CS156 final project:
Motor imagery classification

Gili Karni

April 20, 2018

Introduction

The paper is divided into two sections. The first part - Theory and context- discusses the problem, the challenges, and provides mathematical theory regarding a suggested solution. The second part -The experiment- discussed the implementation, evaluation method, and the results.

The appendix including the code is attached at the end.

Theory and context

Motivation

Non-invasive BMI/BCI. Using an understanding of mental imageries through EEG enables control over devices such Brain Machine Interface (Brain Computer Interface) which can be used to augment human cognitive or sensory function.

The goal

Discriminate between two brain states; it can be task versus rest, hand versus foot, or normal versus abnormal patterns. The discrimination uses the EEG activation patterns as a proxy for the motor imagery thus as a representation of the two classes.

Terminology

EEG An electroencephalogram (EEG) measures electrical activity from the brain using electrodes attached to the scalp. The voltage fluctuations are a result of ionic flows within the neurons which, in combination, are associated the different mental states.

ERP An event-related potential (ERP) is the measured brain response that is the direct result of a specific sensory, cognitive, or motor event. Measured using EEG electrodes.

Motor imagery a mental process of motor action. It captures the electro-waves in one's brain in a given time window. It includes preparation for movement and the mental operations of the motor representation.

An activation channel is an electrode capturing brainwave activity is called an EEG channel. There are different levels of accuracy depends on the number of electrodes.

An event is a neuro-electrophysiological response to a sensory stimulus

EEG epoch is a procedure in which specific time-windows are extracted from the continuous EEG signal. These time windows are called "epochs," and usually are time-locked with respect an event.

ERD/S are event-related desynchronization and event-related synchronization. ERD illustrates a decrease in power in a given frequency band relative to a baseline. Similarly, ERS demonstrates an increase in power.

The data

- M trials signifies the number of individual repetitions of the task.
- N channels; corresponding to number of electrodes.

- T time points are the number of samples taken in the time window chosen to describe the event

The dataset consists of M matrices, each in shape $N * T$ (The labels are a list in size M describing the relevant class)

Feature selection

EEG data, as mentioned above is highly dimensional. To apply classification methods, we apply methods to reduce the dimensionality by extracting relevant features. The first step to avoid the dimensionality curse is manual feature selection-

There are three sources of information that can be used to extract features from EEG signals:

1. Temporal: describes the variation of a signal with time.
2. Spectral: illustrates the variation in activation over band frequency.
3. Spatial: describes variation in activation with its origin channel.

A-priori feature selection:

1. Temporal: Highly depends on the setting of the measurements; knowing when the subjects were relaxed, cued, or active can help discriminate the time window relevant for classification. Usually, measurements will go as follows; pre-movement stage, cue, execution of the task, and rest. The pre-cue time can be used as a baseline.(Schalk et al., 2004)
2. Spectral: Bandpass filter considers only signals between two specific frequencies and discriminates against signals at other frequencies. There are two standard filters; FIR and IIR. We differentiate between five basebands. Delta band (1 – 4 Hz), theta band (4 – 8 Hz), alpha band (8 – 12 Hz), beta band (13 – 24 Hz) and gamma band (> 25 Hz); each illustrates (generally speaking) different types of mental states. (iMotions, 2016)
3. Spatial: Different channels are associated with different cortices (which are related to different cognitive processes). i.e., the electrodes positioned over the motor cortex will pass power variation associated with motor activation.(Schalk et al., 2004)

A substantial caveat is the lack of accuracy regarding this knowledge. It is almost impossible to attain two comparable pictures where only one single aspect has changed thus pointing a single cause for a result; as well as the recognition that each brain is wired slightly different.

Feature extraction

I will discuss one method to preform feature extraction, Common Spatial Pattern (CSP).

Common Spatial Pattern

CSP is a customized method for learning spatial filters for EEG motor imagery classification.

The CSP algorithm finds a spatial filter W such that the variance of the filtered signal is maximal for one class and minimal for the other class. Thus, captures the components that contain the most discriminative information regarding the different motor imagery classes.
To successfully use a CSP, we must follow these assumptions-

- Time window and the frequency band are known (which can be set using apriori knowledge)
- Gaussian distribution: The band-passed signal is a joint Gaussian within the time window (This require choosing a relatively small time window)
- Binary classification: The source activity differs between the two objective classes

How does CSP create the filter W?

X_c denotes the processed EEG matrix per class c; which has dimensions of N*T (N channels and T time samples)
The normalized covariance matrix for class c, R_c, is:[1]

$$R_c = \frac{X_c X_c^T}{trace(X_c X_c^T)}$$

The averaged normalized covariance $\hat{R_c}$ is given by the mean of all the covariances matrices of all the trials (M) in each class.

