Introducción a la Ciencia de Datos

Maestría en Ciencias de la Computación

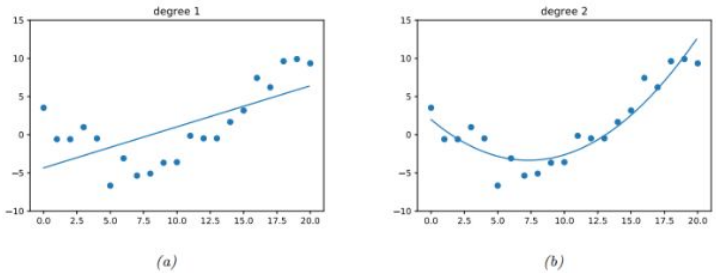
Dr. Irvin Hussein López Nava

04

Clasificación

Métodos paramétricos

- Construyen una descripción general explícita de la función objetivo f.
  - El modelo resume los datos a través de una colección de parámetros.
  - Número fijo de parámetros.
  - Independiente del número de instancias en los datos de entrenamiento.



Métodos no-paramétricos

- No hace suposiciones sobre la función objetivo.
  - Libertad para elegir cualquier función a partir de los datos de entrenamiento.
  - Requieren muchos más datos para estimar la función objetivo.
- Entradas similares tienen salidas similares.
  - Es una suposición razonable: El mundo es suave y las funciones, ya sean densidades, discriminantes o funciones de regresión, cambian lentamente.

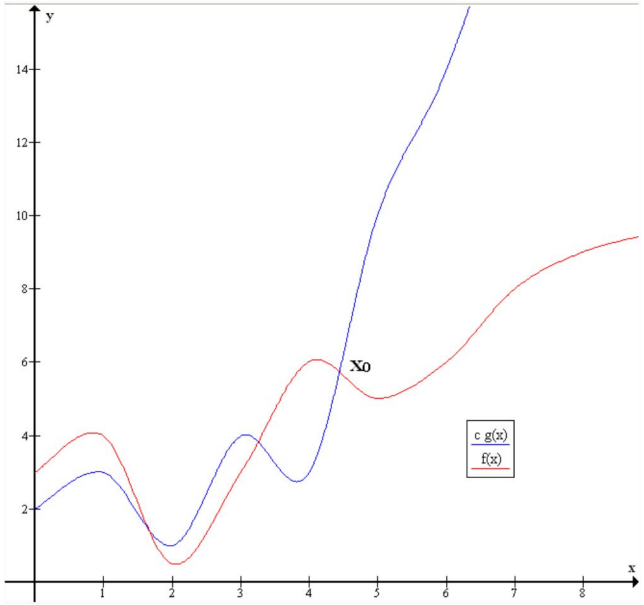
Métodos paramétricos vs no-paramétricos

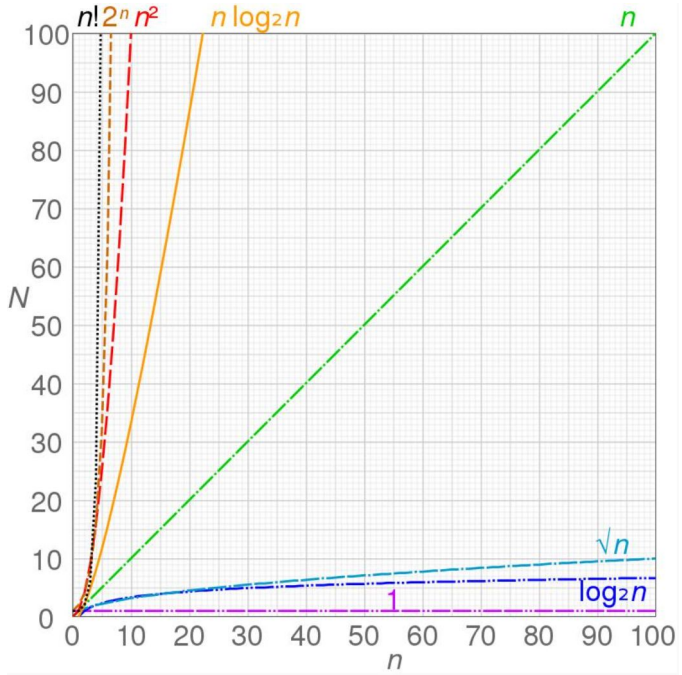
- En un modelo paramétrico, **todas** las instancias de entrenamiento afectan a la estimación global final.
- En un caso no paramétrico, no existe un modelo global único;
  - los modelos locales se estiman a medida que se necesitan, y sólo se ven afectados por las instancias de entrenamiento cercanas.

