Seizure prediction using FFT, eigen values of correlation matrix, artificial neural networks, k-nearest-neighbors and Bayesian model combination

Francisco Zamora-Martínez, Francisco J. Muñoz-Almaraz,
Paloma Botella-Rocamora, Juan Pardo

November 2014

Location: Universidad CEU Cardenal Herrera, Alfara del Patriarca, Valencia, Spain. Embedded Systems and Artificial Intelligence (ESAI) research group.

Email: {francisco.zamora, malmaraz, pbotella, juaparal}@uch.ceu.es

Competition: American Epilepsy Society Seizure Prediction Challenge.

1 Summary

This report presents the solution proposed by ESAI-CEU-UCH team, composed by lecturers from Universidad CEU Cardenal Herrera, at Kaggle American Epilepsy Society Seizure Prediction Challenge [7]. The proposed solution positioned us at the **4th** place at Kaggle leaderboard. Different kind of input features (different preprocessing pipelines) and different statistical models are being proposed. This diversity was motivated to allow model combination to improve the result.

It is important to note that none of the proposed systems use the test set for any calibration or related purpose. The competition rules allow to do a model calibration using test set, but doing it will reduce the reproducibility of the results in a real world implementation.

2 Feature selection and extraction

Several preprocessing pipelines have been performed. The most relevant preprocessing technique according to final performance were Fast Fourier Transform (FFT) plus Principal Components Analysis (PCA) procedure. This different pipelines combine methods implemented in R [9] and others in APRIL-ANN toolkit [10].

2.1 FFT features plus PCA/ICA

This is the most important kind of proposed features, obtaining the best standalone result. This process is similar to [6], and has been implemented into APRIL-ANN toolkit. For each input file it follows the next steps:

1. Extract windows of 60 seconds size with 50% overlapping for every channel in the data, increasing the number of samples in the data set. We obtained 19 windows for every input file. Every sample is filtered by using a Hamming window.

2. For every window, real FFT is computed using an algorithm which needs power of two window sizes. The 60 seconds input window has 24 000 samples for Dogs and 300 000 samples for Patients, leading into 16 384 FFT bins for Dogs and 262 144 FFT for Patients.

3. A filter bank is computed for bands: delta (0.1Hz — 4Hz), theta (4Hz — 8Hz), alpha (8Hz — 12Hz), beta (12Hz — 30Hz), low-gamma (30Hz — 70Hz), high-gamma (70Hz — 180Hz). The mean in the corresponding frequency bands has been computed.

4. The output of the filter bank is compressed by computing the logarithm of the values.

For every file, this FFT preprocess computes a matrix with 19 rows (windows of 60 seconds with 50% overlapping) and $6 \cdot C$ columns being C the number of channels. This transformation has been applied to all the available data (training and test sets). It has been implemented in APRIL-ANN toolkit by using functions `matlab.read` and `matrix.ext.real_fftwh`.

A PCA transformation has been computed over the concatenation by rows of all training data matrices (ignoring test data). Data have been centered and scaled before PCA. After computing centers, scales and PCA matrix transformation, have been applied to all the available data (training and test). In a similar fashion to PCA, Independent Component Analysis (ICA) has been performed. The advantage of PCA/ICA transformation is their ability to remove linear correlations between features.[1] PCA and ICA transformations have been computed using `prcomp` and `fastICA` of R. The transformations have been applied to all the data using APRIL-ANN.

2.2 Eigen values of pairwise channels correlation matrix

This preprocessing was based on previous competition winner report [4] presented by Michael Hills.

2.2.1 Windowed correlation matrix

Correlations between the channels with windows covering 60s were calculated and the eigenvalues of the correlation matrix were taken as features for training with K-Nearest-Neighbors (KNN) and Artificial Neural Network (ANN) models (see Section 3). Every 10 minutes segment provided in the competition data set generates 19 signal subintervals whose lapse is 60s overlapping 30s with the previous and the posterior subintervals. Each subinterval corresponds to a data set that consists in a matrix whose size is the number of channels times the product of the sampling frequency and 60. For every subinterval (window covering 60s) the eigenvalues of its correlation matrix were calculated and stored in a new file. With the former procedure, the number of covariables is $19 \cdot C$ being C the number of channels in the subject. Therefore, the number of features is different depending on the subject. These features jointly with FFT (with and without PCA/ICA transformation) were used as inputs, and denoted as FFT+CORW, PCA+CORW, ICA+CORW.[2] R has been used to implement this step using functions `cor` and `eig`.

