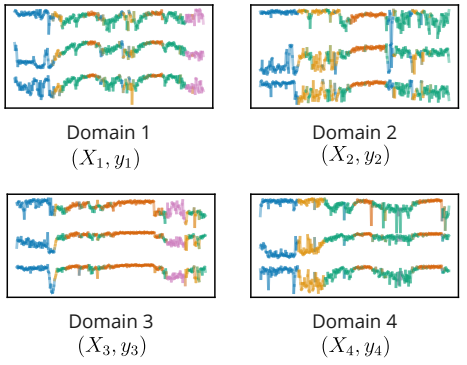
Multi-Source and Test-Time Domain Adaptation on Multivariate Signals using Spatio-Temporal Monge Alignment

Théo Gnassounou, Antoine Collas, Rémi Flamary, Karim Lounici, Alexandre Gramfort

Huawei Seminar, 19-03-2025

Sleep stage classification from EEG signals

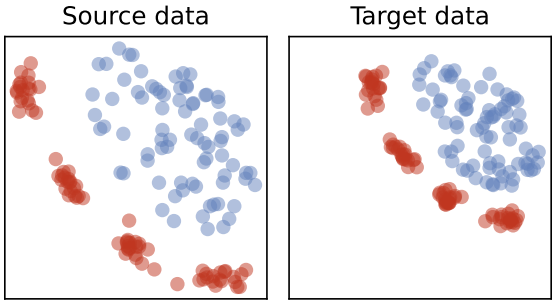
- **Multi-source** EEG signals from subjects/hospitals
- **Target** EEG signals from a new subject/hospital
- **No** access to the target labels
- **Shift** between the domains



$\rightarrow$ **Goal:** **Adapt** the source to the target to **classify** the target signals

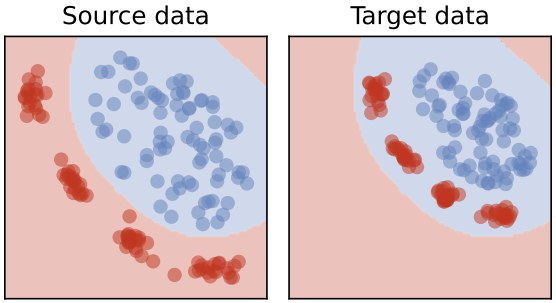
What is Domain Adaptation (DA)?

- Two type of domains: **Source** and **Target** .
- **Source** domains **with label** and **Target** domains **without label** .
- Assumption $\rightarrow$ **shift** between the distribution of the domain's data



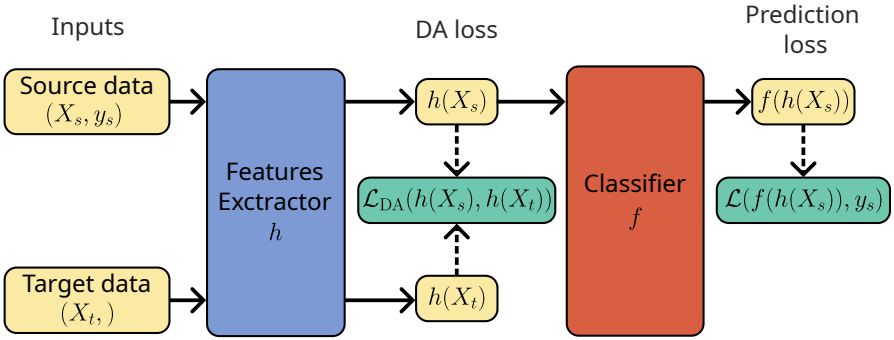
What is Domain Adaptation (DA)?

- Two type of domains: **Source** and **Target** .
- **Source** domains **with label** and **Target** domains **without label** .
- Assumption → **shift** between the distribution of the domain's data



→ **Problem:** **Drop in performance** when applying a model trained on the source to the target.

