Machine Learning en Biomedicina: Aprendizaje no supervisado

Medicina Computacional

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

1. Introducción

En este guion de prácticas vamos a aprender a utilizar técnicas de inteligencia Artificial para el procesamiento de datos biológicos. Concretamente vamos a aprender a utilizar técnicas de Machine Learning – o en español, Aprendizaje Automático.

El aprendizaje automático es una rama de la Inteligencia Artificial que utiliza algoritmos para aprender a reconocer patrones en un conjunto de datos y realizar predicciones a partir de esos patrones. Los patrones que el algoritmo ha aprendido nos pueden proporcionar información sobre las características de las medidas con los que estamos trabajando y hacer predicciones sobre nuevas medidas que realicemos en el futuro.

Hay dos grandes grupos de técnicas de aprendizaje automático: el aprendizaje supervisado y el aprendizaje no supervisado.

Aprendizaje supervisado

En el aprendizaje supervisado disponemos de un conjunto de casos de los cuales conocemos su clasificación. Por ejemplo, si queremos aprender a reconocer si una planta está enferma a partir de una fotografía tendríamos varias (muchas) fotografías y sabríamos cuales corresponden a plantas sanas y cuales a plantas enfermas. En este caso se dice que nuestros datos están *etiquetados*. Al algoritmo que nos permite clasificar los distintos casos que tenemos lo llamamos *clasificador*.

Un ejemplo típico en el campo de la genómica es tener un conjunto de transcriptomas asociados a una cierta enfermedad y trascriptomas de personas sanas. Utilizando técnicas de aprendizaje automático podrías conseguir que tu clasificador aprenda a distinguir los patrones genéticos de esta enfermedad.

El aprendizaje supervisado tiene dos fases: la fase de entrenamiento y la fase de comprobación.

- Entrenamiento: La primera fase del aprendizaje automático supervisado consiste en entrenar a nuestro clasificador utilizando parte de los datos etiquetados de los que disponemos. De esta manera el clasificador aprenderá a reconocer los patrones existentes en los datos que hemos tomado.
- Comprobación: Después de entrenar a nuestro algoritmo debemos comprobar que funciona bien en un conjunto de datos *distinto del original.*

Aprendizaje no supervisado

En el aprendizaje no supervisado disponemos de un conjunto de casos de los cuales pero no conocemos su clasificación. En el ejemplo sobre el reconocimiento de las plantas enfermas varias (muchas) fotografías pero no sabríamos cuales corresponden a plantas sanas y cuales a plantas enfermas.

Los algoritmos de aprendizaje funcionan agrupando casos con características similares, a este proceso de agrupamiento se le llama *clustering*.

El aprendizaje automático en las ciencias de la vida.

Los orígenes del aprendizaje automático se encuentran en la biología. Algunas de los primeros sistemas de aprendizaje automático, conocidos como redes neuronales artificiales, se basan en lo que sabemos del funcionamiento del cerebro.

Hoy en día el aprendizaje automático se utiliza parar resolver una gran cantidad de problemas en el ámbito de ciencias de la vida. Algunas aplicaciones comunes son:

- Análisis de expresión genética para predecir enfermedades.
- Predicción de la interacción entre proteínas.
- Estudio de sistemas neuronales.

En este guion utilizaremos distintas técnicas de aprendizaje automático para analizar un conjunto de datos neuronales asociados a sistema visual humano.

1.1. Aprendizaje no supervisado: Técnicas de clustering

Un algoritmo de clustering es un procedimiento de agrupación de una serie de datos de acuerdo con un criterio. Esos criterios son por lo general distancia o similitud entre los distintos datos. Este tipo de algoritmos son de aprendizaje no supervisado por lo que no necesitamos etiquetar los datos.

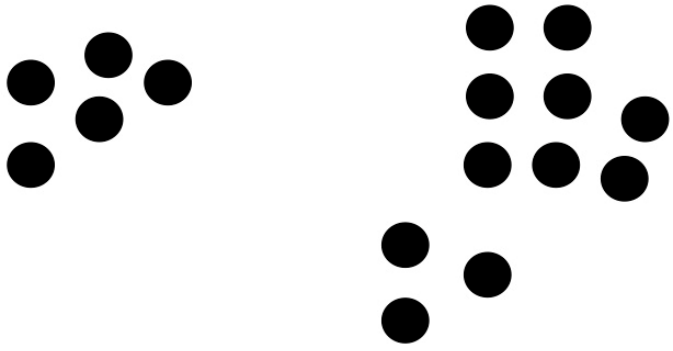
En esta sección vamos a trabajar con un algoritmo de clustering muy común, el algoritmo de k-medias. Este algoritmo divide el conjunto de datos en k diferentes clusters. K es el número de clusters, lo elige el usuario antes de utilizar el algoritmo. El algoritmo de k-medias divide los distintos casos de manera que cada uno se agrupo con los casos más parecidos a él (en terminología técnica cada punto en un cluster tiene la distancia mínima a la media del cluster).