Correlations were also calculated with the FFT transformation of every sequence described in section 2.1, but the results were not stable enough.

2.2.2 Global correlation matrix

A whole segment file provided in the competition was analysed to find correlations between channels. The set of features obtained with this analysis was used with KNN models for training. Not only was considered the original signal in the sequence, but also a differentiated signal consisting in the difference of two consecutive values of the original signal. The original signal is a matrix whose size is the number of channels times the product of the frequency sampling and 600 (60 seconds times 10 minutes). The

[1] ICA transformation needs a random number generator and the seed was random during competition, so it is not possible to reproduce exactly the competition result.

[2] PCA/ICA transformation is only applied to FFT features, not to CORW features.

functions `eigen` and `cor` in R are applied to this matrix and the output are the eigenvalues of the correlation matrix. The number of eigenvalues is equal to the number of channels for each subject.

For each channel, a transformed signal is obtained with the difference between two consecutive values of the signal. The correlations of these differentiated signals form new features that added to the correlation of the original signals increase the predictive performance with KNN models. The differentiated signals of a segment is a matrix whose size is the number of channels times the product of the frequency sampling and 599 (60 seconds times 10 minutes minus one second needed to compute differences). Applying `eigen` and `cor` functions, the eigenvalues of the correlation matrix of these differentiated signals were calculated.

With these procedures, the number of features doubles the number of channels, and they were stored in a new file. This features will be denoted as CORG from now on.

The main flaw observed is that for human patients the differentiated signal does not provide useful information due to patient data have a higher sampling frequency. We suppose that the difference of two values in a human signal, separated by a lapse of time similar to the dog signals (1/400s), could improve the quality of these features. Nevertheless, being a minor feature, in the final track of the competition we payed more attention to the most prominent preprocessing.

2.3 Global statistical features

Exploratory analysis of original data showed different behavior in the signal variability of both classes, pre-ictal and inter-ictal. Standard deviation were calculated for every data set in every channel. Following the study of discriminant statistical features, a fourier basis representation (with 15 elements) was calculated for every data set. Exploratory analysis on their coefficients showed different behavior between pre-ictal and inter-ictal series. Pre-ictal signals coefficients displayed lower coefficients (closer to 0) while inter-ictal signals coefficients presented greater disparity between them. As a summary measure of this behavior, standard deviation of these coefficients was calculated for each data serie, once per channel, and the mean standard deviation was calculated for all channels to reduce the dimensionality and avoid redundant information. This feature will be denoted as COVRED from now on. It has been implemented in R by using the functions `sd`, `mean` and `fdata2fd`,

3 Modelling techniques and training

3.1 Cross-validation algorithm

A cross-validation approach has been followed to compute Area Under Curve (AUC) estimate using the training data. All models have been trained independently, but using the same hyper-parameters and configurations.

The number of folds per subject is the maximum number of full sequences[3] in pre-ictal data. Folds have been build in such a way that all segments in the same sequence belong to the same fold. Both, pre-ictal and inter-ictal data have been splitted into the same number of folds, and following the same sequence constraint. For KNNs, cross-validation allows to compute an estimate of the AUC, but the final model contains all the available training data. For ANNs, one model has been trained for each possible partition, and the test result is the mean of all trained model outputs.

This cross-validation algorithm has been implemented carefully for APRIL-ANN in file `common.lua`, function `common.train_with_crossvalidation`, whose most important arguments are the trainining and classify functions.

[3] A full sequence are those were all sequence numbers are available. Note that not all the given sequences in training data are full.

