Time-series classification & Network analysis

On motor-imagery and earthquake datasets

Satvik Saha, Rudra Mukhopadhyay, Nivedaa Dhandapani

MA4207: Machine learning and network analysis

Abstract

One of the major challenges in machine learning is the *small sample-size problem*, particularly relevant in the domain of time-series data analysis and classification. The current project aims to classify two different time-series data, carrying two distinct flavours- motor imagery using ECoG and earthquake recording. For the ECoG data classification, three independent and meaningful features are proposed (autoregressive correlation, band power of the μ range and log-variance of the CSP-filtered data). Three different classifiers (namely, linear SVM, artificial neural net and linear discriminant, respectively) are implemented on these three features. Finally, a majority vote ensemble is designed to combine these three feature-classifier combinations. Similarly, for the Earthquake data, an ensemble classifier (LD, GNB, Logistic regression, RF) is run on features picked by `tsfresh`. Networks are constructed considering different features and distance metrics to visualize and justify the clustering tendency of the data points of two different classes.

Contents

1 Overview and pre-processing

In machine learning models, the ratio between the number of features and the number of samples gathered/trials is a crucial parameter that determines the power and flexibility of that particular model. Quite a few times, particularly when dealing with time-series data, the number of extractable features surpasses the sample size by quite a degree. This might lead to extreme over-fitting of the train data and poor performance on the test data. The following figure depicts the scenario quite well.

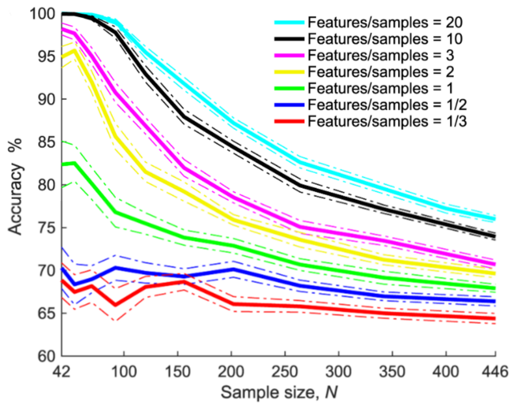


Figure 1: CV accuracy for different nfeat:nsamp (RFE-SVM) [1]

Therefore, the first step is to extract features with variables less than, or comparable to, the number of samples in the dataset. In various multivariate datasets, PCA or LDA serve this exact purpose. However, for time-series data, we oftentimes need to delve into more native features. Various time-domain features (e.g., auto-correlation, auto-regression coefficients, zero-crossings, n-th order moments), frequency domain features (power spectral density-based metrics including band-power, spectral flux), amplitude domain (maximum, minimum, mean amplitudes, peak numbers) and features of filtered data (e.g., using common spatial pattern or independent component analysis techniques) are actively used in different contexts. The next step is to find some classification model for a certain feature. In this project, we have mostly used the most mainstream ML classifiers, including linear SVM, linear discriminant, artificial neural net, random forest, Adaboost-tree ensemble, etc.

In the final step, we construct a network representation of the given data. Based on that, a few network metrics are first reported (using mean degree, average path length, degree distribution, clustering coefficients, etc.). After that, we try to cluster the unsupervised network nodes and report how well correlated these are with the actual labels.

1.1 Motor-imagery data

This ECoG data is collected from an 8×8 platinum grid placed on the subject's right motor cortex area, who was asked to imagine either the movement of the left hand's little finger or the tongue. The recording captures a 3s long time window, starting 0.5s after the cue presentation (recorded at 1000 Hz). Train and test data are collected on two different days. Following is the visualization of the same, without any pre-processing.

We propose the following pre-processing treatments.

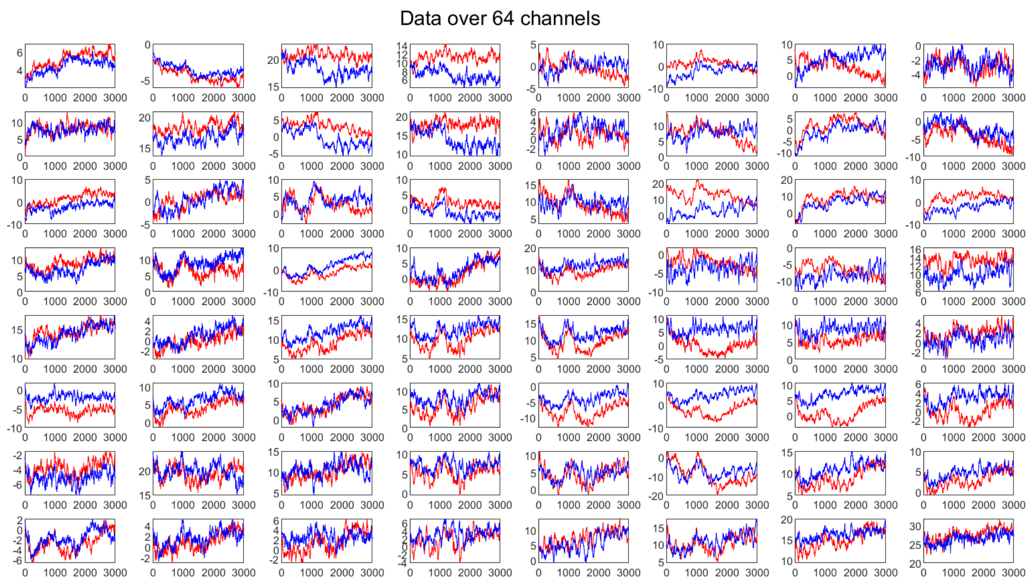


Figure 2: ECoG for 8×8 channels; without pre-processing

- **Removing linear trend:** For a given time-series $x(t) = \{x_1, \ldots, x_t\}$, let $\hat{x}$ be the best-fit regression line on $x(t)$. The de-trended data is given by $x(t) - \hat{x}$.
- **Normalization:** The data set is centred and scaled across time so that $\mu(x(t)) = 0; \sigma(x(t)) = 1$.
- **Bandpass filtering:** The signal is filtered in the frequency range 8-14 Hz (the Mu band range), as the Mu wave is particularly induced in motor execution/imagination[2].

