Riemannian geometry for time series analysis

V. Goncharenko, A. Samokhina, K. Durdymyradov

Moscow, 2022

girafe ai

Outline

1. Time series
    a. Time series
    b. EEG as time series example
    c. Classification problem
2. Riemannian geometry
    a. Quick intro
    b. Why riemann
    c. Useful terms
    d. How to use
3. PyRiemann library
    a. API
    b. Covariances
    c. Metrics
    d. Contents
4. Use case

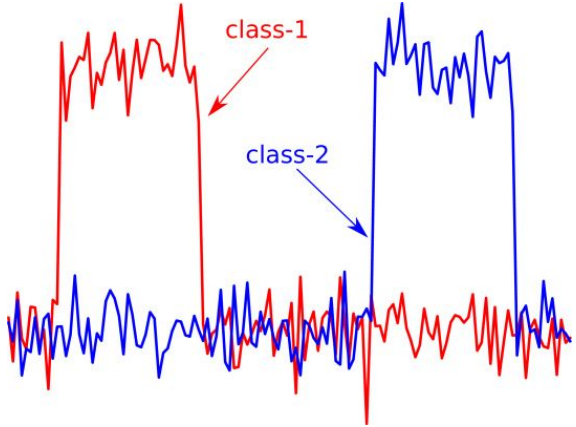
Time series

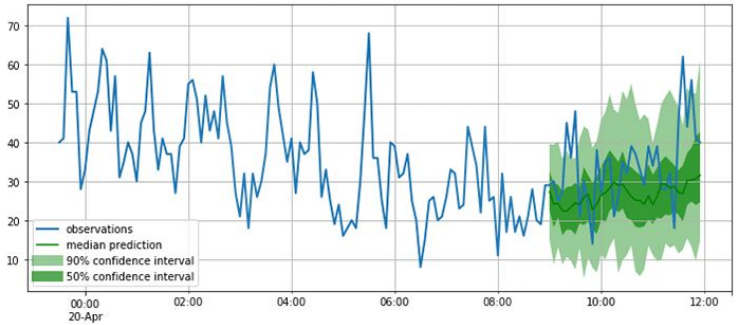
girafe ai

01

Time series

- Practically, every data set that has an order in time is time series data set:
  - sales history, stock prices
  - meteorology data
  - equipment monitoring
  - signals (EEG, ECG, MEG)
- Problems:
  - Forecasting
  - **Classification**





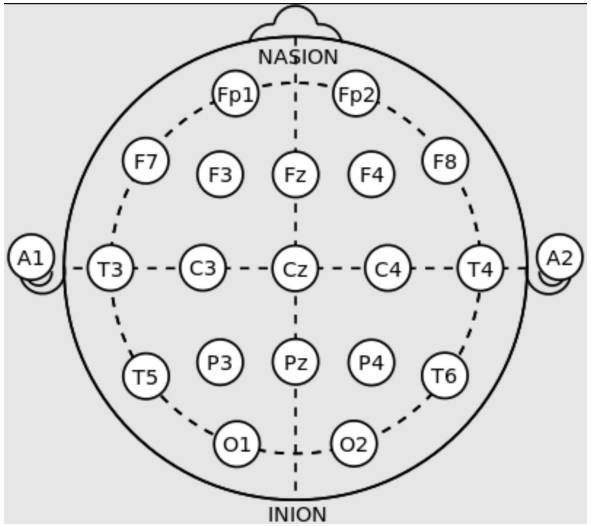
EEG as time series example

EEG - electroencephalogram, detects electrical activity (signals) in your brain

Brain-Computer interfaces (e.g. Speller for disabled people)

Medical diagnosis (ERP and Alzheimer's disease)