3.2 Models

Different models have been tested in APRIL-ANN toolkit. Logistic regression as initial baseline following a simple linear regression approach, which has not been introduced into the final system due to convergence problems. In order to exploit non-linear relations in data, KNNs and dropout ANNs with different number of layers and Rectified Linear Units [1] (ReLUs) as neurons have been trained. And finally, an ensemble of several models has been performed.

3.2.1 General aspects

Two different model specifications are possible, depending on the kind of input features: global features model and windowed features model. The case of global features model is the most simple, correspond to models whose inputs are CORG and COVRED features (see Sections 2.2.2 and 2.3). The model receives as input all the features extracted from one segment file, and computes how likely this file is a pre-ictal sample. In case of windowed features model, the model receives a file segment splitted following a sliding window over temporal axis, and computes how likely every window is a precital sample. Once all window probabilities have been computed, pre-ictal probability of the file segment is computed as the complementary of the geometric mean of inter-ictal probabilities, using APRIL-ANN code `( 1 - stats.gmean(1 - p) )`, following this equation:

$$p(\text{pre-ictal}|x) = 1 - \sqrt[n]{\prod_{t=1}^{n}(1 - p_t)} \tag{1}$$

where x is a file segment, n is the number of time windows in x and p_t is the pre-ictal probability of window t. Different aggregation methods, as arithmetic mean, maximum or harmonic mean, were tested achieving similar or worst results.

3.2.2 K-Nearest Neighbors

KNN models were chosen after convergence problems with logistic regression models. KNNs are non-parametric models and don't need to be trained because training data are the model itself. The KNN model has a hyper-parameter that is the number of neighbors, denoted by K. This hyper-parameter has been set to $K = 40$ after trying different values and looking into cross-validation and public AUC scores. The K neighbors given by the KNN were transformed into probabilities following the implementation given in APRIL-ANN toolkit (function `knn.kdtree.posteriorKNN`), which is based on [3], and basically computes a posterior probability by normalizing the exponential of the negative distances, following this equation:

$$p(\text{pre-ictal}|s) = \frac{\sum_{s' \in \text{pre-ictal}(\mathcal{K})} exp(-||s - s'||_2^2)}{\sum_{s' \in \mathcal{K}} exp(-||s - s'||_2^2)} \tag{2}$$

where s is an input sample[4], $\mathcal{K}$ is the set of K-neighbors given by the KNN, pre-ictal($\mathcal{K}$) is a subset of $\mathcal{K}$ containing only samples belonging to pre-ictal class. The adequate value of K and the computation of probabilities using distances reduce the impact of overfitting in the KNN model.

3.2.3 Artificial Neural Networks

ANN models with different hyper-parameters were tested, but in a non exhaustive way because of the large gap between cross-validation AUC and public test AUC. Overfitting effect was reduced by introducing a

[4]It can be a global feature vector computed over the whole segment file, or a temporal window taken from the file.

50% dropout [5] in all hidden layers and a Gaussian additive noise with mean 0 and variance 0.04 in the inputs. All proposed ANNs have ReLU [1] as activation functions[5] in all hidden layers, and logistic activation function at the output layer. Training was performed by stochastic gradient descent with mini-batches (bunch size) of 128 samples and minimizing the cross-entropy loss function. Validation loss was also cross-entropy but after aggregating all the probabilities computed for each segment file following the Equation (1). The input of the ANN was extended taking into account three windows (the current, the previous one and the following), allowing to capture temporal patterns in data. The learning rate was set to $\eta_0 = 0.2$ with a decay parameter of $\epsilon = 0.001$, momentum was set to $\gamma = 0.1$ and no other regularization term has been used. ANN weights were initialized sampling from a uniform distribution in range $[-3/(\text{fan-in} + \text{fan-out}), 3/(\text{fan-in} + \text{fan-out})]$. In APRIL-ANN one weight w at iteration $t + 1$ is computed following:

$$w^{(t+1)} = w^{(t)} - \frac{\eta_0}{1 + \epsilon t} \cdot \frac{\partial L}{\partial w^{(t)}} + \gamma(w^{(t)} - w^{(t-1)}) \tag{3}$$

being L the cross-entropy loss of the mini-batch.

