Ciencia de Datos 2022

Lista 03

19.marzo.2022

1. Usamos para $t-SNE$ la distancia de Kullback-Leibler. Para distribuciones discretas su definición es:

$$d_{KL}(\mathbb{P}_1||\mathbb{P}_2) = \sum_{\omega} \mathbb{P}_1(\omega) \log \frac{\mathbb{P}_1(\omega)}{\mathbb{P}_2(\omega)}.$$

Calcular $d_{KL}(\mathbb{P}_1||\mathbb{P}_2)$ si $\mathbb{P}_1 \sim Ber(p_1)$ y $\mathbb{P}_2 \sim Ber(p_2)$. Para p_1 fija, graficar $d_{KL}(\mathbb{P}_1||\mathbb{P}_2)$ como función de p_2, y verificar que efectivamente mide de alguna manera la disimilitud entre $\mathbb{P}_1$ y $\mathbb{P}_2$.

Información adicional: En clase mencionamos la propiedad que $d_{KL}(\mathbb{P}_1||\mathbb{P}_2) > 0$. Interesados pueden encontrar una demostración de esta propiedad en el libro de Bishop, sección 1.6.1 para el caso continuo.

No es tan evidente por qué definir $d_{KL}(\mathbb{P}_1||\mathbb{P}_2)$ de esta forma. ¿Por qué no tomar simplemente la distancia L_2, $||\mathbb{P}^1 - \mathbb{P}^2||_2$, donde $\mathbb{P}^i$ es la distribución discreta vista como un vector donde las entradas corresponden a las probabilidades de la distribución $(\mathbb{P}_i(\omega))$?

Una motivación es que con esta distancia se pesa igual una diferencia de 0.9 y 0.8 que una diferencia de 0.1 y 0.2 (en ambos casos, la diferencia es 0.1). Sin embargo de manera relativa en el último caso una es el doble de la otra. Muchas veces se tiene más bien interés en medir errores relativos como aparecen en la Kullback-Leibler.

Otra motivación es que conecta con conceptos como la estimación de máxima verosimilitud. Si $\mathbb{P}_2$ es una distribución $\mathbb{P}_\theta$ con θ un parámetro por estimar y $\mathbb{P}$ es la distribución empírica de una muestra $\{x_i\}$, entonces buscar θ que maximiza la verosimilitud de la muestra bajo $\mathbb{P}_\theta$ es equivalente a buscar θ que minimiza la distancia de Kullback-Leibler $d_{KL}(\mathbb{P}_\theta||\mathbb{P})$ de la muestra. Convencerse de eso (hint: la distribución empírica de una muestra se define como $\mathbb{P}(x) = \frac{n(x)}{n}$ con $n(x)$ el número de veces que x ocurre en la muestra $\{x_i\}$.

2. Sea S un conjunto finito. Definimos como medida de similitud entre dos subconjuntos A y B de S:

$$K(A, B) = \#(A \cap B) = |A \cap B|.$$

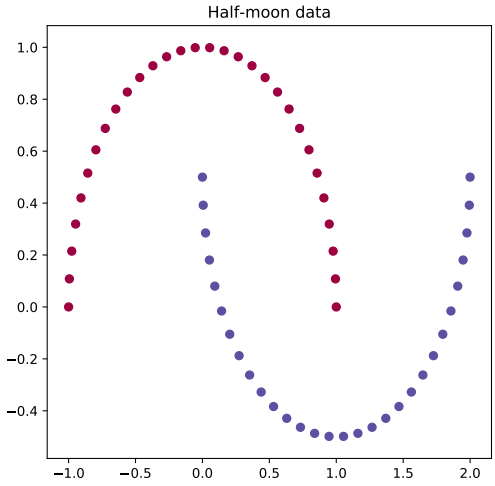
Buscar una función $\Phi(\cdot)$ tal que $K(A, B) = \langle \Phi(A), \Phi(B) \rangle$.

3. Escribir una función que implementa Kernel PCA con un kernel centrado de base radial con parámetro σ, y hacer $\mathbb{K}_c = \mathbb{J}\mathbb{K}\mathbb{J}$ con $K(i, j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/\sigma)$ y $\mathbb{J}$ la matriz para centrar.

Aplícalo para datos en 2D con la siguiente estructura de la Figura (toma en cada grupo 30 observaciones). Muestra cómo las proyecciones sobre los primeros dos componentes cambian en función de σ y cómo Kernel PCA se aproximan a PCA si $\sigma \to \infty$.

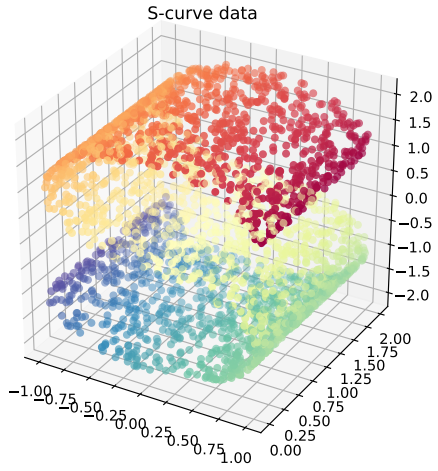
(Cuidado! Algunas funciones ya implementadas en Python o R, en lugar de usar el parámetro σ utilizan el parámetro $\gamma = \frac{1}{\sigma}$.)
La distribución de puntos de la Figura anterior en Python puede llamarse como:

```
from sklearn.datasets import make_moon
X, color = make_moon(n_samples=60)
```



4. Hacer una comparación de distintos métodos de proyección local o *manifold learning*: Linear local embedding LLE, t-SNE, Spectral Embedding, ISOMAP y Escalamiento multidimensional MDS, para comparar diferentes visualizaciones de un conjunto de datos.

   Aplícalo para datos en 3D con la siguiente estructura de la Figura (toma 3000 observaciones). Muestra las proyecciones sobre los primeros dos componentes para cada método. Ilustra todas las proyecciones en un mismo plot, usando subplot, para facilitar la comparación visual.



   La distribución de puntos de la Figura anterior en Python puede llamarse como:

```
from sklearn.datasets import make_s_curve
X, color = make_s_curve(n_samples=3000)
```

5. Usa el método t-SNE, para hacer una proyección del conjunto de datos *Digits*, el cual consiste de 1797 imágenes de dígitos de tamaño 8×8 (similar a MNIST, pero más simple).

   Compara las vistas de las proyecciones a 2D o 3D usando diferentes valores de perplejidad: $\lambda = 0.01, 0.1, 1., 10., 100.$ Agrega etiquetas en las gráficas para ver cómo se distribuyen las nubes de puntos en función de cada dígito.
   Explica cuál es tu intuición sobre qué está haciendo este parámetro de perplejidad.

(Sugerencia: Usar el mismo parámetro `random_state` para todos los experimentos, para que el resultado sólo dependa de λ).

El conjunto *Digits* puede llamarse en Python como:

```
from sklearn import datasets
digits = datasets.load_digits()
X = digits['data']
y = digits['target']
```

6. El índice de felicidad *Happy Planet Index* es un intento de medir el bienestar sostenible para todos, el cual puede encontrarse en el sitio http://happyplanetindex.org/.
   Puedes descargar los datos del índice de felicidad 2016 en http://happyplanetindex.org/resources o también están disponibles en el archivo `hpi-data-2016.xlsx`

   Usa el método SOM para encontrar visualizaciones útiles para el conjunto de datos del índice de felicidad y de sus variables. Complementa estas visualizaciones con otros métodos, y haz un análisis completo a partir de los datos.