Games (neiry-games)



girafe ai

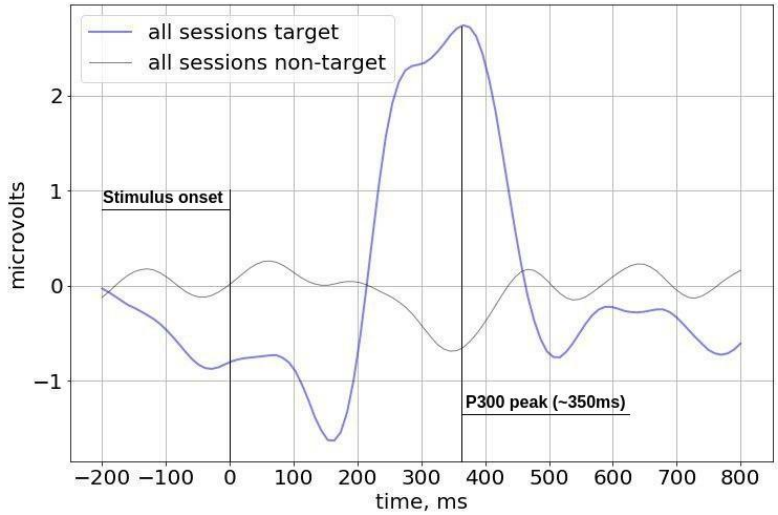
Image was taken from here

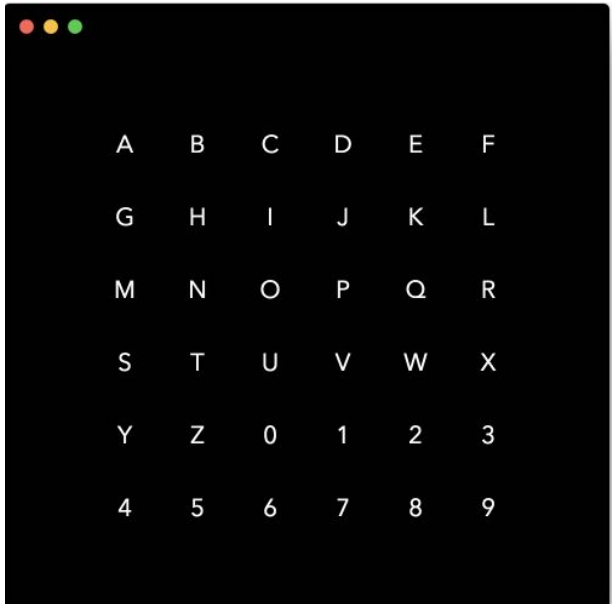
Classification - ERP concept

Event related potential (ERP) — electrophysiological response in very small voltages generated in the brain structures in response to specific events or stimuli.

Stimuli can be:

- visual
- auditory
- tactile





girafe ai

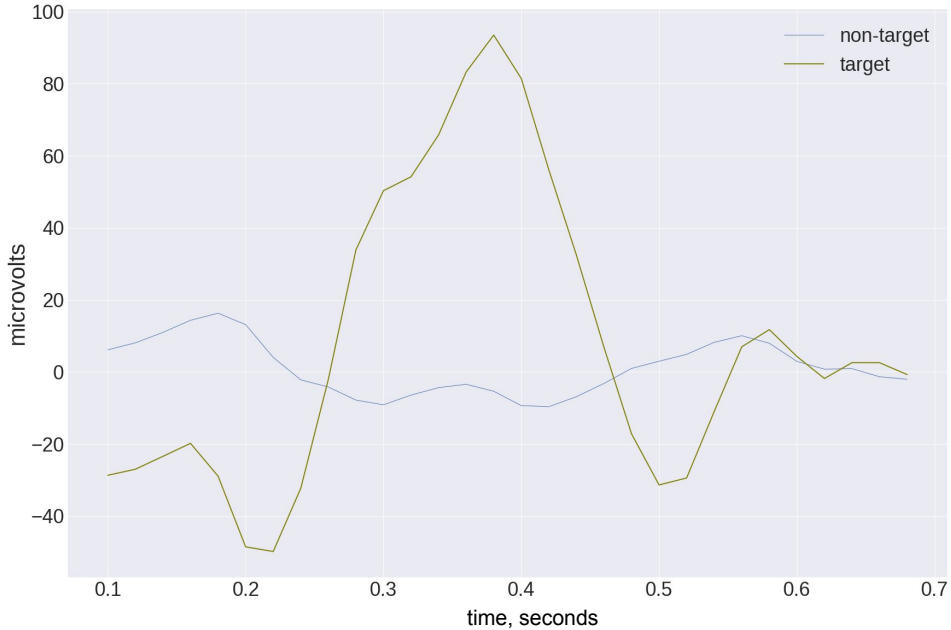
ERP terms

Epoch — EEG segment from stimulus onset. Duration may vary.

