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

MACHINE LEARNING (MEI)

NEURONAL AND FUZZY COMPUTATION (MIEB)

2018-19

Assignment nº 3 – Prediction and detection of epileptic seizures.

(Theoretical content: Chaps. 6, 6A, and 7).

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 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 side 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.

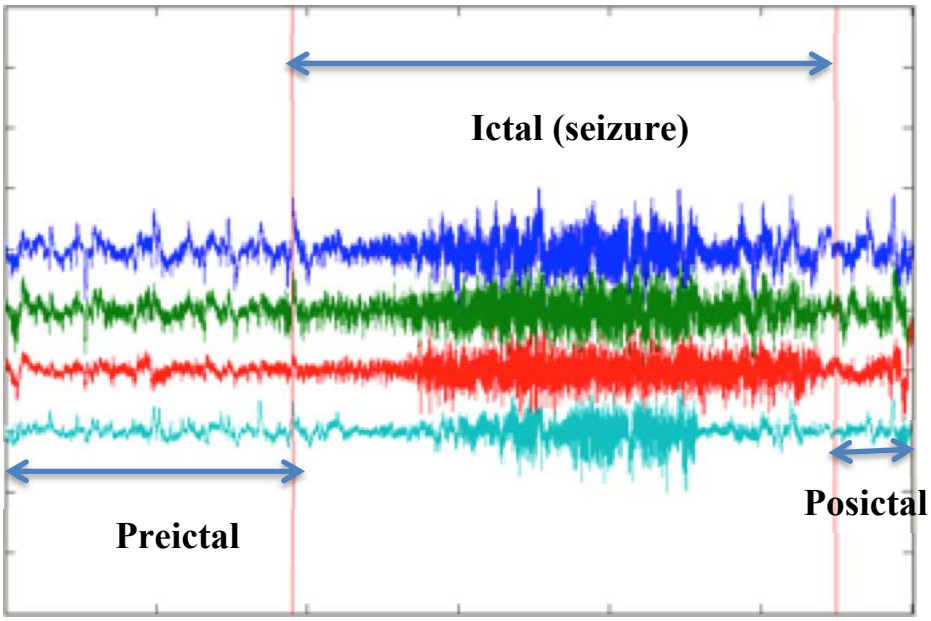

Figure 1. Electroencephalogram (EEG) with one epileptic seizure (view of 4 channels).

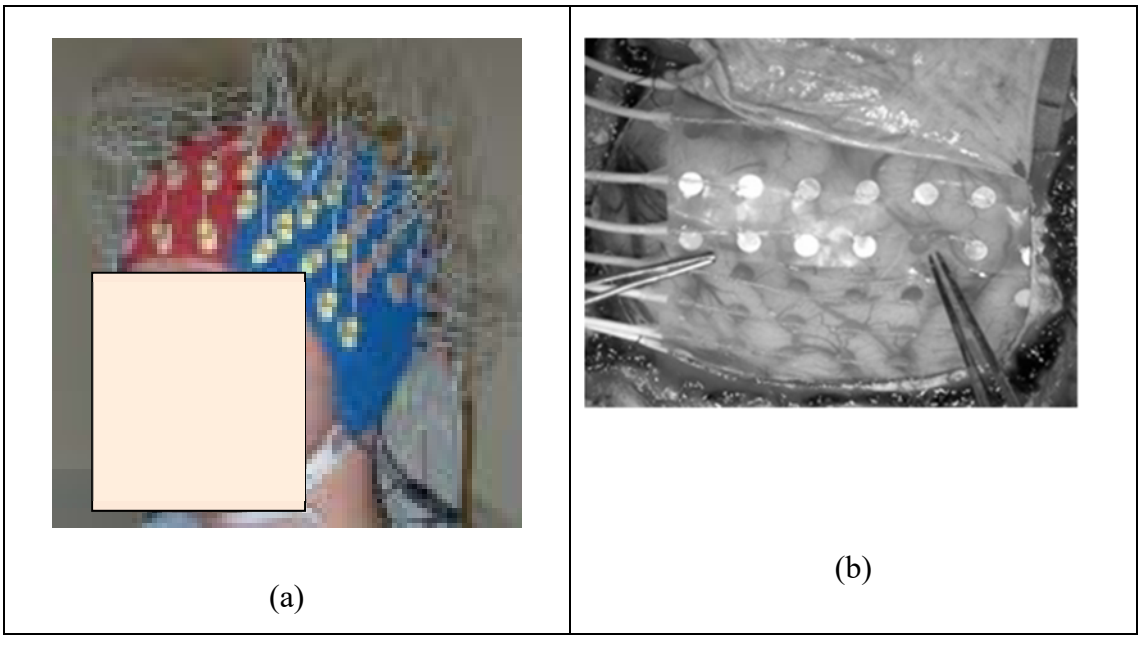

Figure 2. (a) Surface (non invasive) electrodes, (b) Intracranial (invasive) electrodes. Each electrode feeds an EEG channel relative to an electrical reference.