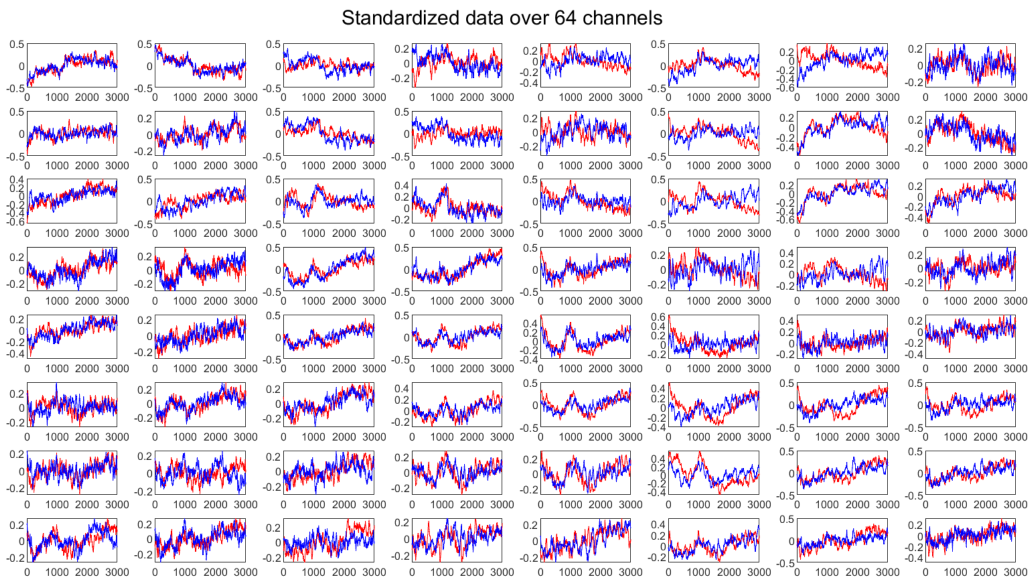


Figure 3: ECoG for 8X8 channels; standardized and de-trended

1.2 Earthquake data

The following data set is taken from Northern California Earthquake Data Center and each data is an averaged reading for one hour. Two class labels- a positive and a negative case are pre-defined.
A **positive case** occurs when a *major event* (Richter recording > 5) is not preceded by another major event for atleast 512 hours. Whereas a **negative case** occurs when at least 20 non-zero readings precede an event with Richter recording < 4 in the past 512 hours. The data set is highly unbalanced, with 264 negative and 58 positive cases in the train set.
In the following visualization, the negative case set is bootstrapped to make the outcome data balanced.

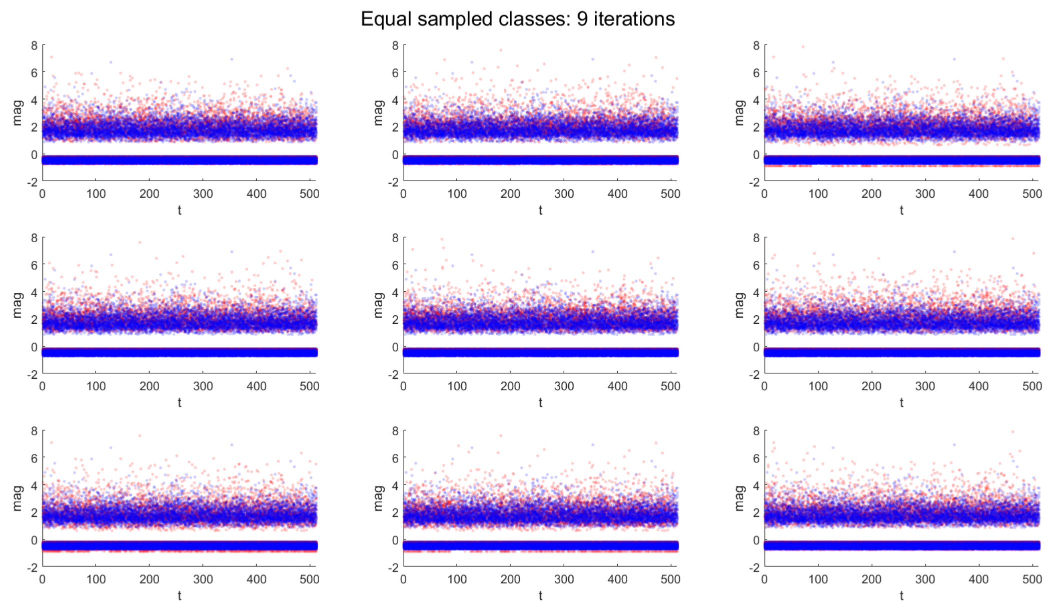


Figure 4: Earthquake data with no pre-processing, 9 bootstrap iterations

The [*]`tsfresh` python package has been used to generate 783 features. The labels from the training data have been used to test each of these and reduce this list to 88 relevant features. These have been sorted by their p-values, and the top 16 features have been retained, forming our feature vectors.