By jointly diagonalizing the covariance matrix and the average covariance matrix, we get U and Σ_c.

The matrix of eigenvectors U and Σ_c the diagonal of the corresponding eigenvalues are used in the whitening transformation matrix. Thus, creating a matrix of uncorrelated variables using the linear transformation values. (The whitening matrix's covariance is the identity matrix)

$$P = \Sigma_c^{-\frac{1}{2}} U^T$$

P is used the transform the average covariance matrices :

$$S_c = P\hat{R_c}P^T$$

Such that both S matrices (for both classes) share the same eigenvectors and their corresponding eigenvalues sums to 1 - $\Sigma_1 + \Sigma_2 = 1$

$$S_c = U\Sigma_c U^T$$

[1] trace is the sum of the elements on the main diagonal (the diagonal from the upper left to the lower right)

The eigenvectors with the largest eigenvalues for one class have the smallest eigenvalues for the second class (and vice versa)- such that it is optimal for separating variance in the two EEG signal matrices (as seen in the below example figure)

The filter matrix W (also known as the projection matrix):

$$W = U^T P$$

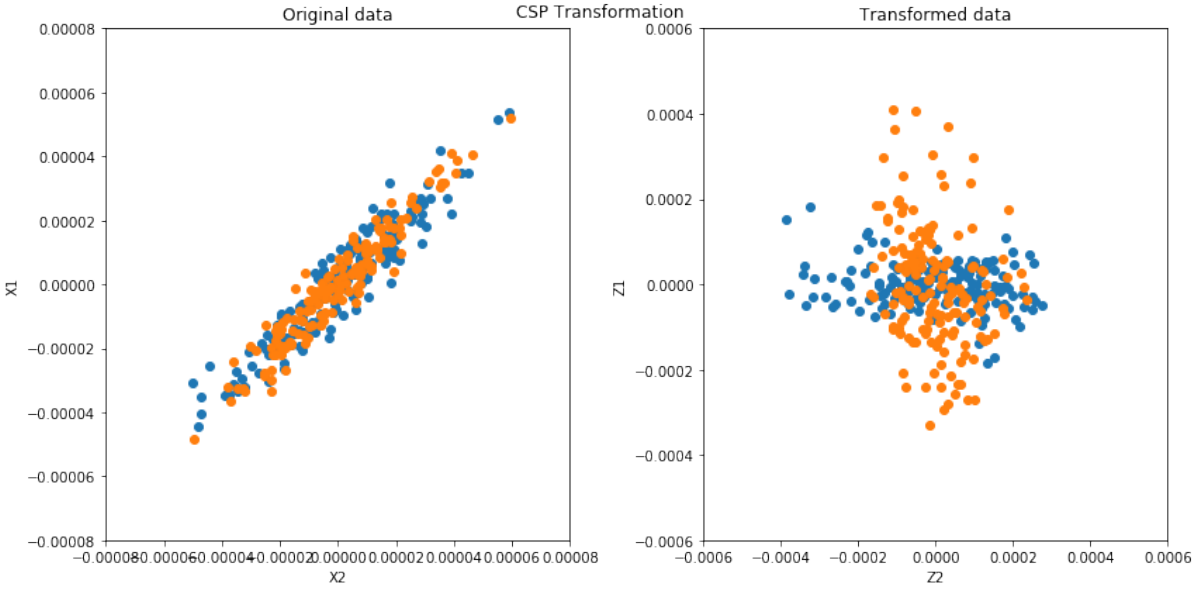
Using the projection matrix to create an uncorrelated components matrix:

$$Z_c = WX_c$$

Z_c represents the source components of the different tasks. Similarly, we can inverse transform Z using W^{-1} to restore the original signals. The Z matrix is in shape (new number of channels, time samples)

(Wang et al., 2005)

Figure 1: Sample data CSP transformation



CSP linear transformation to increase discriminative variation; the two figures present the samples from the two classes (one in orange and the other in blue); in the left figure- each axis represents activation over a channel, in the right channel- each axis represents activation over a transformed channel.

CSP as a method for motor imageries

CSP is a good fit for EEG decoding firstly since it is customized for this cause. It is very simple to implement and use it, and it does not require a lot a computational power for high quality preference. Moreover, since it is dealing with spatial information, it is versatile and can work well with all types of variation (i.e both ERD/S)(Lotte, 2014).

Its flaws are mostly due to its non-robustness to noise and to non-stationarities and its prone to over fitting. It is a good fit for very specific tasks yet sometime hard to generalize from.(Lotte, 2014)

Classification

I will discuss one method to preform feature extraction, Linear Discriminate Analysis (LDA).