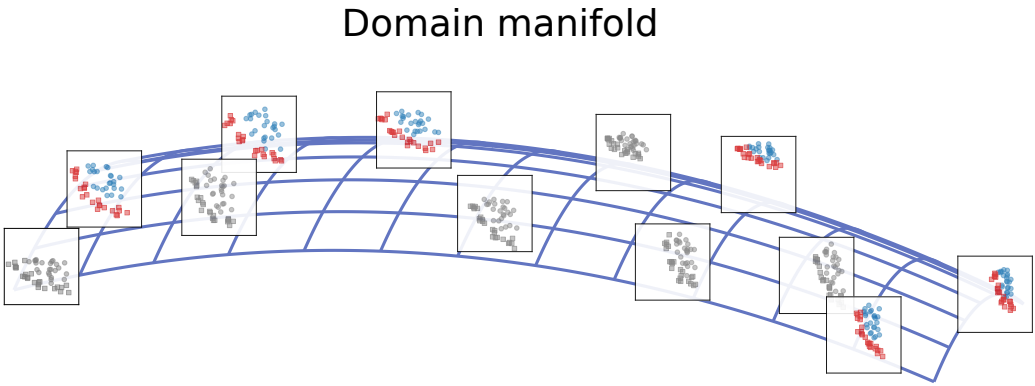
Traditional DA methods: Deep Learning



- Reduce the **divergence** between the source and target features with:
  - Correlation Alignment[1]
  - Domain Adversarial Neural Network[2]
  - Joint Distribution Optimal Transport[3]
  - ...

[1] *Sun et. al., 2016* [2] *Ganin et. al., 2016* [3] *Damodaran et. al., 2018*

Multi-source multi-target Domain Adaptation



$\rightarrow$ Mutliple **subjects** and **hospitals** with **different** EEG signals

Source-free Domain Adaptation (or Test-Time DA)

1. Train-time

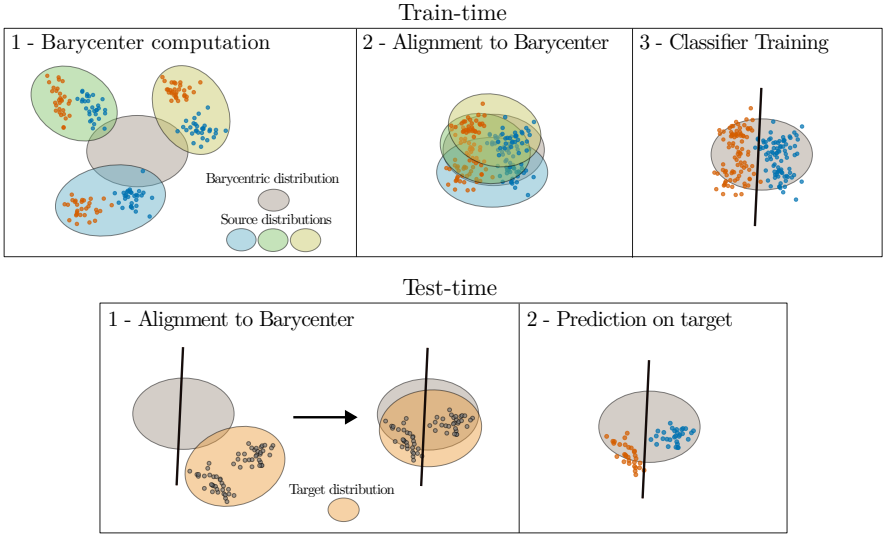
- Acces to **Source** domains with labels
- **No** access to **Target** domains
- **Train** a model on the source domains with labels

2. Test-time

- **No** access to **Source** domains
- Acces to **Target** domains **without** labels
- **Finetune** the model on the target domains without access to the target labels

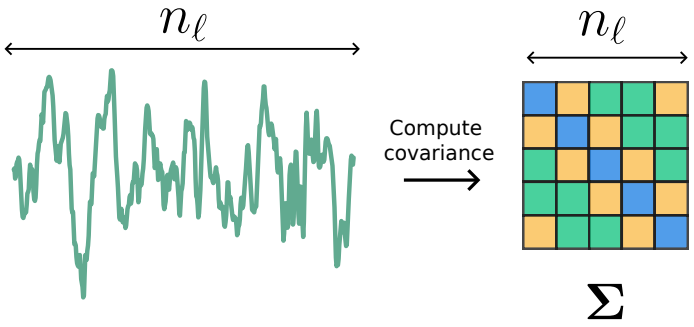
$\rightarrow$ Practionners **only** have access to the target data at test-time