Matematicamente se dice que queremos asignar los datos, a los que llamaremos x, a k conjuntos distintos, a los que llamaremos S, minimizando la siguiente función:

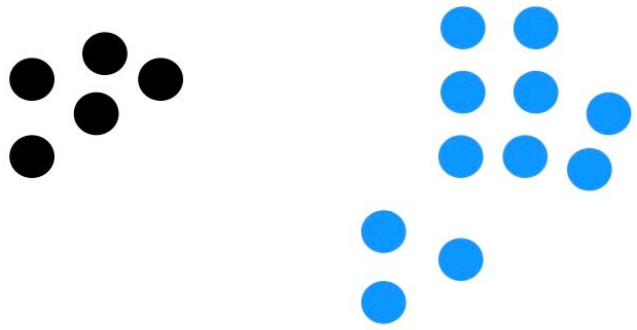
$$\sum_{i=1}^{k} \sum_{x_j \in S} ||x_j - \mu_i||^2$$

Si no la fórmula que acabas de ver no te ha aclarado nada no te preocupes. Vamos a ver ahora un ejemplo intuitivo.

Supongamos que hemos obtenido los siguientes datos en un experimento y los queremos agrupar:

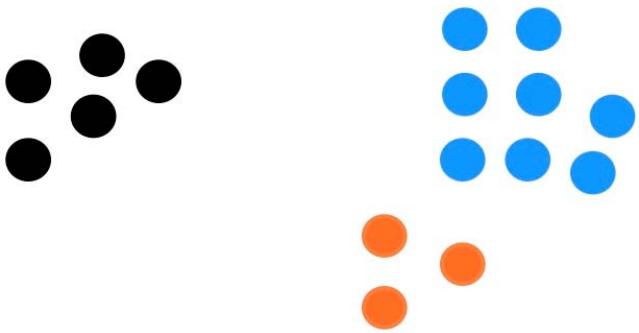


Si pensásemos que sólo hay dos clases distintos entonces agruparíamos los datos en dos clusters asignando a k el valor dos. El algoritmo de k-medias nos agruparía los datos de la siguiente manera:



k = 2

Si luego pensásemos que realmente había tres casos distintos entonces haríamos k = 3, y el resultado sería el siguiente:



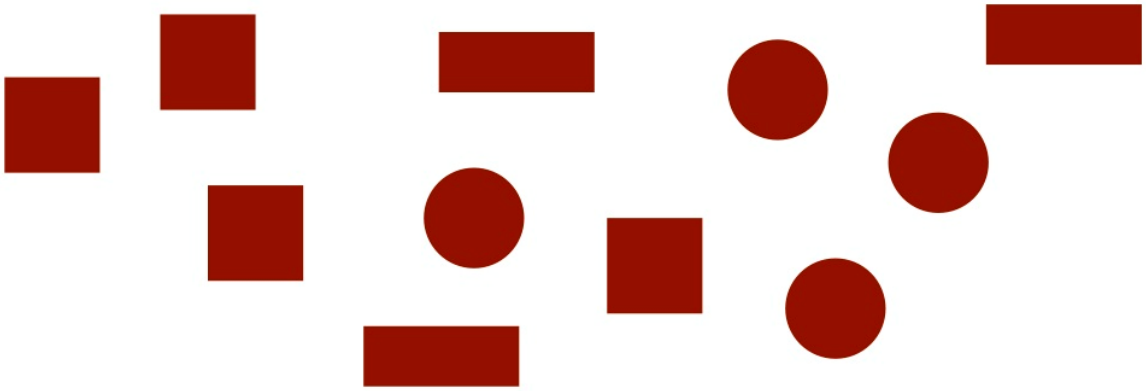
k = 3

Una cosa importante a tener en cuenta con este algoritmo es que la agrupación de los datos depende del número de clusters k. Según el número de clusters el algoritmo k-medias agrupará juntos los casos más parecidos.

Características de los datos

El algoritmo k-medias agrupa los distintos datos en función de sus características. Las características son las propiedades de los datos que utilizamos para clasificarlos.

Por ejemplo, los objetos que ves abajo se podrían clasificar en función de la forma. En teoría también podríamos clasificarlos usando otras propiedades como el área, el color… pero en este caso no resultaría muy útil.



Un ejemplo típico de uso de técnicas de clustering en biología es utilizar la expresión genética de distintos tejidos para agruparlos por su similitud.

Los cuadros azules

Los cuadros azules contienen preguntas o reflexiones. Escribe las preguntas y tus respuestas en un ***Jupyter notebook*** *con nombre* ***`ml_non_supervised.ipynb`*** *que subirás a la plataforma online al final de la clase.*

Incluye también en el notebook el código que desarrolles para resolver las distintas secciones de la práctica.

