EEGraSP: Biblioteca para el procesamiento del electroencefalograma usando procesamiento de señales sobre grafos

EEGraSP: A library for processing the electroencephalogram using graph signal processing

Alejandro Weinstein[1]* https://orcid.org/0000-0003-1486-7003
Julio Rodiño[2] https://orcid.org/0000-0002-8916-6810
Camilo Jara[3] https://orcid.org/0009-0008-9445-5423
Lucas Cortés[4] https://orcid.org/0009-0005-0234-2770
Alejandro Veloz[5] https://orcid.org/0000-0001-9394-6660

Recibido 12 de octubre de 2024, aceptado 07 de noviembre de 2024
Received: October 12, 2024 Accepted: November 07, 2024

RESUMEN

En este trabajo se presenta la biblioteca EEGraSP, desarrollada para el procesamiento y análisis del electroencefalograma (EEG) usando procesamiento de señales sobre grafos (GSP). La biblioteca está desarrollada utilizando el *stack* científico del lenguaje de programación Python, incluyendo el uso de bibliotecas como NumPy, SciPy, Matplotlib, NetworkX, PyGSP2, y MNE. El desarrollo es código abierto, bajo licencia MIT, y se realiza usando la infraestructura disponible en GitHub. También está disponible la documentación respectiva, y los mecanismos que facilitan la instalación de la biblioteca a través de PyPI y de conda-forge. La biblioteca ofrece tres tipos de funcionalidades: creación de grafos, ya sea a partir de la posición de los electrodos o a partir de series de tiempo asociadas a los electrodos (a través de algoritmos de *graph learning*); visualización de los grafos utilizados para modelar los datos del EEG; y mecanismos para imputar datos faltantes en uno o más electrodos. Luego de describir los fundamentos de GSP y de la adquisición y procesamiento del EEG, el trabajo describe, a través de distintos ejemplos, las distintas funcionalidades de EEGraSP. Hasta donde saben los autores de este trabajo, EEGraSP es la primera biblioteca especialmente diseñada para el análisis y procesamiento del EEG usando GSP. En el futuro, EEGraSP incluirá: mejoras en su metodología de desarrollo, incluyendo aspectos como unit testing dentro de su proceso de integración continua; mejoras en la documentación y nuevos ejemplos; y la incorporación de nuevos algoritmos.

Palabras clave: Electroencefalograma, procesamiento de señales sobre grafos, EEG, GSP.

ABSTRACT

This paper presents the EEGraSP library, developed for processing and analyzing electroencephalogram (EEG) data using graph signal processing (GSP). The library is developed using the Python programming language scientific stack, including libraries such as NumPy, SciPy, Matplotlib, NetworkX, PyGSP2, and MNE. The development is open source, under the MIT license, and is done using the infrastructure available on GitHub. The respective documentation is also available, as are the mechanisms that facilitate the

[1] Universidad Técnica Federico Santa María. Departamento de Electrónica, Valparaíso, Chile. E-mail: alejandro.weinstein@usm.cl
[2] Universidad de Valparaíso. Laboratorio de Dinámica Cerebral, Valparaíso, Chile. E-mail: jrodino14@gmail.com
[3] Universidad Técnica Federico Santa María. Laboratorio EEG-GSP, Valparaíso, Chile. E-mail: cjotade@gmail.com
[4] Lanek, Valparaíso, Chile, E-mail: lucas.cortes@lanek.cl
[5] Universidad de Valparaíso. Escuela de Ingeniería Biomédica, Valparaíso, Chile. E-mail: alejandro.veloz@uv.cl
* Autor de correspondencia: alejandro.weinstein@usm.cl

installation of the library through PyPI and conda-forge. The library offers three types of functionalities: the creation of graphs, either from the location of the electrodes or from time series associated with the electrodes (through graph learning algorithms); visualization of the graphs used to model the EEG data; and mechanisms to impute missing data in one or more electrodes. After describing the fundamentals of GSP and EEG acquisition and processing, the paper describes, through several examples, the different functionalities of EEGraSP. To the best of the authors' knowledge, EEGraSP is the first library specifically designed for EEG analysis and processing using GSP. In the future, EEGraSP will include improvements in its development methodology, including unit testing within its continuous integration process, improvements in documentation and new examples, and the incorporation of new algorithms.

Keywords: *Electroencephalogram, graph signal processing, EEG, GSP.*

INTRODUCCIÓN

El electroencefalograma (EEG) es el registro de los potenciales eléctricos sobre el cuero cabelludo. Estos potenciales eléctricos son el resultado, principalmente, de la actividad de la corteza cerebral [1], por lo que el EEG se puede usar para inferir el estado de esta parte del cuerpo [2]. El EEG es una técnica no invasiva de bajo riesgo que se utiliza en entornos clínicos y de investigación. El EEG se utiliza en ambientes clínicos para monitorear la profundidad de la anestesia, diagnosticar epilepsia y trastornos del sueño, para el monitoreo continuo de pacientes de la unidad de cuidados intensivos, entre otros [3], [4]. En contextos de investigación, el EEG se utiliza en estudios relacionados con la neurociencia, la psicología experimental y la neurolingüística, entre otros [5]. El EEG también se ha utilizado en aplicaciones como la detección en tiempo real de la fatiga de conductores [6] y en productos de electrónica de consumo [7].

Por otro lado, durante los últimos años, se ha desarrollado un nuevo paradigma para el procesamiento de señales conocido como Procesamiento de Señales sobre Grafos [8], [9] (GSP, por sus siglas en inglés). GSP extiende las operaciones de procesamiento de señales clásicas [10] (filtrado, interpolación, análisis de Fourier, wavelets, etc.) a señales definidas sobre un grafo, aprovechando así cierta estructura que tiene los datos, que de otra forma no se puede aprovechar. GSP considera un grafo formado por un conjunto de vértices entre los cuales pueden existir aristas que conectan pares de nodos. Una señal definida sobre el grafo es aquella en donde cada nodo tiene asociado un valor numérico [8]. GSP se ha utilizado en redes de sensores para detectar valores atípicos, filtrar y reconstruir lecturas de sensores a partir de mediciones dispersas [11], [12], [13], y en aplicaciones relacionadas al aprendizaje de máquinas y al procesamiento de imágenes [9].

