A. Maragakis and A. Rosengarten. (2015) "Seizure Detection with Single-Channel On-Scalp EEG using SVM-Ensembles," International Mathematics Research Notices, Vol. 2015, Article ID rnn000, 8 pages.
doi:10.1093/imrn/rnn000

Seizure Detection with Single-Channel On-Scalp EEG using SVM-Ensembles

Alexandros Maragakis[1] **and Alexander Rosengarten**[2]

[1]Department of Computer Science and Engineering
University of California, San Diego
9500 Gilman Drive, Mail Code 0404
La Jolla, CA 92093-0404
USA

Correspondence to be sent to: amaragak@ucsd.edu, arosenga@ucsd.edu

This paper presents a mechanism by which seizures are detected using only a single channel of on-scalp EEG. The system has been evaluated through the use of the CHB-MIT on-scalp EEG dataset using only the *FT10-T8* channel. From the onset of the algorithm's development, we experimented with multiple classification primitives and SVM ensemble parameters. Best results were achieved using the method of assigning segments to individual SVMs and having them cast a majority vote. Despite only have a spatial dimension of size 1, and when training on three segments our algorithm labels seizure windows of a patient with temporal lobe epilepsy with an accuracy of over 90%.

1 Introduction

Despite considerable advancement within the discipline of machine learning, the prediction of epileptic seizures is still viewed as an insurmountable task; this is particularly so the case when only on-scalp data is available. This is problematic as, if there were to ever be such a prediction system, it would likely only be of wide public use if it were applicable through a non-invasive procedure; i.e. it would *not* be intracranial.

In an attempt to make the realisation of on-scalp seizure prediction more feasible, our team has created a comfortable in-ear EEG system that can be worn by out-patients on a day-to-day basis that, upon the *detection* of a seizure, will send 20 minutes of pre-seizure EEG to a database for further analysis. Our rationale is that successful, automated prediction of seizures may require patient-specific algorithms and hence a database of patient-specific EEG would be required prior to prediction algorithm development.

2 Housekeeping

For the sake of succinctness within this paper, we shall outline some assumptions that should be made throughout the remainder of the paper.

The *CHB-MIT Scalp EEG Database* and the *Universitat Bonn EEG Data Set* shall be abbreviated to the *CHB-MIT* and *Bonn* data sets respectively. When referring to the *CHB-MIT Scalp EEG Database*, we refer specifically to patient 1.

The terms *Support Vector Machine* and *Bootstrap Aggregating* shall henceforth be mostly referred to *SVM* and *bagging*. *Seizure* and *Non-seizure* labels shall be referred to as S and N labels. Finally, a *classifier label* is a label that is produced by the classifier after testing, whereas a *non-classifier label* is a label that is specified by the data set itself.

Received 11 June 2015; Revised 12 June 2015; Accepted 13 June 2015
Communicated by R. Kastner

The *CHB-MIT* is processed window-by-window due to its large segment length and multi-labelling per segment whilst the *Bonn* data set is processed segment-by-segment for the opposite reasons.

The global true *(S/N)* rates are calculated by dividing the number of correct classifier *(S/N)* labels by the total number of classifier *(S/N)* labels. The per-seizure true S rate is calculated by dividing the number of correct classifier S labels by the number of non-classifier S labels. The per-seizure false N is defined as the number of incorrectly placed N labels by the number of non-classifier S labels. Expect rates to be displayed as [(global true S%, global true N%), per-seizure true S%,]. Per-seizure false N can be inferred from the per-seizure true S.

3 Support Vector Machines

The Support Vector Machine is a promising binary classification and regression technique created by Vapnik at Bell Laboratories. The algorithm works similarly to a single layer perceptron: both algorithms aim to find the best hyperplane to discriminate two groups. The perceptron, however, fails when two groups are not linearly discriminable. SVMs are more robust: they try to maximize the margin while minimizing error.

