FCTUC DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

MACHINE LEARNING (MEI, MEB)

2023-2024

Assignment nº 2 – Prediction and detection of epileptic seizures.

(Theoretical content: Chapters 4, 5)

Learning objectives:

1- To build, train and test multilayer networks for classification of big data sets.
2- To build, train and test dynamic neural networks (with delays) for multidimensional time series prediction.
3- To face the problem of features reduction with autoencoders.
4- To configure, train and test Convolutional Neural Networks for multiclass classification (deep learning).
5- To configure, train and test LSTM (Long Short Time Memory Neural Networks) for multidimensional time series classification.

Before starting the work, study the theoretical materials needed for the assignment, and carefully read this document until the end.

1. The problem

Epilepsy is one of the most prevalent neurologic diseases, affecting about 1% of the population, everywhere (in Portugal about 100 000 patients). In the present state of medical knowledge, about 30% of the epileptic patients are untreatable by drugs or surgery. They can suffer from a sudden seizure, anytime, anywhere, "*like a bolt from the sky*". A seizure may last for some seconds or some minutes, and affects seriously the motor, perception, language, memory and consciousness; consequently, the social and professional abilities of the patients and their families diminished seriously.

The possibility to predict or to detect a seizure, based on the information from brain signals, principally the EEG (ElectroEncephaloGram), is still an actual and challenging research problem. If it would be possible to develop an algorithm to detect on time a seizure, new strategies for disarming the seizure could eventually be developed. On the other hand seizure prediction would allow the patient to take action for his/her own safety and social exposition during the seizure.

An electroencephalogram is a set of electrical signals (Figure 1), in the scale of microvolts, collected by electrodes inserted inside the brain by surgery (Fig. 2b) or glued to the scalp (Fig. 2a).

It is supposed that many neurologic information is embedded in the EEG signals. To predict or detect a seizure, one needs to develop signal analysis methods able to detect patterns in the EEG typical of a preictal or of an ictal state, respectively.

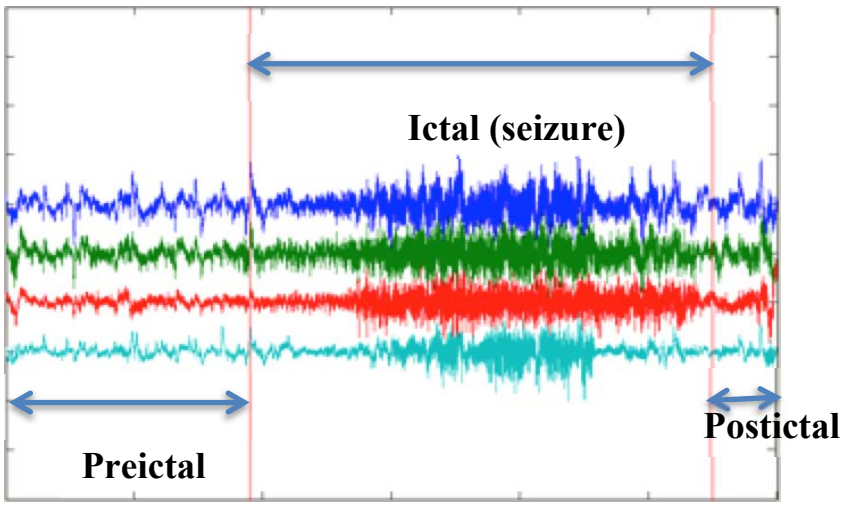
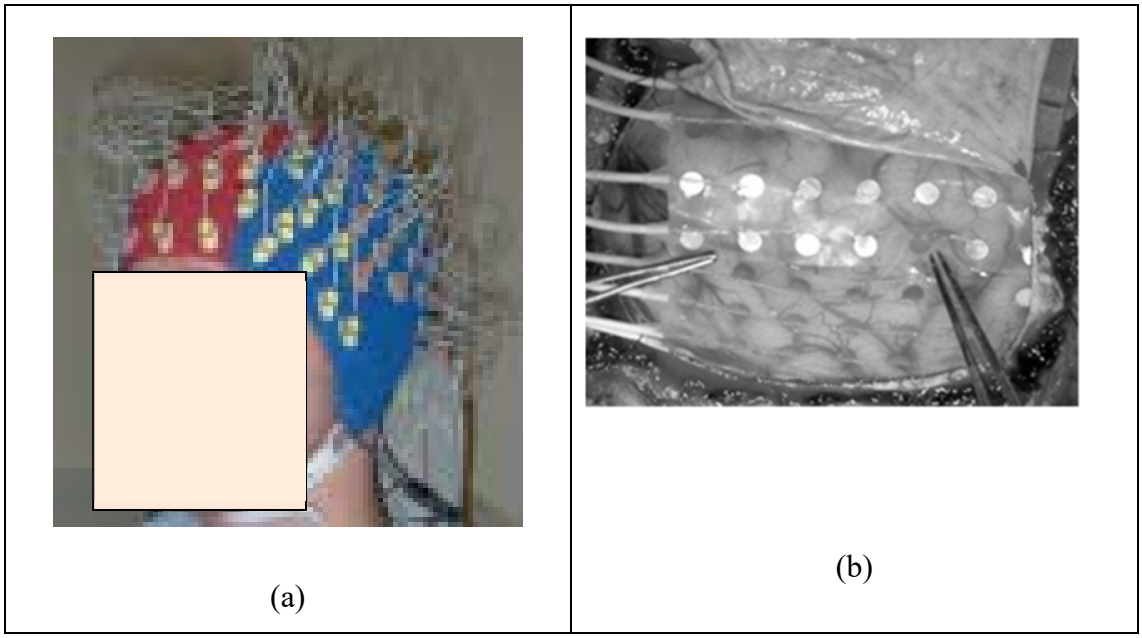


