

Modelos No Paramétricos

Aprenentatge Automàtic

APA/GEI/FIB/UPC - 2025/2026 1Q

 / Javier Béjar

Introducción

- ⊙ En los modelos que hemos visto, seleccionamos un espacio de hipótesis y ajustamos un conjunto fijo de parámetros con los datos de entrenamiento ($f(x, w)$)
- ⊙ Suponemos que los parámetros w resumen los datos de entrenamiento, y podemos olvidarnos de ellos después de ajustar el modelo
- ⊙ Estos métodos se denominan modelos **paramétricos**
- ⊙ Cuando tenemos una pequeña cantidad de datos, tiene sentido tener un pequeño conjunto de parámetros para restringir la complejidad del modelo (evitando el sobreajuste)
- ⊙ Cuando tenemos una gran cantidad de datos, el sobreajuste es un problema menor

- ⊙ Si los datos muestran que la hipótesis tiene que ser compleja, podemos intentar ajustarnos a esa complejidad
- ⊙ Un modelo **no paramétrico** es aquel que no se puede caracterizar por un conjunto fijo de parámetros
- ⊙ Una familia de modelos no paramétricos son los modelos de **Aprendizaje Basado en Instancias** (Instance Based Learning)
- ⊙ En estos modelos, los propios datos se utilizan como representación

- ⊙ El aprendizaje basado en instancias (IBL) se basa en la **memorización** del conjunto de datos
- ⊙ El número de parámetros es ilimitado y crece con el tamaño de los datos
- ⊙ No hay un modelo asociado a los conceptos aprendidos
- ⊙ La clasificación/regresión para nuevos ejemplos se obtiene buscando entre los ejemplos memorizados los más similares
- ⊙ El coste del proceso de aprendizaje es 0, todo el coste está en el cómputo de la predicción
- ⊙ Este tipo de aprendizaje también se conoce como **aprendizaje perezoso**

K-vecinos más cercanos



- ⊙ **K-vecinos más cercanos** usa la vecindad local de un ejemplo para calcular una predicción
- ⊙ Los K ejemplos memorizados más parecidos al que se está clasificando se recuperan de los datos de entrenamiento
- ⊙ Se necesita una función de distancia/similitud para comparar los ejemplos
- ⊙ Esto significa que si cambiamos la función de distancia, cambiamos la forma en la que se predicen los ejemplos.

⊙ Las propiedades de una función de similitud son:

1. $s(p, q) = 1 \iff p = q$

2. $\forall p, q \ s(p, q) = s(q, p)$

⊙ Las propiedades de una función de distancia son:

1. $\forall p, q \ d(p, q) \geq 0 \ y \ \forall p, q \ d(p, q) = 0 \iff p = q$

2. $\forall p, q \ d(p, q) = d(q, p)$

3. $\forall p, q, r \ d(p, r) \leq d(q, p) + d(p, r)$ (desigualdad triangular)

⊙ **Métricas de Minkowski** (Manhattan, euclidiana)

$$d(x_i, x_j) = \left(\sum_{d=1}^D |x_{id} - x_{jd}|^p \right)^{\frac{1}{p}}$$

⊙ **Distancia de Mahalanobis**

$$d(x_i, x_j) = (x_i - x_j)^\top \Sigma^{-1} (x_i - x_j)$$

donde Σ es la matriz de covarianza de los atributos

⊙ **Similaridad del coseno**

$$d(x_i, x_j) = \frac{x_i^\top x_j}{\|x_i\|_2 \|x_j\|_2}$$

Podemos calcular algunas matrices de distancias/similaridad, a partir de operaciones de álgebra lineal.

Siendo $X \in \mathbb{R}^{N \times D}$, podemos calcular:

- ⊙ La matriz de distancias euclidianas como:

$$\mathbf{1} \operatorname{diag}(XX^\top)^\top + \operatorname{diag}(XX^\top) \mathbf{1}^\top - 2XX^\top$$

- ⊙ La matriz de similaridades del coseno, si los ejemplos están divididos por su norma como:

$$XX^\top$$

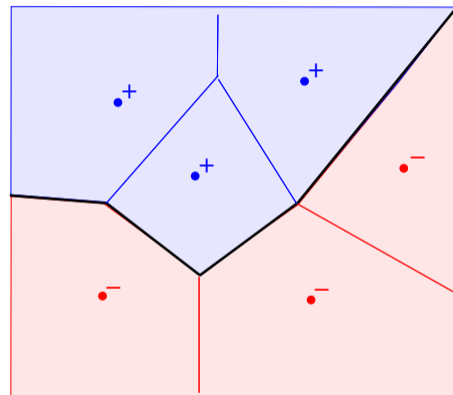
⊙ **Coeficiente de coincidencia**

$$s(x_i, x_k) = \frac{\sum \mathbb{1}[x_{id} = x_{jd} = 0] + \sum \mathbb{1}[x_{id} = x_{jd} = 1]}{D}$$