We suggest the following pre-processing steps.

- **Oversampling:** The Synthetic Minority Oversampling Technique (SMOTE) has been employed to synthesize new feature vectors for positive cases, slightly increasing their proportion to 25% of negative cases.
- **Downsampling:** The negative cases have been further downsampled, bringing the proportion of positive cases to 50% of negative cases. This leaves us with 198 feature vectors for training.
- **Normalization:** The features have been centred and scaled.

[*]https://tsfresh.readthedocs.io/en/latest/index.html

2 ECoG data

Feature extraction, classification, network analysis

2.1 Classification

2.1.1 Autoregressive coefficients

For a signal defined as $x(t)$, the AR model with order p is defined as:

$$x(t) = \sum_{i=1}^{p} \phi_i x(t-i) + \epsilon(t)$$

Where $\epsilon(t)$ is white noise. The vector containing AR coefficients $u = [\phi_i]_{i=1,\ldots,p} \in \mathbb{R}^p$ can be used as a feature for the signal $x(t)$.

In this case, we have used the second-order AR coefficients after standardizing the data set. For each trial, $nchan \cdot 2$-many coefficients are estimated and stacked to be used as features.

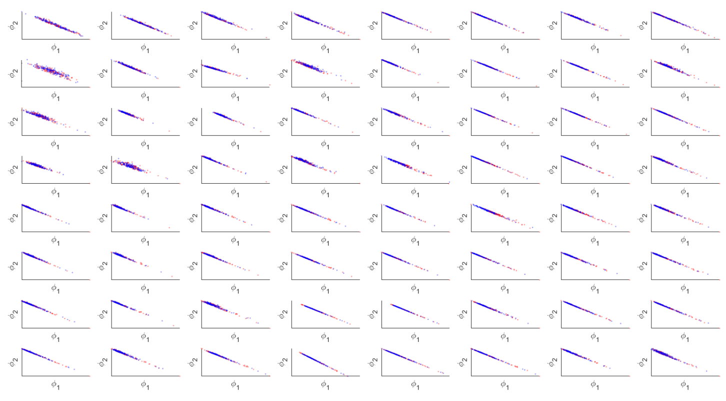


Figure 5: Visualization of the 2nd order AR coeff across channels

We have tested different models to classify the AR features, including linear SVM, linear discriminant, neural net (with a depth of 32 layers), Adaboost + tree ensemble, random forest (100 trees, maximum depth 3), etc. Individual performances are reported below.

Classifier	10-CV	Accuracy
Linear SVM	0.85	0.83
Neural net	0.79	0.76
Random forest	0.76	0.73

Table 1: Three best performing classifiers for AR order 2)

2.1.2 Bandpower

A window of a signal $x(t)$ is defined as

$$x_{\Delta t}(t) = \begin{cases} x(t) & \text{for } |t| < \Delta t/2, \\ 0 & \text{for } |t| \geq \Delta t/2. \end{cases}$$

Let $\hat{x}_{\Delta t}(t)$ be the Fourier transform, if exists. Then the PSD is defined as follows.

$$S_{xx}(f) = \lim_{\Delta t \to \infty} \frac{1}{\Delta t} |\hat{x}_{\Delta t}(f)|^2.$$

Define bandpower of $x(t)$ in the frequency range $f_1 \leq f_2$ as $P_{[f_1,f_2]} = \int_{f_1}^{f_2} S_{xx}\, df$.

In this case, training, as well as the test data, are first band-pass filtered to retrieve the well-characterized frequency bands (refer to the figure below).

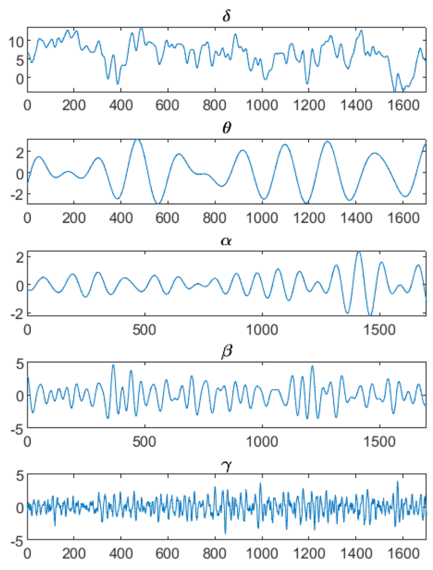


Figure 6: Decomposition into the known cortical rhythms

The invocation and maintenance of the Mu-band (8-14 Hz) during the imagination or anticipation of motor activities are well-studied in the field of brain-computer interface[2] [3]. Therefore, we focus on this particular frequency range. The overall band power is estimated, for each trial. These are included in another set of features. The following figure plots histograms of the class-wise band power log values across all 64 channels.

We have tested different models to classify the BP features (including linear SVM, linear discriminant, neural net, Adaboost + tree ensemble, random forest, etc.). Individual performances are reported below.