Figure. 1.EEG with one epileptic seizure (view of 4 channels).



(a) (b)

Figure 2. (a) Surface (noninvasive) electrodes, (b) Intracranial (invasive) electrodes. Each electrode captures an EEG channel voltage relative to an electrical reference. Voltages are in the scale of microvolts.

2. The classification approach

For the prediction and detection of seizures, the brain state will be classified in this assignment into three states. A three output NN is needed, one output for each class.

1- Class **Interictal**, the normal brain state (output of the classifier NN 1 0 0).

2- Class **Preictal**, a seizure is coming (output of the NN 0 1 0).

3- Class **Ictal,** a seizure is happening (output of the NN 0 0 1).

The Postictal phase will be considered as Ictal. Consider the Postictal the interval of 1 minute after the end of each seizure in the datasets; after it is interictal.

The **5 minutes preceding** a seizure will be considered the **Preictal** state.

The big question, previous to any classification, is to extract from the EEG a good set of features capable to distinguish the 3 classes. Although more than 45 years have passed since this question has been considered in medical and scientific fields, such a set of features has not yet been found. In the present work the set of features to be used exploit the different frequency bands of the electrical signals, and have been extracted by Doctor

Mojtaba Bandarabadi, during his PhD studies in our Department (we thank and acknowledge him for that). They refer to one EEG invasive channel placed in the epileptic focus (the place in the brain where the seizure starts) of patients of the European Epilepsy Database, developed by the FP7 Epilepsiae project (www.epilepsiae.eu).

The features have been extracted using 2 seconds EEG segments with 50% superposition between segments (i.e., consecutive time-windows share half of the information). In practical terms every second a vector of 29 features is obtained. These 29 features are characteristic of the frequency spectrum of the EEG channel (see at the end of this document). For the complete register of a patient, a high number of vectors is obtained, and the objective of the present work is to classify them among the defined three classes, principally to the pre-ictal and ictal classes.

The performance of the classifier is measured by the sensitivity (how many true seizures did it preview or detect, high number corresponds to high sensitivity) and by the specificity (how many false seizures did it predict or detect, low number corresponds to high specificity). For clinical usage, both high sensitivity and high specificity are needed. Probably the performance is different for prediction and detection.

The sensitivity (for prediction or for detection) is defined by

$$SE = Sensitivity = \frac{True\_Positives}{True\_Positives + False\_negatives} = \frac{TP}{TP + FN}$$

And the specificity (for prediction or for detection) is

$$SP = Specificity = \frac{True\_Negatives}{True\_Negatives + False\_positives} = \frac{TN}{TN + FP}$$

Each group will work with the data of two patients.