Linear Discriminate Analysis

Linear Discriminate Analysis (LDA) is a supervised model which classifies based on data separability to classes. LDA uses few normality and ratio assumptions:

1. The categories are sampled from a Gaussian distribution
2. There are meaningfully fewer dimensions than data points (N«D) to enable to invertibility of the B matrix (the matrix describing the within-class separability)
3. There are no elements that consistently do not vary between samples
4. Each attribute has the same variance

The CSP output answers all these assumptions:
The linear transformation does not interrupt with the data distribution, and both classes remain Gaussians. The number of dimensions is limited by the number of channels where the number of samples represents the number of data points and usually is much bigger (EEG machines typically capture hundreds of samples per seconds and the CSP reducing the number of channels). Moreover, it is unlikely for one data point to remain static across all data points since each data point represents activation variation across different channels, no point has the same time with the same channel. Regarding the variance, we assume similar variance due to a similarity in source and limitations

LDA uses the normality assumption to learn the two classes and Bayes theorem to predict a novel data point.

Since the data is normal, its projections and their distributions are also normal; thus

$$y_c^n = W^T x_c^n$$

$$p(y_c) = N(y_c|\mu_c, \sigma_c^2)$$

where

$$\mu_c = W^T m_c$$

$$\sigma_c^2 = W^T S_c W$$

For both classes, where c signifies the class.

We look for W that minimizes the overlap over the two distributions; which is a result of maximum differences between means and as small variances- which is captured by the following objective function.

Using the matrix representation:

$$F(w) = \frac{W^T(m_1 - m_2)(m_1 - m_2)^T W}{W^T(\pi_1 S_1 + \pi_2 S_2)w}$$

The classification prediction is given by Bayes, using the optimized W.

$$p_c(x) = x(\frac{\mu_c}{\sigma_c{}^2}) - (\frac{\mu_c^2}{2\sigma_c^2} + ln(p_{prior}))$$

(Brownlee, 2018) (Barber, 2015)

The experiment

Data

The EEG dataset was contributed from PhysioNet by the developers of the BCI2000 instrumentation system (Schalk et al., 2004)(Goldberger et al., 2000). The EEG signals were obtained from 109 healthy subjects that were asked to perform different motor/imagery tasks while being rerecorded using 64 electrodes (EEG 20-10 system standard).

In this experiment, I focus on the task asking subjects to open and close their right or left fist corresponding to a figure showed on the screen.

Feature selection

As mentioned above, the EEG picture is a whole is very noisy, yet extracting only the relevant features can lead to loss of important data; which varies between individuals. The challenge is to create a generalizable model that captures the signals associated with the event. I follow a strategy of elimination rather than addition, to try and reach that goal.

Since CSP is a spatial algorithm and following its assumptions; we will have to specify the frequency (spectral) and time window(temporal).

The frequencies relevant for motor activation are α and β. α ranges approximately between 8-12Hz and β ranges roughly between 13-24Hz.

Alpha, α, waves have several functional correlates reflecting sensory, motor and memory functions. You can see increased levels of alpha band power during mental and physical relaxation with eyes closed. By contrast, alpha power is reduced, or suppressed, during mental or bodily activity with eyes open. (Pfurtscheller & Aranibar, 1977).

Beta, β, band power becomes stronger as we plan or execute movements, particularly when reaching or grasping requires fine finger movements and focused attention.Interestingly, this increase in beta power is also noticeable as we observe others' bodily movements. Mirror neurons (Zhang et al., 2008).

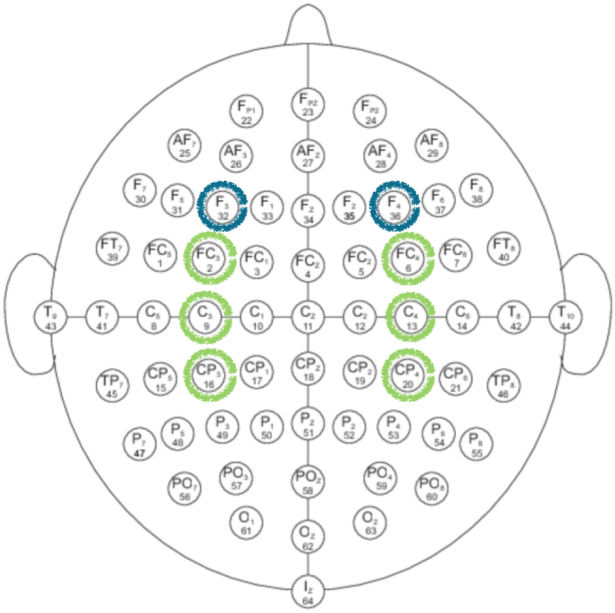
The experiment was structured such that the cue was given at time t=0, and the activity lasted 4 seconds. Such that the time window relevant for classification is 0-4. (Alomari et al., 2013)

The time window before cue (-2s up to -.5s) can be used as a baseline. A baseline is a form of normalization of the transform matrix. The baseline correction process computes the mean of the baseline period and subtracting it from the data.(Alomari et al., 2013)

I expect to see ERD/S variation in the channels responsible for motor control.