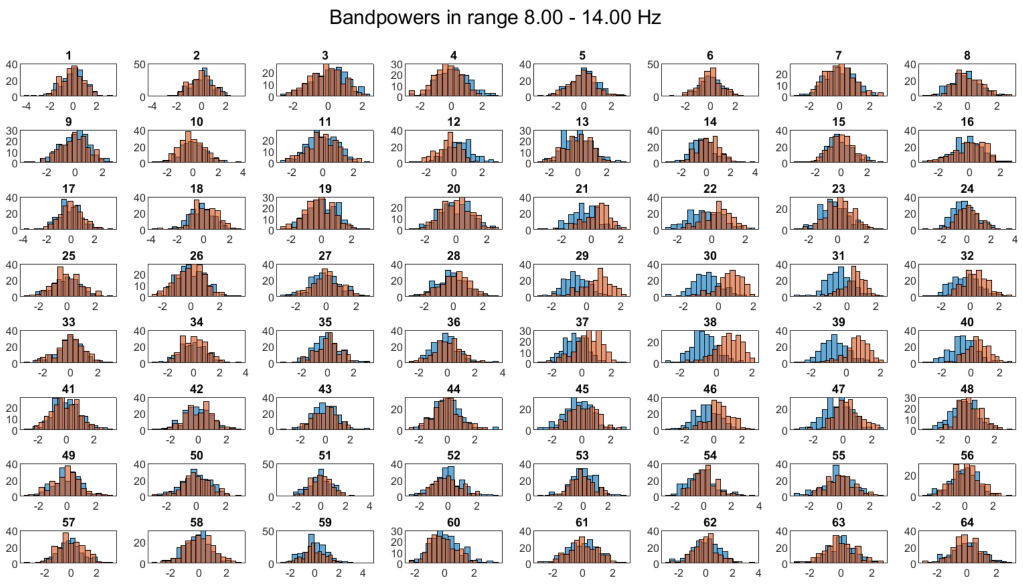


Figure 7: Distribution of the class-wise μ band power values

Classifier	**10-CV**	**Accuracy**
Neural net	0.83	0.78
Linear discriminant	0.82	0.73
Linear SVM	0.83	0.58
Adaboost + tree	0.83	0.57

Table 2: Four classifiers tested for the BP feature

2.1.3 Log-variance of CSP features

For two given classes of signals $X_i(n, t_i), i \in \{1, 2\}$ (for n many channels and t_i many samples), the CSP algorithm intends to find a spatial filter $w \in \mathbb{R}^n$ such that:

$$w := \arg\max_{w} \frac{w^t \Sigma_1 w}{w^t \Sigma_2 w}$$

where Σ_1, Σ_2 are trial-averaged for their respective class.

The intuitive idea is to maximize the *difference in variance* between two signals, by creating a spatial filter projection. The idea is quite similar to LDA, where a filter is applied to maximize the difference in class means. In the following example, two Gaussian classes are generated synthetically. The CSP algorithm finds a projection, where the variance of one class is maximized along one particular axis.

The algorithm is presented below.

i For $X_i \in \mathbb{R}^{ntrial,nchan,nsamp}$, calculate trial-averaged $\Sigma_i \in \mathbb{R}^{ncha,nchan}$.
ii Solve for $W : \Sigma_1 W = \lambda \Sigma_2 W$.
iii Sort columns of W based on $D(\lambda)$.
iv Choose first and last m many columns of W.
v $Y_{csp} = W_{csp}^t Y, W_{csp} \in \mathbb{R}^{n,m}, Y \in \mathbb{R}^n$

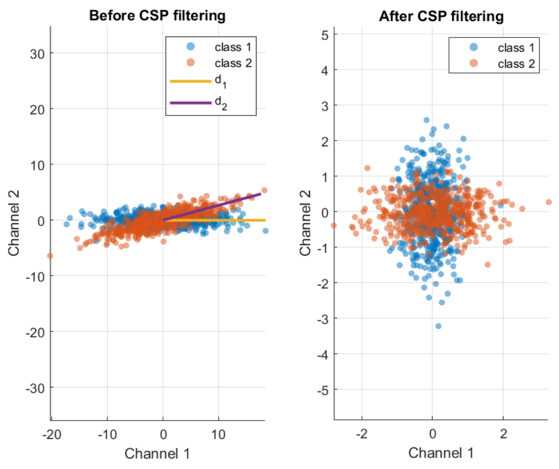


Figure 8: Example usage of the CSP filter

The following figure shows the separability of the log-variance values of the transformed signal. The CSP weights estimated from the train data are used to find the projection for the test data.

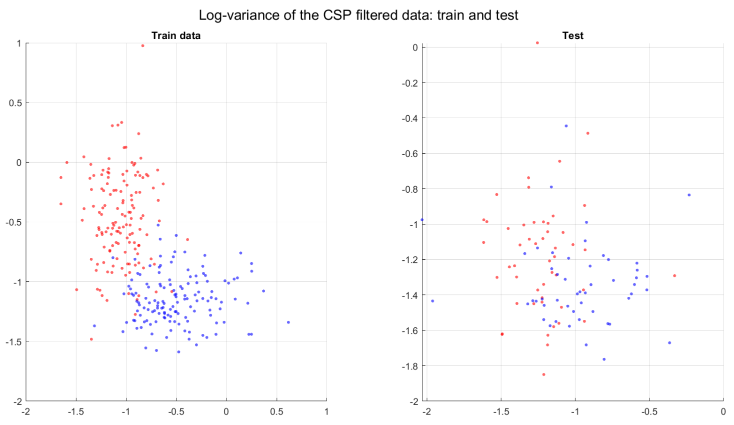


Figure 9: Logvar of the individual time series after CSP-filtering

We have tested different models to classify the CSP-based features (including linear SVM, linear discriminant, neural net, Adaboost + tree ensemble, random forest, etc.). Individual performances are reported below.