Different hyper-parameters, including L1 and L2 regularization were tested without good results. Again, it is important to take into account that all these hyper-parameters were selected in a non-exhaustive way.

3.2.4 Ensemble of models

All the proposed models have been trained independently, computing cross-validation probabilities for every training file, and test output submission file, and the final system was a combination of these outputs. Initially, a simple uniform linear combination of probability outputs in submission files has been tested. In order to improve such result, a Bayesian Model Combination (BMC) algorithm [8] has been implemented in Lua for APRIL-ANN [10] toolkit. BMC algorithm uses cross-validation output probabilities to optimize the likelihood of the combination, and uses these weights to combine output probabilities in submission files.

3.3 Results

The best submitted model is the BMC ensemble of the described models above. It achieved a **0.7935** AUC in private test.

4 Dependencies and solution recipe

This system uses the following open source software:

- APRIL-ANN toolkit v0.4.0 [10]. It is a toolkit for pattern recognition with Lua scripting and a C/C++ core. Because this tool is very new, the installation and configuration has been written in the pipeline.
- R project v3.0.2 [9]. For statistical computing, a wide spread tool in Kaggle competitions. Packages R.matlab, MASS, fda.usc, fastICA, stringr and plyr are necessary to run the system.
- GNU BASH v4.3.11 [2] with standard command line tools.

The system is prepared to run in a Linux system with Ubuntu 14.04 LTS, but it can be run in other Debian based distributions, but not tested.

It is possible to run the system (training and test) by executing:

[5] ReLU follows the expression $r(x) = max(0, x)$.

```
$ ./train.sh
```

And it is possible to run only test by using:

```
$ ./test.sh
```

See the README file at GitHub repository[6] for deeper information about the solution recipe.

5 Code description

The code is a bunch of different scripts for APRIL-ANN and R, plus bash-scripts for the recipe automation. The folder `scripts/` contains the different programs, which have been organized depending in their purpose.

The following scripts are for configuration and common functions:

- `scripts/configure.sh`: is a bash-script which downloads APRIL-ANN, install dependencies, and compiles it using ATLAS library by default.
- `scripts/configure.R`: is an R script which install the needed R packages at the user home folder.
- `scripts/cmd.lua`: contains Lua code for APRIL-ANN with command line options of the training scripts.
- `scripts/common.lua`: contains Lua code for APRIL-ANN with several functions which were shared between different preprocessing and training programs.

The following ones are for preprocessing:

- `scripts/PREPROCESS/compute_fft.lua`: is a Lua script for APRIL-ANN which computes the FFT preprocess, before PCA/ICA transformation, indicated at Section 2.1.
- `scripts/PREPROCESS/compute_pca.R`: is an R script which computes PCA transformation matrices: rotation, center and scale matrices.
- `scripts/PREPROCESS/compute_ica.R`: is an R script which computes ICA transformation matrices: rotation, center and scale matrices.
- `scripts/PREPROCESS/apply_pca.lua`: is a Lua script for APRIL-ANN which takes the PCA transformation matrices and computes PCA rotation of all data.
- `scripts/PREPROCESS/apply_ica.lua`: is a Lua script for APRIL-ANN which takes the ICA transformation matrices and computes ICA of all data.
- `scripts/PREPROCESS/correlation_60s_30s.R`: is a R script which computes windowed correlation matrix as described at Section 2.2.1.
- `scripts/PREPROCESS/compute_corg.R`: is a R script which computes global correlation matrix as described at Section 2.2.2.
- `scripts/PREPROCESS/compute_covarred.R`: is a R script which computes global statistical features as described at Section 2.3.

[6] `https://github.com/ESAI-CEU-UCH/kaggle-epilepsy`

Note that this preprocessing scripts produces intermediary output results in disk storage.

The following are for training+testing:

- `scripts/MODELS/train_one_subject_knn.lua`: is a Lua script for APRIL-ANN which computes cross-validation results for KNN model and produces the test outputs. This script has a lot of configuration options, the most promising were selected during competition and fixed in the solution at `scripts/MODELS/confs/knn*.lua`.
- `scripts/MODELS/train_one_subject_mlp.lua`: is a Lua script for APRIL-ANN which trains a MLP following the cross-validation scheme and produces its outputs. This script has a lot of configuration options, the most promising were selected during competition and fixed in the solution at `scripts/MODELS/confs/ann*.lua`.
- `scripts/MODELS/train_all_subjects_wrapper.lua`: is a Lua script for APRIL-ANN which takes one of the two above scripts and trains the model for all the available subjects, producing the submission for Kaggle.