Métodos paramétricos vs no-paramétricos

- En ML, los métodos **no paramétricos** también se denominan algoritmos de aprendizaje basados en instancias o memoria, ya que almacenan las instancias de entrenamiento en una tabla de consulta e “interpolan” a partir de ella.
  - Dado que todas las instancias de entrenamiento deben ser almacenadas, se requieren memoria $O(n)$.
  - Dada una entrada, encontrar otras similares requiere el cálculo de $O(n)$.

Métodos paramétricos vs no-paramétricos





Métodos paramétricos vs no-paramétricos

- En el caso de un enfoque **paramétrico**, el modelo puede ser bastante simple y puede tener un pequeño número de parámetros, de orden $O(d)$ o $O(d^2)$.
- En general, n es mucho mayor que d o d^2.

Métodos paramétricos vs no-paramétricos

- Los métodos **no paramétricos** también se denominan algoritmos de aprendizaje perezoso (*lazy*), porque a diferencia de los modelos paramétricos ansiosos (*eager*), no calculan un modelo cuando se les da el conjunto de entrenamiento, sino que posponen el cálculo hasta que se les da una instancia de consulta.

Tell me about your friends, and ...

... I will tell you who you are.

4.4 Instance-based learning


¿Qué es el
Aprendizaje basado
en Instancias?

Definiciones

ChatGPT

Es un enfoque en aprendizaje automático que se centra en la utilización de ejemplos específicos o "instancias" para realizar tareas de clasificación. Se basa en la retención y recuperación de casos o ejemplos individuales para resolver problemas.

https://chat.openai.com/

Wikipedia

Es una familia de algoritmos de aprendizaje que, en lugar de realizar una generalización explícita, compara nuevas instancias de problemas con instancias vistas en el entrenamiento, que han sido almacenadas en la memoria.

https://en.wikipedia.org/wiki/Instance-based_learning

Gemini

Es un enfoque de aprendizaje automático que almacena ejemplos de datos de entrenamiento y los utiliza para realizar predicciones.
Es diferente del aprendizaje basado en modelos, que construye un modelo matemático de los datos de entrenamiento.

https://gemini.google.com/

¿Cuándo optar por este enfoque?

- Menos de 20 atributos por instancia.
- Muchos datos de entrenamiento.

Ventajas:

- El entrenamiento es muy rápido.
- Aprende funciones objetivo complejas.
- No se pierde información.

Desventajas:

- Lento en el momento de la consulta.
- Fácilmente engañado por atributos irrelevantes.