3. The plan of the assignment

The following challenges are proposed:

- Build and train multilayer NNs, without and with delays (dynamics), with good performance, for each patient, one in predicting and another in detecting (it is very unlikely that one will be good for both, but you can try). You will use all the features and, alternatively, a reduced number of features to be extracted by autoencoders.

- Train autoencoders to reduce the number of features for the classifiers. You can choose the number of features for example by trial and error.

- Build and train deep networks (CNN and LSTM) for predicting or detecting the seizures.

The training and testing datasets must be representative of all the possibilities, so they must include all the three classes, namely seizures. If you use validation, the validation set must also have seizures. Use indexes to divide the dataset in training, validation and testing, since the data must be temporarily ordered, except in the training data when class balancing is made (see Note 3).
Take full freedom to choose your NN, even if it is outside of those studied in the class, but allowing to fulfill the 5 learning objectives.

For example, one alternative approach, for seizure detection, would be to consider only two classes, ictal and non-ictal. More enthusiastic groups may find interesting to try it.

The report must describe in detail each used NN: type, number of layers, number of neurons per layer, activation function, learning and training function, etc.

4. Some remarks:

4.1 – Choosing the architecture

In a first approach, one can use a normal feedforward NN (with the function *feedforwardnet*). However, considering that the brain is a dynamical system, it has memory, and as such it will be interesting to face the possibility to introduce delays in some features, or in some layers. For example, the layer Recurrent Network can be used:

net=layrecnet(layerDelays,hiddenSizes,trainFcn) (see help in Matlab)

In this architecture, the output of each layer is fed back, with one delay, to the input of the same layer. The toolbox has appropriate training algorithms for these architectures.

The final application to be developed by each group must have a GUI for:

- choice of the of the NN to be trained and tested (shallow and deep),
- choice of a NN already trained that is accessible to the user (shallow and deep),

- selection of the datasets (for training and testing, and also for validation if it is used),
- presentation of the results of training and of testing, with the computed sensitivity and specificity shown in the GUI.

The *guide* can be used to built the GUI or by the **App Designer->**`appdesigner`, as in the first assignment. It allows to develop quickly and easily any GUI. Just type *guide* or *appdesigner* in the command line. There is an online manual for both.

4.2. Building the training and testing sets, class balancing, using error weights

One important issue in problems involving the detection of rare events, as it is the case of seizures, is the short duration of an event compared to the long duration without events. In the present case, a seizure may last in average 90 seconds, while the average registered time per patient is 162 hours. From the classification point of view, this means that the number of data points for the ictal class is much lower that for the other classes, specially the interictal class that contains more than 90% of the points. Without some precaution, the results can be good if the NN classifies well all the interictal points and classifies wrongly all the ictal (or preictal) points. This means, a good result, from the point of view of statistics, but a catastrophic result from the point of view of the aim of the problem.

One common way of preventing this effect, is to equilibrate the number of points of the several classes in the training set, but not in the testing set. This is the class balancing approach.

For a patient, a number of interictal points at most equal to the sum of the points of the other classes should be chosen, for example randomly. This corresponds to an undersampling of the interictal phase.

Moreover, even with (or without) class balancing, it may be interesting to give more importance to the ictal (for detection) or preictal (for prediction) instants. This may be done by the use of different weights (penalization) of the error of the NN in the different instants. Higher weights mean that the optimization algorithm will give more importance to that instant and find a solution with small error in that instant. See the toolbox User's Guide 2023b, page 24-41 or Matlab online help "Train Neural Networks with Error Weights".

Without class balancing, if for example there are 1000 more interictal instants that ictal ones, then the weights of the errors in ictal instants should be at leas 1000 greater than the weights of the interictal instants. Even with class balancing better results can be expected if the weights for preictal (prediction) or ictal (detection) are higher. Let us say that an adequate choice of the weights "specializes" the network for the class with higher weights.

