MEG decoding using Riemannian Geometry and Unsupervised classification.

Alexandre Barachant

Location : Grenoble, France
Email : alexandre.barachant@gmail.com

1 Summary

The solution is composed by two classification steps. The first one is supervised and use data from the training subjects to build a generic model. This generic model will be applied on each test subjects to obtain a first estimation of the test labels. The second step is unsupervised and is applied independently on each test subjects. It will use the labels from the first step as an initialization of an iterative unsupervised algorithm, similar to a k-means clustering.

These two steps are build upon methods originally devoted to classification of P300 Evoked potential for EEG Brain-Computer Interfaces (BCI) :

- A dedicated **Spatial filtering** is applied on the data in order to increase the signal to noise ratio and reduce the dimensionality of the signal.
- Special form of the **covariance matrices** of the trials are used as features, and manipulated with tools from **Riemannian Geometry.** Indeed, covariance matrices are Symmetric and Positive-Definite Matrices (SPD), and therefore belong to a Riemannian manifold. A dedicated metric should be used in order to take into account the structure of the manifold.

2 Introduction

It is well known that EEG and MEG signals are very specific to each subject. Indeed, the organization and orientation of the cortical dipoles is slightly different for each individual. As a result, establishing a generic model, which gives high classification performance, is a very difficult task. With the exception of some special cases, a subject-specific model is always better than a generic model.

In addition, MEG signals are really noisy, and are recordings of the whole brain activity, including from parts of the brain which are not related to the task. This makes the classification task more difficult, since the unwanted activities share some common statistical properties with the signal of interest. Some of the channels contains more useful information than others, for example those around the visual cortex. However, the other channels are also useful to some

degree, since they can give insights on the nature of the noise or on non-task related activity. For this reason, the MEG/EEG community has developed methods called **spatial filtering**. Spatial filtering is an operation (usually, a linear combination of the original channels) that produce virtual channels by focusing on a particular part of the brain (or a particular activity) and discarding the irrelevant informations. Those methods could be blind, i.e. based only on statistical properties of the signal of interest, or not, by using a head model and inverse solution. Spatial filters are very specific to each subjects, and generalize badly between subjects.

For all those reasons, the wining solution should use adaptation or unsupervised training to fit the data of the test subject. Given the fact that the class are not well separated and that the signal is high dimensional, developing an efficient and stable clustering method is tricky.

During the past years, I have developed an innovative frameworks for classification of EEG signal using Riemannian Geometry [1, 2, 3]. The general idea is to use covariance matrices as features for the classification, and a Riemannian metric to manipulate them. This framework features good generalization between subjects and robustness to artefacts and bad labels (among other interesting properties). It is the perfect candidate to build an efficient unsupervised method.

3 Method

3.1 preprocessing

The original data consist in MEG recording on 306 channels sampled at 250Hz. The duration of the signal is 1.5s, and a stimulus is presented at 0.5s. The first step of preprocessing is to discard the first 0.5 second of the signal, resulting in a 1s trial. The second step is to apply a 5-order Butterworth band-pass filter between 1 and 20 Hz.

3.2 Spatial Filtering

For each class, a set of 4 spatial filters are build in order to enhance the signal to noise ratio of the evoked potential. Therefore, the resulting signal is composed by $2 \times 4 = 8$ virtual channels. The spatial filters are estimated using an algorithm derived from the xDawn algorithm [4]. Let $\mathbf{X}_i \in \Re^{C \times N}$ denotes a trial of index i, with C the number of channels and N the number of time samples, and y_i the class of this trial. Let denotes $\mathbf{P}^{(k)}$ the average trial of the class k :

$$\mathbf{P}^{(k)} = \frac{1}{|\mathcal{I}^{(k)}|} \sum_{i \in \mathcal{I}^{(k)}} \mathbf{X}_i, \tag{1}$$

where $\mathcal{I}^{(k)}$ is the set of indices of the trials belonging to the class k, i.e. $\mathcal{I}^{(k)} = \{i \mid y_i = k\}$. Let $\mathbf{X}$ be the matrix representing the whole signal, build by concatenation of all the trials (from both classes).