Classifier	**10-CV**	**Accuracy**
Linear discriminant	0.93	0.72
Linear SVM	0.93	0.69
Neural net	0.92	0.51
Random forest	0.85	0.50

Four classifiers tested on different combinations (with the 1st and last CSP feature)

2.1.4 Hard voting ensemble

A voting ensemble considers the predictions from multiple classifiers and returns the class label with the maximum *vote*. In other words, the ensemble's predicted target label is the mode of the distribution of individually predicted labels. Following is a general schematic of this technique.

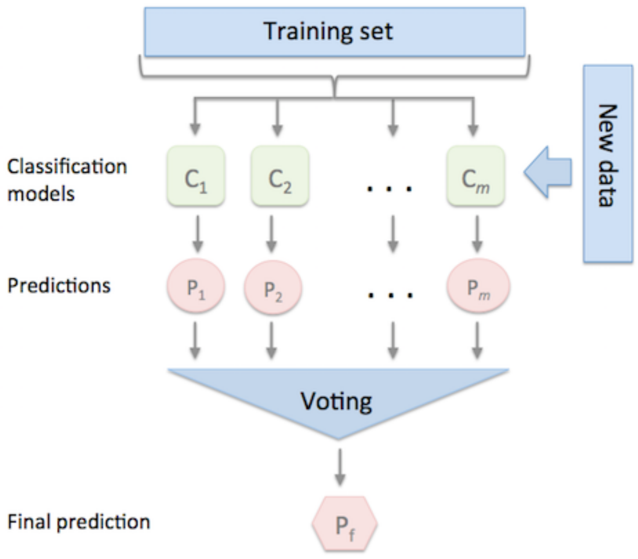


Figure 10: A general schematic of the Hard voting ensemble

Here, we combine the three features and the best-performing classifier for each. It is important to note that the features used are independent by default. The autoregressive coefficient is a time-domain feature, whereas band power comes under the frequency domain. Logvariance, even though is a time-domain metric, is estimated after CSP-filtering the raw data. The result of the final hard voting ensemble classifier is presented below.

Classifier	10-CV	Accuracy
Linear SVM (AR coefficients, order 2)	0.82	0.81
Neural net (Bandpower, 8-14 Hz)	0.81	0.80
Linear discriminant (Logvar CSP, 1 feature)	0.92	0.71
Majority vote	**0.92**	**0.87**

Performance of the hard voting ensemble.

2.2 Network analysis

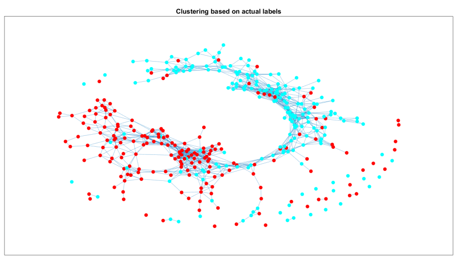
In this part, the train and the test datasets are merged. After dropping off the labels, this compilation is clustered for a particular feature, using the K-means algorithm. The synthesized class labels are compared against the real class identities using the Kendall's rank correlation coefficient.

2.2.1 Using BP features

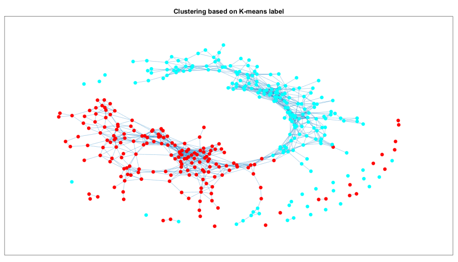
In order to construct the graph $G = (V, E)$, we first define the vertices. One vertex V_i in this graph represents one particular trial in the merged dataset (i.e. $i = 1, \ldots, 378$. Two vertices V_i, V_j shares an edge $e_{i,j}$ iff $d(bp_{[8,14]}(V_i), bp_{[8,14]}(V_j)) \in (0, \mathcal{U})$, where d is the Euclidean distance metric, $bp_{[8,14]}(V_j)$ denotes the Mu-range band power for the i-th trial and $\mathcal{U} = 1$ is merely an upper limit (chosen for an optimum visualization of the network).