GSP también se ha utilizado en investigaciones relacionadas con la resonancia magnética funcional (fMRI, por sus siglas en inglés). En este contexto, cada región del cerebro se considera como un vértice del grafo. La conectividad entre estos vértices se infiere a partir de la conectividad anatómica o funcional de estas regiones [14], [15]. La señal sobre el grafo es el nivel de actividad de cada región, determinado mediante fMRI. Por ejemplo, en los trabajos de [16], [17], [18], se combina fMRI con la noción de frecuencia de la señal definida sobre el grafo para caracterizar los niveles de adaptabilidad y exposición a una tarea en un sujeto. Otro ejemplo es el trabajo de [19], donde se combina el uso de fMRI con imágenes de resonancia magnética de difusión para capturar redes espacio-temporales transitorias de conectividad. GSP también se ha usado para clasificar y comprender la etiología de enfermedades mentales [20], [21], [22], [23].

También se ha propuesto usar GSP para el procesamiento del EEG. Mathur and Chakka utilizan GSP para detectar epilepsia a partir del EEG [24], construyendo el grafo a partir de la serie de tiempo del EEG. Rui, Nejati, Safavi and Cheung también construyen un grafo usando la serie de tiempo del EEG, para luego mostrar cómo las técnicas de GSP permiten reducir la dimensionalidad de los datos y mejorar el desempeño de un clasificador en comparación con la técnica clásica [25]. GSP se ha usado en el procesamiento del EEG para la supresión del ruido, reducción de dimensionalidad, clasificación, y estimación de la conectividad

[26]-[29]. Recientemente, se ha propuesto utilizar GSP para decodificar la actividad motora a partir del espectro de la señal sobre un grafo inducido por datos de EEG [30]; y para imputar datos faltantes en los electrodos del EEG, construyendo el grafo a partir de la ubicación espacial de los electrodos y formulando el problema de la imputación de los datos faltantes como un problema de regresión [31].

Existen distintos paquetes computacionales que implementan las técnicas relacionadas con GSP, incluyendo a PyGSP [32], GSPBOX [33], y GRaSP [34] (el primero implementado en Python y los dos últimos en MATLAB). Sin embargo, ninguna de estas bibliotecas está diseñada para procesar el EEG usando GSP. Es por esto que en este trabajo presentamos un nuevo paquete computacional, llamado EEGraSP, desarrollado con el fin de facilitar el uso de las técnicas basadas en GSP para el procesamiento y análisis del EEG. Hasta donde sabemos, esta es la primera y única biblioteca desarrollada específicamente para este fin.

En las siguientes dos secciones se presentan, de manera resumida, los fundamentos de la teoría detrás de GSP y de las técnicas de procesamiento de la señal de EEG. A continuación, se describe en detalle a EEGraSP, incluyendo sus funcionalidades principales y el proceso utilizado para su desarrollo. El artículo finaliza con una sección de discusión y conclusiones.

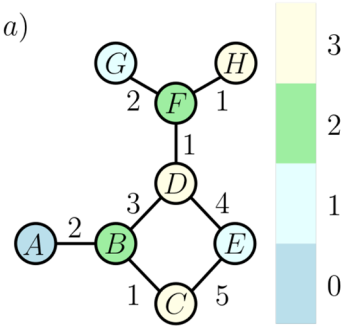
PROCESAMIENTO DE SEÑALES SOBRE GRAFOS

Los dos objetos matemáticos sobre los que se construye la teoría de GSP son el grafo y las funciones definidas sobre éste. Un grafo ponderado y no dirigido está definido por la tupla $g = \{V, \varepsilon, W\}$, donde V es el conjunto de vértices, ε es el conjunto de artistas, y W es la matriz de adyacencia ponderada. El caso de un grafo no ponderado y no dirigido se considera como un grafo ponderado en donde todos los pesos son 1. La teoría de GSP también se puede extender a grafos dirigidos [35]. En esta matriz se asocia, de manera arbitraria, cada fila y columna a un vértice, y cada elemento de la matriz corresponde al peso entre un par de vértices. Si dos vértices no están conectados, se les asocia un peso de valor cero. También se define la matriz diagonal de grados D, donde el valor de la diagonal es la suma de los pesos correspondientes a cada vértice. En base a estas dos matrices se define el Laplaciano como $L = D - W$. Esta definición corresponde a lo que se conoce como Laplaciano combinatorial. También existe el Laplaciano normalizado, definido como $L = D^{-1/2}WD^{-1/2}$. En este artículo usaremos solamente el Laplaciano combinatorial. La descomposición espectral del Laplaciano posee un conjunto de propiedades [36], [37] que permiten extender muchas de las técnicas clásicas del procesamiento de señales al marco de GSP [8], [9], [35]. La función $f : v \rightarrow \mathbb{R}$ definida sobre un grafo le asigna a cada vértice del grafo un número real. Es decir, ésta es una función cuyo dominio corresponde a los vértices del grafo. En la Figura 1 se ilustran estos conceptos a través de un ejemplo. El panel a) muestra un grafo con ocho vértices *A, B, …, H*. El peso entre los vértices se indica con un número asociado a la línea que conecta un par de vértices. El color de cada vértice corresponde al valor que retorna la función para cada vértice. El panel b) muestra los conjuntos v y ε, y el Laplaciano L correspondiente al grafo mostrado en el panel a). Los trabajos de [9], [35], [8] resumen la teoría asociada a GSP.