Usually a linear hyperplane is found to separate the two classes: $f(w,b) = sign(w \cdot x + b)$ where w is the weight vector, x is the input vector, and b is the starting point.

For separable cases, the SVM aims to minimize the following:

$$argmin_w 1/2||w||^2$$

subject to the constraint:

$$y_i(w \cdot x_i + b) \geq 1$$

For linearly non-separable cases, the minimization problem is adjusted to allow for misclassified data points. The SVM penalizes errors by introducing new variables, $\xi_{i=1}^{l}$(these are known as support vectors):

$$argmin_{(w,C,k)} 1/2||w||^2 + C(\Sigma_{i=1}^{L}\xi_i)^k$$

Subject to the constraint:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

where C and k are used to weight the error variables, ξ_i, and L is the number of training examples.

4 Module Ensembles

An ensemble of classifiers is a collection of several classifiers whose individual decisions are combined in some way to classify the test examples. It is known that combining multiple classifiers improves performance over an individual classifier, but why exactly is this the case?

Hansen et. al. explains with the following: Assume that there are n classifiers $f_1, f_2, ..., f_n$ and some test data x. If all the classifiers are identical, they will show the same performance as an individual classifier. However, if some classifiers are different and their errors are uncorrelated, then when $f_i(x)$ is wrong, the majority of classifiers $f_{\sim i}(x)$ may be right. If the error of an individual classifier, p, is even slightly better than chance or $p < 1/2$, and each classifier is independent, then the probability of error p_{total} that results from majority voting is as presented in equation 1. Thus, when n becomes sufficiently large, the total error, p_{total},becomes very small.

$$\sum_{k=\lceil n/2 \rceil}^{n} p^k (1-p)^{(n-k)} < \sum_{k=\lceil n/2 \rceil}^{n} (1/2)^k (1/2)^{(n-k)} = \sum_{k=\lceil n/2 \rceil}^{n} (1/2)^n \tag{1}$$

The method of combining the decisions of the individual SVMs that is used in this paper is majority voting.

4.1 **Bagging**

Bagging is a technique used to train classifiers in an ensemble so that they are produce uncorrelated errors. In order to train k classifiers in such a way, we need k unique training datasets. In order to guarantee that the classifiers are independent and uncorrelated, the datasets need to be as different as possible whilst remaining in accordance with the labelling schema incorporated.

Bagging builds k training datasets by randomly sampling with replacement from an original training dataset; here, by data set, we mean a feature vector as randomly sampled from a feature matrix. Each example in the newly generated training set may appear a repeated number of times whilst some examples from the original dataset may not appear at all in a given generated training set. Each of the k generated training sets are then used to train the k classifiers.

4.2 **Training Segment-SVM Pairing**

Here we present an novel alternative to this paper. Training segment-SVM pairing is the method by which each module in the SVM ensemble is assigned its own training segment. When training our detection algorithm using the *CHB-MIT* data set, each segment is an hour in length, each containing an average of about a minute of active seizure. This method

5 **Data**

5.1 **Data Set Selection**

The first step in creating a seizure detection system is to select an appropriate EEG data set. This task is made simpler upon identifying the specifications of the ideal data set; one that best mimics the EEG that would be extracted through the ear. Namely, the requirements with which our third-party EEG data would have to cohere would have to be that it is intracranial and single-channel and contains only S and N labels. Preferably, this single-channel data set is also be extracted from a location in or close to the ear.

Ultimately we selected the *CHB-MIT*, and for the following reasons: the EEG is recorded on scalp, contains both S and N labels and, although initially multi-channel, coheres to the international 10-20 standard of electrode placement and hence is reducible to a channel of arbitrary selection from a discrete set of on-scalp locations. It is important to note that not all of the subjects that were included in the *CHB-MIT* data set had temporal epilepsy, hence we further reduce the dimensionality of the data-set by using data from patients with epilepsy localised in the temporal cortex only.