Target epoch — epoch with potential evoked by stimulus

Non-target epoch — epoch where nothing happened

We want to learn to distinguish them from each other.



Time series covariance matrix

Let's denote EEG epochs as: $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots \mathbf{X}_N\}, \mathbf{X}_i \in \mathbb{R}^{E \times T} \; \forall i \in \{1, N\}$

N — number of epochs
E — number of electrodes
T — number of sampled time points

If we have multichannel time series, then matrix form of signal:

$$\mathbf{X}_i = \begin{bmatrix} X^i_{1,1} & X^i_{1,2} & \dots & X^i_{1,T} \\ \dots & \dots & \dots & \dots \\ X^i_{E,1} & X^i_{E,2} & \dots & X^i_{E,T} \end{bmatrix} = \begin{bmatrix} t_1 \\ \dots \\ t_E \end{bmatrix}$$

Time series covariance matrix

Covariance matrix is a square matrix giving the covariance between each pair of elements of a given random vector.
Covariance - is a measure of how much two random variables vary together.

Then the covariance matrix looks like this:

$$Var(\boldsymbol{x}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

$$\text{Cov}(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{(N-1)}$$

$$\text{Cov}(\mathbf{X}_i) = \begin{bmatrix} \text{Var}(t_1) & \text{Cov}(t_1, t_2) & \dots & \text{Cov}(t_1, t_E) \\ \text{Cov}(t_1, t_2) & \text{Var}(t_2) & \dots & \text{Cov}(t_1, t_E) \\ \dots & \dots & \dots & \dots \\ \text{Cov}(t_E, t_1) & \text{Cov}(t_E, t_2) & \dots & \text{Var}(t_E) \end{bmatrix}$$

Riemannian geometry

girafe ai

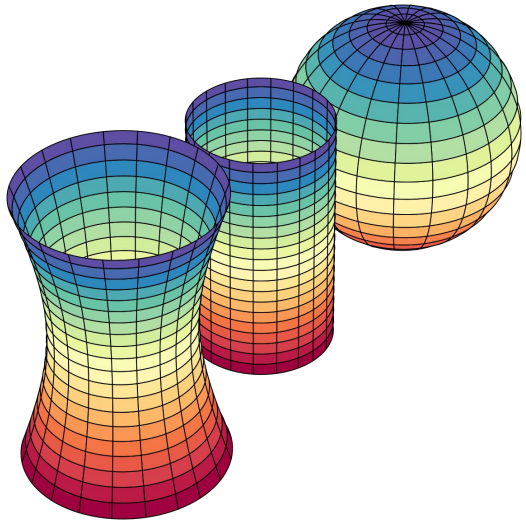
02

Riemannian geometry

Riemannian geometry is the branch of differential geometry that studies Riemannian manifolds, smooth manifold **M** equipped with a positive-definite inner product **g_p** on the tangent space **TpM** at each point **p**.

Riemannian metric is the family **g_p** of inner products.

Tangent space of a manifold generalizes to higher dimensions the notion of tangent planes to surfaces in three dimensions and tangent lines to curves in two dimensions



girafe ai

If you want more: there is a short introduction and a very big book on geometry.

Why Riemannian geometry

- Space of symmetric positive definite (SPD) matrices forms a Riemannian manifold.
  In time series world we have covariance matrices which are SPD matrices.

- Since covariance matrices are SPD, the correct manipulation of these matrices relies on the Riemannian geometry.

