Ciencia de Datos 2021

Lista 03

01.marzo.2021

1. En aula vimos el Teorema de Rao o de Eckart-Young: Si $\mathbb{X}$ es una matriz simétrica de rango d y con SVD

$$\mathbb{X} = USV = \sum_{i=1}^{d} \lambda_i \mathbf{u}_i \mathbf{v}_i^T,$$

la matriz $\mathbb{X}^*$ de rango $r < d$ que minimiza el error $||\mathbb{X} - \mathbb{X}^*||_F$ es

$$\mathbb{X}^* = \sum_{i=1}^{r} \lambda_i \mathbf{u}_i \mathbf{v}_i^T.$$

Muestra que para esta elección, el error de reconstrucción es

$$||\mathbb{X} - \mathbb{X}^*||_F^2 = \sum_{i=r+1}^{d} \lambda_i.$$

(*sugerencia*: usar las propiedades de $\mathbf{v}_i$ y recordar que $||A||_F^2 = tr(A^T A)$).

2. 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 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 $\{x_i\}$.

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

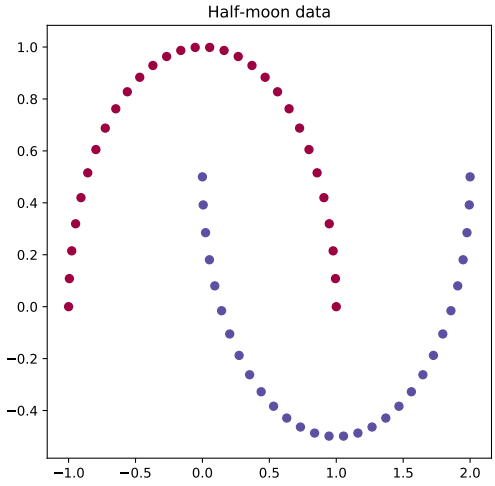
$$K(A, B) = \#(A \cap B) = |A \cap B|.$$

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

4. 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)
```

5. Aplicar los siguientes métodos PCA, ICA y NNMF a una parte del conjunto MNIST. Para ello, vamos a descargar el conjunto MNIST de la siguiente forma

```
from tensorflow.keras.datasets import mnist
(Xtrain, ytrain), (Xtest, ytest) = mnist.load_data()
```

   Vamos a trabajar sólo con la parte (Xtest, ytest), la cual consiste de 10,000 imágenes de tamaño 28×28. Vectorice las imágenes para obtener una matriz de datos $\mathbb{X}$ de tamaño 10000×784.

   Para ese conjunto de datos, hacer los siguiente: Aplicar los tres algoritmos anteriores, con $k = 25$ (proyectamos al espacio de las primeras 25 componentes).

   a) Mostrar en un subplot de 5×5 cómo se ven las primeras 25 componentes principales (un plot para PCA, otro ICA y otro para NNMF). Cada una de estas componentes es una imagen de 28×28.
   b) Tratar de dar una interpretación sobre qué están haciendo estas componentes, y discutir cuáles son las diferencias entre cada método. Discutir también la diferencia cuando en NNMF se aumenta o se reduce el valor de k.

   Opcional: Puede ser interesante comparar también con otros métodos como FactorAnalysis, KernelPCA y DictionaryLearning, también incluidos en el módulo `sklearn.decomposition`.