1. Instalacion

Estos scripts están implementados y probados en python 2.7, pero deberían funcionar en python 3 sin problemas.

Para su funcionamiento hace falta tener instaladas las librerias de python:

- scipy
- numpy
- scikit-learn
- h5py
- neo (para leer ficheros ABF)
- joblib (para el paralelismo)

2. Formato de los datos

Los datos de un experimento y diferentes resultados se guardan en un fichero con formato HDF5. La estructura en el fichero es la siguiente:

- Cada tramo del experimento se guarda en una carpeta con el nombre del fichero del que se extrajo
- Dentro de una carpeta de un experimento hay una tabla `Raw` donde están los datos importados (por lo general solo se importan los datos de los sensores lumbares). Esta tabla tendrá además como atributos la lista de los nombres de los sensores y el muestreo de los datos
- En el caso de que se aplique un filtro a la señal raw para eliminar algunas frecuencias también habrá una tabla `RawFiltered`. La tabla tendrá dos atributos que indican el rango mínimo y máximo de frecuencias que tiene la señal
- Dentro de la carpeta de cada tramo del experimento habrá una carpeta para cada sensor (los nombres son los que se han usado en la definición del experimento)
- Dentro de cada carpeta de un sensor habrá:
  - Una tabla `Time` que contendrá los tiempos del centro de los picos detectados en la señal para ese sensor. Esta tabla tiene como atributos la longitud de las ventanas (en milisegundos) usadas para extraer los picos, la frecuencia mínima y máxima usada en el suavizado FFT de la señal en el algoritmo de detección de picos y la amplitud mínima usada para aceptar un pico.
  - Una tabla `Peaks` que contendrá los picos detectados extraídos directamente de la señal original con una longitud idéntica a la usada en la ventana del algoritmo de identificación. Esta tabla tiene los mismos atributos que `Time`.
  - Una tabla `PeaksFiltered` que contendrá los picos detectados extraídos de la señal filtrada con una longitud idéntica a la usada en la ventana del algoritmo de identificación. Esta tabla tiene los mismos atributos que `Time`.
  - Una tabla `PeaksResample` que tienen los picos aplicando un remuestreo y extrayendo un tamaño de ventana de los picos. Tiene como atributos el tamaño de ventana usado en la extracción (puede ser diferente de la usada en la identificación) y el factor de remuestreo empleado.

- Una tabla `PeaksResamplePCA` con los picos remuestreados, suavizados utilizando un número de componentes del PCA y con la media de una ventana inicial de los datos substraída para nivelar la línea base (baseline) de los picos. La tabla tiene como atributos el número de componentes usados en el PCA para el suavizado (0 si no se ha aplicado PCA) y la longitud de la ventana inicial usada para nivelar el baseline.
- En el caso de que se haya guardado un clustering de los datos, habrá una carpeta `Clustering` que contendrá una tabla `Centers` con los centroides de los clusters ordenados de menor a mayor amplitud

3. Definición de un experimento

Los experimentos se definen usando la clase `Experiment`. Un experimento se compone de:

- `dpath`: Directorio raíz donde se encuentran los datos de los experimentos. Se asume que los datos estan dentro de este directorio en una carpeta que tiene el nombre del experimento
- `name`: Nombre del experimento
- `sampling`: Muestreo de la señal en Hz
- `datafiles`: Nombres de los ficheros originales donde están los datos (los ABF), estos nombres se usaran para organizarlos en el fichero HDF5 y para los resultados
- `sensors`: Lista con los nombres de los sensores. Se supone que estan en ese orden en el fichero original y que el fichero tiene al menos ese numero de columnas de datos.
- `clusters`: Lista con el número de clusters a usar en cada uno de los tramos del experimento. La lista tiene que tener la misma longitud que `datafiles`.
- `colors`: Lista de colores a usar en el histograma que compara la frecuencia de picos a partir de los clusters. La lista tiene que tener la misma longitud que `datafiles`.
- `peaks_id_params`: Diccionario con los parámetros a usar por el algoritmo de detección de picos: `wtime`, `low`, `high`, `threshold`.
- `peaks_resampling`: Diccionario con los parámetros a usar por el algoritmo de resampling: `wsel`, `rsfactor`

Por conveniencia, el fichero `Config/experiments.py` tiene el diccionario `experiments` que contiene las definiciones de todos los experimentos.

4. Preproceso

El preproceso de los datos se compone de los siguientes pasos:

1. `ConvertABF.py`: Importación de los ficheros ABF en un fichero HDF5 (en el caso de usar otro formato para guardar los datos originales se creará un script equivalente). Esto añadirá al fichero del experimento las carpetas para cada tramo del experimento y las tablas `Raw`
2. `PeaksIdentification.py`: Ejecuta el algoritmo de identificación de picos para todo el experimento generando para cada señal las tablas `Time` y `Peaks`. En `Time` esta el tiempo del centro del pico, en `Peaks` están los picos extraídos con un tamaño de ventana igual al tamaño de la ventana de identificación.

Los parámetros de la identificación se definen en el experimento como un diccionario con las claves:

- `wtime`: ventana temporal de detección de picos en milisegunos
- `low`: frecuencia inferior para el filtro FFT
- `high`: frecuencia superior para el filtro FFT
- `threshold`: Amplitud mínima para seleccionar un pico (¿micro voltios?)

El algoritmo usa paralelismo y ejecuta tantos hilos por tramo de experimento como cores tenga la maquina.

3. `PeaksFilterRaw.py`: Opcionalmente se puede filtrar la señal para dejar solo un rango de frecuencias y extraer los picos de la señal filtrada. Este proceso genera las tablas `RawFiltered` para cada tramo del experimento y las tablas `PeaksFiltered` para cada sensor con los picos sacados de la señal filtrada.

4. `PeaksResamling.py`: Hace un remuestreo de la señal y extrae una ventana de los picos (puede ser más pequeña que la de identificación). Este proceso genera las tablas `PeaksResample`.

   Estos parámetros se definen en el experimento usando un diccionaro con las claves:

   - `wsel`: ventana de selección en milisegundos
   - `rsfactor`: factor de resampling
   - `filtered`: si se usa la señal original o la filtrada (`True` o `False`)

   El script usa paralelismo.

5. `PeaksPCA.py`: Aplica un suavizado mediante PCA y mueve la señal para que su baseline inicial este alrededor del cero (restando la media de una ventana inicial de la señal). Estos parámetros se definen en el experimento usando un diccionaro con las claves:

   - `pcasmooth`: Si se aplica el suavizado mediante PCA
   - `componentsw`: Cuantos componentes de PCA usar para la reconstrucción
   - `baseline`: Tamaño de ventana desde el inicio del pico (en puntos) a usar para modificar el baseline