The following are for models ensemble:

- `scripts/ENSEMBLE/naive_ensemble.lua`: is a Lua script for APRIL-ANN which implements an uniform linear combination of several test outputs.
- `scripts/ENSEMBLE/bmc_ensemble.lua`: is a Lua script for APRIL-ANN which implements BMC ensemble for models trained following the cross-validation scheme, and produces the corresponding test output.

The following are only for testing:

- `scripts/MODELS/test_one_subject_knn.lua`: is a Lua script for APRIL-ANN which computes test results for a KNN model and one subject.
- `scripts/MODELS/test_one_subject_mlp.lua`: is a Lua script for APRIL-ANN which computes test results for a MLP model and one subject.

6 Additional comments and observations

Among the indicated as submitted best system, different models and features combinations have been tested. Table 1 summarizes the cross-validation AUC and public test AUC for the most important combinations. In Table 1 first column contains the model, being one of the following:

- LR: Logistic regression.
- KNN: K-Nearest-Neighbors with $K = 40$.
- ANN: Artificial Neural Networks with 50% dropout and ReLU [1] neurons, 128 units in every hidden layer, and a logistic activation function at the output. ANN2 for two hidden layers, ANN3 for three hidden layers, and so on. ANN2p for two hidden layers with 64 units in every layer,
- UNIFORM or BMC: ensemble of the most promising systems, indicated in bold face. UNIFORM is a linear combination with uniformly distributed weights; BMC is linear combination where weights are estimated following BMC optimizing cross-validation likelihood.

The second column indicates which features were used as input of the model, being one of the following list:

Model	Features	CV AUC	Pub. AUC
LR	FFT	0.9337	0.6784
KNN	FFT	0.8008	0.6759
KNN	FFT+CORW	0.7994	0.7040
KNN	**PCA+CORW**	**0.8104**	**0.7288**
KNN	**ICA+CORW**	**0.8103**	**0.6840**
ANN2	FFT+CORW	0.9072	0.7489
ANN2	PCA+CORW	0.9082	0.7815
ANN2p	**PCA+CORW**	**0.9175**	**0.7895**
ANN2	**ICA+CORW**	**0.9104**	**0.7772**
ANN3	PCA+CORW	0.9188	0.7690
ANN4	PCA+CORW	0.9268	0.7772
ANN5	**PCA+CORW**	**0.9283**	**0.7937**
ANN6	PCA+CORW	0.9291	0.7722
KNN	**CORG**	**0.7097**	**0.6552**
KNN	**COVRED**	**0.6900**	**0.6901**
UNIFORM	ENSEMBLE	—	0.8048
BMC	ENSEMBLE	0.9271	0.8249

Table 1: Cross-validation and public AUC results for the most important systems tested during the competition.

- FFT: the output of the proposed filter bank plus logarithm compression.
- FFT+CORW: the previous one plus windowed eigen values of correlation matrix.
- PCA+CORW: the PCA transformation of FFT features plus windowed eigen values of correlation matrix.
- ICA+CORW: idem as previos one, but using ICA instead of PCA.
- CORG: correlation using 10 minutes window.
- COVRED: different statistical measures over the original signal.
- ENSEMBLE: output probabilities of the bold faced systems.

Table 1 shows that logistic regression model was the worst in terms of generalization ability. The application of PCA allows to improve the public AUC results by **0.025** for KNN model and **0.033** for ANN model. ICA obtains improvements similar to PCA. The advantage of deep ANNs (with more than two hidden layers) is not clear, but the best public AUC was obtained by an ANN with five hidden layers, improving by **0.004** the public AUC of the best ANN2. Global features as CORG and COVRED show strong correlation between cross-validation AUC and public test AUC. Finally, the use of BMC ensemble method allow to improve the public AUC in **0.031** points compared to the ANN5 model.