Distribution alignment to barycenter to tackle domain shift



Assumptions on the signals

- Centered **Gaussian** distributions $\rightarrow \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ with $\mathbf{\Sigma} \in \mathcal{S}_{n_\ell}^{++}$
- $\mathbf{\Sigma}$ is the "auto-covariance", computed with time-lagged. $\mathbf{\Sigma}_{i,j} = \mathbf{X}_i \mathbf{X}_j$
- **Stationarity+Periodicity** $\rightarrow$ Covariance matrices are **Toeplitz circulant** matrices.

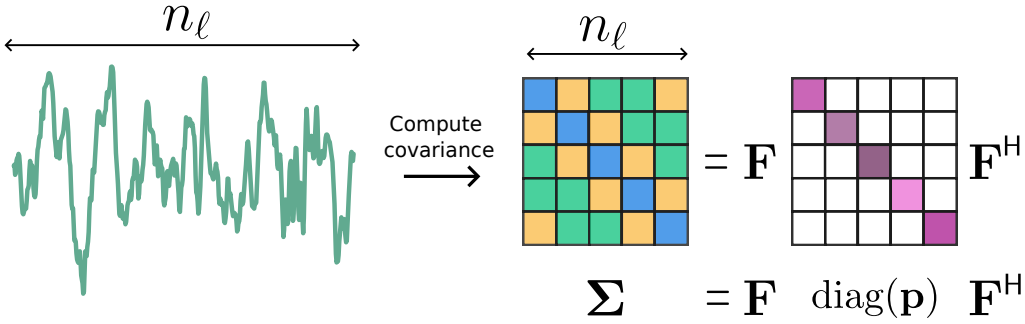


Assumptions on the signals

The Discrete Fourier Transform (DFT) can diagonalize the circulant matrix

$$\mathbf{\Sigma} = \mathbf{F}\mathrm{diag}(\mathbf{p})\mathbf{F}^* \ ,$$

with $\mathbf{F}$ and $\mathbf{F}^*$ the Fourier transform operator and its inverse, and $\mathbf{p}$ the Power Spectral Density (PSD) of the signal.



Monge mapping for Gaussian distributions

Let consider Gaussian distributions $\mu_d = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_d)$ with $d \in \{s, t\}$. The OT cost, also called the **Bures-Wasserstein distance** when using a quadratic ground metric, is

$$\mathcal{W}_2^2(\mu_s, \mu_t) = \mathrm{Tr}\left(\boldsymbol{\Sigma}_s + \boldsymbol{\Sigma}_t - 2\left(\boldsymbol{\Sigma}_t^{\frac{1}{2}}\boldsymbol{\Sigma}_s\boldsymbol{\Sigma}_t^{\frac{1}{2}}\right)^{\frac{1}{2}}\right) . \tag{1}$$

The OT mapping, also called **Monge mapping** , can be expressed as the following affine function :

$$m(\mathbf{x}) = \mathbf{A}\,\mathbf{x}, \quad \text{with} \quad \mathbf{A} = \boldsymbol{\Sigma}_s^{-\frac{1}{2}}\left(\boldsymbol{\Sigma}_s^{\frac{1}{2}}\,\boldsymbol{\Sigma}_t\,\boldsymbol{\Sigma}_s^{\frac{1}{2}}\right)^{\frac{1}{2}}\boldsymbol{\Sigma}_s^{-\frac{1}{2}} = \mathbf{A}^{\mathsf{T}} . \tag{2}$$

Map

Target Covariance

Source Covariance

Wasserstein barycenter between Gaussian distributions

Considering multiple Gaussian distributions μ_k. The barycenter $\bar{\mu}$ is expressed as

$$\bar{\mu} = \arg\min_{\mu} \frac{1}{K} \sum_{k=1}^{K} \mathcal{W}_2^2(\mu, \mu_k) \, . \tag{3}$$

Wasserstein barycenter between Gaussian distributions

Considering multiple Gaussian distributions μ_k. The barycenter $\bar{\mu}$ is expressed as

$$\bar{\mu} = \arg\min_{\mu} \frac{1}{K} \sum_{k=1}^{K} \mathcal{W}_2^2(\mu, \mu_k) \, . \tag{3}$$

The barycenter is still a Gaussian distribution $\bar{\mu} = \mathcal{N}(\mathbf{0}, \bar{\mathbf{\Sigma}})$.

