T5 Modelos basados en ejemplos

Problemas

1. 3-NN 2d L2
2. 5-NN 2d L1

Cuestiones

1. Paramétricos vs no paramétricos
2. DML
3. KDE
4. KDE h

3-NN 2d L2: Sea un problema de clasificación de datos 2d en tres clases, $c \in \{0, 1, 2\}$. Se tienen los siguientes datos de aprendizaje:

n	1	2	3	4	5	6
x_{n1}	2	3	4	5	6	7
x_{n2}	3	0	4	1	5	3
c_n	1	1	2	0	0	0

Clasifica la muestra $\boldsymbol{x} = (4, 3)^t$ por los 3-vecinos más próximos en distancia Euclídea.

3-NN 2d L2: Sea un problema de clasificación de datos 2d en tres clases, $c \in \{0, 1, 2\}$. Se tienen los siguientes datos de aprendizaje:

n	1	2	3	4	5	6
x_{n1}	2	3	4	5	6	7
x_{n2}	3	0	4	1	5	3
c_n	1	1	2	0	0	0

Clasifica la muestra $\boldsymbol{x} = (4, 3)^t$ por los 3-vecinos más próximos en distancia Euclídea.

Solución:

Hallamos las distancias (Euclídeas) entre $\boldsymbol{x}$ y cada dato:

n	1	2	3	4	5	6
$d(\boldsymbol{x}_n, \boldsymbol{x})$	2	$\sqrt{10}$	1	$\sqrt{5}$	$\sqrt{8}$	3

El conjunto de 3 vecinos más próximos de $\boldsymbol{x}$ es $N_3(\boldsymbol{x}) = \{3, 1, 4\}$. Cada vecino vota a una clase distinta de las tres que tenemos. Como hay empate a tres, decidimos por el vecino más cercano entre los tres, que es el 3, de la clase 2. Por tanto, 3-NN clasifica $\boldsymbol{x}$ en la clase 2.

3-NN 2d L2: Sea un problema de clasificación de datos 2d en tres clases, $c \in \{0, 1, 2\}$. Se tienen los siguientes datos de aprendizaje:

n	1	2	3	4	5	6
x_{n1}	2	3	4	5	6	7
x_{n2}	3	0	4	1	5	3
c_n	1	1	2	0	0	0

Clasifica la muestra $\boldsymbol{x} = (4, 3)^t$ por los 3-vecinos más próximos en distancia Euclídea.

Solución a máquina:

In [1]:
```
import numpy as np; from sklearn.neighbors import NearestNeighbors
Xy = np.array([ [2, 3, 1], [3, 0, 1], [4, 4, 2], [5, 1, 0], [6, 5, 0], [7, 3, 0]], dtype=float)
X = Xy[:, :2]; y = Xy[:, 2]; x = np.array([4, 3], dtype=float); K = 3
KNN = NearestNeighbors(n_neighbors=K).fit(X)
dist, ind = KNN.kneighbors([x]); print('n =', ind + 1, '  d =', dist, '  c =', y[ind])
classes, votes = np.unique(y[ind], return_counts=True); print('votes for', classes, ': ', votes)
```

```
n = [[3 1 4]]   d = [[1.         2.         2.23606798]]   c = [[2. 1. 0.]]
votes for [0. 1. 2.] :  [1 1 1]
```

El NN entre las clases empatadas al máximo de votos es de la clase 2.

5-NN 2d L1: Sea un problema de clasificación de datos 2d en tres clases, $c \in \{0, 1, 2\}$. Se tienen los siguientes datos de aprendizaje:

n	1	2	3	4	5	6	7	8
x_{n1}	5	6	7	2	2	2	2	4
x_{n2}	1	1	3	0	3	4	5	4
c_n	0	0	0	1	1	1	1	2

Clasifica la muestra $\boldsymbol{x} = (4, 3)^t$ por los 5-vecinos más próximos en distancia L1 (Manhattan).

5-NN 2d L1: Sea un problema de clasificación de datos 2d en tres clases, $c \in \{0, 1, 2\}$. Se tienen los siguientes datos de aprendizaje:

n	1	2	3	4	5	6	7	8
x_{n1}	5	6	7	2	2	2	2	4
x_{n2}	1	1	3	0	3	4	5	4
c_n	0	0	0	1	1	1	1	2

Clasifica la muestra $\boldsymbol{x} = (4, 3)^t$ por los 5-vecinos más próximos en distancia L1 (Manhattan).