Initial testing also made use of another set, namely the *Bonn* data set that took readings using electrodes placed in the closest proximity to the seizures localisation region.

5.2 **Pre-Processing Data**

The 20 or so size spatial dimension of the CHB-MIT data set is stripped down to be of a single channel; the *FT10-T8* channel. The *FT10-T8* channel gives the difference between the signals of *FT10*, Frontal-Temporal channel 10, and *T8*, Temporal channel 8. This is the closest difference channel with respect to the inner-ear and hence is best representative.

The resulting single-channel data is then sent to an 8th order low-pass filter with cut-off frequency 100Hz; a normalised frequency of 0.4 given that the sample rate of the data is 256Hz. We do this in order to quickly and tentatively remove recording noise from the stream.

EEG is typically classified in terms of its activity at specific octaves, namely the theta, delta, alpha, beta and gamma bands that are localised at 0-4Hz, 4-8Hz, 8-16Hz, 16-32Hz and 32-64Hz respectively. As to coincide with such tradition, we follow the filtering procedure by also performing the described frequency band isolation. These bands are calculated, or rather approximated, through the use of a 5-level wavelet transform.

6 Features

6.1 Feature Extraction

A very important phase in the design of a classifier is the selection of appropriate features to extract from the incoming data. The initial iteration of the algorithm made use of very primitive features indeed, namely the log of variance and energy. Each of the listed features were extracted for each of the previously specified dyadic frequency bands.

Using a basic SVM classifier, *CHB-MIT* gave a result vector of [(79.7%, 99.1%), 58.8%], again with a 5-feature reduction. This outcome is respectable, however, we ideally want to bring the global true S closer to 100%.

In order to improve our results, extra features were added. The feature list of the current implementation of the classifier is as follows:

- Log of Variance and Energy
- Log of Skewness and Kurtosis
- Raw Mobility and Complexity
- Log of Power and Frequency of maximal intensity band in power spectrum

Again, each feature is calculated for each of the dyadic frequency bands and, again, the feature-space reduction algorithm can be applied to reduce overfitting. When re-running *CHB-MIT* with the above features and a 35-feature reduction, such as to bring the space down to a size of 5 as before, a result vector of [(83.5%, 99.2%), 64.5%] is produced; an improvement from before. Further improvement is achieved by adding windows to the feature matrix stochastically; see `AddChance_N` and `AddChance_S` in 1.

6.2 Feature-Space Reduction

In order to reduce over-fitting, a feature-space reduction algorithm is applied post-feature extraction, hence is adaptive depending on the given training set. This procedure involves two steps, the first of which is to identify the features that produce the least distinction between classes when considered on a one-dimensional plane. The method by which this is done depends on the classification primitive used. In the case were SVM is used, the least distinct feature is that which has the highest number of points from each of the S and N sets that fall between the maximal and minimal values of each of the opposing set. When LDA is used, the number of points that have higher probability of belonging to the opposing classification set when assuming that each set has Gaussian distribution.

The second stage is to reduce the feature spaces of both the training and test data sets by k, the specified number of features to remove. Hence, the resulting sets are as original but have had the k least desirable features, as determined through running stage 1 of the procedure on the training set, removed.

Without feature-space reduction, the *Bonn* data set had roughly 70% correct labeling whereas with feature-space reduction we achieved 100% correct labeling.

7 Classifier

7.1 Parameters

The classifier's parameters are specified in table 1. A standard SVM classifier, as mentioned in section *6.1*, can be implemented by setting `numModules` = 1 and `segSpecific` = 1.

Parameter Name	Notes
Feature Matrix Construction	—
`windowSlide_sec`	Number of seconds by which feature extraction window slides.
`windowSize_sec`	Number of seconds encapsulated within a single feature extraction window.
`num_ft2Rmv`	Number of features to remove from feature-space
`AddChance_N`	The probability with which an N window is added to the training feature matrix.
`AddChance_S`	The probability with which an S window is added to the training feature matrix.
Ensemble Construction	—
`mode`	Enum: 0-standard 1-bagging, 2-SVM-Window pairing
`numModules`	Size of ensemble - can only be set if bagging.