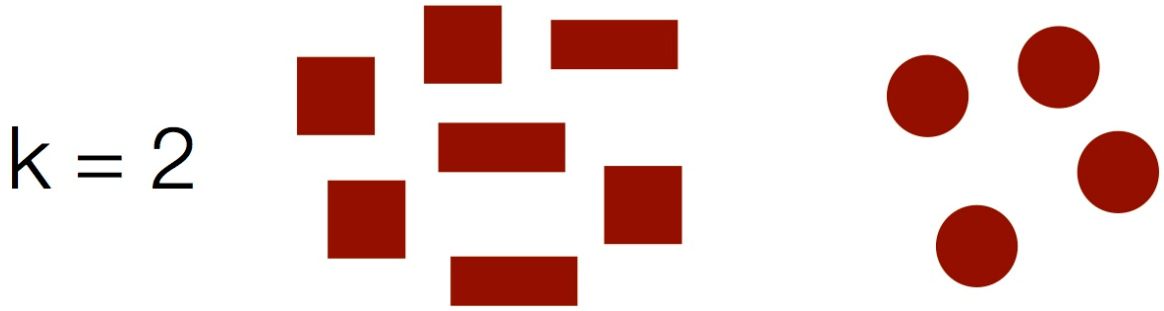
Para hacer la práctica más interesante ***hemos incluido varios errores en el código****. Encuéntralos, corrígelos e índica en el Jupyter notebook donde los has encontrado.*

Pensemos un poco

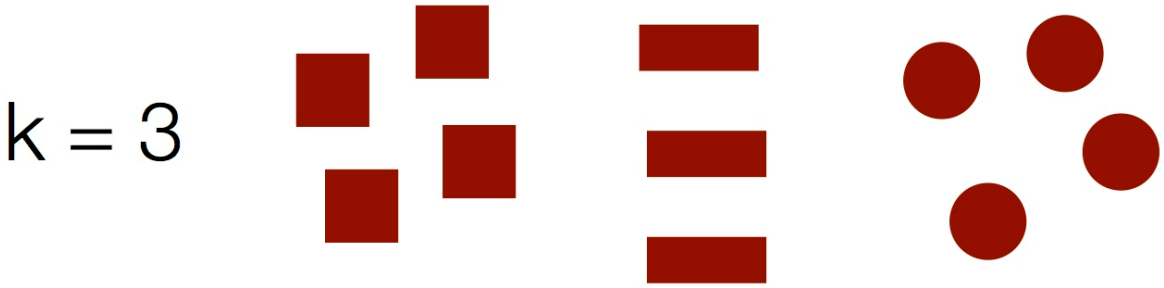
Si quisiésemos agrupar los polígonos que hemos visto arriba utilizando el algoritmo k-medias con k = 2 ¿Cuáles serían los dos clusters?

- Círculos y cuadrados en un grupo y rectángulos en el otro.
- Círculos y rectángulos en un grupo y cuadrados en el otro.
- Rectángulos y cuadrados en un grupo y círculos en el otro.

Los cuadrados y los rectángulos tienen la forma más parecida y se agruparían juntos. Por lo que los dos grupo serían:



Si en vez de dos grupos quisiésemos tres entonces círculos, rectángulos y cuadrados tendría cada uno su grupo. Es importante que comprendas que los resultados de un algoritmo de clustering dependen del número de clusters que especifiques.



Pensemos un poco

¿Qué gran inconveniente le ves a los algoritmos de clustering?

Uno de los principales inconvenientes de utilizar este tipo de algoritmos es que necesitas saber a priori cuantos grupos distintos hay. Si dispones de esa información este tipo de algoritmos es muy potente.

2. Ejercicio: Machine Learning para analizar medidas neuronales.

En esta sección vamos a utilizar las técnicas que hemos visto previamente para analizar los datos de un experimento con un magnetoencefalograma (MEG).

Un MEG utiliza 306 sensores en torno a la cabeza para medir los campos magnéticos que se crean cuando varias neuronas se disparan simultáneamente. En la foto que hay abajo puedes ver a un sujeto sentado en el escáner MEG. El instrumento se coloca sobre su cabeza y durante el experimento ve imágenes en la pantalla que está frente a él.

En el experimento que vamos a analizar el sujeto observó 25 paisajes distintos que pertenecen a cinco categorías: playa, ciudad, bosque, autopista y montaña. Mientras veía cada una de estas imágenes se medía su actividad cerebral.

Vamos a analizar las medidas obtenidas con el MEG aplicando las técnicas de aprendizaje automático que acabamos de ver. Posteriormente intentaremos predecir a partir de una medida que tipo de datos el sujeto está viendo.

Dimitrios Pantazis, PhD, Research Scientist, Director of MEG Lab, McGovern Institute for Brain Research

2.1. Formato de los datos experimentales

En este ejercicio utilizaremos Jupyter para analizar los datos del experimento. Siempre que trabajemos con datos experimentales lo primero que tenemos que hacer es cargarlos en Jupyter y *comprender bien su organización.*

En nuestro caso tenemos tres ficheros distintos: 'MEG_data.csv', 'stim_ID.csv', and 'cat_ID.csv'

- MEG_data.csv contiene una matriz con 125 filas que corresponde a 125 visionados de una imagen. Esta matriz tiene 306 columnas, que corresponden a las medidas de los 306 sensores del MEG.