Finally, the private AUC for the two selected models is shown at Table 2. Note that this table shows private AUC for competition system, and private AUC for post-competition system using MKL or ATLAS library. As it is indicated in the README at GitHub repository, the competition system couldn't be reproduce in an exact way due to a non fixed seed number. Now, the code has this number fixed, and its AUC is shown in the table. In addition to this problem, our system uses Intel MKL library for the best performance in experimentation. However, the code is configured by default to compile using ATLAS

Model	FEATURES	Priv. AUC MKL	Post. priv. AUC MKL	ATLAS
ANN5	PCA+CORW	0.7644	0.7644	0.7405
BMC	ENSEMBLE	**0.7935**	0.7983	0.7910

Table 2: Private AUC for the two selected submissions. Post. priv. AUC refers to AUC results in private test after the competition.

library, which is open source and standard in Linux systems. The post-competition private AUC for MKL and ATLAS options are also shown in the table for comparison purposes.[7]

The private AUC of the BMC ensemble is the second best if we compare it to all our submissions. However, the difference with the best private AUC is not significant, the best one has a private AUC of **0.7958**.

The experimentation has been performed in a cluster of **three** computers with same hardware configuration:

- Dell PowerEdge 210 II server rack.
- Intel Xeon E3-1220 v2 at 3.10GHz with 16GB RAM (4 cores).
- 2.6TB of NFS storage.

7 Simple features and methods

The best single features plus model was ANN5, an artificial neural network with 5 hidden layers using ReLUs [1] as activation function, with 128 neurons in every layer and logistic activation function as output. The input features were FFT+CORW as described above. This system achieves a **0.7644** AUC in the private test.

Post-competition software revision: v1.0

A version 1.0 of the software has been prepared, just solving bugs which have been introduced during competition. This new version removes from the ensemble combination the ANN models with PCA features which lead to a worst public AUC score. So, compiled with Intel MKL option, the BMC ensemble of ANN2 with ICA features and KNNs with ICA, PCA, CORG and COVRED features achieves a public AUC of **0.8222** and a private AUC of **0.7947**. The models selection for the ensemble has been performed maximizing the public AUC.

References

[1] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey J. Gordon and David B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011.

[2] GNU. Bash: Bourne again shell, 2007. `http://www.gnu.org/software/bash`.

[7] Note that the system hyper-parameters have been optimized using MKL compilation flag, so it is expected worst results compiling with ATLAS library.

[3] Jacob Goldberger, Sam Roweis, Geoffrey Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems (NIPS)*, pages 513–520. MIT Press, 2005.

[4] Michael Hills. Seizure detection using FFT, temporal and spectral correlation coefficients, eigenvalues and Random Forest, 2014. `https://github.com/MichaelHills/seizure-detection`.

[5] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[6] J. Jeffry Howbert, Edward E. Patterson, S. Matt Stead, Ben Brinkmann, Vincent Vasoli, Daniel Crepeau, Charles H. Vite, Beverly Sturges, Vanessa Ruedebusch, Jaideep Mavoori, Kent Leyde, W. Douglas Sheffield, Brian Litt, and Gregory A. Worrell. Forecasting seizures in dogs with naturally occurring epilepsy. *PLoS ONE*, 9(1):e81920, 01 2014.

[7] Kaggle. American epilepsy society seizure prediction challenge, 2014. `https://www.kaggle.com/c/seizure-prediction`.

[8] K. Monteith, J.L. Carroll, K. Seppi, and T. Martinez. Turning bayesian model averaging into bayesian model combination. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2657–2663, July 2011.

[9] R Development Core Team. R: A language and evironment for statistical computing, 2005. `http://www.R-project.org`.

[10] Francisco Zamora-Martínez, Salvador España-Boquera, Jorge Gorbe-Moya, Joan Pastor-Pellicer, and Adrián Palacios-Corella. APRIL-ANN toolkit, A Pattern Recognizer In Lua with Artificial Neural Networks, 2014. `https://github.com/pakozm/april-ann`.