5. Shallow neural network training styles (as in the OCR assignment)

There are two learning styles for shallow networks: the incremental and the batch. This is true for any number of layers in the NN.

a) **Incremental learning**

One input at a time is presented to the network, and the weights and bias are updated after each input is presented. There are several ways to do it:
net.trainFcn='trainc'
net=train(net,P,T): "trainc trains a network with weight and bias learning rules with incremental updates after each presentation of an input. Inputs are presented in cyclic order."
net.trainFcn='trainr'
net=train(net,P,T) : "trainr trains a network with weight and bias learning rules with incremental updates after each presentation of an input. Inputs are presented in random order."
When these learning methods are used, the algorithms must be iterative and they are implemented in the toolbox with names started by *learn* as for example:
learngd – gradient rule
learngdm – gradient rule improved with momentum (see help)
learnh – hebb rule
learnhd- hebb rule with decaying weight (see help)
learnwh- Widrow-Hoff learning rule

The learning function is specified by
net.adaptFcn='learngd', for example.
Incremental learning can also be done by

net=adapt(net,P,T), but it is mandatory in this case that P and T be cell arrays. If they are matrices then adapt is in batch, the same as train.

b**) Batch training**

net.trainFcn=’trainb’“trainb trains a network with weight and bias learning rules with batch updates. The weights and biases are updated at the end of an entire pass through the input data.”
net=train(net,P,T) , train by default is in batch mode.

In these methods the algorithms are in batch implementation, and their names start by train, as for example

traingd	gradient descent
traingda	gradient descent with adaptive leaning rate
traingdm	gradient with moment
trainlm	Levenberg- Marquardt
trainscg	scaled conjugate gradient

Note that learngd and traingd both implement the gradient descent technique, but in different ways. The same for similar names. However trainlm has no incremental implementation, only batch. They are specified by *net.trainFcn=’traingd’* for example.
Basically, the training methods that improve the gradient use second order information (second derivative, or the Hessian) building the Quasi-Newton methods family, that combine successive gradients in order to improve convergence, as the conjugate gradient family (Chapt. 9 of Hagan and Coll.).

6. Using parallelism and graphical processing unit (GPU)

The backpropagation training algorithms can be run in parallel and using the graphical processing units (GPU). The Parallel Computing Toolbox must be installed.

The process is quite simple, needing only the specification of some calling arguments of the train method, such as for example (from help):

```
net = feedforwardnet(140,'trainscg');
net = train(net,O,T,'UseParallel','yes','UseGPU','yes');
Y = net(P,'UseParallel','yes','UseGPU','yes');
```

For more details type > *help train*.

You may also create the network with

net=network, which is the most general way.

And then define the properties and options of the *net.* See Chaper 31 of nnug (user's guide).

7. The supplied datasets have the following variables inside:

- FeatVectSel: Features matrix, where each column represents an extracted feature, and each row a vector of features. So this matrix must be transposed to make the P matrix defined in class (one column is a vector of features).
- Trg: A column matrix that identifies the seizures. It is made with 0's and 1's. The value 0 means non-ictal class (so interictal, preictal, or postictal), and the value 1 ictal.
- The points of the preictal are easily defined: 300 points before the first 1 for each seizure, corresponding to 300 seconds=5 minutes. Similarly, the postictal points are those 60 after the last 1 of each seizure (1 minute); they are joined to the ictal class.
- The Trg vector must be changed to identity all the three classes: use 1 for interictal, 2 for preictal, 3 for ictal.
- After that, the target matrix T defined in the class must be built in accordance to the changed Trg. Using the suggested values, the points 1 in Trg correspond to [1 0 0]' in T, the points 2 in Trg to [0 1 0]' in T, the 3 to [0 0 1] in T.

8. Postprocessing:

In an initial approach, one classifies point by point, i.e., each point must be assigned by the classifier to one of the three classes. The errors and the performance are computed point by point.

In a more elaborated approach, one may consider the following hypothesis: a seizure is predicted only if, for example, one finds 10 consecutive points classified as preictal, and a seizure is detected only if the classifier finds 10 consecutive ictal points. However, a

more relaxed hypothesis can be followed, and instead of 10 consecutive one may accept 5 among the last 10 points as a tentative threshold.