Wasserstein barycenter between Gaussian distributions

Considering multiple Gaussian distributions μ_k. The barycenter $\bar{\mu}$ is expressed as

$$\bar{\mu} = \arg\min_{\mu} \frac{1}{K} \sum_{k=1}^{K} \mathcal{W}_2^2(\mu, \mu_k) \,. \tag{3}$$

The barycenter is still a Gaussian distribution $\bar{\mu} = \mathcal{N}(\mathbf{0}, \bar{\mathbf{\Sigma}})$.

$\Longrightarrow$ **No closed-form** for computing the covariance $\bar{\mathbf{\Sigma}}$.

Wasserstein barycenter between Gaussian distributions

Considering multiple Gaussian distributions μ_k. The barycenter $\bar{\mu}$ is expressed as

$$\bar{\mu} = \arg\min_{\mu} \frac{1}{K} \sum_{k=1}^{K} \mathcal{W}_2^2(\mu, \mu_k) \,. \tag{3}$$

The barycenter is still a Gaussian distribution $\bar{\mu} = \mathcal{N}(\mathbf{0}, \bar{\mathbf{\Sigma}})$.

$\Longrightarrow$ **No closed-form** for computing the covariance $\bar{\mathbf{\Sigma}}$.

One uses the following optimality condition from[1]:

$$\bar{\mathbf{\Sigma}} = \frac{1}{K} \sum_{k=1}^{K} \left(\bar{\mathbf{\Sigma}}^{\frac{1}{2}} \mathbf{\Sigma}_k \bar{\mathbf{\Sigma}}^{\frac{1}{2}} \right)^{\frac{1}{2}} , \tag{4}$$

Barycenter Covariance — Domain k Covariance

[1] *Agueh et. al., 2011*

Previoulsy for Univariate Gaussian stationary signals

- **Stationary** Gaussian signals with **PSD** $\mathbf{p}_s$ and $\mathbf{p}_t$
- **Monge mapping** between the two **univariate** signals
- After **diagonalization** of the covariance matrix, the mapping is expressed as a **convolution** [1]:

$$m(\mathbf{x}) = \underbrace{\mathbf{h}}_{\text{Filter}} * \mathbf{x}\,, \quad \text{with} \quad \mathbf{h} = \mathbf{F}^* \left(\underbrace{\mathbf{p}_t}_{\text{Target PSD}}{}^{\odot \frac{1}{2}} \odot \underbrace{\mathbf{p}_s}_{\text{Source PSD}}{}^{\odot -\frac{1}{2}} \right) . \tag{5}$$

[1] *Flamary et. al., 2018*

Wasserstein barycenter between Gaussian stationary signals

Lemma from[1]

Consider K centered stationary Gaussian signals of PSD $\mathbf{p}_k$ with $k \in [K]$, the Wasserstein barycenter of the K signals is a centered stationary Gaussian signal of PSD $\bar{\mathbf{p}}$ with:

$$\bar{\mathbf{p}} = \left(\frac{1}{K} \sum_{k=1}^{K} \mathbf{p}_k^{\odot \frac{1}{2}} \right)^{\odot 2} . \tag{6}$$