The broad algorithm is presented below.

```
   i  normalize the BP features for trial and test
  ii  start with a set of channels $\{c_i\}; n(\{c_i\}) = N$ & set $\tau_0 = 0$
 iii  for $i = 1, \ldots, N$
  iv      choose $\mathcal{C} \subset \{c_i\}; n(\mathcal{C}) = i$
   v      for $j \in \mathcal{C}$
  vi          select $bp_j := bp[., j]$ & kmeans_label = kmeans$(bp_j, 2, d)$
 vii          $\tau_j > \tau_0 \implies \tau_0 \leftarrow \tau_j$ & best_label $\leftarrow$ kmeans_label
viii      end for
  ix  end for
```



(a) V_i highlighted using the *given* labels



(b) V_i highlighted using the *K-means* labels

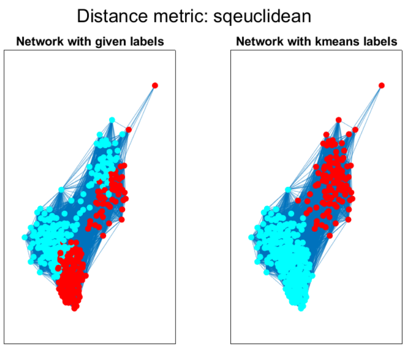
The channels selected in this way are 29, 38 and 39, with a $\tau = 0.6935$. The network thus constructed is presented above. We report the mean clustering coefficient given by $CC_m = \langle CC_i \rangle = \langle \frac{2n\{e_{jk}:v_j, v_k \in N_i, e_{jk} \in E\}}{k_i(k_i-1)} \rangle = \mathbf{0.4896}$, and the mean degree $\langle k \rangle = \mathbf{8.011}$.

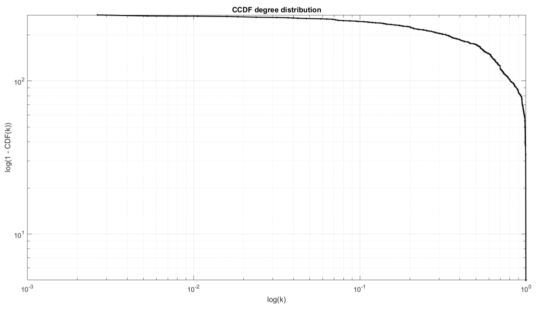
2.2.2 Using CSP-based features

A similar treatment is performed with the log-variance of the data, after CSP-based filtering. Considering first two and last two CSP filters, the ECoG data is projected to a $ntrial \times 4$ feature matrix. We construct the graph $G = (V, E)$ by defining $u_i = W^t_{csp} D_{nchan,t}; i = 1, \ldots, ntrial$ and defining an edge $e_{i,j}$ between V_i, V_j iff $d(u_i, u_j) \in (0, \mathcal{U})$, where d is a particular distance metric and $\mathcal{U}$ is an upper limit as before.

In this case, three different distance metrics are used while performing the K-means clustering. Individual τ and clustering accuracy values are reported in the following table.

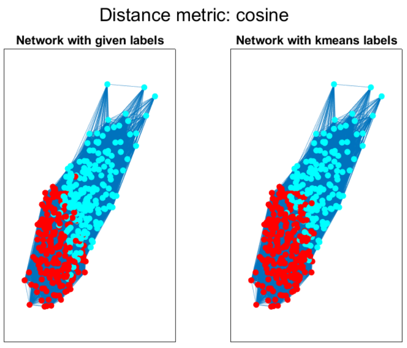
Distance metric	τ	**(tp $\wedge$ tn)/ntrial**
Euclidean	0.0059	50%
Cosine	0.7678	89%
Correlation	0.8557	93%

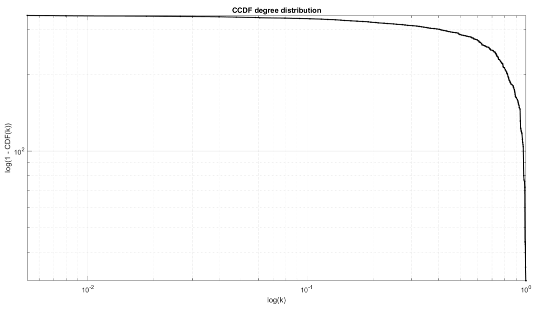

Distance metric: sqeuclidean
Network with given labels
Network with kmeans labels



(a) $\mathcal{U} = 1$ (b) CCDF of the constructed network

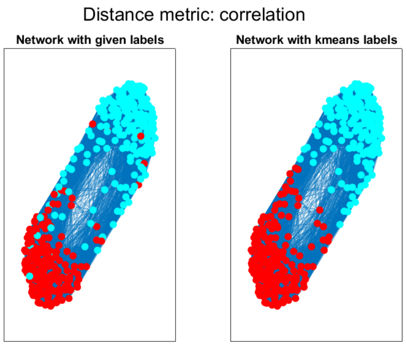
Figure 12: Distance: Euclidean, mean CC = 0.8179

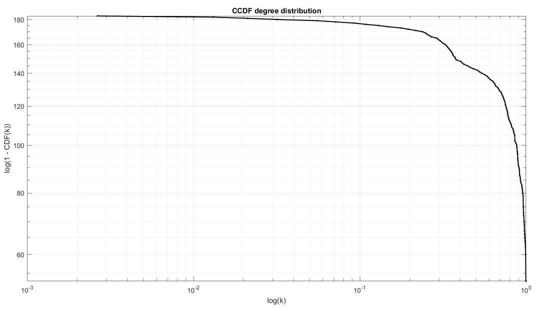




(a) $\mathcal{U} = .2$ (b) CCDF of the constructed network

Figure 13: Distance: Cosine, mean CC = 0.8668


Distance metric: correlation
Network with given labels
Network with kmeans labels



(a) $\mathcal{U} = .6$ (b) CCDF of the constructed network

Figure 14: Distance: Correlation, mean CC = 0.7179

The networks constructed and the associated degree distribution diagram are presented above.