Table 1: Classifier Parameters

7.2 Architecture

The CHB-MIT training data is parsed and pre-processed as specified in section *5.2*. The augmented data is passed to the feature extraction stage of the system, outputting a feature matrix F, whose columns represent features and rows represent windows of the training segments. The dimensionality of F is reduced using adaptive feature-space reduction.

Depending on the ensemble method, F is sampled in a particular way such that each SVM is trained with an augmentation of the original data set. In the case where bagging is applied as the selected ensemble assembly method, each SVM is trained with sub-feature matrices, each of size `numWindows` by `numFeatures`, that are created by randomly sampling from the rows of F `numWindows` times. If, on the other hand, segment-SVM pairing is applied, F is divided into horizontal chunks by row such that $\texttt{SVM}_i$ is trained on the feature matrix produced by $\texttt{training\_segment}_i$ and are each of size `numWindows/numModules` by `numFeatures`. As is implied, to 'train on the feature matrix' is to train on both the feature and respective label matrix.

The testing feature matrix is then extracted from the test set. Unlike in training, here we do not create any sub-matrices per module and do not add windows stochastically. Instead, we maintain a constant feature matrix that is passed to all of the `numModules` SVMs and that adds windows with 100% certainty as to allow all test windows to be classified.

7.3 Results

The parameters listed in table 1 are manipulated such as to demonstrate the effect of altering said parameters on the output. Results are presented with the corresponding parameters of the classifier that produced them. Parameters are specified as [`num_ft2Rmv`, (`AddChance_N`, `AddChance_S`), `mode`, `numModules`] and results are specified as usual. For the all iterations, `windowSlide_sec`, `windowSize_sec` are both set to 4.

Parameters	Results
[5, (1,1), 0, 1]	[(84.9%, 99.2%), 62.6%]
[20, (1,1), 0, 1]	[(80.0%, 99.2%), 61.4%]
[35, (1,1), 0, 1]	[(83.5%, 99.2%), 64.5%]

Table 2: Results - Standard SVM classification with varying reductions in feature space.

Table 2 shows the effect of varying the degree of feature space reduction in a standard SVM classifier. Interstingly, as was the case with several runs, the results did not improve nor degrade in a monotonic fashion with increasing feature space reduction.

Parameters	Results
[20, (1,1), 0, 1]	[(80.0%, 99.2%), 61.4%]
[20, (0.5,1), 0, 1]	[(90.5%, 99.1%), 59.4%]
[20, (1,0.5), 0, 1]	[(75.4%, 98.9%), 53.6%]

Table 3: Results - Standard SVM classification with varying probabilities of feature matrix expansion.

The results presented in 3 show how altering ratio of the number of S and N rows in the feature matrix affect correct classification rates. By decreasing the ratio of the number N windows to the number of S windows, we see an improvement in the global true S rate.

Parameters	Results
[20, (1,1), 1, 1]	[(82.3%, 99.1%), 60.4%]
[20, (1,1), 1, 3]	[(82.2%, 99.2%), 64.5%]
[20, (1,1), 1, 9]	[(83.1%, 99.2%), 63.4%]

Table 4: Results - Bagged SVM classification with varying numbers of modules in ensemble.

Table 4 shows that varying the number of modules in the bagged ensemble produces negligible improvement. The lack of improvement in classification accuracy despite increasing the number of modules implies that the problem lies within the feature extraction. We can infer this by considering that even though the different SVMs are being trained using varying data sets, the features that these data sets represent are not enough to correctly classify certain parts of the seizure signals. This could be due to the low spatial dimension. Although they are all voting, they will not give an improvement if they do not have the right features available.

