Machine Learning
Assignment 2 - Eplepsiae

UNIVERSIDADE D
COIMBRA

André Carvalho
andrecarvalho@student.dei.uc.pt
no. 2019216156

Paulo Cortesão
paulocortesao@student.dei.uc.pt
no. 2019216517

Class PL2

November 2022

Contents

1 Introduction

This report aims to describe the process undertaken in the development of a classifier for the detection and prediction of epileptic seizures. The various relevant phases for the completion of this project will be described, including input processing, code development and structure, and results obtained.

2 Input Processing

The data to process was given as a feature vector and a binary matrix. These structures had to be processed so that a more concrete classification was obtained, i.e., with the 4 classes mentioned in the statement. As well as this, given the constraints presented by the MATLAB documentation, they had to be transformed into cell matrices and categorical vectors. As well as this, the 29*29 images to work with in the CNN architectures had to be generated. All this preprocessing is performed in several files, including *make_classification* and *prepare_images*. The final result was wrapped in *prepare_data*.

3 Code Development

After the main preprocessing tasks were implemented, the code for building the various networks was developed, and a file for each different architecture was created. The standard conventions were applied and, after a number of experiments was performed, the various structures crystallised into well-defined parameters.

3.1 Data Balancing and Error Weights

As mentioned in the statement, the data to use for training and classification is unbalanced and the most relevant classes (2 for prediction and 3 for detection) are underrepresented. As such, random subsampling of the dominant, interictal instances was undertaken so that they would constitute only at most 50% of the dataset, in opposition to the 90%. As well as this, error weights were defined, so that changes in a network were more significant after training with a more rare class. In an initial phase, the weights were defined in inverse proportionality to the number of instances of each class, but then other combinations were tried, including extracting the squared root and multiplying by fixed factors. After this definition, given that the inclusion of this concept varies with architecture, the various networks were adapted in order to apply it.

3.2 CNN architecture

To allow for the use of this kind of networks as was commanded in the statement, two-dimensional, 29-by-29 matrices were generated from the inputs. Each matrix was obtained from 29 consecutive samples (a window), all belonging to the same category. In order to obtain the whole dataset, the window from which each matrix would slide by 1 unit, yielding a number of images slightly smaller than the number of samples. The sequence of layers for the CNNs would be comprised of a convolutional layer with 3 filters of size 4-by-4; followed by a reLu layer, a max-pooling layer of 2-by-2 windows with stride 2, a fully connected layer, a softmax layer and a classification layer.

3.3 LSTM architecture

An important parameter in this kind of neural network is the number of hidden units. In an experimental stage, various numbers were tested (including 5, 20, 50 and 100), and it was concluded that 20 hidden units would be adequate for the classification of the dataset. The training time was not too long and the relevant metrics of sensitivity and specificity would be similar to the ones respective to the higher numbers. As far as the network is concerned, after the LSTM layer, a fully connected layer would lead to a softmax and, finally, to a classification layer. The major variations in the tests conducted with this architecture had to do with the error weights. In an initial stage, they were set to be inversely proportional to the frequency of the respective classes; but then

other combinations were tested. For example, the square root or the square of the elements was used as a weight. It is also important to mention that the ADAM optimizer was adopted, as preliminary tests showed its superiority.

3.4 Recurrent Network

In order to test the performance of recurrent neural networks in this kind of data, the *layrecnet* function was used. Various values for number of delays were tested, including 5, 20, 30 and 100. In the last two cases, the program ran out of memory, so there was a somewhat narrow limit to what could be tested. Error weights were also used in order to compensate for the imbalancing of the data.

3.5 Autoencoders

As suggested in the statement, some autoencoders were assembled and tested to draw conclusions on how dimensionality reduction affects the loss of information and network predictive power. For this, autoencoders with 15, 10 and 3 features as output were built, and their outputs were fed to neural networks as training data. As well as this, clustering algorithms were applied to the 3D points resulting from one of the autoencoders. The following graph allows to see their dispersion throughout space:

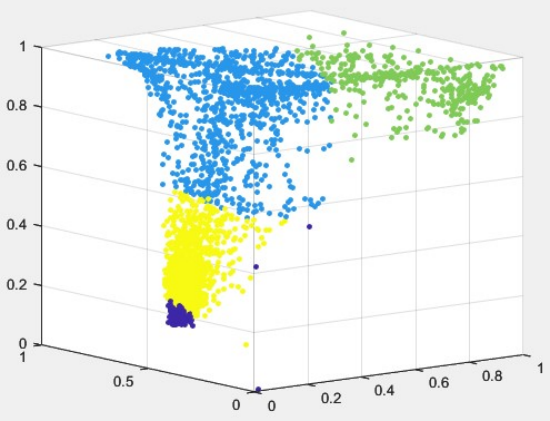

Figure 1: Kmeans algorithm results when applied to data with 3 features.