- FC3, FC4, C3, C4, CP3, CP4 - location above the motor sensorimotor cortex and the lateral premotor cortex, associated with hand movement, located above the area related to hands.
- F4, F3 - located above the frontal lobe, associated with motor planning (Alomari et al., 2013)(Zama and Shimada, 2015)

Figure 2: EEG montage representing system 10-10



The channels highlighted are FC3, FC4, C3, C4, CP3, CP4 (green) and F4, F3 (blue)

Feature extraction

I used CSP (Martinos.org, 2018), modified the parameters as follows:

1. FIR bandpass filter is chosen due to its higher stability. (Martinos.org, 2018)
2. The regularization for covariance estimation using Oracle Approximating Shrinkage since it operates better with Gaussian data. (Scikit-learn.org, 2017)

Classification

I used an LDA with its default parameters. (Scikit-learn.org, 2018)

Evaluation and testing

I used a classification report to estimate the accuracy of the model. Precision marks the ratio between the true positive and the predicted positive and recall stands for the ratio between the true positives and the relevant elements. Such that the probability to miss of a true positive is captured by the recall

score whereas the probability to mis-classify an irrelevant element is captured by the precision.

Using EEG classifier for BCI, I believe the precision is of a bigger importance, one would not want a BCI to act out of the desired action, yet a miss of an action is usually less critical. Yet, in the case of use for screening of abnormal brain activity, the opposite is the true. As many other disease detectors, one would rather a false-positive rather than missing a true diagnosis.

I used a train set and a test set to evaluate the preference of the classifier with novel data points. The split divides the data into 90% train and 10% test stratified over the two classes.

I added to each results the relevant CSP components montage which illustrates the patterns of highest variance, these patterns hint the channel that was either activated or suppressed in a discriminative manner. Attaching these can help recognize the source of activation and using the apriori assumptions, provide an explanatory nature to it. The pattens plot includes the four most discriminative components, by their importance order.

Results and Discussion

I conducted three steps experiment using the same algorithm and the same parameters. The first targeted the whole dataset, the second targeted a set of 20 subjects and the last a single subject. The agenda was to demonstrate the sensitivity the algorithm has for individual differences, as mention above.

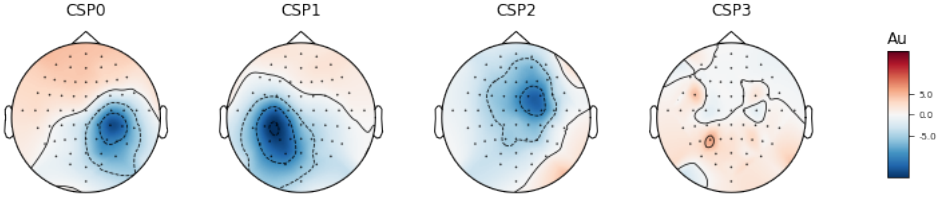
The results are as follows:

1. The entire dataset: 109 subjects

Table 1: Classification report: 109 subjects

	precision	recall	f1-score	support
left	0.67	0.63	0.65	237
right	0.61	0.65	0.63	212
avg/ total	0.64	0.64	0.64	449

Figure 3: Pattern EEG montage - entire dataset



The results show a balanced around 64%.

The pattern plot tells us that the two most discriminative patterns were a ERD in channels CP4 and CP3, followed by a ERD in the right frontal motor cortex.

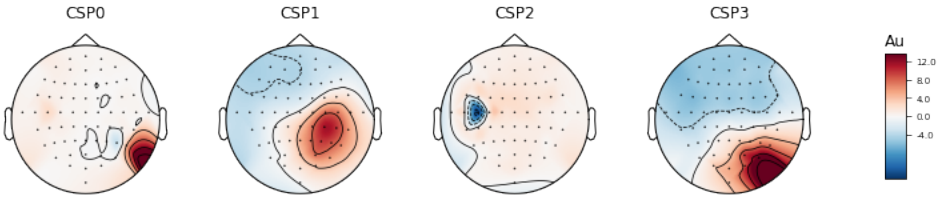
We can interpret these results as a ERD of β waves due to a motor event; specifically- hand movements. (Zama and Shimada, 2015)