Parameters	Results
[20, (1,1), 0, 1]	[(80.0%, 99.2%), 61.4%]
[20, (1,1), 1, 3]	[(82.2%, 99.2%), 64.5%]
[20, (1,1), 2, 3]	[(87.5%, 99.2%), 63.4%]

Table 5: Results - Standard SVM, Bagged SVM and Segment-Specific SVM

From table 6 we can see that the altering the ensembling method does have a considerable impact on the correctness of classification, with Segment-Specific SVM pairing yielding the most desirable global true S label rate.

Parameters	Results
[20, (1,1), 0, 1]	[(80.0%, 99.2%), 61.4%]
[5, (0.5,1), 2, 3]	[(95.0%, 99.0%), 59.6%]

Table 6: Results - Standard SVM (row 1) vs optimal parameters (row 2)

Here we present the optimal parameters for achieving the highest global true S label rate. That is, for every seizure label placed, 95% of them are correct.

7.4 **Result Augmentation**

Classification results, without any post-processing, are somewhat granular. By this we mean that there we sometimes may come across the placement of anomalous false S and N labels which are clearly incorrect upon inspection due to the unrealistically short time scale with which they span.

Although no result augmentation was applied in the presented results of section *7.3*, it is vital that it is applied prior to sending data to the database for later analysis. We do not want anything in our database that has been sent as a result of an incorrectly placed S label. Hence we take a very humanistic approach to augmenting our results. And no, we do not do it by hand. We write an algorithm that discards S as a human examiner would, if it appears to be anomalous, remove it. Apparent error is determined depending on whether or not the S is surrounded by a continuous stream of k S labels where k is some small, hard-coded value that specifies the minimal length of continuous S labels that can be considered a single seizure. We will not go further into the details of its implementation due to its simplicity.

An example seizure detection result set before, table 7, and after, table 8, augmentation is presented. NB, each segment should only have one seizure present, hence the presented augmentation has been successful. All of the presented post-augmentation seizure times fall within the non-classifier S labels, 9 and hence have located the seizure correctly.

Segment Number	Seizure Start Time	Seizure End Time
4	1013	1025
4	1029	1049
5	1725	1769
6	333	381
7	1869	1941
7	1945	1961

Table 7: Detected Seizures without augmentation.

Segment Number	Seizure Start Time	Seizure End Time
4	1013	1049
5	1725	1769
6	333	381
7	1869	1961

Table 8: Detected Seizures with augmentation.

Segment Number	Seizure Start Time	Seizure End Time
4	1015	1066
5	1720	1810
6	327	420
7	1862	1963

Table 9: Given seizure labels.

By comparison with our post-augmentatino results we can see that the problem with our classifier is in not necessarily in that it lags in detecting a seizure, rather that it seizes to recognise the remainder of the seizure past a certain point in its development. As mentioned before, this likely an issue with feature selection and the lack of spatial dimensionality.

8 Milestones

At the beginning of the quarter, we set out to create an epileptic seizure prediction system using only a single channel of scalp EEG. In our project specification, we laid out four main milestones for the quarter: First, create a neural network model that correctly classifies third-party training data. Second, create a communication link between our prototype in-ear device (an OpenBCI board and traditional electrodes) and a laptop. Third, perform live processing and correct classification of original data. Fourth, migrate classifier to a mobile application for data processing given model parameters trained offline on a laptop.

We also created contingency plans for the third and fourth milestones in case Murphy's Law took effect: If the third milestone became unachievable, we would adapt our system to simulate real-time data using an offline data source. If the fourth milestone was unreachable (and we had time left in the quarter), the end product would be a desktop application that uses live data from the hardware teams in-ear device and performs both classification and model updating.