	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Sensor 7	Senso
Experiment 1	2,81E-10	8,36E-09	-4,86E-09	-5,15E-11	1,99E-08	1,83E-08	4,33E-10	
Experiment 2	1,71E-10	2,64E-09	3,30E-09	-5,25E-10	2,96E-08	-1,38E-08	-3,61E-10	
Experiment 3	4,73E-10	8,61E-09	8,29E-09	5,17E-11	1,34E-08	1,27E-08	3,43E-10	
Experiment 4	-1,22E-10	-1,06E-08	2,69E-09	2,90E-10	2,15E-09	-2,40E-08	-2,33E-11	
Experiment 5	-4,19E-10	5,63E-09	1,30E-08	-9,69E-10	1,36E-08	-1,91E-08	-7,60E-10	
Experiment 6	1,98E-10	-4,69E-09	2,33E-08	-7,87E-11	-4,67E-09	-2,47E-08	-5,10E-10	
Experiment 7	-8,52E-10	-7,75E-13	3,73E-09	1,15E-10	-1,97E-08	-1,52E-09	7,78E-11	
Experiment 8	-7,53E-10	-9,32E-09	7,48E-09	3,00E-10	-1,58E-08	-2,75E-08	-4,36E-10	
Experiment 9	-1,83E-11	-3,11E-09	1,17E-08	-6,04E-12	1,04E-08	-7,13E-09	-3,24E-10	
Experiment 10	-5,25E-10	3,84E-09	-1,81E-08	7,56E-10	-2,90E-08	2,93E-09	1,53E-10	
Experiment 11	-9,50E-10	-1,00E-08	-2,55E-08	5,08E-10	-3,24E-08	4,43E-09	9,79E-11	
Experiment 12	-3,94E-10	9,39E-09	2,17E-08	-7,53E-10	-2,43E-09	-1,70E-08	-7,85E-11	
Experiment 13	-3,27E-10	-1,24E-08	-2,48E-08	6,93E-10	1,51E-09	-8,80E-09	9,76E-12	
Experiment 14	-6,10E-11	-1,63E-08	-3,09E-08	2,56E-10	-4,45E-09	-1,68E-08	-2,68E-11	
Experiment 15	-9,71E-10	2,54E-09	1,77E-08	-5,27E-10	-2,51E-09	-8,83E-09	-4,03E-10	
Experiment 16	-3,42E-11	4,87E-09	1,09E-08	-2,80E-10	-3,00E-09	-5,04E-09	-3,11E-10	

- stim_ID.csv contiene un vector de longitud 125 que contiene qué imagen de las veinticinco (valores 1-25) estaba viendo el sujeto en esa ocasión.

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5	Experiment 6	Experiment 7	Exper
Imagen	1	1	1	1	1	2	2	

- cat_ID.csv contiene un vector de longitud 125 que contiene la categoría a la que corresponde la imagen en stim_ID.csv ('playa', 'ciudad', 'bosque', 'autopista', o 'montaña', ordenado de 1 a 5).

	Image 1	Image 2	Image 3	Image 4	Image 5	Image 6	Image 7	Image
Type	1	1	1	1	1	1	1	

Comenzaremos cargando los distintos ficheros utilizando la librería pandas y el comando: `read_csv`.

```
# Cargamos la libreria pandas con alias 'pd'
import pandas as pd

# Leemos los datos del fichero MEG_data.csv
MEG_data = pd.read_csv("data/MEG_Data.csv", header=None)

# Comprobamos que lo hemos cargado correctamente
MEG_data.head()
```

La fila i de la matriz `MEG_data`, `MEG_data.iloc[i,:]`, contiene las 306 medidas de la fila i. Es decir las medidas asociadas a la imagen etiquetada con `stim_ID.iloc[i]` y perteneciente a la `cat_ID.iloc[i]`.

La columna j de la matriz `MEG_data`, `MEG_data.iloc[:,j]`, contiene la respuesta del sensor j a las 125 imágenes.

El valor `MEG_data.iloc[i,j]` contiene la respuesta del sensor j cuando se le muestra al sujeto la imagen i.

Si queremos obtener los datos de las imágenes de i a k-1 escribiremos `MEG_data.iloc[i:k,:]`. Si queremos obtener los datos de las imágenes i, j y k escribiremos `MEG_data.iloc[[i,j,k],:]`.

Ejercicios

1. Muestra las medidas asociadas a la imagen 33
2. Muestra las medidas tomadas por el sensor 45
3. Muestra la medida tomada por el sensor 128 para la imagen 89
4. Muestra de una sola vez las medidas asociadas a las imágenes 56 a 62
5. Muestra de una sola vez las medidas tomadas por los sensores 22 a 27
6. Muestra de una sola vez las medidas asociadas a las imágenes 17 y 25
7. Muestra de una sola vez las medidas tomadas por los sensores 34 a 77

Exploremos los datos

```
# Cargamos la librería pyplot
import matplotlib.pyplot as plt
%matplotlib inline

# Realizamos unas gráfica de exploración
plt.ylim(-10**-11, 10**-11)
plt.plot(MEG_data.iloc[:,200])
plt.show()
plt.plot(MEG_data.iloc[:,233])
plt.show()
```

Pensemos un poco

¿Qué puedes deducir sobre los datos a partir de las gráficas que hemos obtenido?

2.2. Analicemos las medidas utilizando k-medias

Vamos a usar el algoritmo k-medias para agrupar nuestros datos. Para simplificar los resultados vamos a trabajar únicamente con dos sensores que miden zonas de procesamiento visual: el sensor 200 y el sensor 233.

Para ello creamos la matriz X que contendrá únicamente la medida de estos dos sensores.

Pensemos un poco

Si queremos agrupar las distintas categorías de imágenes ¿Qué valor debe tomar k?