EL PROCESAMIENTO DEL ELECTROENCEFALOGRAMA

El EEG es la señal eléctrica que se registra a través de un conjunto de electrodos posicionados sobre el cuero cabelludo de una persona. Esta señal es el resultado de la actividad eléctrica de la corteza cerebral. Al EEG también se le superponen [38] señales generadas por el corazón (ECG), músculos (EMG), el movimiento ocular (EOG), llamados artefactos, y el ruido inducido por la red eléctrica. Para inferir la actividad cerebral a partir del EEG es necesario procesar esta señal para remover estos artefactos y ruido, y para extraer de ésta la información relevante. Existen, además, distintos paradigmas que permiten enfocar un registro dado en la obtención de la información específica de interés.

Una secuencia de procesamiento estándar del EEG incluye las siguientes etapas: adquisición de la señal, filtraje pasa-banda, reducción de la taza de muestreo, eliminación de los artefactos, y cuantificación de la señal. Los detalles de esta última etapa dependen del paradigma utilizado y el tipo de análisis a realizar. Los paradigmas se pueden clasificar en dos grandes clases. La primera clase corresponde al paradigma



b)

$$\mathcal{V} = \{A, B, C, D, E, F, G, H\}$$

$$\mathcal{E} = \{(A,B), (B,C), (B,D), (E,C) \\ (E,D), (D,F), (F,G), (F,H)\}$$

$$L = \begin{bmatrix} 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 6 & -1 & -3 & 0 & 0 & 0 & 0 \\ 0 & -1 & 6 & 0 & -5 & 0 & 0 & 0 \\ 0 & -3 & 0 & 8 & -4 & -1 & 0 & 0 \\ 0 & 0 & -5 & -4 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 4 & -2 & -1 \\ 0 & 0 & 0 & 0 & 0 & -2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

Figura 1. a) Ejemplo de un grafo y de una función definida sobre él. El color de cada vértice indica el número asociado a éste por la función. b) Conjuntos v y ε y el Laplaciano L correspondiente.

del potencial relacionado con evento (ERP, por sus siglas en inglés). Bajo este paradigma la persona es sometida a estímulos repetitivos bajo distintas condiciones de experimentación. Luego se realiza un promedio de ventanas temporales del EEG sincronizadas con estos estímulos. El resultado de este promedio es el ERP, a través del cual se pueden inferir distintos aspectos relacionados al experimento realizado [38]. El segundo paradigma es el registro del EEG en estado de reposo, en donde la persona no es sometida a estímulos externos ni realiza una tarea en específico. Para ambos paradigmas existen distintas técnicas de análisis para llevar a cabo la etapa de cuantificación. Estas incluyen técnicas basadas en análisis en el dominio de la frecuencia [39], tiempo-frecuencia [40], y conectividad cerebral [41].

Uno de los aspectos relacionados a la etapa de adquisición del EEG es la definición del número de electrodos a utilizar, su ubicación en el cuero cabelludo, y la nomenclatura utilizada para identificar cada electrodo. A esta configuración se le conoce como montaje. Existen montajes estándar definidos por la comunidad [42], como el sistema internacional 10/20, el sistema internacional 10/10 y el sistema internacional 10/5. También existen montajes definidos por los distintos fabricantes de equipos para registrar EEG, como, por ejemplo, los montajes de Biosemi (con 16, 32, 64, 128, 160 ó 256 electrodos) y de Magnetism EGI denominados como HydroCel GSN (con 32 a 257 electrodos y distintas configuraciones). La Figura 2 muestra, a modo de ejemplo, tres montajes: estándar 10/20 (panel de la izquierda), Biosemi con 32 electrodos (panel central), y GSN con 32 electrodos (panel de la derecha).

EEGraSP

Como se observa en la Figura 2, la señal del EEG consiste en un conjunto de valores de potenciales eléctricos definidos en cada uno de los electrodos del montaje. Si consideramos al conjunto de electrodos, y las relaciones de estos entre sí, como un grafo, podemos pensar en el EEG como en una señal definida sobre un grafo. Este punto de vista permite utilizar las técnicas asociadas a GSP para el procesamiento del EEG. Esto motiva el desarrollo de la biblioteca EEGraSP.

EEGraSP es una biblioteca de código abierto (bajo licencia MIT) desarrollada en el lenguaje de programación Python y el *stack* de programación científica SciPy [43]. Las tareas relacionadas al manejo de la señal de EEG (lectura y escritura de datos, procesamiento, etc.) se basan en la biblioteca MNE [44]. Las funcionalidades principales relacionadas con GSP se basan en la biblioteca PyGSP2 [45], la que a su vez utiliza la biblioteca NetworkX [46] para el manejo de las estructuras de datos y algoritmos relacionados a los grafos. Todas las visualizaciones son implementadas usando Matplotlib [47].

Las funcionalidades de EEGraSP se pueden agrupar en tres grupos: (i) creación de grafos, ya sea a partir

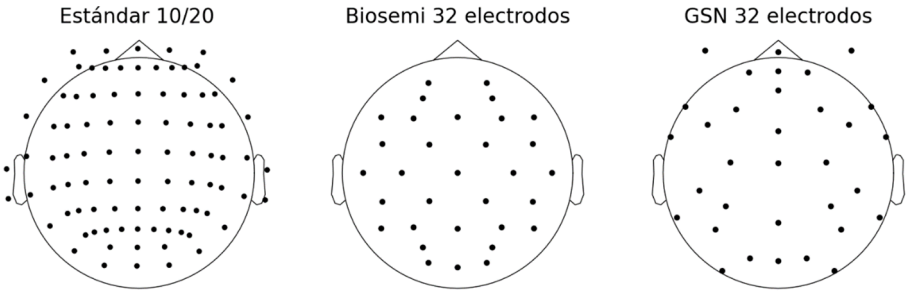


Figura 2. Ejemplos de montaje de EEG. Cada punto representa la proyección de la posición de cada electrodo sobre una representación plana de la cabeza. Se muestran tres montajes: Estándar 10/20 (panel de la izquierda), Biosemi con 32 electrodos (panel central), y GSN con 32 electrodos (panel de la derecha). Los puntos fuera de la circunferencia corresponden a electrodos posicionados fuera la parte superior de la cabeza (frente, sienes, etc.).

de la posición de los electrodos o a partir de series de tiempo asociadas a los electrodos (a través de la técnica conocida como *graph learning*); (ii) visualización; y (iii) imputación de datos faltantes. La interfaz a estas funcionalidades es a través de la clase EEGrasp, la cual se inicializa, opcionalmente, con los datos asociados a los electrodos, la posición de los electrodos correspondientes al montaje utilizado durante el registro, y la etiqueta de los electrodos. A continuación, se ilustra funcionalidades de la biblioteca a través de distintos ejemplos. En todos los ejemplos que siguen usamos el registro de EGG IVa del conjunto de datos "BCI Competition III", correspondiente a una tarea motora imaginada.