3 Earthquake data

Feature extraction, classification, network analysis

3.1 Classification

3.1.1 Random Forest on raw data

Upon using the Random forest classifier on raw unbalanced training data, the class predictions were all that of the majority class, namely 0. Since the test data consists of 104 members of class 0 out of a total of 139, we get a classification rate of 74.820%. However, this is a misleading statistic, given the extremely unbalanced nature of the classification task. A better metric to optimize would be a macro F1 score. With this metric, the classifier performs extremely poorly, with a score far less than 0.5. Indeed, a classifier that assigns every example to the majority class earns a precision score of zero with respect to the minority class. In a rare event classification problem such as this, this kind of classification has little value.

Since the number of minority class members in the training set is 58 out of 322 training data set, we downsample the majority class member to the same size, and repeat the same procedure. This yields a macro F1 score of 0.56. The classification accuracy here is 57%.

We note that using a raw time series as a feature vector is problematic, as it is sensitive to small perturbations or phase shifts.

3.1.2 Hard voting ensemble

The final features, obtained using the `tsfresh` feature vectors as described in Section 1.2, include the following.

- Intercepts and standard errors from linear regression (calculated on chunked data, aggregated by minimums and maximums)
- Fractions of amplitudes which fall a certain number of standard deviations away from the mean.
- Lempel-Ziv signal complexities
- Maximum, or absolute maximum signal amplitudes
- Number of peaks, sustained over various supports
- Sample entropies
- Quantiles

We reiterate that these features were chosen procedurally instead of being handpicked, and consequently are difficult to interpret or assign meaning.

The processed feature vectors have been used to train a Linear Discriminant classifier, a Gaussian Naïve Bayes classifier, a Logistic Regression classifier, and a Random Forest classifier (128 estimators, maximum depth 3). The predictions from each of these models has been combined using majority vote.

Unlike the ensemble in the ECoG setup, all four sub-models are fed with the same features. Thus, a significant performance boost is not to be expected from the majority vote. Instead, the role of the hard voting ensemble is to minimize errors from the individual models and produce stable predictions. It has been observed that the performance of each individual model can fluctuate greatly over executions of the pipeline; this is likely due to the upsampling/downsampling process. Hard voting works to even out these discrepancies.

The final hard-voting ensemble achieves a test accuracy rate of 73%, with a macro F1 score of 0.66. This is the average of the F1 scores 0.82 for the negative cases, and 0.51 on the positive cases. Keeping in mind the high unbalances of the dataset, we report the F1 scores (F1:= $HM(\frac{tp}{tp+fp}, \frac{tp}{tp+fn})$), instead of the accuracy values in the following table.

Classifier	**10-CV**	**Test**
Linear Discriminant	0.63	0.66
Gaussian Naïve Bayes	0.59	0.53
Logistic Regression	0.67	0.65
Random Forest	0.68	0.64
Majority vote	**0.68**	**0.66**

Cross validation and test performance of each classifier on the pre-processed `tsfresh` feature vectors, reported as macro F1 scores.

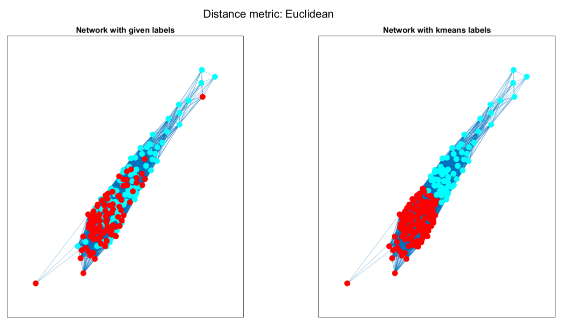
3.2 Network analysis

As mentioned earlier, the train and test data sets are merged. Using K-means clustering, labels are generated, after down-sampling the number of negative cases. Via an iterative method of boosting Kendall's τ, the best features are selected before constructing the network.

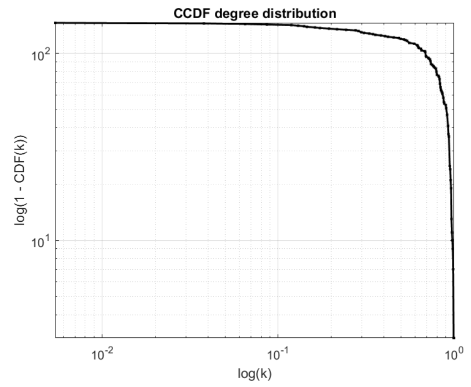
```
   i  $X_0^{368,16}, X_1^{93,16}$ be the features for labels $\{0,1\}$
  ii  $X_{00}^{93,16} \xleftarrow{\text{down-sample}} X_0^{368,16}$ & combine $X \leftarrow [X_{00}, X_1]$
 iii  start with features $\{c_i\}_{i=1,\dots,N}$ & set $\tau_0 = 0$; best_feat = []
  iv  for $i = 1, \dots, N$
   v      define $\mathcal{C} := \{c_i\} \setminus \{i\}$
  vi      for $j \in \mathcal{C}$
 vii          select $x_j := X[., j]$ & kmeans_label = kmeans($x_j, 2, d$)
viii          if $\tau_j > \tau_0$
  ix              $\tau_0 \leftarrow \tau_j$ & best_feat $\leftarrow$ j
   x          end if
  xi      end for
 xii  end for
```

Using the best 15 features thus obtained, the network is constructed with $\tau = \mathbf{0.3271}, (tp \wedge tn)/ntrial \approx \mathbf{0.32}, \langle CC_i \rangle = \mathbf{0.8472}, \langle k \rangle = \mathbf{107.31}$. Note that the construction procedure remains the same: V_i represents one particular trial and an edge between V_i, V_j exists iff $d(\mathcal{F}(i,.), \mathcal{F}(j,.) \in (0, \mathcal{U}$ where $\mathcal{F}$ is the feature matrix.