La librería scikit-learn contiene una función para realizar agrupamiento por k-medias. Dicha función se llama `kmeans`.

```
# Seleccionamos los datos que nos interesan
samples = MEG_data.iloc[:,[200,233]]

# Cargamos la función KMeans de la libreria scikit-learn
from sklearn.cluster import KMeans

# Especificamos el número de clusters que queremos utilizar
model = KMeans(n_clusters = 5)

# Realizamos el clustering
model.fit(samples)

# Obtenemos el cluster para cada elemento
labels = model.predict(samples)

# Le damos un formato adecuado
clustering = pd.DataFrame(labels)

# Mostramos el resultado
print(clustering)
```

Una vez ejecutado el comando ¿Qué contiene labels?

Ahora vamos a dibujar los resultados. Queremos mostrar en un gráfico todos los puntos del experimento dándole colores distintos a cada uno de los grupos. Os damos una plantilla que podéis utilizar como guía.

```
# Cargamos la librería pyplot
import matplotlib.pyplot as plt
%matplotlib inline

# Obtenemos la coordenada x e y de cada muestra
xs = samples.iloc[:,0]
ys = samples.iloc[:,1]

# Realizamos la gráfica
plt.ylim(0, 10**-11)
plt.xlim(-10**-11, 10**-11)
plt.scatter(xs,ys, c = labels)
```

Pensemos un poco

¿Qué muestra el gráfico?

¿Qué conclusión sacas?

Supongamos ahora que tenemos una nueva muestra. Vamos a intentar predecir qué tipo de imagen es

```
# Definimos la nueva muestra
new_sample = [[7.282100e-12, 4.252600e-12]]

# Vemos a ver a que cluster pertenece
new_labels = model.predict(new_sample)

# Mostramos el resultado
print(new_labels)
```

2.3. Reproducibilidad de los resultados

En ciencia se dice que un experimento es reproducible si los resultados que ha obtenido un investigador pueden ser reproducidos por otro investigador en otro laboratorio. Vamos a ver si los resultados obtenidos en nuestro ejercicio son reproducibles.

Vuelve a ejecutar el programa del apartado anterior desde el principio ¿Ves alguna diferencia?

Pensemos un poco

Ejecuta el programa del apartado anterior diez veces ¿Qué hechos interesantes observas?

¿A qué crees que se puede deber? Si no lo sabes consulta la documentación de la función kmeans:
http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

2.4. Vamos a guardar los resultados del análisis

Para concluir vamos a guardar los resultados que hemos obtenido. Vamos a guardar las muestras que hemos utilizado, el clustering obtenido y la gráfica que hemos generado.

Procedemos a guardar los resultados

```
# Almacenamos en el fichero samples.csv las muestras utilizadas
samples.to_csv("results/samples.csv",index=False)

# Almacenamos en el fichero clustering.csv los clusters obtenidos
clustering.to_csv("results/clustering.csv",index=False)


# Guardamos la gráfica generada
plt.ylim(0, 10**-11)
plt.xlim(-10**-11, 10**-11)
plt.scatter(xs,ys, c = labels)
plt.savefig ("results/clustering.png")
```

2.5. Una última cuestión.

Hemos comprobado que podíamos dividir las medidas de actividad cerebral en cinco clusters.

Pero para ver la precisión del sistema nos falta un paso: comprobar si los clusters que hemos obtenido coinciden con las categorías de las imágenes del experimento.

Mira en qué grado los clusters coinciden con las categorías de imágenes. ¿Qué conclusión sacas de estos resultados?

Pista: Mira la ayuda de la función `crosstab`

```
# Leemos los datos del fichero cat_ID
cat_ID = pd.read_csv("data/cat_ID.csv", header=None)

# Realizamos una crostabulación de las categorías con los
clusters
pd.crosstab(cat_ID.iloc[0], labels)
```

3. Ejercicio: El cambio diaúxico de la Saccharomyces.

Uno de los primeros organismos vivos que los seres humanos consiguieron domesticar fue la levadura. Ya hace 6000 años, en Armenia se producía vino: unos investigadores encontraron en una cueva armenia una bodega completa con una prensa de vino, recipientes para la fermentación e incluso copas para beberlo.

La producción del vino requiere un conocimiento profundo sobre cómo controlar al saccharomyces, el género de la levadura que se utiliza en la producción de alcohol y pan. En este apartado vamos a trabajar con la Saccharomyces cerevisiae, capaz de producir vino porque puede metabolizar la glucosa que se encuentra en la fruta en etanol.

Empezaremos este apartado con una pregunta sencilla: Si la S. cerevisiae vive con frecuencia en los viñedos *¿por qué es necesario sellar los barriles herméticamente para producir vino?*

La razón es sencilla: una vez que la glucosa disponible se agota la S. cerevisiae debe consumir algún otro producto para sobrevivir. Para ello transforma su metabolismo y convierte el etanol que acaba de producir en su nueva fuente de alimento.

Este cambio metabólico, conocido como cambio diaúxico solo se puede dar en presencia de oxígeno. Si al agotarse la glucosa no hay presencia de oxígeno S. cerevisiae entra en estado de hibernación.

En nuestro caso, los productores de vino tienen que cerrar herméticamente los barriles para que no haya presencia de oxígeno. Si el oxígeno entra en el barril la levadura entra en cambio diaúxico y comienza metabolizando el etanol estropeando el vino.