2. A subset of the dataset: Twenty subjects

Table 2: Classification report: 20 subjects

	precision	recall	f1-score	support
left	0.68	0.73	0.71	41
right	0.74	0.70	0.72	46
avg/ total	0.71	0.71	0.71	87

Figure 4: Pattern EEG montage - Twenty subjects



The results are balanced around 71

The pattern plot shows an activation and a ERD in channels C4 and C3, in the second and third patterns, demonstrates hand movement in the left and right sides (Zama and Shimada, 2015). The first and last recalls an activation in the right partial cortex perhaps regarding verbal events or it can be an activation in the occipital lobe reflecting eye movement. Either way, this is a noisy interruption. (Sontisirikit, 2018)

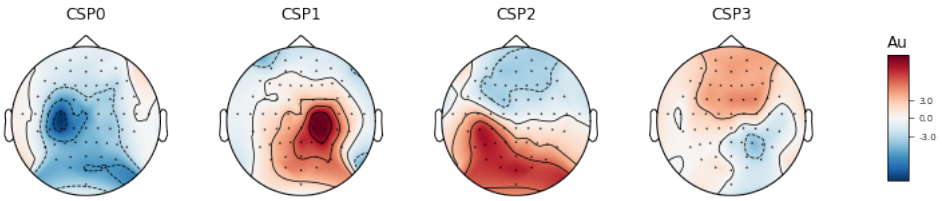
3. One subject The results are balanced around 100%.

Table 3: Classification report: 1 subject

	precision	recall	f1-score	support
left	1.0	1.0	1.0	3
right	1.0	1.0	1.0	2
avg/ total	1.0	1.0	1.0	5

The two most discriminative patterns were ERD/S patterns in channels C4/CP4 and C3/CP3, followed by a broad ERS in the left lateral cortex and a broad ERD of the right partial cortex.

Figure 5: Pattern EEG montage - One subjects



These are, again, areas related to hand movement illustrated by α and β waves due to motor imagery and motor execution.(Zama and Shimada, 2015)

The increasing accuracy corresponding to a decrease in sample size matches the knowledge regarding the lack of generalization ability of EEG analysis. As mentioned both in the theory part and in the explanation about CSP specifically, generalizations are not easy, both due to natural variations between people and the CSP algorithm lack of robustness to noise. It is probable to fit a model for a specific person with higher accuracy than it is for a group. However, newer models using neural networks can potentially mark individual differences (in an unsupervised manner) and increase the ability of the classification to generalize. (Gandhi et al., 2014)

References

[1] Barber, D. (2015). Bayesian reasoning and machine learning. Cambridge: Cambridge University Press.

[2] Brownlee, J. (2018). Linear Discriminant Analysis for Machine Learning. Machine Learning Mastery. Retrieved 20 April 2018, from https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/

[3] Brunner, C., Naeem, M., Leeb, R., Graimann, B., & Pfurtscheller, G. (2007). Spatial filtering and selection of optimized components in four class motor imagery EEG data using independent components analysis. Pattern Recognition Letters, 28(8), 957-964. http://dx.doi.org/10.1016/j.patrec.2007.01.002

[4] EEG (Electroencephalography): The Complete Pocket Guide. (2016). imotions. Retrieved 20 April 2018, from https://imotions.com/blog/eeg/

[5] Gandhi, V., Prasad, G., Coyle, D., Behera, L., & McGinnity, T. (2014). Quantum Neural Network-Based EEG Filtering for a Brain–Computer Interface. IEEE Transactions On Neural Networks And Learning Systems, 25(2), 278-288. http://dx.doi.org/10.1109/tnnls.2013.2274436

[6] Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P., & Mark, R. et al. (2000). PhysioBank, PhysioToolkit, and PhysioNet : Components of a New Research Resource for Complex Physiologic Signals. Circulation, 101(23), e215-e220. http://dx.doi.org/10.1161/01.cir.101.23.e215

[7] H. Alomari, M., Samaha, A., & AlKamha, K. (2013). Automated Classification of L/R Hand Movement EEG Signals using Advanced Feature Extraction and Machine Learning. International Journal Of Advanced Computer Science And Applications.