In this work, a spatial filter is a vector $\mathbf{w} \in \Re^{C \times 1}$. The spatial filter is estimated in order to increase the signal to noise ratio of a given class, i.e. for the class k we have :

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{P}^{(k)} \mathbf{P}^{(k)^T} \mathbf{w}}{\mathbf{w}^T \mathbf{X}\mathbf{X}^T \mathbf{w}}. \tag{2}$$

This Equation is a generalized Rayleigh quotient, the solutions can be found using an eigenvalue-eigenvector decomposition of the matrix $[(\mathbf{P}^{(k)}{\mathbf{P}^{(k)}}^T)(\mathbf{X}\mathbf{X}^T)^{-1}]$. This will give a total of C solution, ranked by the value of the eigenvalues. For each class, only the 4 best spatial filters, corresponding to the 4 highest eigenvalue, are selected.

Let denotes by $\mathbf{W}^{(k)} \in \Re^{C\times 4}$ the selected spatial filters for the class k. Since we have two classes, the total number of spatial filters is 8. The spatial filters could be aggregated in a single matrix $\mathbf{W} = [\mathbf{W}^{(0)}, \mathbf{W}^{(1)}] \in \Re^{C\times 8}$. Then, the spatial filtering operation is simply the linear projection of the trial by the matrix $\mathbf{W}$:

$$\mathbf{Z}_i = \mathbf{W}^T\mathbf{X}_i \tag{3}$$

Note that this step is supervised.

3.3 Features

Covariance matrices are used as feature. A special estimation of the covariance matrix is used in order to take into account the shape of the evoked potentials. First, we build a new trial $\tilde{\mathbf{Z}}_i \in \Re^{16\times T}$ by the concatenation of the spatially filtered average evoked potential $\mathbf{P}^{(k)}$ and the spatially filtered trial $\mathbf{Z}_i$:

$$\tilde{\mathbf{Z}}_i = \begin{bmatrix} {\mathbf{W}^{(0)}}^T\mathbf{P}^{(0)} \\ {\mathbf{W}^{(1)}}^T\mathbf{P}^{(1)} \\ \mathbf{Z}_i \end{bmatrix}. \tag{4}$$

Then, we simply estimate the spatial covariance matrix $\mathbf{\Sigma}_i \in \Re^{16\times 16}$ form the new trial $\tilde{\mathbf{Z}}_i$:

$$\mathbf{\Sigma}_i = \frac{1}{N}\tilde{\mathbf{Z}}_i\tilde{\mathbf{Z}}_i^T \tag{5}$$

Note that the features are not in the usual form of vectors, they are matrices. We can not simply vectorize these matrices, because we want to keep their special structure (SPD). Instead, we will use tool from Riemannian Geometry to manipulate them.

3.4 Riemannian Geometry

Riemannian Distance : For two covariance matrix ($\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$), the Riemannian distance according to information geometry is given by [5]

$$\delta_R(\mathbf{\Sigma}_1, \mathbf{\Sigma}_2) = \|\log\left(\mathbf{\Sigma}_1^{-1/2}\mathbf{\Sigma}_2\mathbf{\Sigma}_1^{-1/2}\right)\|_F = \left[\sum_{c=1}^{C}\log^2\lambda_c\right]^{1/2}, \tag{6}$$

where $\lambda_c, c = 1\dots C$ are the real eigenvalues of $\mathbf{\Sigma}_1^{-1/2}\mathbf{\Sigma}_2\mathbf{\Sigma}_1^{-1/2}$ and C the number of electrodes. Note that this distance is also called *Affine-invariant* [6]

Riemannian Mean : The Riemannian geometric mean of I covariance matrices (denoted by $\mathfrak{G}(.)$), also called Fréchet mean, is defined as the matrix minimizing the sum of the squared Riemannian distances [7], i.e.,

$$\mathfrak{G}\left(\mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_I\right) = \arg\min_{\mathbf{\Sigma}} \sum_{i=1}^{I} \delta_R^2\left(\mathbf{\Sigma}, \mathbf{\Sigma}_i\right). \tag{7}$$

There is no closed form expression for this mean, however a gradient descent in the manifold can be used in order to find the solution. A matlab implementation is provided in a Matlab toolbox[1].