El primer paso para utilizar GSP es definir los vértices sobre los que está definida la señal de interés. Para el EEG, esto corresponde a los electrodos utilizados en el registro de éste. En general, existen tres estrategias para determinar las aristas y los pesos correspondientes. La primera es utilizar alguna estructura intrínseca que posean los vértices del grafo. La segunda es utilizar la distancia, euclidiana o de otro tipo, entre los vértices. La tercera es usar alguna serie de tiempo asociada a los vértices para inferir la estructura del grafo. En el caso del EEG, la primera estrategia no aplica, por lo que en EEGraSP implementamos las dos últimas.

El siguiente es un ejemplo en donde se obtiene el grafo a partir de la posición dada por el montaje Biosemi de 64 electrodos utilizado en un registro de EEG. En este caso el grafo se construye usando un kernel Gaussiano. Se ilustra el efecto que el parámetro tiene en la obtención del grafo mostrando los resultados para dos valores distintos del parámetro. El resultado se muestra en la Figura 3. El panel A) muestra la posición, en 3 dimensiones, de los 64 electrodos según la definición dada por el montaje utilizado en este ejemplo. Los paneles B) y C) muestran la matriz de pesos obtenidas al utilizar $\sigma = 0{,}3$ y $\sigma = 0{,}5$, respectivamente. En ambos casos se usó $\varepsilon = 0{,}5$ (este parámetro corresponde a un umbral en donde toda distancia más grande que este valor se hace 0). Podemos observar que al aumentar el factor de dispersión σ el grafo se hace más denso, asignándole pesos mayores a cero a vértices más distantes. El siguiente es el segmento de código utilizado en este ejemplo.

```
montage = mne.channels.make_standard_
montage('biosemi64')
ch_names = montage.ch_names
eeg_pos = montage.get_positions()['ch_pos']
eeg_pos = np.array([pos for _, pos in
eeg_pos.items()])
eegrasp = EEGrasp(coordinates=eeg_pos)
Z = eegrasp.compute_distance(method=
'Euclidean')
G = eegrasp.compute_graph(epsilon=0.5,
sigma=0.1)
W1 = eegrasp.graph_weights
G = eegrasp.compute_graph(epsilon=0.5,
sigma=0.3)
W2 = eegrasp.graph_weights
```

En este código se observa que la posición de los electrodos se obtiene a partir de la información

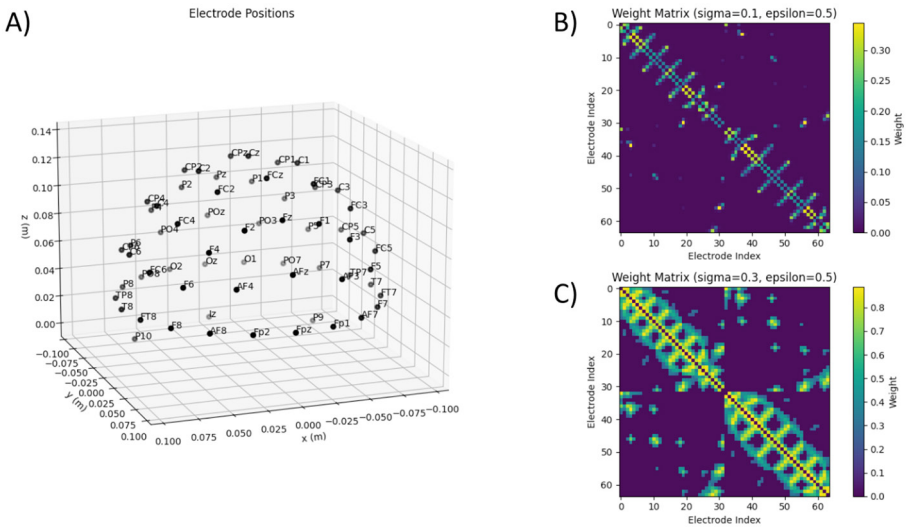


Figura 3. Grafo creado a partir de la posición de los electrodos y el kernel Gaussiano. A) Visualización en el espacio tridimensional del montaje Biosemi de 64 electrodos. B) y C) Matriz de pesos obtenidos con parámetro $\sigma = 0{,}1$ y $\sigma = 0{,}3$, respectivamente. En ambos casos se usó $\varepsilon = 0{,}5$.

disponible en la biblioteca MNE. Esta información se usa para inicializar la clase EEGrasp. Luego se usan los métodos compute_distance para calcular la distancia, en este ejemplo Euclidiana, y compute_graph para determinar el grafo usando el kernel Gaussiano con los parámetros deseados. Finalmente se obtienen la matriz de pesos a través del atributo graph_weights.