El cambio diaúxico es un cambio muy complejo que involucra una gran cantidad de genes. Si queremos encontrar qué genes están involucrados y cual es su función es necesario analizar la expresión genética de los distintos genes durante el proceso.

Gene	Expression Vector						
YLR361C	0.14	0.03	-0.06	0.07	-0.01	-0.06	-0.01
YMR290C	0.12	-0.23	-0.24	-1.16	-1.40	-2.67	-3.00
YNR065C	-0.10	-0.14	-0.03	-0.06	-0.07	-0.14	-0.04
YGR043C	-0.43	-0.73	-0.06	-0.11	-0.16	3.47	2.64
YLR258W	0.11	0.43	0.45	1.89	2.00	3.32	2.56
YPL012W	0.09	-0.28	-0.15	-1.18	-1.59	-2.96	-3.08
YNL141W	-0.16	-0.04	-0.07	-1.26	-1.20	-2.82	-3.13
YJL028W	-0.28	-0.23	-0.19	-0.19	-0.32	-0.18	-0.18
YKL026C	-0.19	-0.15	0.03	0.27	0.54	3.64	2.74
YPR055W	0.15	0.15	0.17	0.09	0.07	0.09	0.07

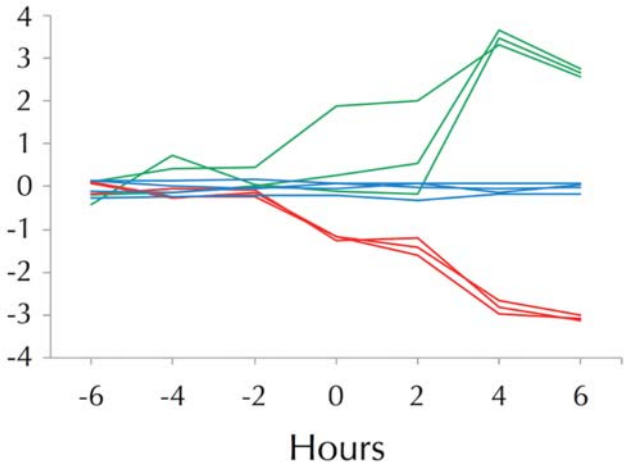
Si analizamos la expresión genética de distintos genes podemos observar que hay distintos tipos de comportamiento. Por ejemplo, el gen YPL012W disminuye su expresión durante el cambio diaúxico, el gen YLR258W aumenta su expresión durante el cambio diaúxico y el gen YPR055W permanece más o menos constante.

Realizar este tipo de análisis a mano para todos los genes sería costoso y daría lugar a una poca reproducibilidad de los resultados. Podemos utilizar las técnicas de análisis no supervisado para realizar el proceso de manera automática. Así es más rápido y reproducible.

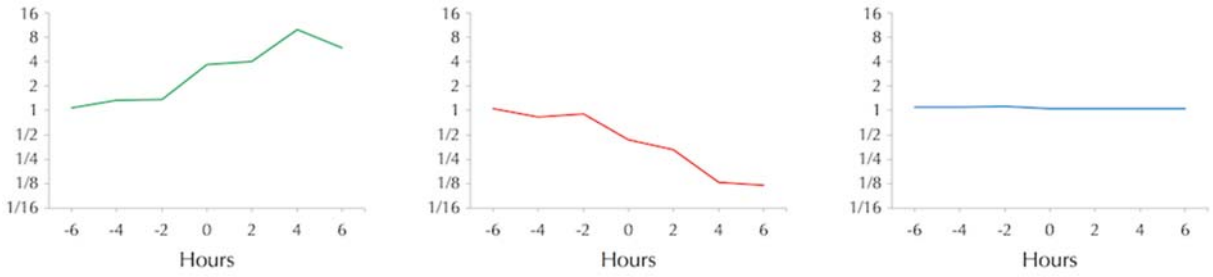
Si utilizamos técnicas de clustering sobre el conjunto de los genes podremos clasificarlos en varios tipos. Por ejemplo, si fijamos como objetivo tres clusters obtendríamos una clasificación similar a esta:

YLR361C	**0.14**	0.03	-0.06	0.07	-0.01	-0.06	-0.01
YMR290C	0.12	-0.23	-0.24	-1.16	-1.40	-2.67	**-3.00**
YNR065C	-0.10	**-0.14**	-0.03	-0.06	-0.07	-0.14	-0.04
YGR043C	-0.43	-0.73	-0.06	-0.11	-0.16	**3.47**	2.64
YLR258W	0.11	0.43	0.45	1.89	2.00	**3.32**	2.56
YPL012W	0.09	-0.28	-0.15	-1.18	-1.59	-2.96	**-3.08**
YNL141W	-0.16	-0.04	-0.07	-1.26	-1.20	-2.82	**-3.13**
YJL028W	-0.28	-0.23	-0.19	-0.19	**-0.32**	-0.18	-0.18
YKL026C	-0.19	-0.15	0.03	0.27	0.54	**3.64**	2.74
YPR055W	0.15	0.15	**0.17**	0.09	0.07	0.09	0.07

Podemos observar que los genes coloreados de verde son genes que aumentan su expresión durante el cambio diaúxico, que los genes coloreados de azul permanecen constantes y los genes coloreados de rojo disminuyen su expresión.