Tangent Space Mapping : The tangent space mapping is an operation that project matrices from the manifold in a vector space named Tangent space. This tangent space is Euclidean and locally homomorphic to the manifold and Riemannian distance computations in the manifold can be well approximated by Euclidean distance computations in the tangent space. The idea is very similar to a kernel-mapping. $n \times n$ covariances matrices are represented by vectors of dimension $n(n+1)/2$ in the tangent space. For more details about how this mapping is done, please refer to [2].

3.5 Generic Model

A generic model is build in order to achieve the best classification accuracy in Leave-One-Subject-Out cross-validation (LOSO-CV). The test labels obtained with the generic model will be usedas an initialisation for the unsupervised classification step. This generic model is build as follow :

- For each of the 16 training subjects, an set of 8 spatial filters are trained. The spatial filters are applied on the 16 subjects. Special form covariances matrices are estimated and projected in the tangent space. Thus, for each subject, we have a feature subspace of dimension [136$\times$ Total Number of Trial] = 136×9414 (approximately 588 trial per subjects)
- The 16 individual features subspace are aggregated to build a new feature space. The dimension of this feature space is $16 * 136 \times 9414 = 2176 \times 9414$
- A Regularized logistic regression is trained on this feature space (function *lasso* from matlab)

This generic model gives a accuracy of 69.7% in LOSO-CV, with a public leaderboard score of 0.698 (private 0.65)

[1] http://github.com/alexandrebarachant/covariancetoolbox

3.6 Unsupervised training

For each test subjects, an unsupervised algorithm is applied, using only data from the subject at hand, and initialized with the labels from the generic model. The procedure is inspired from the k-means clustering algorithm, in the Riemannian manifold. The procedure is iterative, and stop when the convergence is reached or after 10 iterations. Lets denote by $y_{i,n}$ the label of the trial i at the iteration n. The procedure is the following :

1. Train spatial filters given the labels $y_{i,n}$.
2. Apply the spatial filters on the trials.
3. Estimate special form covariance matrices.
4. For each class, estimate the mean covariance matrix from Eq.(7). We obtain two mean covariance matrix : $\mathbf{\Sigma}^{(0)}$ and $\mathbf{\Sigma}^{(1)}$.
5. Classify each trial according to the Riemannian distance from Eq.(6) to obtain new labels :
$$y_{i,(n+1)} = \begin{cases} 0 & \text{if } \delta_R(\mathbf{\Sigma}^{(0)}, \mathbf{\Sigma}_i) < \delta_R(\mathbf{\Sigma}^{(1)}, \mathbf{\Sigma}_i) \\ 1 & \text{otherwise} \end{cases} \tag{8}$$
6. Stop if $y_{i,(n+1)} = y_{i,n}$ or if $n > 10$. Restart to step 1 with the new labels $y_{i,(n+1)}$ if not.

This second classification step allow to reach 75.8% of accuracy in LOSO-CV, with a public leaderboard score of 0.774 (0.755 private). There is a risk that the algorithm does not converge for some subject. The critical part is the initialization and it is important that the generic model gives the best possible initialization. Note that if the labels are know, this procedure could be applied in a supervised way (or semi-supervised way), with only one iteration.

4 Results

The results are given in Table 1. Single stands for the method described in section 3.6 applied in 6-fold CV using data of a single subject with known labels. This represent the maximum achievable accuracy if we have training data with labels for the subject at hand. Generic is the method described in section 3.5, i.e the generic model only. Gen + Unsup is the combination of 3.5 and 3.6, i.e. the winning submission. Note that for two subjects, marked with a star, the unsupervised method did not converge, leading to a decrease in accuracy.

5 Function and Code

The code is written in Matlab and is available here :

- https://github.com/alexandrebarachant/DecMeg2014

The code as a dependency to my covariance toolbox, available here :

Subject	Single	Generic	Gen + Unsup
1	87.0	75.5	64.3*
2	82.2	67.5	72.7
3	82.8	61.0	63.3
4	90.7	75.9	86.5
5	81.5	68.4	74.2
6	83.8	67.6	73.8
7	88.7	69.7	79.9
8	88.8	68.2	76.6
9	89.9	69.5	79.6
10	86.7	71.5	78.9
11	76.0	65.5	59.9*
12	85.4	78.5	83.7
13	85.8	68.2	73.4
14	91.6	74.8	87.0
15	91.3	73.9	87.0
16	87.1	58.8	72.5
Mean (std)	**86.2** (4.2)	**69.7** (5.3)	**75.8** (8.3)
public LB	NA	69.8	77.8
private LB	NA	65.0	75.5

Table 1: Classification accuracy for the different methods. The $*$ corresponds to the subjects where the usupervised method did not converge.