⊙ **Coeficiente de Jaccard**

$$s(x_i, x_k) = \frac{\sum \mathbb{1}[x_{id} = x_{jd} = 1]}{\sum \mathbb{1}[x_{id} = x_{jd} = 0] + \sum \mathbb{1}[x_{id} \neq x_{jd}]}$$

- ⊙ El espacio de hipótesis de K-NN corresponde a aproximaciones locales de las fronteras de decisión
- ⊙ La clasificación se obtiene mediante una combinación de los ejemplos más cercanos al punto de predicción
- ⊙ Las fronteras de decisión no son lineales
- ⊙ **Ejemplo:** Frontera de decisión para 1 vecino más cercano



Entrenamiento

Almacenar todos los ejemplos

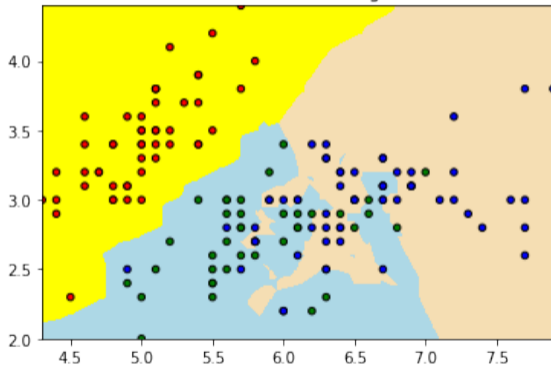
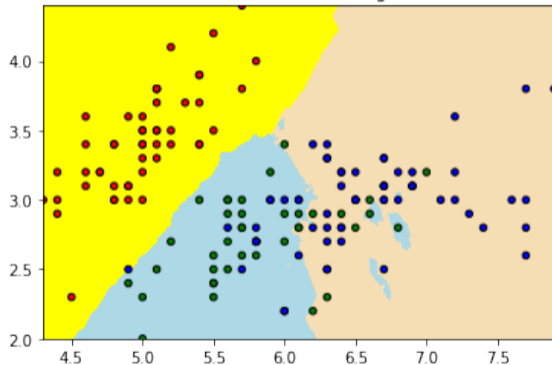
Predicción

- ⊙ Dado x_n
- ⊙ Calcular la similitud/distancia de x_n con los datos almacenados
- ⊙ Sean x_1, \dots, x_k los k ejemplos más parecidos a x_n
- ⊙ $f(x_n) = \text{combinar_predicciones}(x_1, \dots, x_k)$

- ⊙ K-NN tiene **tres hiperparámetros**
 - La función de distancia/similitud
 - El número k de vecinos para hacer la predicción
 - El procedimiento para combinar las predicciones de los k ejemplos
- ⊙ Estos parámetros deben ajustarse mediante un procedimiento de selección de modelos
 - Podemos usar la validación cruzada k-Fold
 - La función de error para clasificación es la pérdida 0-1, para regresión podemos usar error cuadrático medio
- ⊙ El valor de k suele ser el más crítico
 - Podemos sobreajustar si k es demasiado bajo (predicciones demasiado locales)
 - Podemos subajustar si k es demasiado alto (la función es demasiado suave)

- ⊙ Hay diferentes posibilidades para calcular la clase a partir de los k vecinos más cercanos
 - Voto mayoritario
 - Voto ponderado por distancia
 - Inverso de la distancia
 - Inverso del cuadrado de la distancia
 - Funciones del kernel (kernel gaussiano, kernel tricube...)
- ⊙ Una vez que usamos pesos para la predicción, podemos relajar la restricción de usar solo k vecinos
 1. Podemos usar k ejemplos (modelo local)
 2. Podemos usar todos los ejemplos (modelo global)

- ⊙ Buscar los ejemplos K más cercanos puede ser costoso
- ⊙ El algoritmo directo tiene un coste $O(n \log(k))$, no es bueno si el conjunto de datos es grande
- ⊙ Podemos usar estructuras de datos de indexación como *kd trees* (árboles multidimensionales similares a los árboles de búsqueda binaria)
 - Son buenos solo si tenemos alrededor de 2^D ejemplos, por lo que funcionan peor con alta dimensionalidad
- ⊙ Podemos usar algoritmos de vecinos más cercanos aproximados para reducir el costo computacional (aumentando el error)

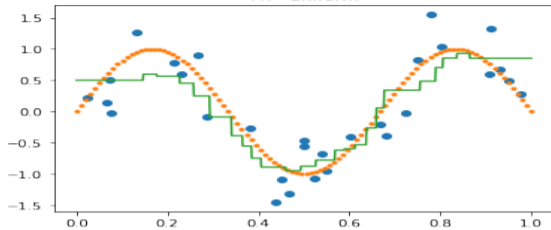
3-Class classification ($k = 3$, weights = 'uniform')3-Class classification ($k = 15$, weights = 'uniform')