Podemos aumentar la potencia del análisis si hacemos la media de todos los genes de cada cluster. De esta manera disminuimos el ruido y hacemos más potente la señal detectada.



Este análisis que hemos descrito lo realizaremos utilizando las técnicas aprendidas en este guion.

3.1. Formato de los datos experimentales

Al igual que en la sección anterior comenzaremos nuestro

- levadura_data.csv contiene una matriz con 230 filas que corresponde a 230 genes. Esta matriz tiene 7 columnas, que corresponden a las medidas de tomadas en siete instantes en torno al cambio diaúxico.

0,1361	-0,1110	-0,1890	-0,7824	-0,7570	-0,8560	-2,3045
-0,2869	-0,0841	0,1844	0,1361	0,5353	2,3219	1,2515
-0,0426	-0,0976	-0,0144	-0,7991	-0,8400	-2,2479	-2,3868
-0,0566	-0,1243	-0,0704	-0,5460	-0,5558	-2,1043	-2,3674
0,0145	0,0740	0,0589	0,0145	0,2515	2,3959	1,6897
0,1361	-0,0426	-0,2987	-0,8718	-1,1440	-2,4222	-2,7225
0,0740	0,1361	0,4344	0,8365	0,9434	3,6439	3,3219
-0,1890	-0,0286	0,5995	0,8890	0,8365	2,5564	2,6439
-0,2510	0,1361	-0,2265	0,8625	0,6215	2,5564	2,3959
-0,1110	0,0589	0,0145	0,1047	0,3040	2,7370	3,3219
-0,0286	-0,0976	0,3585	0,4344	0,3771	2,5564	1,2515
0,0740	0,3040	0,4941	0,4739	0,3771	2,3219	1,4739
0,1844	0,4344	0,9434	1,5146	1,6439	2,3219	1,5995
0,1520	0,7612	1,2863	1,9434	1,5146	2,1844	2,6439
0,0439	0,0145	0,3959	-0,1635	-0,2750	-1,1110	-2,4436
0,1047	0,1520	0,1047	0,0589	0,0439	-0,6041	-2,3646
-0,0144	-0,0841	-0,1243	-1,0356	-1,1243	-2,0215	-2,7991

- levadura_ID.csv contiene un vector de longitud 230 que contiene la identidad de los 230 genes que se han analizado en este estudio.

YDR025W		
YDR031w		
YDR060w		
YDR064W		
YDR070c		
YDR144C		
YDR171W		
YDR178W		
YDR258C		
YBL015W		
YDR272W		
YDR272W		
YDR342C		
YDR343C		
YBL027W		
YDR382W		

Comenzaremos cargando los distintos ficheros utilizando la librería pandas y el comando: `read_csv.`

```
# Cargamos la libreria pandas con alias 'pd'
import pandas as pd

# Leemos los datos del fichero levadura_data.csv
levadura_data = pd.read_csv("data/levadura_data.csv",
header=None)

# Comprobamos que lo hemos cargado correctamente
levadura_data.head()
```

La fila i de la matriz `levadura_data`, `levadura_data.iloc[i,:]`, contiene las 7 medidas de la fila i. Es decir las medidas asociadas al gen `levadura_ID.iloc[i]`.

La columna j de la matriz `levadura_data`, `levadura_data.iloc[:,j]`, contiene las medidas en el instante j en los 230 genes.

El valor `levadura_data.iloc[i,j]` contiene la respuesta del gen i en el instante j.

Exploremos los datos

```
# Realizamos unas gráfica de exploración
plt.ylim(-4,4);
plt.plot(levadura_data.iloc[55,:])
plt.show()
plt.plot(levadura_data.iloc[120,:])
plt.show()
```

Pensemos un poco

¿Qué puedes deducir sobre los datos a partir de las gráficas que hemos obtenido?

3.2. Analicemos las medidas utilizando k-medias

Vamos a usar el algoritmo k-medias para agrupar nuestros datos. El proceso es muy similar al seguido en el apartado anterior, por lo que no entraremos muy en detalle.

Realizamos el clustering utilizando la función `kmeans` de la librería scikit-learn.

```
# Seleccionamos los datos que nos interesan
samples_lev = levadura_data.iloc[:,:]
# Cargamos la función KMeans de la libreria scikit-learn
from sklearn.cluster import KMeans

# Especificamos el número de clusters que queremos utilizar
model = KMeans(n_clusters = 6)
# Realizamos el clustering
model.fit(samples_lev)

# Obtenemos el cluster para cada elemento
labels_lev = model.predict(samples_lev)
# Le damos un formato adecuado
clustering_lev = pd.DataFrame(labels_lev)
# Mostramos el resultado
print(clustering_lev)
```

Pensemos un poco

¿Por qué todas las variables que estoy utilizando llevan el sufijo _lev?