También es posible inferir el grafo a partir de datos asociados a cada electrodo usando *graph learning* [48]. En el siguiente ejemplo se utilizan los *trials* del EEG en un registro con 128 electrodos, previamente filtrados en el rango de frecuencias de 8 a 30 Hz. Estos datos son utilizados por el algoritmo de graph learning, en este caso con parámetros $a = 0{,}34$ y $b = 0{,}4$. El parámetro a controla el valor de los pesos inferidos, en donde valores más grandes del parámetro resultan en pesos de mayor valor. El parámetro b controla la densidad de los pesos, en donde valores más grandes del parámetro resultan en una matriz de adyacencia más densa (más coeficientes distintos de cero). El resultado se muestra en la Figura 4. En el panel de la izquierda se observa el grafo, con los vértices ubicados en la posición definida según el montaje usado en el experimento. En el panel de la derecha se muestra la matriz de adyacencia correspondiente. El siguiente es el ejemplo de código usado para este ejemplo.

```
gsp = EEGrasp() # Instantiate EEGraSP
# Carga y preprosamiento del EEG
# ...
# Epoch and Crop epochs
epochs = mne.Epochs(data, events2, tmin=0.0,
tmax=2.5,
        baseline=(0, 0.5), preload=True)
epochs = epochs.crop(0.5, None)
epochs_data = epochs.get_data(copy=False)
# Compute the average euclidean distance
between the channels
gsp.data = epochs_data
gsp.coordinates = pos
W, Z = gsp.learn_graph(a=0.34, b=0.4)
gsp.compute_graph(W)
```

Luego de crear el objeto EEGrasp, se realiza la carga y preprocesamiento de los datos (se omiten los detalles de estos pasos), y la obtención de los *trials*, a través de la clase mne.Epochs de la biblioteca MNE. Luego se establecen los atributos data y coordinates del objeto EEGrasp con los datos de los *trials* y la posición de los electrodos (notar que este ejemplo la posición de los electrodos solo se utiliza para efectos de la visualización del grafo).

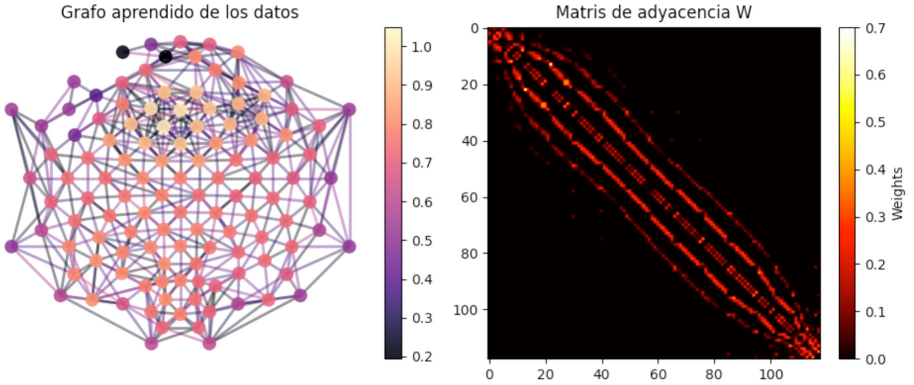


Figura 4. Grafo aprendido a partir de los datos. El panel de la izquierda muestra el grafo, con la ubicación de los vértices definida en función de la posición de los electrodos. El panel de la derecha muestra una representación alternativa del grafo a través de la matriz de adyacencia correspondiente.

Finalmente se llama al método learn_graph y se calcula la matriz de adyacencia.

La biblioteca EEGrasSP también cuenta con funcionalidades que facilitan la visualización de los datos. En el siguiente ejemplo se genera el grafo correspondiente al montaje Biosemi de 64 electrodos en base al kernel Gaussiano, y luego se utiliza el método plot para visualizar este grafo tanto en dos como en tres dimensiones. El resultado se muestra en la Figura 5. En el panel de la izquierda se observa el grafo proyectado en dos dimensiones. La posición de los vértices está basada en la ubicación de los electrodos según el montaje. Se superpone además el contorno de la cabeza. El color de las aristas del grafo indica el valor del peso correspondiente. En el panel de la derecha se muestra el mismo grafo en una visualización en tres dimensiones. El siguiente es el segmento de código utilizado para crear la figura.

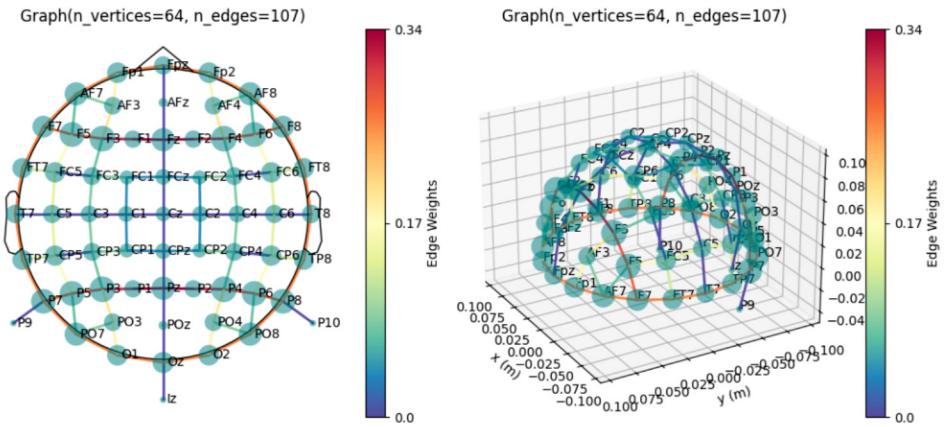


Figura 5. Visualización de un grafo inducido por el montaje Biosemi de 64 electrodos. Cada vértice corresponde a un electrodo, con la posición determinada por el montaje utilizado. En este ejemplo el grafo fue determinado usando un kernel Gaussiano. En color de las aristas indica el peso asociado a ésta. En el panel de la izquierda se muestra el grafo en dos dimensiones, con el contorno de la cabeza superpuesto en el gráfico. En el panel de la derecha se muestra el grafo visualizado en tres dimensiones.

```
montage = mne.channels.make_standard_
montage('biosemi64')
ch_names = montage.ch_names
EEG_pos = montage.get_positions()['ch_pos']
# Restructure into array
EEG_pos = np.array([pos for _, pos in EEG_
pos.items()])
gsp = EEGrasp(coordinates=EEG_pos,
labels=ch_names)
Z = gsp.compute_distance(EEG_pos)
G = gsp.compute_graph(sigma=0.1, epsilon=0.2)
fig, ax = gsp.plot()
fig, ax = gsp.plot(kind='3d')
```