If repeated considering correlation-based distance metric, the same analysis yields a slight boost in the Kendall's correlation value ($\tau = \mathbf{0.4086}, (tp \wedge tn)/ntrial \approx \mathbf{0.33}$) with 7 best features. Network matrics include $\langle CC_i \rangle = \mathbf{0.8338}, \langle k \rangle = 86.94$.

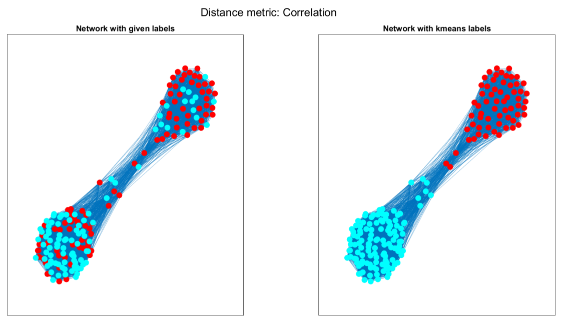

Distance metric: Euclidean
Network with given labels
Network with kmeans labels

(a) $\mathcal{U} = 4$

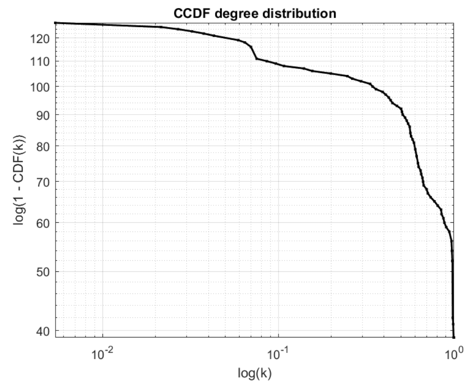


(b) CCDF of the constructed network

Figure 15: Distance: Euclidean


Distance metric: Correlation
Network with given labels
Network with kmeans labels

(a) $\mathcal{U} = .85$



(b) CCDF of the constructed network

Figure 16: Distance: Correlation

4 Conclusions

4.1 ECoG data

The goal was to extract meaningful (and interpretable, if possible) features and develop a robust classifier. We propose an ensemble classifier combining a linear SVM, neural net and linear discriminant, individually performing on AR coefficients, the Mu band's band power, and the CSP-filtered data's log variance. The accuracy of the test data is around 87%, quite on a par with the maximum accuracy ever attained on this data set (91%).
However, question regarding the *interpretability* and *extendibility* is always a concern. It is quite difficult to interpret the autoregressive coefficients in the context of any time-series data. So is the question of why a neural model is one of the best classifiers for the band power features. Extendibility raises concerns about how generally usable these features are and addresses the issue where the CSP-based method fails to generalise the test data.

One immediate improvement to our methodology is incorporating a recursive channel selection/elimination method, particularly for features like AR coefficients or band power. Additionally, multiple other features can be probed into, including, entropy, discrete wavelet transformation, mutual information, CSP-based analysis on band powers, etc.

The network construction categorically shows that with proper distance metrics and features, the ECoG data points display clustering behaviour and tend to form discrete populations.

4.2 Earthquake data

Even brute-force feature extraction and an extensive ensemble (operating on linear discriminant, Gaussian Naïve Bayes, logistic regression, random forest) fail to attain a substantial accuracy. The maximum F1 score we could report is 0.66 on the test data. Furthermore, the features extracted in this way are hardly interpretable.

This result is quite strongly supported by the poor clustering tendency of the data, even after recursively selecting the best features (based on Kendall's τ comparison). After using a correlation-based distance metric, the best K-means label displays a $\tau \approx 0.41$ with the actual label.

References

[1] Vabalas, A., Gowen, E., Poliakoff, E., & Casson, A. J. (2019). *Machine learning algorithm validation with a limited sample size.* In E. Hernandez-Lemus (Ed.), PLOS ONE (Vol. 14, Issue 11, p. e0224365). Public Library of Science (PLoS). https://doi.org/10.1371/journal.pone.0224365

[2] Aleksandrov, A. A., & Tugin, S. M. (2012). *Changes in the Mu Rhythm in Different Types of Motor Activity and on Observation of Movements.* In Neuroscience and Behavioral Physiology (Vol. 42, Issue 3, pp. 302–307). Springer Science and Business Media LLC. https://doi.org/10.1007/s11055-012-9566-2

[3] Yu, H., Ba, S., Guo, Y., Guo, L., & Xu, G. (2022). *Effects of Motor Imagery Tasks on Brain Functional Networks Based on EEG Mu/Beta Rhythm.* In Brain Sciences (Vol. 12, Issue 2, p. 194). MDPI AG. https://doi.org/10.3390/brainsci12020194

[4] Bagnall, A., Davis, L., Hills, J., & Lines, J. (2012). *Transformation Based Ensembles for Time Series Classification.* In Proceedings of the 2012 SIAM International Conference on Data Mining. Proceedings of the 2012 SIAM International Conference on Data Mining (SDM). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611972825.27