Manifolds and tangent space

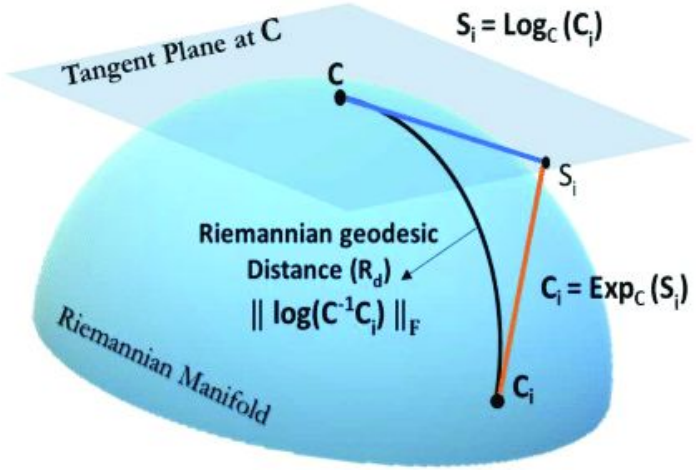
At every point **C** of the manifold we have a tangent plane with an inner product, depending on that point.

For each covariance matrix $\mathbf{C}_i$ we have its tangent vector $\mathbf{S}_i$

Mapping between $\mathbf{C}_i$ and $\mathbf{S}_i$ looks like this:

$$\mathrm{Exp}_{\mathbf{C}}\left(\mathbf{S}_i\right) = \mathbf{C}_i = \mathbf{C}^{\frac{1}{2}} \exp\left(\mathbf{C}^{-\frac{1}{2}} \mathbf{S}_i \mathbf{C}^{-\frac{1}{2}}\right) \mathbf{C}^{\frac{1}{2}}$$

$$\mathrm{Log}_{\mathbf{C}}\left(\mathbf{C}_i\right) = \mathbf{S}_i = \mathbf{C}^{\frac{1}{2}} \log\left(\mathbf{C}^{-\frac{1}{2}} \mathbf{C}_i \mathbf{C}^{-\frac{1}{2}}\right) \mathbf{C}^{\frac{1}{2}}$$



girafe ai

Image was taken from this paper, explained well in this paper

How to choose the reference point

To project covariance matrices to tangent space we have to choose one reference point.

The best choice would be geometric mean of our matrices.

The Riemannian (geometric) mean of SPD matrices is given by

$$C = \mathfrak{G}\left(\mathbf{C}_1, \ldots, \mathbf{C}_I\right) = \underset{\mathbf{C} \in C(n)}{\arg\min} \sum_{i=1}^{I} \delta_R^2\left(\mathbf{C}, \mathbf{C}_i\right)$$

Where the metric is riemannian geodesic distance.

$$\delta_R\left(\mathbf{C}_1, \mathbf{C}_2\right) = \left\|\log\left(\mathbf{C}_1^{-1}\mathbf{C}_2\right)\right\|_F = \left[\sum_{i=1}^{n} \log^2 \lambda_i\right]^{1/2}$$

λ_i — are the real eigenvalues of $\mathbf{C}_1^{-1}\mathbf{C}_2$

- **Geodesic distance** - the curve connecting two points that has the smallest length.

girafe ai

To dive into details

Available metrics

In this package we have some pretty simple classification algorithms, e.g. Minimum Distance to Mean. It can be interesting due to different metrics we can use to measure distance.

- Euclidean $d = \|A - B\|_F$
- Log Euclidean $d = \|\log(A) - \log(B)\|_F$
- Log-det $d = \sqrt{(\log(\det(\frac{A+B}{2})) - 0.5 \times \log(\det(A)\det(B)))}$
- Riemannian $d = (\sum_i \log(\lambda_i)^2)^{-1/2}$
- Wasserstein $d = (tr(A + B - 2(A^{1/2}BA^{1/2})^{1/2}))^{1/2}$
- Symmetrized kullback leibler divergence $d = D_{KL}(A, B) + D_{KL}(B, A)$

Mean covariance estimation

Reference point and mean covariance matrices for ERP Covariances can be computed using different metrics, such as:

- riemannian $C = \arg\min_C \sum_i \delta_R^2 (C, C_i)$
- log-euclidean $C = \exp(\frac{1}{N} \sum_i \log C_i)$
- euclidean $C = \frac{1}{N} \sum_i C_i$
- wasserstein $K = (\sum_i (K C_i K)^{1/2})^{1/2},\ K = C^{1/2}$
- there are others (e.g. harmonic, symmetrized kullback), read the docs

However, the experiments show that, it is crucial to process covariance matrices with dedicated Riemannian tools, impacting the efficiency of the classification.