Groups with even numbers work with patients 54802 (31 seizures), 112502 (14 seizures).

Groups with odd number work with the patients 44202 (22 seizures), 63502 (19 seizures).

If one group wants to test in more than two patients, feel free to do it. Data from four patients is available for all.

9. Remark about the used features

The information given by the frequency content of the EEG signals is considered indicative of the brain state: different states produce different spectra, meaning that the energy of the signal is distributed among the several frequencies depending on the brain state.
The spectra measure the contribution of each frequency for the total energy of the system. The spectral power of a frequency band is given by the sum of the powers of each frequency in the band. It is supposed that the distributions of the spectral powers are different in the four considered states.
In the present study are used, by empiric choice, the normalized power spectra (in such a way that their average is null, and variance is unit) of the following 29 frequency (Hz) bands, covering the total band of 0.5 to 512 Hz:

0.5-3, 3-5, 5-8, 8-10, 10-12, 12-14, 14-16, 16-18, 18-22, 22-26, 26-30, 30-35, 35-40, 40-48, 52-60, 60-70, 70-80, 80-90, 90-98, 102-125, 125-148, 152-175, 175-198, 202-248, 252-298, 302-348, 352-398, 402-512, 0.5-512Hz.

10. Reducing the number of features with autoencoders

The 29 features may be reduced to a lower number by a simpler autoencoder or a stack autoencoder. The number of used features is the number of neurons in the smaller middle layer. You can try for example 15, 10, and 3 features and compare the results. The reduced set of features may be given to a multilayer NN or to a LSTM as time series.

With three features you can see the data in 3D graphics. Although the number of points is high, and so the figure will not be very clear, you can try to cluster the 3D features into 3 classes (using k-means and DBSCAN, for example) and see if there is some useful information from clustering.

11. Preparing the data for deep learning with CNN

CNN can be used as image classifier if the inputs are given in appropriate format. To convert the features time series into 2D images, we can define (square) data windows as images if all the instants in these windows correspond to a single class. For example, take a matrix composed by 29 instants of the interictal period. Since we have 29 features, this will result in a matrix 29x29 of real numbers that can be given to the CNN as a grey 2D image, eventually after some preprocessing (normalization to (0,1) for example). In this way a number of interictal images can be generated by successive 29 s windows (in a first approach non-overlapping). The same for pre-ictal, ictal windows. The problem of class balance appears here in the same way as before. The target matrix for classification is composed of categorical vectors corresponding to the classes of the images, using the same coding as before.

The training data is an array with four dimensions: 29 x 29 x 1 x NumberOfImages. Each image is in the fourth dimension. The 1 is there because the images are grey (monocolor), so there is only one channel. Colored images have 3 channels - Red, Green, Blue- and the data for training would be 29 x 29 x 3 x NumberOfImages.
The array is similar to the ones in

```
> [XTrain, YTrain] = digitTrain4DArrayData;
```

Where Xtrain is the training data (a collection of 5000 images, each one given by a matrix 28 x 28 real numbers) and Ytrain is the target- a categorical vector with numbers 0-9. In our case we have a categorical matrix with three columns.

You are free to create the architecture of the CNN (how many convolutional layers, other types of layers, etc.). The last fully connected layer must have four outputs, and similarly the softmax and classification layers.

12. Preparing the data for deep learning with LSTM

LSTM performs sequence learning. In this case, the sequences have 29 components. To train the LSTM it is only needed to specify that 29 in the number of components of the time series. The extension of the series is extracted from the dimensions of the number of instants in the training set. You may find inspiration in the example of classification of Japanese vowels spoken by several persons. See in help "Sequence classification using deep learning" and look at `[XTrain,YTrain] = japaneseVowelsTrainData;`
This assignment will challenge you with a dataset with a dimension and complexity already representatives of real problems for machine learning.

The report must be delivered to inforestudante in a file named
ML2023EpiReportPLxGy.pdf
All the classifiers, all the code, and the report, must be delivered in a compressed file **ML2023EpilepPLxGy.zip** (or .rar).

Each file*.m you deliver must have an introduction describing what it does and the authorship.