Barycenter PSD

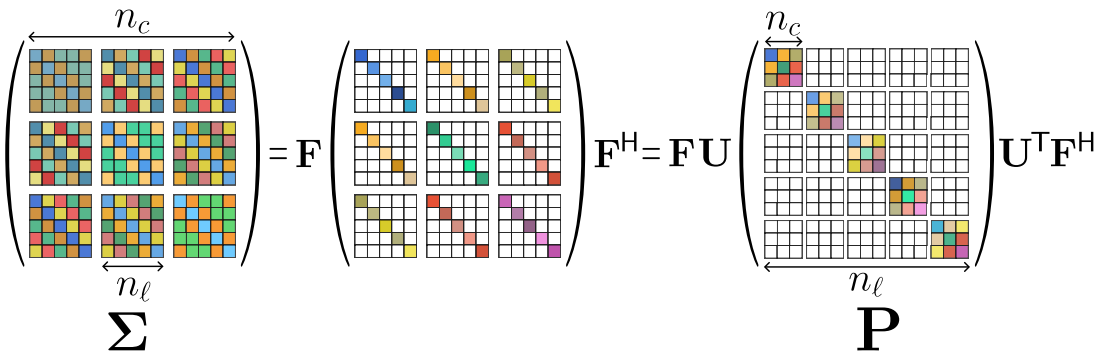
Domain k PSD

Sketch of proof: the proof directly applies the optimality condition (7) of the barycenter. With factorized covariances, the matrix square root and the inverse can be simplified as element-wise square root and inverse.

[1] *Gnassounou et. al., 2023*

New assumption on the signals

- **Multivariate** signals with cross covariance $\mathbf{\Sigma} \in \mathcal{S}^{++}_{n_\ell n_c}$
- Reduction of parameters from $(\mathbf{n_\ell} \times \mathbf{n_c})^2 \rightarrow \mathbf{n_\ell} \times \mathbf{n_c^2}$
- **P** is the cross-PSD matrix of the signal



Monge mapping for Multivariate Gaussian stationary signals

- Let consider Gaussian distributions $\mu_d = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_d)$ with $d \in \{s, t\}$ and $\mathbf{\Sigma}_d = \mathbf{F}\mathbf{U}\mathbf{Q}_d\mathbf{U}^\mathsf{T}\mathbf{F}^\mathsf{H}$

$$\begin{pmatrix} \mathrm{diag}(\mathbf{p}_{1,1}) & \dots & \mathrm{diag}(\mathbf{p}_{1,n_c}) \\ \dots & \dots & \dots \\ \mathrm{diag}(\mathbf{p}_{n_c,1}) & \dots & \mathrm{diag}(\mathbf{p}_{n_c,n_c}) \end{pmatrix} \triangleq \mathbf{U}\mathbf{P}_s^{-\frac{1}{2}} \left(\mathbf{P}_s^{\frac{1}{2}} \mathbf{P}_t \mathbf{P}_s^{\frac{1}{2}} \right)^{\frac{1}{2}} \mathbf{P}_s^{-\frac{1}{2}} \mathbf{U}^\mathsf{T} \in \mathcal{H}_{n_c f}^{++}$$

Target Cross-PSD

Source Cross-PSD

- Given a signal $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_{n_c}]^\mathsf{T} \in \mathbb{R}^{n_c \times n_\ell}$ the Monge mapping is a sum of convolutions

$$m(\mathbf{X}) = \left[\sum_{j=1}^{n_c} \mathbf{h}_{1,j} * \mathbf{x}_j, \dots, \sum_{j=1}^{n_c} \mathbf{h}_{n_c,j} * \mathbf{x}_j \right]^\mathsf{T}$$

where $\mathbf{h}_{i,j} = \frac{1}{\sqrt{f}} \mathbf{F}^\mathsf{H} \mathbf{p}_{i,j} \in \mathbb{R}^f$.