What to do next

What we have done so far:

- for each epoch we got covariance matrix
- got a tangent space at the reference point
- projected each covariance matrix to this tangent space

In the end we have vector representation of EEG epochs.

And now we take these vectors as features and... just do the classification.

PyRiemann

girafe ai

03

Scikit-learn interface

```python
from pyriemann.estimation import ERPCovariances
from pyriemann.tangentspace import TangentSpace
from sklearn.linear_model import LogisticRegression

def get_classifier():
    lr_params = {
        'solver': 'liblinear',
        'max_iter': 1000,
        'C': 1.0,
        'class_weight': 'balanced',
        'penalty': 'l1',
    }
    return make_pipeline(
        ERPCovariances(estimator='oas'),
        TangentSpace(),
        LogisticRegression(**lr_params),
    )
```

ERPCov TS LR :

- Estimation of ERP covariances
- Projection to the tangent space
- Classification with logistic regression

Covariance estimation

- **Covariance estimation**

  Perform a simple covariance matrix estimation for each given input

  `pyriemann.estimation.`**`Covariances`**`(`***`estimator='scm'`***`)`

  $$\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots \mathbf{X}_N\}, \mathbf{X}_i \in \mathbb{R}^{E \times T} \; \forall i \in \{1, N\}$$

  Input:
  X [ndarray, shape (n_matrices, n_channels, n_times)] Multi-channel time-series.
  Transformation:
  Estimate covariance matrices of time-series
  Output:
  covmats [ndarray, shape (n_matrices, n_channels, n_channels)] Covariance matrices.

  For more detail.

Covariance estimation

- **ERP Covariance estimation**

`pyriemann.estimation.ERPCovariances`

Estimation of special form covariance matrix dedicated to ERP processing. For target class, a prototyped response is obtained by average across trial :

$$\mathbf{P} = \frac{1}{N}\sum_{i}^{N}\mathbf{X}_i$$

and a super trial is build using the concatenation of P and the trial X :

$$\tilde{\mathbf{X}}_i = \begin{bmatrix} \mathbf{P} \\ \mathbf{X}_i \end{bmatrix}$$

This super $\tilde{\mathbf{X}}_i$ will be used for covariance estimation. This allows to take into account the time structure of the signal.only the covariance matrix for the classification leads to the loss of the information carried by the temporal structure of the ERP. For more detail

Covariance estimation

XDawn covariance:

`pyriemann.estimation.XdawnCovariances`

Estimate special form covariance matrix for ERP combined with Xdawn
This is similar to ERPCovariances but data are spatially filtered with Xdawn

The advantage of this estimation is to reduce dimensionality of the covariance matrices efficiently.

XDawn maximizes the signal to signal + noise ratio using signal projection's first N components.

Covariance estimation

XDawn covariance:

The data filtered by this filter:

$$\tilde{X} = XW$$

Where:

$X \in R^{n \times d}$ - The EEG signal

$$\widehat{W} = \underset{w}{\operatorname{argmax}} \frac{\operatorname{Tr}\left(W^T \hat{A}^T D^T D \hat{A} W\right)}{\operatorname{Tr}\left(W^T X^T X W\right)}$$

$$\widehat{A} = \underset{A}{\operatorname{argmin}} \|X - DA\|_2^2 = \left(D^T D\right)^{-1} D^T X$$

$D \in R^{e \times n}$ - The position of the P300 component in the signal

Tangent Space

class `pyriemann.tangentspace.`**`TangentSpace`**

Tangent space projection map a set of covariance matrices to their tangent space. After projection, each matrix is represented as a vector of size N(N+1)/2 where N is the dimension of the covariance matrices.

Tangent space projection is useful to convert covariance matrices in euclidean vectors while conserving the inner structure of the manifold. After projection, standard processing and vector-based classification can be applied.

Input:

X [ndarray, shape (n_trials, n_channels, n_channels)] ndarray of SPD matrices.

Transformation:

- Estimates the reference point(by using geometric mean)
- By using this reference point convert matrix to vector

$$\mathbf{S}_i = \mathbf{C}^{\frac{1}{2}} \log\left(\mathbf{C}^{-\frac{1}{2}} \mathbf{C}_i \mathbf{C}^{-\frac{1}{2}}\right) \mathbf{C}^{\frac{1}{2}}$$

Output:

ts [ndarray, shape (n_trials, n_ts)] the tangent space projection of the matrices.