The report must also have a table synthetizing the report: type of training networks, how many you trained, how many trained networks you delivered, the best among all for prediction, the best among all for detection.

Wishing you a nice work. 23/10/2023.
ADC

Annex. Details about the patients (from The European Epilepsy Database**,** built by the European Project FP7 **EPILEPSIAE, www.epilepsiae.eu).**

ID	Sex	Patient age (y)	Onset age (y)	Localization of seizures	Seizure type	Total EEG Recording (h)	No. of seizure	Seizure duration (s)		
								Mean	Min	Max
44202	M	21	5	RLT, RMT	CP(19), UC(3)	170.6	22	131.6	19	199
54802	M	17	1	LMT, LLT, L-T	SP(25), SG(6)	142	31	118.4	33	274
63502	F	63	30	LMT, R-T	CP(15), UC(4)	118.9	19	102.8	8	156
112502	F	11	3	RMT	CP(4), SP(4) ,UC(6)	155	14	122.7	56	171

- Localization of seizures: ABC; A (R: right, L: left), B (-: none, B: basal, L: lateral, M: mesial), C (F: frontal, T: temporal, C: central). E.g. RMT (right mesial temporal lobe), L-F (left frontal lobe), RBF (right basal frontal lobe).
- Seizure type: type of the clinical seizures; CP: Complex Partial, SP: Simple Partial, SG: Secondarily Generalized, UC: Unclassified. The numbers given in parentheses represent the number of seizures for each type.
- Seizure duration: mean, minimum, and maximum values of seizure durations, considering electrographic onsets and offsets of the seizures.

DADOS PARA LSTM E CNN

G1 PL2 2018

```
if neuralNetwork == "LSMT"
        XTrain = num2cell(newFeatVectSel,1);
        YTrain = categorical(newTrg3);
        YTrain = YTrain';
```

newFeatVecSel 29x42934, por exemplo

newTrg3 1x42943, ex.

newTrg 4x42943 ex.

G5PL2 2018

```
aux = treinoTrg(:,1);
         trg = categorical(aux);

         trainCell = {};
         for i=1:length(treinoFeat)
             trainCell{i,1} = treinoFeat(:,i);
         end


        net = trainNetwork(trainCell,trg,net,options);
```

```
G6 PL2 2018

    yTrainLSTM = vec2ind(T);    % T= 1000'…
    yTrainLSTM = categorical(yTrainLSTM');
    xTrainLSTM = num2cell(FeatVectSel, 1)';
```

PL2 G14 2018

```
v1=[1;0;0;0];
v2=[0;1;0;0];
v3=[0;0;1;0];
v4=[0;0;0;1];

if tipoRede==3 %LSTM
    inputSize=29;
    features=num2cell(features,1);
    % target=num2cell(target,1);
    targetStr=[];

for i=1:length(target)
        if target(:,i)==v1
            targetStr=[targetStr "interictal"];
        elseif target(:,i)==v2
            targetStr=[targetStr "pre-ictal"];
```

```
        elseif target(:,i)==v3
            targetStr=[targetStr "ictal"];
        elseif target(:,i)==v4
            targetStr=[targetStr "pos-ictal"];
        end
    end

    targetStr=categorical(targetStr);
    targetStr=transpose(targetStr);

… e depois
net = trainNetwork(features,targetStr,layers,options);
```

PL3 G6

```
function [Pn, Tn] = data_for_lstm(P, aux)
    size_P = size(P);
    Pn = mat2cell(P, ones(1, size_P(1)), 29);
    Pn = cellfun(@transpose,Pn,'UniformOutput',false);
    Tn = categorical(aux);
end
```

PL3G6

LSTM

```
function [xTrain, yTrain] = transformInputsAndTarget(p,t)
    xTrain = cell([29 1]);
    for features = 1:length(p)
        xTrain{features} = p(:,features);
    end

    yTrain = categorical(t);
end
```

CNN

matrizes de dados de treinamento e teste, estão na ordem ATRIBUTOS X TAMANHO(Num. de Observações).

```
    xTrain = reshape(p, [size(p,1),1,1,size(p,2)]);
    xTest = reshape(p, [size(p,1),1,1,size(p,2)]);
```