Vamos a analizar los resultados obtenidos.

```
# Queremos separar los genes por su tipo de cluster
lev_cluster_1 = samples_lev.iloc[labels_lev == 0,:]
lev_cluster_2 = samples_lev.iloc[labels_lev == 1,:]
lev_cluster_3 = samples_lev.iloc[labels_lev == 2,:]
lev_cluster_4 = samples_lev.iloc[labels_lev == 3,:]
lev_cluster_5 = samples_lev.iloc[labels_lev == 4,:]
lev_cluster_6 = samples_lev.iloc[labels_lev == 5,:]


# Ahora generamos las gráficas
import matplotlib.pyplot as plt

plt.ylim(-4,4); plot1 = plt.plot(lev_cluster_1.transpose())
plt.show();
plt.ylim(-4,4); plot2 = plt.plot(lev_cluster_2.transpose())
plt.show();
plt.ylim(-4,4); plot3 = plt.plot(lev_cluster_3.transpose())
plt.show();
plt.ylim(-4,4); plot4 = plt.plot(lev_cluster_4.transpose())
plt.show();
plt.ylim(-4,4); plot5 = plt.plot(lev_cluster_5.transpose())
plt.show();
plt.ylim(-4,4); plot6 = plt.plot(lev_cluster_6.transpose())
plt.show();
```

Pensemos un poco

¿Qué puedes ver en los gráficos? ¿Cómo describirías cada uno de los clusters?

Ejercicio

¿Cómo obtendrías el nombre de los genes que pertenecen a cada cluster?

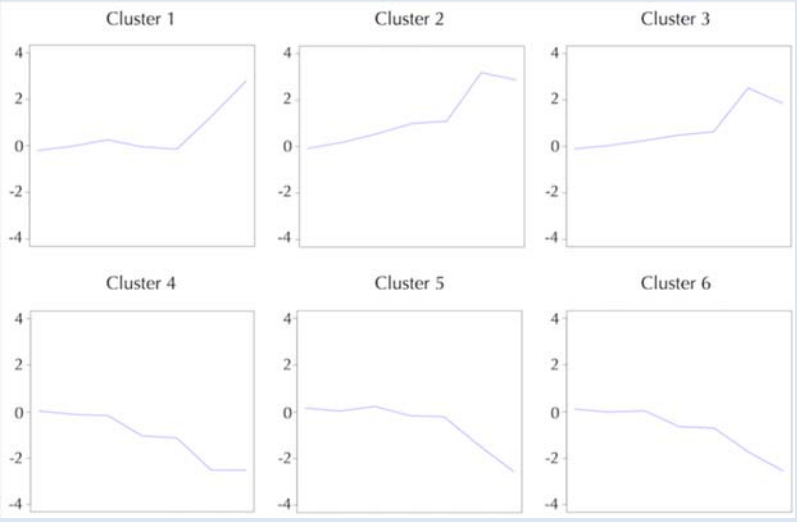
Procesemos algo más los datos.

```
# Vamos a dibujar la media de todos los genes de cada cluster
# Primero calculamos la media para cada cluster.
lev_cluster_1_mean = lev_cluster_1.mean()
lev_cluster_2_mean = lev_cluster_2.mean()
lev_cluster_3_mean = lev_cluster_3.mean()
lev_cluster_4_mean = lev_cluster_4.mean()
lev_cluster_5_mean = lev_cluster_5.mean()
lev_cluster_6_mean = lev_cluster_6.mean()

# Ahora generamos las gráficas
plt.ylim(-4,4); plot1 = plt.plot(lev_cluster_1_mean.transpose())
plt.show()
plt.ylim(-4,4); plot2 = plt.plot(lev_cluster_2_mean.transpose())
plt.show()
plt.ylim(-4,4); plot3 = plt.plot(lev_cluster_3_mean.transpose())
plt.show()
plt.ylim(-4,4); plot4 = plt.plot(lev_cluster_4_mean.transpose())
plt.show()
plt.ylim(-4,4); plot5 = plt.plot(lev_cluster_5_mean.transpose())
plt.show()
plt.ylim(-4,4); plot6 = plt.plot(lev_cluster_6_mean.transpose())
plt.show()
```

Pensemos un poco

¿Las conclusiones que obtienes ahora son similares a antes de realizar la media? ¿Coinciden tus gráficas con estas gráficas?



¿A qué crees que se puede deber la diferencia?

3.3. Para terminar.

Guarda los resultados que hemos obtenido. Estamos interesados en las muestras y el clustering. También queremos guardar las gráficas de la expresión media de cada cluster.

Una vez completado el ejercicio deberías tener dos ficheros csv y seis gráficas png.

4. ¿Qué hemos aprendido?

- Hemos aprendido qué es el Aprendizaje Automática (Machine Learning).
- Conocemos las diferencias entre el aprendizaje supervisado y el no supervisado.
- Conocemos los fundamentos del algoritmo de k-medias.
- Sabemos cómo realizar un análisis de clustering de datos biológicos utilizando Jupyter.

Referencias

La primera parte de este guion se basa en parte en la clase *Neuroscience using Machine Learning Concepts* impartida por Leyla Isik en el workshop *7.QBWx: the Quantitative Biology Workshop del MIT*. Leyla Isik en un estudiante de doctorado en el Poggio Lab en MIT.

La segunda parte de este guion se basa en parte en Compeau, P., & Pevzner, P. (2015). *Bioinformatics algorithms: an active learning approach* (Vol. 1). La Jolla, California: Active Learning Publishers.