[8] Lotte, F. (2018). Tutorial on EEG Signal Processing Techniques for Mental State Recognition in Brain-Computer Interfaces F. In E. Miranda & J. Castet, Guide to Brain-Computer Music Interfacing,. Springer. Retrieved from https://hal.inria.fr/hal-01055103/document

[9] MNE — MNE 0.15 documentation. (2018). Martinos.org. Retrieved 20 April 2018, from https://martinos.org/mne/stable/index.html

[10] Schalk, G., McFarland, D., Hinterberger, T., Birbaumer, N., & Wolpaw, J. (2004). BCI2000: A General-Purpose Brain-Computer Interface (BCI) System. IEEE Transactions On Biomedical Engineering, 51(6), 1034-1043. http://dx.doi.org/10.1109/tbme.2004.827072

[11] Shrinkage covariance estimation: LedoitWolf vs OAS and max-likelihood — scikit-learn 0.19.1 documentation. (2018). Scikit-learn.org. Retrieved 20 April 2018, from http://scikit-learn.org/stable/auto_examples/covariance/plot_covariance_estimation.html#sphx-glr-auto-examples-covariance-plot-covariance-estimation-py

[12] sklearn discriminant analysis Linear Discriminant Analysis documentation. (2018). Scikit-learn.org. Retrieved 20 April 2018, from http://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html

[13] Sontisirikit, S. (2018). Guideline To Develop Brain Computer Interface. Sralife.com. Retrieved 20 April 2018, from http://sralife.com/?page=1_workblog_eeg-2014

[14] Yijun Wang, Shangkai Gao, & Xiaornog Gao. (2005). Common Spatial Pattern Method for Channel Selelction in Motor Imagery Based Brain-computer Interface. 2005 IEEE Engineering In Medicine And Biology 27Th Annual Conference. http://dx.doi.org/10.1109/iembs.2005.1615701

[15] Zama, T., & Shimada, S. (2015). Simultaneous measurement of electroencephalography and near-infrared spectroscopy during voluntary motor preparation. Scientific Reports, 5(1). http://dx.doi.org/10.1038/srep16438

MI classification using CSP-LDA

April 20, 2018

0.1 Common spatial patterns

```
In [1]: import glob

        import numpy as np
        import matplotlib.pyplot as plt

        from sklearn.pipeline import Pipeline
        from sklearn.model_selection import  train_test_split

        from sklearn import decomposition
        from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
        from sklearn.metrics import classification_report, recall_score, precision_score

        import mne
        from mne import Epochs, pick_types, find_events
        from mne.io import concatenate_raws, read_raw_edf
        from mne.channels import read_layout
        from mne.decoding import CSP
```

0.2 Set constants

```
In [2]: ## set params ##
        event_id = dict(left=0, right=1) # names of the two classes
        SEED = 7

        # the task chosen represent open and close left or right fist
        tasks = ["03", "07", "11"]

        #layout matches the EEG measurement layout
        layout = read_layout('EEG1005')
```

0.3 Utility function

```
In [68]: def extract_files (files):
             # read the edf files from disk
             raw_files=[]
             for f in files:
```

```
            raw_files.append(read_raw_edf(f, preload=True, stim_channel='auto',
            verbose =False))

    print ('extracting EDFs')
    return raw_files


def preprocess (raw_files):
    # Concatenate raw instances
    raw_data = concatenate_raws(raw_files)

    # rename channels
    raw_data.rename_channels(lambda x: x.strip('.'))


    print ('setting channels')
    #Pick channels by type and names; The data contains only eeg samples
    picks = pick_types(raw_data.info, eeg=True)

    print('setting filters')
    ## filtering by wave frequency ##
    raw_data.filter(8, 24, fir_design='firwin2', skip_by_annotation='edge',
                    picks=picks, verbose=False)

    print('extracting events')
    ## exctracting events ##
    events = find_events(raw_data, consecutive=True,
                         stim_channel='STI 014', verbose=False)
    # fix labels
    events[:, -1] -= 2

    print('extracting epochs')
    ## exctract ephochs ##
    epochs = Epochs(raw_data, events, event_id, -2, 4.1,
                proj=True, picks=picks,
                baseline=(-2,-.5),
                preload=True,
                verbose=False)

    #X = epochs.copy().crop(tmin=1., tmax=2.).get_data()
    X = epochs.get_data()
    y = epochs.events[:, -1]

    return(X,y, epochs)


def classify(X, y, clf):
```

```
    X_train, X_test, y_train, y_test = train_test_split(X, y,
                        test_size=0.1, random_state=SEED, stratify=y)

    print('fitting clf')

    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)


    print('preparing report')
    print('\n -------------------- \n')


    print (classification_report(y_pred, y_test,
                                    target_names= event_id))
```