Regresión K-nn

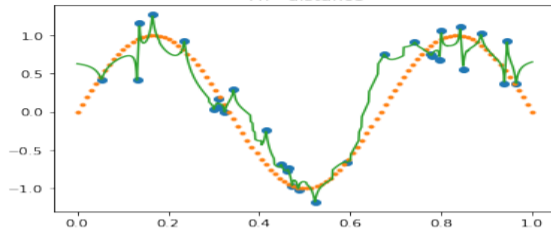
- ⊙ Podemos extender K-vecinos más cercanos a tareas de regresión
- ⊙ En lugar de combinar las predicciones discretas de k-vecinos, tenemos que combinar predicciones continuas
- ⊙ Estas predicciones se pueden obtener de diferentes maneras:
 - Promedio
 - Promedio ponderado (por distancia)
 - Kernel smoothing (función sobre la distancia)
 - Ajustar un modelo, por ejemplo una regresión lineal (pero es más caro)

K-vecinos más cercanos - Ejemplo de regresión

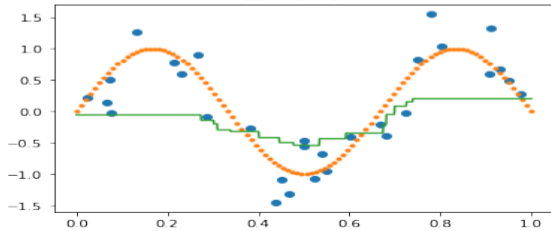
7n - uniform



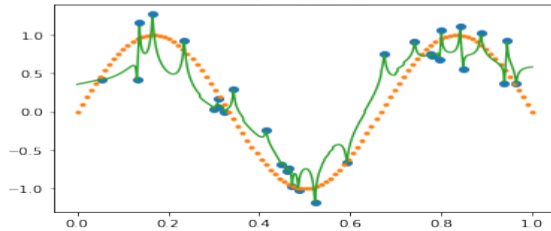
7n - distance



15n - uniform



15n - distance



Ventajas

- ⊙ El coste del proceso de aprendizaje es cero
- ⊙ No hace suposiciones sobre la distribución de los datos
- ⊙ Los conceptos complejos se pueden aprender por aproximación local usando procedimientos simples dados suficientes datos

Inconvenientes

- ⊙ Calcular los k vecinos cercanos es costoso cuando el conjunto de datos es muy grande
- ⊙ El rendimiento depende del número de dimensiones (<20) (*maldición de dimensionalidad*) \implies Selección de atributos/Reducción de dimensionalidad



Este **notebook** muestra el efecto de la dimensionalidad de los datos en las distancias y el efecto de diferentes estructuras de indexación en el rendimiento de KNN

También podéis ver un **video** explicando el contenido del notebook

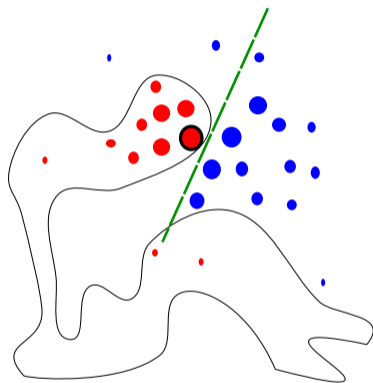
Interpretabilidad/Explicabilidad



- ⊙ No es posible una interpretación global del modelo ya que no hay parámetros relacionados con las características
- ⊙ Se puede obtener una explicación de los ejemplos recuperados
 - Se puede usar una explicación basada en ejemplos (contrafactuales, ejemplos contradictorios)
 - La distancia/similitud se puede usar para obtener el efecto de las características en la predicción (qué atributos son más similares)
- ⊙ Se pueden usar métodos de interpretación independientes del modelo para obtener más información

- ⊙ La **Importancia de permutación** (*Permutation importance*) obtiene la relevancia global de los atributos de un modelo
- ⊙ Usa solo predicciones, por lo que es independiente de las características del modelo
- ⊙ Para cada atributo mide cómo cambia el rendimiento si sus valores se permutan aleatoriamente
- ⊙ Rompemos la asociación de los atributos con el resto para verificar cuanto afecta esto a las predicciones
- ⊙ Debemos permutar los atributos múltiples veces y promediar los resultados para tener una buena estimación
- ⊙ **Problema:** Si los atributos están correlacionados, la permutación puede crear asociaciones poco realistas que pueden sesgar los resultados

- Se puede usar un modelo sustituto local para explicar predicciones individuales localmente
- La idea es ajustar localmente al ejemplo a explicar un modelo simple (por ejemplo, regresión/clasificación lineal) que se pueda interpretar
- LIME usa perturbaciones locales para obtener una muestra alrededor del ejemplo a explicar



1. Generar un conjunto de datos usando perturbaciones locales alrededor del ejemplo y obtener las predicciones del modelo
2. Usar como pesos la distancia desde los ejemplos generados hasta el ejemplo a explicar
3. Calcular un modelo interpretable (regresión/clasificación) usando el conjunto de datos ponderado (limitando opcionalmente los atributos usados)

- ⊙ La longitud de la explicación se puede restringir usando un modelo disperso (por ejemplo, LASSO) o usando las características más importantes del modelo sustituto



Este **notebook** muestra un ejemplo de K-nn con una explicación de los modelos