- https://github.com/alexandrebarachant/covariancetoolbox

There is 2 scripts and 4 functions. The scripts :

- **preprocessing.m** : This script reads the raw data from the data folder, applies the preprocessing steps described in section 3.1, and saves the preprocessed data in the preproc folder.
- **final_submission.m** : This script contains all the steps to generate the final solution.

The 4 function are :

- **trainfilter.m** : This function generates the spatial filters and the averaged evoked potentials.
- **applyfilter.m** : Apply the filters and return a spatially filtered version of the trials.
- **extract_features.m** : extract the features for the generic model.
- **mdmfilter.m** : this function contains all the steps for 1 iteration of the unsupervised classification.

6 How to generate the solution

This code has been developed using a 16GB, intel i7-3632QM, linux laptop. The code is not optimized in terms of memory usage and can require up to 16GB of RAM. There is two dependency : my covariance toolbox (for all the riemannian geometry related functions) and the statistical toolbox from matlab (for the function "lasso"). To generate the solution :

1. Download and install the covariance toolbox[2]
2. Download the code[3]
3. Place data from the competition in the "data" folder.
4. Run the preprocessing script : 'run preprocessing.m'
5. Run the final submission script : 'run final_submission.m'

7 Additional information

As mentioned in the introduction, the best model is the model derived from the data of the subject itself. At the beginning of the competition, I focus on developing the best possible method for single subject classification. I came up pretty quickly to the solution given in the function "mdmfilter.m". I have tried to keep thing as simple as possible to be able to implement an unsupervised procedure with this model. Then I have run some simulation to evaluate the behaviour of the unsupervised classification. It turns out that the method was robust to missclassified labels, and if you provide a good initialization (in the range of 65-70% of accuracy), you have a very good chance of improving your results. The main challenge was to provide this good initialization. I tried several solutions, with very similar results (including more classical methods without Riemannian geometry). The generic model presented in this document was the best i could find in this limited time window. But there is room for improvement, for example by using covariate shift on the tangent space.

For the generic model, I was not very happy to use spatial filtering. Spatial filtering generalize badly between subjects, and usually leads to a loss of information. But the Riemannian geometry frameworks can't be applied with a high number of electrodes. I developed method using electrodes selection or by combining classifier on random sub-set of electrodes. One of these methods gives very good results (0.766 on the private LB), but wasn't very stable.

There is probably room for improvement, for example by applying my unsupervised procedure with an initialization provided by the methods of other competitors.

[2] https://github.com/alexandrebarachant/covariancetoolbox
[3] https://github.com/alexandrebarachant/DecMeg2014

References

[1] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten, "Multiclass Brain-Computer interface classification by riemannian geometry," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 4, pp. 920–928, 2012.

[2] ——, "Classification of covariance matrices using a Riemannian-based kernel for bci applications," *Neurocomputing*, vol. 112, pp. 172–178, 2013.

[3] M. Congedo, A. Barachant, and A. Andreev, "A new generation of brain-computer interface based on riemannian geometry," *Arxiv*, 2013.

[4] B. Rivet, A. Souloumiac, V. Attina, and G. Gibert, "xdawn algorithm to enhance evoked potentials: application to brain–computer interface," *Biomedical Engineering, IEEE Transactions on*, vol. 56, no. 8, pp. 2035–2043, 2009.

[5] M. Moakher, "A differential geometric approach to the geometric mean of symmetric Positive-Definite matrices," *SIAM J. Matrix Anal. Appl.*, vol. 26, no. 3, pp. 735–747, 2005.

[6] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Geometric means in a novel vector space structure on symmetric positive-definite matrices," *SIAM journal on matrix analysis and applications*, vol. 29, no. 1, pp. 328–347, 2007.

[7] X. Pennec, P. Fillard, and N. Ayache, "A riemannian framework for tensor computing," *International Journal of Computer Vision*, vol. 66, no. 1, pp. 41–66, 2006.