Solución:

Hallamos las distancias L1 entre $\boldsymbol{x}$ y cada dato:

n	1	2	3	4	5	6	7	8
$d(\boldsymbol{x}_n, \boldsymbol{x})$	3	4	3	5	2	3	4	1

El conjunto de 5 vecinos más próximos de $\boldsymbol{x}$ es $N_5(\boldsymbol{x}) = \{8, 5, 1, 3, 6\}$. Los 5 vecinos votan: 1 a la clase 2, 2 a la 1, y 2 a la 0. Observamos que no existe una única clase más votada; tenemos dos clases empatadas a dos votos, la 0 y la 1, por lo que debemos desempatar con el NN entre las clases empatadas. Entre los vecinos de las clases 0 y 1, el más cercano es el 5, que pertenece a la clase 1. Por tanto, 5-NN clasifica $\boldsymbol{x}$ en la clase 1.

5-NN 2d L1: Sea un problema de clasificación de datos 2d en tres clases, $c \in \{0, 1, 2\}$. Se tienen los siguientes datos de aprendizaje:

n	1	2	3	4	5	6	7	8
x_{n1}	5	6	7	2	2	2	2	4
x_{n2}	1	1	3	0	3	4	5	4
c_n	0	0	0	1	1	1	1	2

Clasifica la muestra $\boldsymbol{x} = (4, 3)^t$ por los 5-vecinos más próximos en distancia L1 (Manhattan).

Solución a máquina:

In [4]:
```
import numpy as np; from sklearn.neighbors import NearestNeighbors
Xy = np.array([[5,1,0],[6,1,0],[7,3,0],[2,0,1],[2,3,1],[2,4,1],[2,5,1],[4,4,2]],dtype=float)
X = Xy[:, :2]; y = Xy[:, 2]; x = np.array([4, 3], dtype=float); K = 5
KNN = NearestNeighbors(n_neighbors=K, p=1).fit(X)
dist, ind = KNN.kneighbors([x]); print('n =', ind + 1, '  d =', dist, '  c =', y[ind])
classes, votes = np.unique(y[ind], return_counts=True); print('votes for', classes, ': ', votes)
```

```
n = [[8 5 3 6 1]]   d = [[1. 2. 3. 3. 3.]]   c = [[2. 1. 0. 1. 0.]]
votes for [0. 1. 2.] :  [2 2 1]
```

El NN entre las clases empatadas al máximo de votos es de la clase 1.

Paramétricos vs no paramétricos: Una dicotomía clásica entre modelos de aprendizaje automático distingue entre modelos paramétricos y no paramétricos. En relación con esta dicotomía, indica la respuesta incorrecta (o escoge la última opción si las tres primeras son correctas).

1. Los paramétricos estiman un vector de parámetros de dimensión fija a partir de datos (de aprendizaje) y luego, en inferencia, prescinden de los datos.
2. Los no parámetricos mantienen los datos (tras el aprendizaje, en inferencia), por lo que puede decirse que el número efectivo de parámetros crece con el número de datos.
3. El clasificador por los K vecinos más próximos es un ejemplo clásico de modelo paramétrico pues se define en términos de una medida de (di)similitud o función distancia cuyos parámetros debemos aprender.
4. Todas son correctas.

Paramétricos vs no paramétricos: Una dicotomía clásica entre modelos de aprendizaje automático distingue entre modelos paramétricos y no paramétricos. En relación con esta dicotomía, indica la respuesta incorrecta (o escoge la última opción si las tres primeras son correctas).

1. Los paramétricos estiman un vector de parámetros de dimensión fija a partir de datos (de aprendizaje) y luego, en inferencia, prescinden de los datos.
2. Los no parámetricos mantienen los datos (tras el aprendizaje, en inferencia), por lo que puede decirse que el número efectivo de parámetros crece con el número de datos.
3. El clasificador por los K vecinos más próximos es un ejemplo clásico de modelo paramétrico pues se define en términos de una medida de (di)similitud o función distancia cuyos parámetros debemos aprender.
4. Todas son correctas.

Solución:

La 3 es incorrecta. KNN es un ejemplo clásico de modelo no paramétrico.