0.4 Import data

```
In [69]: files = []
         files_20 = []
         file_1 = []

         for i in tasks:
             task_file_21 = glob.glob('data/S0*1/*R{}.edf'.format(i))
             files_20.append(task_file_21)
             task_file_22 = glob.glob('data/S0*5/*R{}.edf'.format(i))
             files_20.append(task_file_22)

             task_file = glob.glob('data/S*/*R{}.edf'.format(i))
             files.append(task_file)

             task_file_1 = glob.glob('data/S007/*R{}.edf'.format(i))
             file_1.append(task_file_1)

         #flatten lists of files
         files = [item for sublist in files for item in sublist]
         files_20 = [item for sublist in files_20 for item in sublist]
         file_1 = [item for sublist in file_1 for item in sublist]
```

0.5 Results for all of the dataset

```
In [70]: # read the edf files
         raw_files=[]
         for f in files:
             raw_files.append(read_raw_edf(f, preload=True, stim_channel='auto',
                                           verbose =False))
```

```
        print ('files imported')


        n_components = 4
        csp = CSP(n_components=n_components, reg='oas', log=True,
                  norm_trace=False)
        lda = LDA()
        clf = Pipeline([('CSP', csp), ('classification', lda)])


        raw_files=extract_files(files)
        X,y,epochs = preprocess(raw_files)

        classify(X,y,clf)

        # Printing the results
        csp.fit_transform(X, y)
        # plot CSP patterns estimated on full data for visualization

        csp.plot_patterns(epochs.info, layout=layout, ch_type='eeg',
                          units='Au', size=1.5)

        plt.show()
```

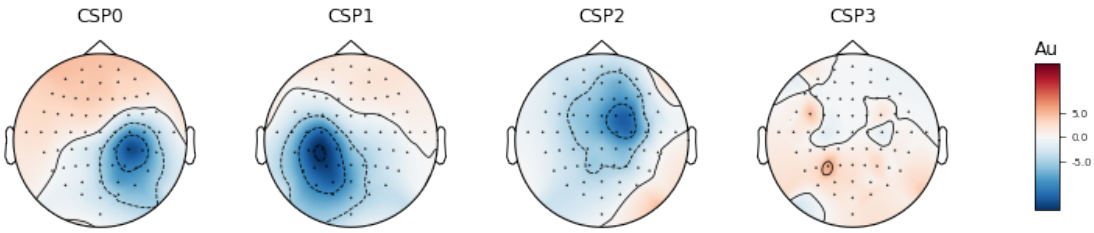
```
files imported
extracting EDFs
setting channels
setting filters
extracting events
extracting epochs
fitting clf
preparing report

 --------------------

             precision    recall  f1-score   support

       left       0.67      0.63      0.65       237
      right       0.61      0.65      0.63       212

avg / total       0.64      0.64      0.64       449
```



0.5.1 Results for 20 subjects

```
In [71]: # read the edf files
         raw_files_20=[]
         for f in files_20:
             raw_files_20.append(read_raw_edf(f, preload=True, stim_channel='auto',
                                              verbose =False))

         n_components = 4
         csp = CSP(n_components=n_components, reg='oas', log=True,
                   norm_trace=False)
         lda = LDA()
         clf = Pipeline([('CSP', csp), ('classification', lda)])


         raw_files_20=extract_files(files_20)
         X,y,epochs = preprocess(raw_files_20)

         classify(X,y,clf)

         # Printing the results
         csp.fit_transform(X, y)
         # plot CSP patterns estimated on full data for visualization

         csp.plot_patterns(epochs.info, layout=layout, ch_type='eeg',
                           units='Au', size=1.5)

         plt.show()
```

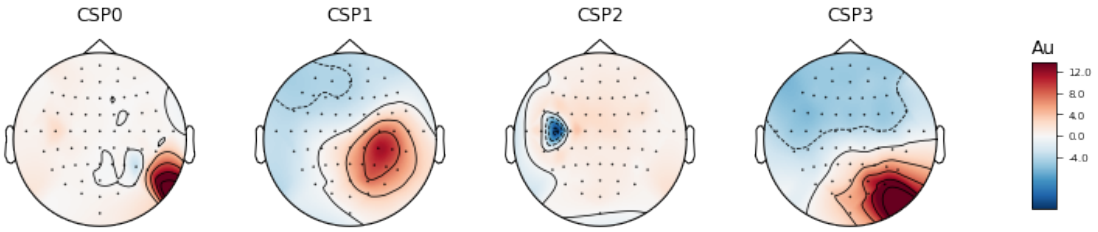
```
extracting EDFs
setting channels
setting filters
extracting events
extracting epochs
fitting clf
preparing report
```

```
 --------------------

             precision    recall  f1-score   support

       left       0.68      0.73      0.71        41
      right       0.74      0.70      0.72        46

avg / total       0.71      0.71      0.71        87
```