As well as this, stacked autoencoders were used. Three layers from 29 features to 3 features were applied, and the ratio of feature reduction between layers was meant to remain as constant as possible. With a 29-14-6-3 autoencoder, when representing the points obtained in space, a straight line appeared, and even though there was some correlation between point location and category, there was not a clear cluster separation between them, so unsupervised learning methods were not applied in these cases.

4 Results

The following tables aims to describe some of the results registered when testing the various networks. It is important to state that most have been omitted, given their redundancy

LSTM								
Wts.	Pat.	No. feat	Units	L. Rate	Sens.2	Spec.2	Sens.3	Spec.3
1	1	29	100	10^{-3}	0	1	**.999**	**.975**
1	1	29	10	10^{-3}	0	1	.999	.974
w	1	29	20	10^{-4}	0.592	.502	.998	.603
$w^{3/2}$	1	29	20	10^{-3}	.729	.502	.996	.515
1	1	29	50	10^{-3}	0	1	.999	.972
1	1	29	5	10^{-3}	0	1	.998	.970
w	2	29	10	10^{-3}	**0.916**	**1**	0.981	0.590

CNN					
Wts.	Pat.	Sens.2	Spec.2	Sens.3	Spec.3
1	1	0.83	0.62	1	.800
1	1	0.802	0.507	1	.726
1	1	0.840	0.516	1	.846

As far as the recurrent networks are concerned, the results dispense a table: they would always consist of a class, which could vary according to error weights.

Shallow NN					
Wts.	Pat.	Sens.2	Spec.2	Sens.3	Spec.3
0	1	0	1	.996	.536
1	1	0	.493	.998	.97
1	1	0	.493	.998	.97
1	1	.963	.50	.998	.939
1	1	1	.5061	.998	.939
1	1	.962	.505	.998	.927
1	1	.947	.505	.998	.956
1	1	.936	.505	.998	.927

5 Discussion

Given the results displayed by the various architectures, it is possible to draw some conclusions. They are subject to the fact that the data in analysis corresponds to spectral coefficients in a time series. The main conclusions are listed here:

- The main regularities in the dataset can be learned. Given the high specificity and sensibility in some cases, there are networks that detect them, which gives them a high classification power.

- Time considerations are a relevant factor in the classification of this dataset. Proof of this is the fact that the training of shallow networks would not yield good results and would normally always return the most frequent, inter-ictal category; or just one category, if weights were used. This means that the context that surrounds a given sample is paramount for determining its category.

- The LSTM architectures have the best classification power (in the best case 99% of specificity and 98% of sensibility). Given their capacity to analyse time series, they can infer correlations between feature vectors in different instants, something that is inherently impossible in shallow networks, for example.

- There is a relative predictive power in convolutional neural networks. This is shown by their acceptable performance when compared to the other architectures. Given the various filters trained and employed in the convolution phase, they can detect some patterns that span through time and correlate to some classes. This allows them a specificity of 99% and a sensitivity of 80% in the best cases.

- From the training of networks, some actions can be performed to improve the results. It was concluded that, if the gradient associated to the error function increases with no improvement in the performance function, then the learning rate must be decreased: this is evidence of the exploding gradient problem. On the other hand, if there is no change in both performance and gradient, even though the gradient is positive, this means that the learning rate is too low and needs to be increased.

- The use of error weights has a significant impact on the results obtained. In general, when they were not used, only the majority category would be obtained. Only after using them would there be different and better results. It was using this method that the best networks were obtained, both in prediction and detection. However, in some cases

- Recurrent networks had a poor performance in classification, as only one category was obtained as output.

- The autoencoders seem to be able to induce some information loss in the data. When considering analogous architectures, the performance dropped sharply when using only 3 features as input ($value_1$ vs. $value_2$). Still, if the performance when 15 features were used as input, it can be seen that there is not a significant drop in predictive power.

6 Conclusion

From this work, it was possible to achieve good results, both in prediction and detection of epileptic seizures, even though there is normally a compromise

between both within a network. The relevance of the neural networks used, especially the ones most fit for time series, became evident, as they were applied to a real-world situation involving human health. As well as this, a greater familiarity with the methods used for research in this field and associated metrics (sensitivity and specificity) as attained. As such, it is considered that the goals proposed for this work were achieved.