In this work, for the prediction and detection of seizures, the brain state will be classified into four states. A four output NN is needed, one output for each class.

1- Class **inter-ictal**, corresponding to the normal brain state (output of the classifier NN 1 0 0 0).
2- Class **Pré-Ictal**, a seizure is coming (output of the NN 0 1 0 0).
3- Class **Ictal,** a seizure is happening (output of the NN 0 0 1 0).
4- Class **Pos-Ictal,** a seizure has just finished (output of the NN 0 0 0 1).

The 10 minutes preceding a seizure will be considered the preictal state. The 5 minutes after the seizure ended will be considered the postictal state.
The big question, previous to any classification, is to extract from the EEG a good set of features capable to distinguish the 4 classes. Although more than 30 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. They refer to one EEG invasive channel placed in the epileptic focus (the place in the brain where the seizure starts).
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 four 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

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

Doctor Mojtaba Bandarabadi (we thank and acknowledge him for that) has kindly extracted features from 11 patients.

Data from two patients will be given to each group.

The following challenges are proposed:

- Build and train shallow NNs with the best success, one in predicting and another in detecting (it is very unlikely that one will be good for both, but you can try).

- Build and train a deep network (a CNN or a LSTM) for predicting the seizures

The training and testing datasets must have the usual properties, so they must include seizures. If you use validation, the validation set must also have seizures. If your patients do not have enough seizures for training, validation and testing, consider only training and testing. For example, for a patient with 9 seizures, you may use 70% (6 seizures) for training and 30% (3 seizures) for testing. 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 chose your NN, even if it is outside of those studied in the class.

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.

Some remarks:

1 – **Choosing the architecture**

In a first approach, one can use a normal feedforwatd 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. RBF NN can also be used. 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
- choice of a NN already trained that is accessible to the user
- 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 showed in the GUI.

The *guide* can be used to built the GUI or by the **App Designer->** `appdesigner`**)**. 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.

3. Building the training and testing sets.

One important issues 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 penalizations of the error of the NN in the different instants. See the toolbox User's Guide 2018b, page 6-43.

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.

4. Neural network training styles

There are two learning styles: the incremental and the batch.

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=trainc(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=trainr(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.inputweights{i,j}.learnFcn='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=trainb(net,X,P) "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,X,P) , 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.
Basically, the training methods that improve the gradient use second order information (second derivative, or the Hessian) building the Quasi-Newton methods family, of combine successive gradients in order to improve convergence, as the conjugate gradient family.

4. 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 read the User's Guide Chap 9, pages 9-2 to 9-10 of type > help train.

5- The supplied datasets have the following variables inside:

- FeatVectSel: Featuress 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 classes.
- 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: 600 points before the first 1 for each seizure, corresponding to 600 seconds=10 minutes. Similarly, the postictal points are those 300 after the last 1 of each seizure (5 minutes).
- The Trg vector must be changed to identity all the four classes. For example 1 for interictal, 2 for preictal, 3 for ictal, 4 for postictal.
- After that, the target matrix T defined in the classes must be built in accordance to the changed Trg. Using the suggested values, the points 1 in Trg correspond to [1 0 0 0]' in T, the points 2 in Trg to [0 1 0 0]' in T, the 3 to [0 0 1 0] in T, and the 4 to [0 0 0 1]' in T.

6- Postprocessing:

In an initial approach, one classifies point by point, i.e., each point must be assigned by the classifier to one of the four 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.

If one group wants to test in more than two patients, just ask for more data.

7. 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.

8. 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 (squared) 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. 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, pos-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 four columns.

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

9. 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;`

9. 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
107702	F	29	10	RMT, RLT	CP(1), SP(7), UC(1)	183	9	82.3	13	172
109602	F	32	1	LMT	CP(8), UC(1)	162.6	9	121.9	74	157
112502	F	11	3	RMT	CP(4), SP(4) ,UC(6)	155	14	122.7	56	171
115002	F	32	8	RBF, RMT, LMT	CP(2), SP(2), UC(5)	151.6	9	122.5	38	210
132502	F	18	6	L-T, L-F	CP(13)	127.8	13	86.5	68	105
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

59002	M	18	11	LBT, LLT, RBT	CP(4), SP(4), SG(4), UC(1)	246.2	13	100.2	36	137
63502	F	63	30	LMT, R-T	CP(15), UC(4)	118.9	19	102.8	8	156
92202	M	39	8	L-F, L-C	CP(2), SP(3), UC(25)	110.6	30	15.4	6	43
95802	F	14	13	L-T, LLT	CP(6), SG(4), UC(4)	217.1	14	52.5	7	123
Sum	7F/4M	-	-	-	CP(74), SP(45), SG(14), UC(50)	1785.4	183	-	-	-
Mean	-	26.7	8.7	-	-	162.3	16.6	91.8	32.5	158.8

- 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.

Wishing you a nice work. 15/10/2018.
ADC