La imputación de datos faltantes es un problema que ocurre a menudo en el análisis del EEG. Esta situación puede ocurrir por una falla en uno o más electrodos durante el registro de los datos, o por el movimiento de un electrodo en un momento específico. En EEGraSP abordamos la imputación de datos faltantes como un problema de interpolación formulado como una regresión de Tikhonov. Ilustramos este proceso a través de un ejemplo en donde se simula la ausencia de los datos de uno de los electrodos del EEG. Estos datos faltantes son luego imputados usando el método interpolate_channel. En este ejemplo el grafo correspondiente se determina usando el kernel Gaussiano con la distancia euclidiana entre los electrodos. El resultado de la imputación se observa en la Figura 6, en donde se muestra la señal original (en azul) y la señal reconstruida (en naranja). El segmento de código utilizado para esto es el siguiente:

```
MISSING_IDX = 5
lost_ch = data[MISSING_IDX, :].copy()
data[MISSING_IDX, :] = np.nan # Simula electro
de faltante
eegsp = EEGrasp(data, eeg_pos, ch_names) #
Inicializa una instancia de EEGraSP
# Calcula la distancia entre electrodos
dist_mat = eegsp.compute_distance
(normalize=True)
eegsp.compute_graph(epsilon=0.5, sigma=0.1)
# Calcula la estructura del grafo
# Interpola en canal faltante
interpolated = eegsp.interpolate_channel
(missing_idx=MISSING_IDX)
```

Finalmente, es importante destacar que EEGraSP ha sido desarrollado bajo una metodología ágil. Todo el código de la librería se desarrolla bajo un sistema de control de versiones basado en git, con el repositorio alojado en GitHub (https://github.com/gsp-eeg/EEGraSP). A través de esta plataforma se cuenta con un esquema de integración continua. Este esquema genera de manera automática, cada vez que se actualiza la rama *main* del repositorio, la

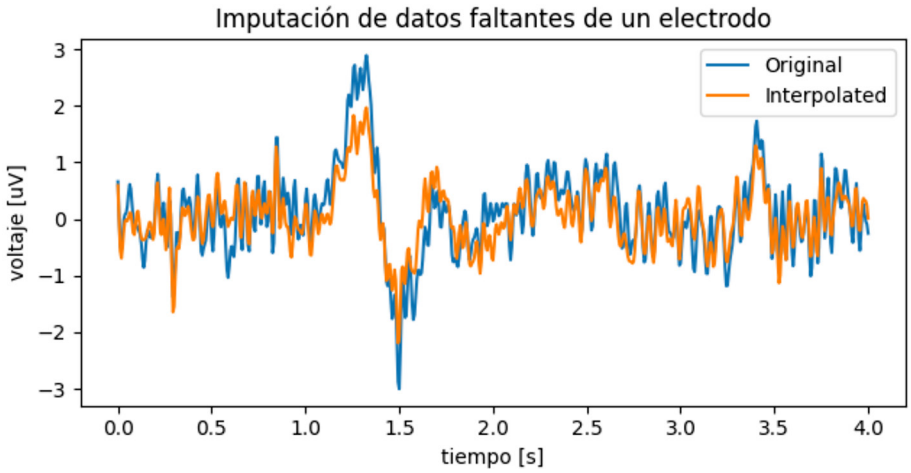


Figura 6. Interpolación realizada por EEGraSP. En este ejemplo se simula que los datos de uno de los electrodos no están disponibles. Luego se utiliza el método interpolate_channel para imputar estos datos faltantes. En la figura se pueden comparar la señal original (en azul) con la señal obtenida por el método de interpolación (en naranja).

documentación, la versión de la librería disponible en el Python Package Index (para la instalación de la librería a través de pip) y la versión de la librería disponible en conda-forge (para la instalación a través de conda). La plataforma GitHub es también utilizada para registrar los *issues* del proyecto.

CONCLUSIÓN

Este trabajo ha presentado a EEGraSP, una biblioteca para el procesamiento del EEG basado en GSP. La biblioteca cuenta con tres funcionalidades principales: construcción de grafos, visualización, e imputación de datos faltantes. Cada funcionalidad se ha presentado mediante un ejemplo que permite comprender la esencia de estas.

El desarrollo de EEGraSP es relevante ya que es, hasta donde sabemos, la primera biblioteca que permite aplicar las técnicas de GSP en el análisis y procesamiento del EEG.

EEGraSP posee varias ventajas. Primero, permite utilizar las técnicas de GSP para el análisis del EEG con facilidad, sin necesidad de tener que implementar una solución desde cero, permitiendo así avanzar más rápido en obtener una solución al problema de interés. Segundo, al usar como estructura de datos para el EEG las clases provistas por la biblioteca MNE (estándar de facto para el procesamiento del EEG en Python) facilita su adopción por parte de los investigadores que ya usan esta biblioteca. Finalmente, la tercera ventaja de EEGraSP es su desarrollo bajo un paradigma ágil, en donde se aprovecha la infraestructura disponible para proyectos de código abierto (GitHub, *Read the Docs*, PyPi, y conda-forge). Esto facilita tanto su uso como su desarrollo. Por otro lado, una de las limitaciones de la biblioteca es que esta está en una etapa temprana de desarrollo, donde los avances mostrados en este artículo son el reflejo de cuatro meses de trabajo, realizado por cuatro desarrolladores. Es por esto por lo que hay funcionalidades en EEGraSP que aún no han sido implementadas.

La perspectiva para el futuro es consolidar EEGraSP como una herramienta sólida técnicamente, amigable con el usuario, y completa. Para lograr el primer punto, prontamente agregaremos al modelo de desarrollo el uso de *unit testing* dentro del proceso de integración continua, para mejorar la validación del código fuente. Para lo segundo, continuaremos mejorando la documentación, incluyendo más ejemplos que permitan entender las capacidades y las distintas formas de usar EEGraSP. Para lo último, tenemos una hoja de ruta que define varias funcionalidades que debemos incorporar, incluyendo más algoritmos de interpolación y el uso de distintas métricas de conectividad.