DML: Sea $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) : i = 1 : N\}$ un conjunto de datos etiquetados y sea $\mathcal{S} = \{(i, j) : y_i = y_j\}$ un conjunto de pares similares derivado. De acuerdo con el propósito de **deep metric learning,** si $(i, j) \in \mathcal{S}$ pero $(i, k) \notin \mathcal{S}$, entonces $\boldsymbol{x}_i$ y $\boldsymbol{x}_j$ deben estar:

1. Lejos en el espacio de embedding, con independencia de la cercanía entre $\boldsymbol{x}_i$ y $\boldsymbol{x}_k$.
2. Lejos en el espacio de embedding, pero $\boldsymbol{x}_i$ y $\boldsymbol{x}_k$ deben estar cerca.
3. Cerca en el espacio de embedding, con independencia de la cercanía entre $\boldsymbol{x}_i$ y $\boldsymbol{x}_k$.
4. Cerca en el espacio de embedding, pero $\boldsymbol{x}_i$ y $\boldsymbol{x}_k$ deben estar lejos.

DML: Sea $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) : i = 1 : N\}$ un conjunto de datos etiquetados y sea $\mathcal{S} = \{(i, j) : y_i = y_j\}$ un conjunto de pares similares derivado. De acuerdo con el propósito de **deep metric learning,** si $(i, j) \in \mathcal{S}$ pero $(i, k) \notin \mathcal{S}$, entonces $\boldsymbol{x}_i$ y $\boldsymbol{x}_j$ deben estar:

1. Lejos en el espacio de embedding, con independencia de la cercanía entre $\boldsymbol{x}_i$ y $\boldsymbol{x}_k$.
2. Lejos en el espacio de embedding, pero $\boldsymbol{x}_i$ y $\boldsymbol{x}_k$ deben estar cerca.
3. Cerca en el espacio de embedding, con independencia de la cercanía entre $\boldsymbol{x}_i$ y $\boldsymbol{x}_k$.
4. Cerca en el espacio de embedding, pero $\boldsymbol{x}_i$ y $\boldsymbol{x}_k$ deben estar lejos.

Solución:

La 4.

KDE: La **Parzen window** o **kernel density estimator (KDE)** puede verse como una generalización de la mixtura de N Gaussianas "empírica" (con una Gaussiana hiperesférica centrada en cada dato) a N kernels de amplitud dada por un ancho de banda h. En relación con este estimador, indica la respuesta incorrecta (o escoge la última opción si las tres primeras son correctas).

1. No requiere ajuste, salvo la elección de h, y no es necesario escoger el número de centros de clúster.
2. Requiere mucha memoria y tiempo de evaluación.
3. Cuanto menor sea h, más suave será KDE.
4. Todas son correctas.

KDE: La **Parzen window** o **kernel density estimator (KDE)** puede verse como una generalización de la mixtura de N Gaussianas "empírica" (con una Gaussiana hiperesférica centrada en cada dato) a N kernels de amplitud dada por un ancho de banda h. En relación con este estimador, indica la respuesta incorrecta (o escoge la última opción si las tres primeras son correctas).

1. No requiere ajuste, salvo la elección de h, y no es necesario escoger el número de centros de clúster.
2. Requiere mucha memoria y tiempo de evaluación.
3. Cuanto menor sea h, más suave será KDE.
4. Todas son correctas.

Solución:

La 3 es incorrecta; cuanto mayor sea h, más suave será KDE.

KDE h: Un kernel densidad es una función $\mathcal{K} : \mathbb{R} \to \mathbb{R}^{\geq 0}$ que integra a uno y simétrica. Su amplitud puede controlarse mediante un parámetro ancho de banda (**bandwidth**), $h > 0$, como sigue:

1. $\mathcal{K}_h(x) = h\mathcal{K}(x)$
2. $\mathcal{K}_h(x) = h\mathcal{K}\left(\frac{x}{h}\right)$
3. $\mathcal{K}_h(x) = \frac{1}{h}\mathcal{K}(x)$
4. $\mathcal{K}_h(x) = \frac{1}{h}\mathcal{K}\left(\frac{x}{h}\right)$

KDE h: Un kernel densidad es una función $\mathcal{K} : \mathbb{R} \to \mathbb{R}^{\geq 0}$ que integra a uno y simétrica. Su amplitud puede controlarse mediante un parámetro ancho de banda (**bandwidth**), $h > 0$, como sigue:

1. $\mathcal{K}_h(x) = h\mathcal{K}(x)$
2. $\mathcal{K}_h(x) = h\mathcal{K}\left(\frac{x}{h}\right)$
3. $\mathcal{K}_h(x) = \frac{1}{h}\mathcal{K}(x)$
4. $\mathcal{K}_h(x) = \frac{1}{h}\mathcal{K}\left(\frac{x}{h}\right)$

Solución:

La 4.