures, but trained with class weights, we reach a slightly worse overall accuracy, but with some improvements for certain classes (for example class 4 from 12% to 27%).

6 Conclusion

In conclusion we can say that the main difficulty of this competition is to manage the unbalance of the dataset and the scarcity of the samples for some classes. Thanks to an ensemble model we could reduce the variance of the misclassification error, reaching 75% of overall accuracy. Possible future improvements are: a more suitable data augmentation and the implementation of attention mechanisms to weight more the most important parts of the input.