AGRADECIMIENTOS

La investigación descrita en este artículo la financió, en parte, la Agencia Nacional de Investigación y Desarrollo (ANID): proyectos Basal AFB240002, Anillo ACT210053, y Fondecyt Regular 1231132.

REFERENCIAS

[1] A.F. Jackson and D.J. Bolger, "The neurophysiological bases of EEG and EEG measurement: a review for the rest of us," *Psychophysiology*, vol. 51, no. 11, pp. 1061-1071, Nov. 2014, doi: 10.1111/psyp.12283.

[2] A.J. Casson, M. Abdulaal, M. Dulabh, S. Kohli, S. Krachunov, and E. Trimble, "Electroencephalogram," in *Seamless Healthcare Monitoring*, T. Tamura and W. Chen, Eds., 2018, pp. 45-81, doi: 10.1007/978-3-319-69362-0_2.

[3] E. Niedermeyer and F.L. da Silva, *Electroencephalography: basic principles, clinical applications, and related fields,* Philadelphia, PA, USA: Lippincott Williams & Wilkins, 2005.

[4] C. Rubiños and D.A. Godoy, "Electroencephalographic monitoring in the critically ill patient: What useful information can it contribute?," *Medicina Intensiva (English Edition)*, vol. 44, no. 5, pp. 301-309, Jun. 2020, doi: 10.1016/j.medine.2019.06.008.

[5] J.T. Cacioppo, L.G. Tassinary, and G. Berntson, *Handbook of psychophysiology*, 3rd ed., Cambridge, UK: Cambridge University Press, 2007.

[6] H. Abbood, W. Al-Nuaimy, A. Al-Ataby, S.A. Salem, and H. S. AlZubi, "Prediction of driver fatigue: Approaches and open challenges," in *2014 14th UK Workshop on Computational Intelligence (UKCI)*, Sep. 2014, pp. 1-6. doi: 10.1109/UKCI.2014.6930193.

[7] G.A.M. Vasiljevic and L. C. de Miranda, "Brain-Computer Interface Games Based

on Consumer-Grade EEG Devices: A Systematic Literature Review," *International Journal of Human-Computer Interaction*, vol. 36, no. 2, pp. 105-142, Jan. 2020, doi: 10.1080/10447318.2019.1612213.

[8] A. Ortega, *Introduction to Graph Signal Processing*, Cambridge, UK: Cambridge University Press, 2022.

[9] A. Ortega, P. Frossard, J. Kovačević, J.M. F. Moura, and P. Vandergheynst, "Graph Signal Processing: Overview, Challenges, and Applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808-828, May 2018, doi: 10.1109/JPROC.2018.2820126.

[10] S.J. Orfanidis, *Introduction to signal processing*, Piscataway, NJ, USA: Rutgers University, 2010.

[11] R.S. Wagner, R.G. Baraniuk, S. Du, D.B. Johnson, and A. Cohen, "An architecture for distributed wavelet analysis and processing in sensor networks," in *Proceedings of the 5th international conference on Information processing in sensor networks*, IPSN '06, Nashville, Tennessee, Apr. 2006, pp. 243-250, doi: 10.1145/1127777.1127816.

[12] R.K. Jain, J.M.F. Moura, and C.E. Kontokosta, "Big Data + Big Cities: Graph Signals of Urban Air Pollution [Exploratory SP]," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 130-136, Sep. 2014, doi: 10.1109/MSP.2014.2330357.

[13] X. Zhu and M. Rabbat, "Graph Spectral Compressed Sensing for Sensor Networks," in 2012 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, 2012, pp. 2865-2868, doi: 10.1109/ICASSP.2012.6288515.

[14] E. Bullmore and O. Sporns, "The economy of brain network organization," *Nature Reviews Neuroscience*, vol. 13, no. 5, Art. no. 5, May 2012, doi: 10.1038/nrn3214.

[15] O. Sporns, *Networks of the Brain*, London, UK: MIT Press, 2010.

[16] W. Huang, L. Goldsberry, N.F. Wymbs, S. T. Grafton, D.S. Bassett, and A. Ribeiro, "Graph Frequency Analysis of Brain Signals," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 7, pp. 1189-1203, Oct. 2016, doi: 10.1109/JSTSP.2016.2600859.

[17] L. Goldsberry, W. Huang, N.F. Wymbs, S. T. Grafton, D.S. Bassett, and A. Ribeiro, "Brain signal analytics from graph signal processing perspective," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans,* LA, USA, 2017, pp. 851-855, doi: 10.1109/ICASSP.2017.7952276.

[18] D.S. Bassett, N.F. Wymbs, M.A. Porter, P.J. Mucha, J.M. Carlson, and S.T. Grafton, "Dynamic reconfiguration of human brain networks during learning," *PNAS*, vol. 108, no. 18, pp. 7641-7646, May 2011, doi: 10.1073/pnas.1018985108.

[19] A. Griffa *et al.*, "Transient networks of spatio-temporal connectivity map communication pathways in brain functional systems," *NeuroImage*, vol. 155, pp. 490-502, Jul. 2017, doi: 10.1016/j.neuroimage.2017.04.015.

[20] C. Hu *et al.*, "A Spectral Graph Regression Model for Learning Brain Connectivity of Alzheimer's Disease," *Plos One*, vol. 10, no. 5, Art. no. 0128136, May 2015, doi: 10.1371/journal.pone.0128136.

[21] C. Hu, J. Sepulcre, K.A. Johnson, G.E. Fakhri, Y.M. Lu, and Q. Li, "Matched signal detection on graphs: Theory and application to brain imaging data classification," *NeuroImage*, vol. 125, pp. 587-600, Jan. 2016, doi: 10.1016/j.neuroimage.2015.10.026.