0.6 Results for 1 subject

```
In [73]: # read the edf files
         raw_file_1=[]
         for f in file_1:
             raw_file_1.append(read_raw_edf(f, preload=True, stim_channel='auto',
                                            verbose =False))

         n_components = 4
         csp = CSP(n_components=n_components, reg='oas', log=True,
                   norm_trace=False)

         lda = LDA()
         clf = Pipeline([('CSP', csp), ('classification', lda)])


         raw_file_1=extract_files(file_1)
         X,y,epochs = preprocess(raw_file_1)

         classify(X,y,clf)

         # Printing the results
         csp.fit_transform(X, y)
         # plot CSP patterns estimated on full data for visualization

         csp.plot_patterns(epochs.info, layout=layout, ch_type='eeg',
```

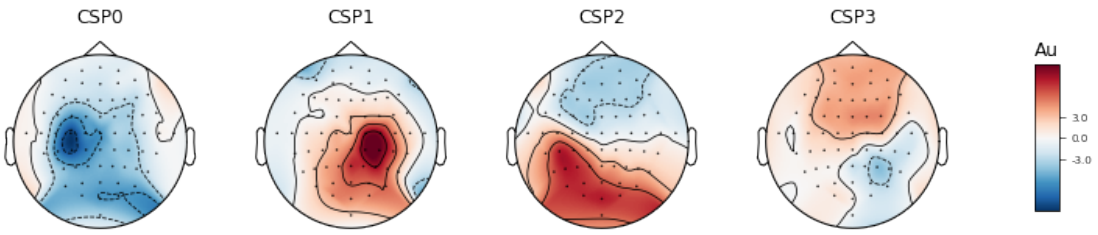
```
                        units='Au', size=1.5)

        plt.show()
```

```
extracting EDFs
setting channels
setting filters
extracting events
extracting epochs
fitting clf
preparing report
```

```
 --------------------
```

	precision	recall	f1-score	support
left	1.00	1.00	1.00	3
right	1.00	1.00	1.00	2
avg / total	1.00	1.00	1.00	5



CSP transformation visualization

April 20, 2018

In [112]:
```
import numpy as np
import matplotlib.pyplot as plt

from mne import Epochs, pick_types, find_events
from mne.channels import read_layout, Montage
from mne.io import concatenate_raws, read_raw_edf
from mne.datasets import eegbci
from mne.decoding import CSP
import mne

from sklearn.model_selection import  train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
```

0.1 Import sample files

In [116]:
```
# sample code from documentation
tmin, tmax = -1., 4.
event_id = dict(rest=1, activity=2)
subject = 1
runs = [1,4]

raw_fnames = eegbci.load_data(subject, runs)
raw_files = [read_raw_edf(f, preload=True, stim_channel='auto',
verbose=False) for f in
             raw_fnames]
raw = concatenate_raws(raw_files)

raw.rename_channels(lambda x: x.strip('.'))

raw.filter(7., 30., fir_design='firwin', skip_by_annotation='edge', verbose=False)

events = find_events(raw, shortest_event=0, stim_channel='STI 014', verbose=False)

picks = pick_types(raw.info, meg=False, eeg=True, stim=False, eog=False,
                   exclude='bads')
epochs = Epochs(raw, events, event_id, tmin, tmax, proj=True, picks=picks,
                baseline=None, preload=True, verbose=False)
```

```
epochs_train = epochs.copy().crop(tmin=1., tmax=2.)
labels = epochs.events[:, -1] - 2
```

0.2 Create figure

In [118]:
```
n_components = 4
# CSP model
csp = CSP(n_components=n_components, reg=None, log=None,
          norm_trace=True, transform_into='csp_space')

transform = csp.fit_transform(X, y)

#original data
o_c11 = X[0][1]
o_c12 = X[0][0]

o_c21 = X[-1][1]
o_c22 = X[-1][0]

# transformation
t_c11 = transform[0][1]
t_c12 = transform[0][0]

t_c21 = transform[-1][1]
t_c22 = transform[-1][0]

# figure
plt.figure(figsize=(12, 6))

plt.suptitle('CSP Transformation')

plt.subplot(121)
plt.title('Original data')
plt.scatter(o_c11, o_c12)
plt.scatter(o_c21, o_c22)

plt.ylim(-0.00008,0.00008)
plt.xlim(-0.00008,0.00008)
plt.ylabel('X1')
plt.xlabel('X2')

plt.subplot(122)
plt.title('Transformed data')

plt.scatter(t_c11, t_c12)
plt.scatter(t_c21, t_c22)
plt.ylim(-0.0006,0.0006)
plt.xlim(-0.0006,0.0006)
```

```
plt.ylabel('Z1')
plt.xlabel('Z2')
plt.tight_layout()
plt.show()
```