For more detail.

Use case

girafe ai

04

Scikit-learn interface

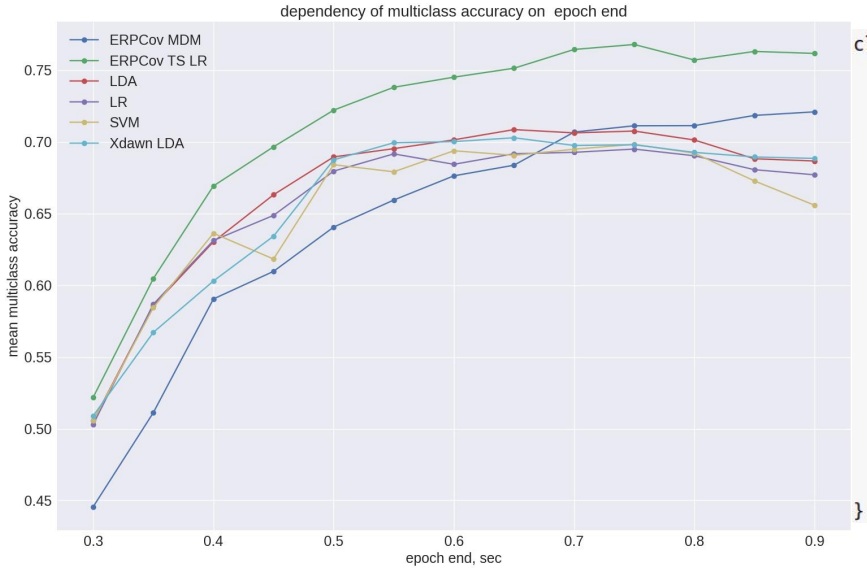
```python
from pyriemann.estimation import ERPCovariances
from pyriemann.tangentspace import TangentSpace
from sklearn.linear_model import LogisticRegression

def get_classifier():
    lr_params = {
        'solver': 'liblinear',
        'max_iter': 1000,
        'C': 1.0,
        'class_weight': 'balanced',
        'penalty': 'l1',
    }
    return make_pipeline(
        ERPCovariances(estimator='oas'),
        TangentSpace(),
        LogisticRegression(**lr_params),
    )
```

ERPCov TS LR :

- Estimation of ERP covariances
- Projection to the tangent space
- Classification with logistic regression

Comparison to other algorithms

ERPCov TS LR outperforms the best non-riemannian classifier
by more than 5% in multiclass accuracy




```
clfs = {
    'LR': (make_pipeline(Vectorizer(), LogisticRegression()),
        {'logisticregression__C': np.exp(np.linspace(-4, 4, 5))},),
    'LDA': (make_pipeline(Vectorizer(), LDA(shrinkage='auto', solver='eigen')),
        {},),
    'SVM': (make_pipeline(Vectorizer(), SVC()),
        {'svc__C': np.exp(np.linspace(-4, 4, 5)),
         'svc__kernel': ('linear', 'rbf')},),
    'CSP LDA': (make_pipeline(CSP(), LDA(shrinkage='auto', solver='eigen')),
        {'csp__n_components': (6, 9, 13), 'csp__cov_est': ('concat', 'epoch')},),
    'Xdawn LDA': (make_pipeline(Xdawn(2, classes=[1]), Vectorizer(),
                                LDA(shrinkage='auto', solver='eigen')),
        {},),
    'ERPCov TS LR': (make_pipeline(ERPCovariances(estimator='oas'),
                                   TangentSpace(), LogisticRegression()),
        {'erpcovariances__estimator': ('lwf', 'oas')},),
    'ERPCov MDM': (make_pipeline(ERPCovariances(), MDM()),
        {'erpcovariances__estimator': ('lwf', 'oas')},),
}
```

[P300 classifier using open data medium post]

girafe ai

Summary

1. Time series
2. Riemannian geometry
3. PyRiemann library
4. Use case

If you classify time series -

Riemannian geometry is a must

Thanks for attention!

Questions? Additions? Welcome!

---

girafe
ai