[22] C. Hu, X. Hua, J. Ying, P.M. Thompson, G. E. Fakhri, and Q. Li, "Localizing Sources of Brain Disease Progression with Network Diffusion Model," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 7, pp. 1214-1225, Oct. 2016, doi: 10.1109/JSTSP.2016.2601695.

[23] M. Ménoret, N. Farrugia, B. Pasdeloup, and V. Gripon, "Evaluating graph signal processing for neuroimaging through classification and dimensionality reduction," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Montreal, QC, Canada,* 2017, pp. 618-622, doi: 10.1109/GlobalSIP.2017.8309033.

[24] P. Mathur and V.K. Chakka, "Graph Signal Processing of EEG signals for Detection of Epilepsy," in *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, Noida, India, 2020, pp. 839-843, doi: 10.1109/SPIN48934.2020.9070326.

[25] L. Rui, H. Nejati, S.H. Safavi and N.M. Cheung, "Simultaneous low-rank component

and graph estimation for high-dimensional graph signals: Application to brain imaging," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, 2017, pp. 4134-4138, doi: 10.1109/ICASSP.2017.7952934.

[26] L. Rui, H. Nejati and N.M. Cheung, "Dimensionality reduction of brain imaging data using graph signal processing," in *2016 IEEE International Conference on Image Processing (ICIP)*, Phoenix, AZ, USA, 2016, pp. 1329-1333, doi: 10.1109/ICIP.2016.7532574.

[27] U. Graichen, R. Eichardt, P. Fiedler, D. Strohmeier, F. Zanow, and J. Haueisen, "Sphara - A Generalized Spatial Fourier Analysis for Multi-Sensor Systems with Non-Uniformly Arranged Sensors: Application to EEG," *Plos One*, vol. 10, no. 4, Art. no. 0121741, Apr. 2015, doi: 10.1371/journal.pone.0121741.

[28] K. Smith *et al.*, "Locating Temporal Functional Dynamics of Visual Short-Term Memory Binding using Graph Modular Dirichlet Energy," *Scientific Reports*, vol. 7, no. 1, Art. no. 1, Feb. 2017, doi: 10.1038/srep42013.

[29] S. Mortaheb *et al.*, "A Graph Signal Processing Approach to Study High Density EEG Signals in Patients with Disorders of Consciousness," in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Berlin, Germany, 2019, pp. 4549-4553, doi: 10.1109/EMBC.2019.8856436.

[30] M. Miri, V. Abootalebi, H. Saeedi-Sourck, D. Van De Ville, and H. Behjat, "Spectral representation of EEG data using learned graphs with application to motor imagery decoding," *Biomedical Signal Processing and Control*, vol. 87, p. 105537, Jan. 2024, doi: 10.1016/j.bspc.2023.105537.

[31] A. Weinstein, "Imputing missing electroencephalography data using graph signal processing," in *18th International Symposium on Medical Information Processing and Analysis, SPIE, Valparaiso, Chile,* Mar. 2023, pp. 417-424. doi: 10.1117/12.2669735.

[32] Michaël Defferrard, Lionel Martin, Rodrigo Pena, and Nathanaël Perraudin, "*PyGSP: Graph Signal Processing in Python," Zenodo*, 2017, doi: 10.5281/zenodo.1003158.

[33] N. Perraudin *et al.*, "GSPBOX: A toolbox for signal processing on graphs," 2016, arXiv: 1408.5781.

[34] B. Girault, S. S. Narayanan, A. Ortega, P. Gonçalves, and E. Fleury, "GRASP: A matlab toolbox for graph signal processing," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 6574-6575.

[35] D.I. Shuman, S.K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83-98, May 2013, doi: 10.1109/MSP.2012.2235192.

[36] F.R.K. Chung, *Spectral graph theory (CBMS Regional Conference Series in Mathematics, No. 92)*, Rhode Island, USA: American Mathematical Society, 1997.

[37] D. Spielman, "Spectral graph theory," in *Combinatorial Scientific Computing*, U. Naumann and O. Schenk, ed., Boca Raton, Florida, USA: Chapman and Hall/CRC, 2012.

[38] S.J. Luck, *An introduction to the event-related potential technique,* Cambridge, MA, USA: MIT press, 2014.

[39] P. Mitra, *Observed brain dynamics*, Oxford, England: Oxford University Press, 2007.

[40] M.X. Cohen, *Analyzing neural time series data: theory and practice*, 1st ed. Cambridge, MA, USA: MIT press, 2014.

[41] C.J. Stam, G. Nolte, and A. Daffertshofer, "Phase lag index: Assessment of functional connectivity from multi channel EEG and MEG with diminished bias from common sources," *Human Brain Mapping*, vol. 28, no. 11, pp. 1178-1193, 2007, doi: 10.1002/hbm.20346.

[42] V. Jurcak, D. Tsuzuki, and I. Dan, "10/20, 10/10, and 10/5 systems revisited: Their validity as relative head-surface-based positioning systems," *NeuroImage*, vol. 34, no. 4, pp. 1600-1611, Feb. 2007, doi: 10.1016/j.neuroimage.2006.09.024.

[43] P. Virtanen *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nat Methods*, vol. 17, no. 3, pp. 261-272, Mar. 2020, doi: 10.1038/s41592-019-0686-2.

[44] A. Gramfort *et al.* "MEG and EEG data analysis with MNE-Python," *Frontiers in Neuroscience*, vol. 7, Art. 267, 2013, doi:10.3389/fnins.2013.00267.

[45] A. Weinstein, L. Cortés, J. Rodiño, and C. Jara, PyGSP2: Graph Signal Processing in Python. Zenodo, 2024. doi: 10.5281/zenodo.13136296.

[46] A. Hagberg, P.J. Swart, and D.A. Schult, "Exploring network structure, dynamics, and function using NetworkX," *Los Alamos National Laboratory (LANL)*, Los Alamos, NM, USA, 2008.

[47] J.D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, May 2007, doi: 10.1109/MCSE.2007.55.

[48] V. Kalofolias, "How to Learn a Graph from Smooth Signals," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, PMLR*, May 2016, pp. 920-929.