Filter between sensor i and j

Wasserstein barycenter for Multivariate Gaussian stationary signals

- No more closed form for the barycenter covariance $\bar{\mathbf{\Sigma}}$.
- The barycenter PSD $\bar{\mathbf{P}}$ is expressed as

$$\bar{\mathbf{P}} = \frac{1}{K}\sum_{k=1}^{K}\left(\bar{\mathbf{P}}^{\frac{1}{2}}\,\mathbf{P}\,\bar{\mathbf{P}}^{\frac{1}{2}}\right)^{\frac{1}{2}}, \tag{7}$$

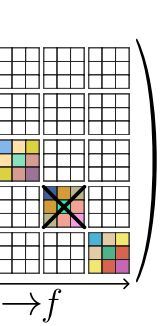
Barycenter cross-PSD

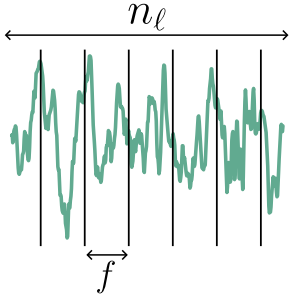
Domain k cross-PSD

How to reduce the the number of parameters of the filter?

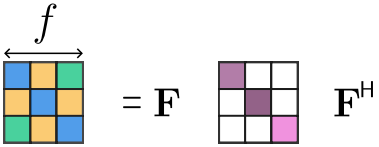
- $n_\ell \times n_c^2$ parameters for $\mathbf{h}$ still highlight
- $\mathbf{n_\ell \to f}$
- Use **Welch** method to estimate the PSD
  - Cut the signal into segments
  - Compute the PSD of each segment
  - Average the PSD

$$\mathbf{P} = \left(\quad \right)$$





Compute covariance $\longrightarrow$



$$\mathbf{\Sigma} = \mathbf{F} \ \mathrm{diag}(\mathbf{p}) \ \mathbf{F}^{\mathsf{H}}$$

Convolutional Monge Mapping Normalization (CMMN)

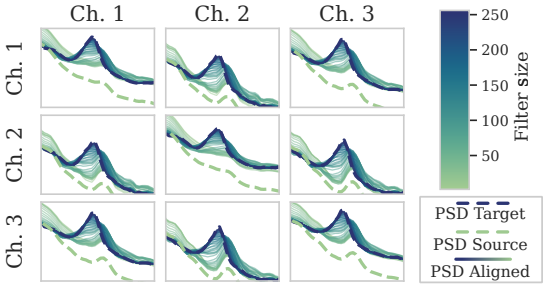
1. Train-time: Access to source domains

- Compute **cross-PSD** $\hat{\mathbf{P}}_k$ for each source domain.
- Compute **barycenter** $\hat{\bar{\mathbf{p}}}$ with the PSD $\hat{\mathbf{P}}_k$.
- Compute the **convolutional filters h**
- **Train a predictor** g on the normalized source data.

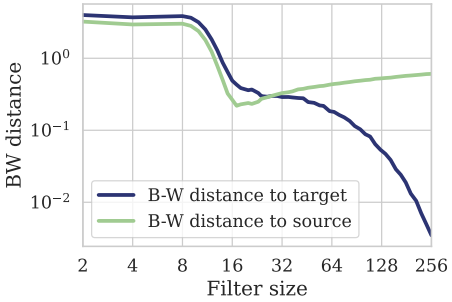
2. Test-time: Access to unseen target data

- Compute the **cross-PSD** $\hat{\mathbf{P}}_t$ for the target domain.
- Compute the **convolutional filter h** between target domain and the barycenter $\bar{\mathbf{P}}$.
- Predict target labels with trained predictor g.

Illustration of the monge mapping for two signals

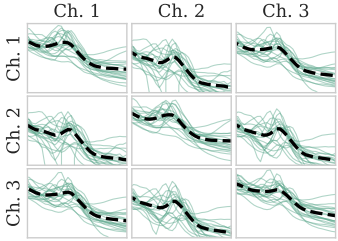


(a) Cross-PSD alignment

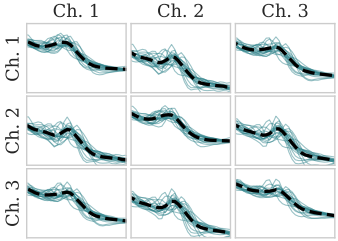


(b) Bures-Wasserstein distance

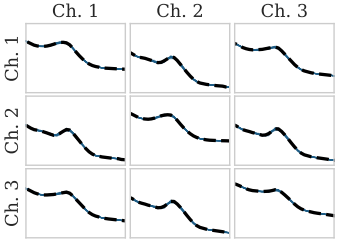
Cross-PSD alignment to barycenter for different filters size



(a) Source cross-PSD



(b) Alignment with $f = 32$



(c) Alignment with $f = 256$

Domain adaptation in biosignals

Possible variability in biosignals:

- Variability in the **patient population** : age, gender, height, diseased or healthy, different sleep stage proportion.