We started by working with the resources we had available: we began our search for an adequate dataset we could use to simulate ear-EEG. The dataset we had available for building a predictive model came from a Kaggle competition and consisted of intracranial EEG. This was not a suitable dataset for the construction of our classifier for the following reasons: First, intracranial data is very different from scalp-EEG, in that the skull acts as a low-pass filter for the electrical signals (plus, there are more sources of artifacts, or noise, in scalp EEG). Second, the dataset did not provide us with enough examples of seizures; it consisted of ten minute examples of pre-ictal (pre-seizure) but not seizure data. This would make it very difficult to create a practical, real-time model.

We attempted to use the Bonn dataset, which consisted of single channel recordings; however it was also recorded from an intracranial source. In addition, we were getting unrealistically high accuracies for our classifier. Eventually, we found an excellent dataset for our needs CHB-MIT dataset, described above.

We worked on developing the physical infrastructure of our system in parallel to acquiring a dataset. We connected OpenBCI with our laptops (via the Lab-Streaming-Layer) and updated its firmware to guarantee proper time synchronization properties. We also investigated various technologies to perform EEG signal processing and machine learning, namely BCILab. A lot of time was spent teaching ourselves how to use such technologies. Lastly, in the initial phase of our project, we reached out to professionals in the EEG industry as well as researchers in various labs at UCSD to ask for advice in how we might accomplish our goals.

By the middle of the quarter, we made a big shift in our project: instead of attempting to build a prediction system, we pivoted our project and set out to build a seizure detection system. The change came after we received an email from an industry expert warning us that prediction is a very difficult problem and would be near impossible to accomplish without more resources and time. Seizure detection, he explained, would be a very worthwhile alternative for us to pursue. Switching to seizure detection still fit our aspirations since seizure detection is an important intermediate step that would enable us to pursue prediction later.

With this in mind, we added an additional milestone to our project: the creation of a database system to collected epilepsy data for offline analysis. By this time we had accomplished the following: We developed a full working support vector machine model that classified the Bonn Epileptologie dataset with close to 100% accuracy (its worth noting that when we applied the methods mentioned in their paper, we could only achieve 64% classification accuracy). We also created a communication link between the in-ear device and laptop. In other words, we had accomplished our first and second milestones. And by mid quarter, we started creating the database system: The SQLite API was built and the python server was under way. We had set out to finish the database system, develop a classifier that would work on more realistic data (the MIT dataset), and apply our detection system to a live subject – a friend who has epilepsy.

In the final sprint of our project, we were not able to perform a live recording. We were focused on boosting the accuracy of our SVM model by trying various ensemble strategies as well as finishing up the database. Though we had at some point enough of a working version of our software to be able to collect live data, the subject was unavailable for a recording. By the end of the quarter, however, we did accomplish a lot: we created a seizure detection system that classified seizures with 90% accuracy and were able to simulate the performance of our system with offline data, thus accomplishing the contingency plan of our third milestone. The database was also fully operational, complete with Matlab client compatible with the classifier.

9 Conclusion

We have successfully identified SVM ensemble parameters that produce the best seizure classification results and, in the process, have learnt a great deal about time-series signal classification. This, in combination with the implementatino of our database that automatically stores 20 minutes of pre-seizure data, will hopefully allow us to move on to the task of seizure prediction at a later time.

We would like to thank Ryan Kastner for running the most fulfilling class we have taken at UCSD!

10 References

Cortes, C., Vapnik, V.: Support vector network. Machine Learning. 20 (1995) 273297 397
Kim, H. C., Pang, S., Je, H. M., Kim, D., & Bang, S. Y. (2002). Support vector machine ensemble with bagging. In Pattern recognition with support vector machines (pp. 397-408). Springer Berlin Heidelberg.
Hansen et. al. [Hansen, L., Salamon, P.: Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12 (1990) 9931001 400
Gutierrez-Osuna, R. Linear discriminants analysis [PowerPoint slides]. Retrieved from http://research.cs.tamu.edu/prism/lectures/pr/pr_l10.pdf