Domain adaptation in biosignals

Possible variability in biosignals:

- Variability in the **patient population** : age, gender, height, diseased or healthy, different sleep stage proportion.
- Variability in **recording and preprocessing** : different sensors, sensor positions, impedance, noise, interference, sampling rates, filtering.

Domain adaptation in biosignals

Possible variability in biosignals:

- Variability in the **patient population** : age, gender, height, diseased or healthy, different sleep stage proportion.
- Variability in **recording and preprocessing** : different sensors, sensor positions, impedance, noise, interference, sampling rates, filtering.
- Variability in **data interpretation by specialist** : different scoring criteria, subjectivity, tiredness.

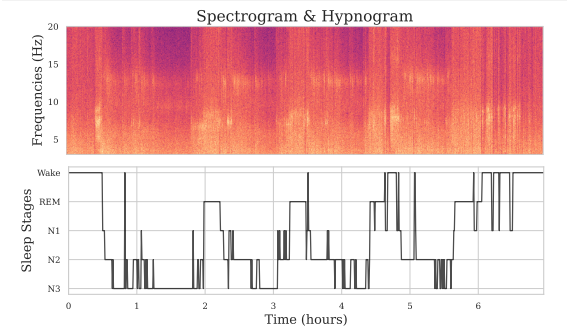
Domain adaptation in biosignals

Possible variability in biosignals:

- Variability in the **patient population** : age, gender, height, diseased or healthy, different sleep stage proportion.
- Variability in **recording and preprocessing** : different sensors, sensor positions, impedance, noise, interference, sampling rates, filtering.
- Variability in **data interpretation by specialist** : different scoring criteria, subjectivity, tiredness.

$\rightarrow$ **Domain Adaptation** problem.

Sleep Staging



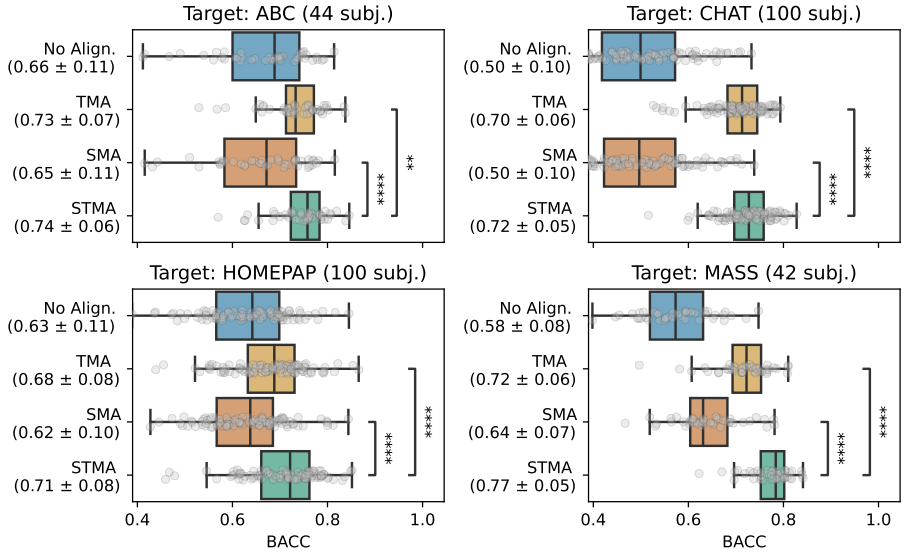
- **Classification** problem with **five** classes: Wake, N1, N2, N3, REM
- **Frequency** helps to classify sleep stage

Experimental setup

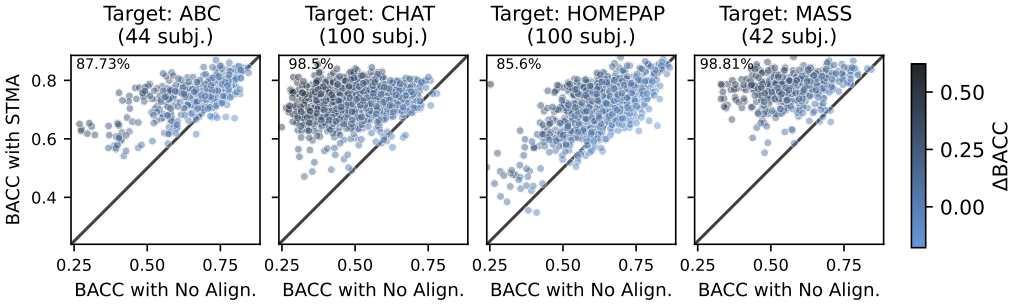
- **Four** different datasets: **ABC** , **CHAT** , **HOMEPAP** and **MASS**
- Around **300** subjects in total
- One **domain** = One **subject**
- **Seven** EEG channels
- Use **CNN** architecture from[1]

[1] *Chambon et. al., 2018*

Results on Sleep data



Results on Sleep data



How to use DA?

Skada [1] is a **Python** library to **easily** use DA methods.

- **Homogeneous API** for all DA methods (Shallow and Deep learning).
- **Sklearn-like API** with estimator class (.fit, .predict, ...), pipeline, grid search ...
- **DA scorer** to validate hyper-parameters without using target label.

[1] *Gnassounou et. al., 2024*

Data format in Skada

- `X` → 2D array of shape `(n_samples, n_features)`
- `y` → 1D array of shape `(n_samples,)`
- `sample_domain` → 1D array of shape `(n_samples,)` giving the **domain** of each **sample**

```python
from skada.datasets import make_shifted_datasets

X, y, sample_domain = make_shifted_datasets(
    20, 20, shift='covariate_shift', random_state=42
)
```

- All shift are available in `make_shifted_datasets` function

Shallow DA in Skada

- Initialize the estimator
- Fit the model
- Don't forget to give the **sample domain**

```python
from skada import LinOT

estimator = LinOT()
estimator.fit(X, y, sample_domain=sample_domain)
```

- $\sim$ 20 shallow methods available in Skada

Pipeline DA in Skada

- Can be used with **Pipeline**

```python
from skada import make_da_pipeline
from skada import LinOTAdapter, GaussianReweightAdapter
from sklearn.linear_model import LogisticRegression

pipeline = Pipeline(
    LinOTAdapter(),
    LogisticRegression()
)
pipeline.fit(X, y, sample_domain=sample_domain)
```

- Possibility to mixed DA adapters

```python
pipeline = Pipeline(
   LinOTAdapter(),
   GaussianReweightAdapter(),
   LogisticRegression()
)
```

DA scorer in Skada

- Possibility to use `cross_val_score` with **DA scorers**
- DA scorers are used to **validate** the **hyperparameters** without using the target labels

```
from skada.scorers import ImportanceWeightedScorer

scorer = ImportanceWeightedScorer()
score = cross_val_score(pipeline, X, y, sample_domain=sample_domain,
↪  scoring=scorer)
```

- 6 DA scorers available in Skada

Deep DA method in Skada

- Use **Skorch** → **Pytorch** wrapper for **Sklearn**
- Give an **architecture** and **hyperparameters**

```python
from skada.deep import DeepCoral
from skada.deep.modules import ToyCNN

model = DeepCoral(
    ToyCNN(),
    batch_size=32,
    max_epochs=5,
    lr=1e-3,
    reg=1,
    layer_name="feature_extractor",
)
model.fit(X, y, sample_domain=sample_domain)
```

- $\sim$ 10 Deep DA methods available in Skada

Conclusion

- Distribution shift is a **challenging** problem in **biosignals**
- Alignment of the **cross-PSD** is a **powerful** tool to tackle the problem
- Try **Skada** to easily use DA methods
- Don't hesitate to contribute to the library!