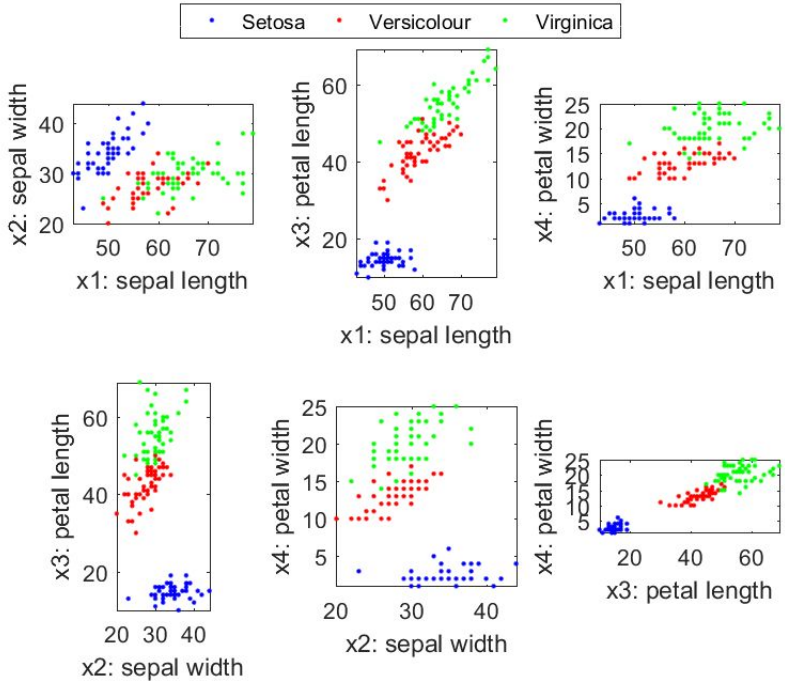
Iris dataset

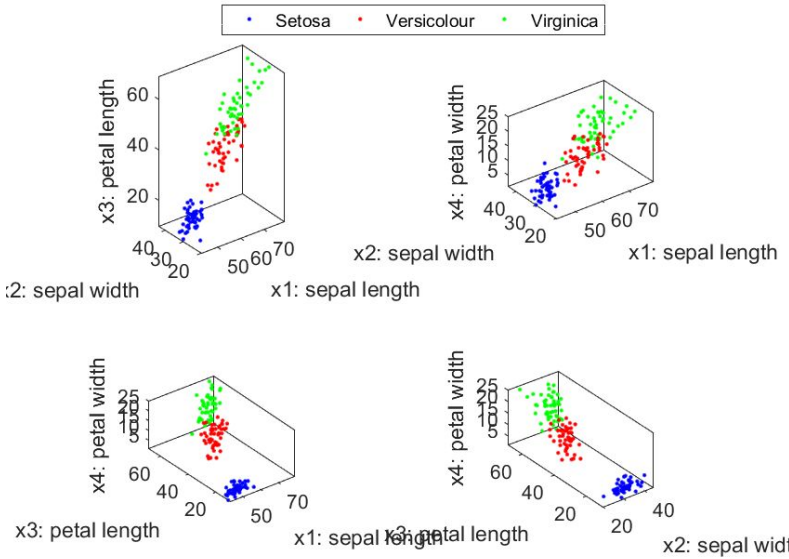
Setosa

Versicolor

Virginica

Iris dataset





Idea principal

Vecino más próximo (*nearest neighbor*):

- Dada la instancia de consulta x_q, primero se localiza el ejemplo de entrenamiento más cercano x_n, después se estima $\hat{f}(x_q) \leftarrow f(x_n)$

k-vecinos más próximos (*k-nearest neighbor*):

- Dado x_q, se vota entre sus *k* vecinos más cercanos (si la función objetivo es de valor discreto), o
- Se toma la media de los valores *f* de los *k* vecinos más cercanos (si son de valor real).

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

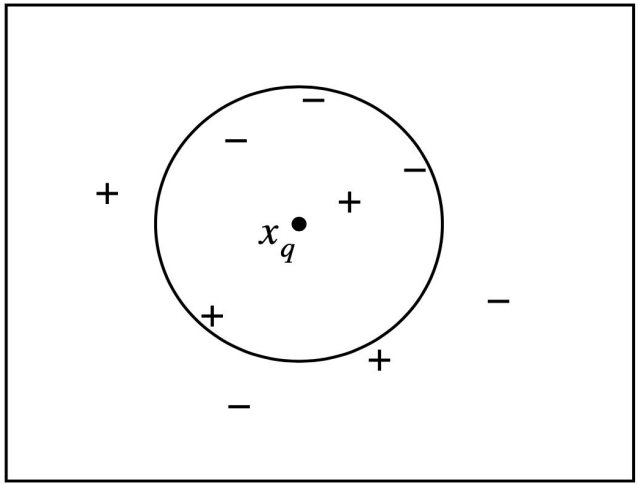
Suposiciones iniciales

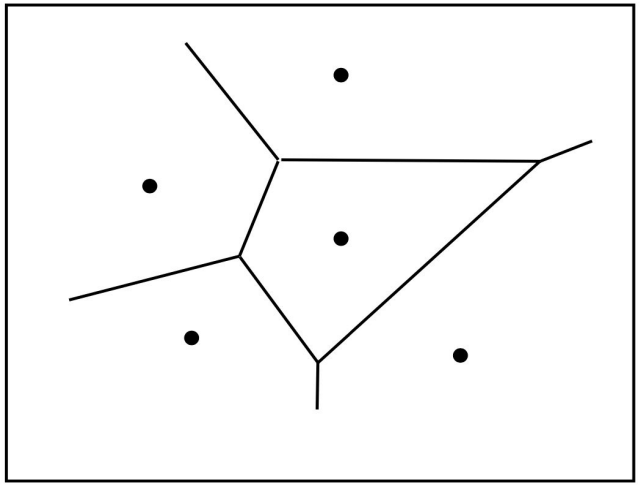
Todas las instancias corresponden a puntos en un espacio n-dimensional R^n.

- Los vecinos más cercanos de una instancia se definen en términos de la distancia euclidiana estándar.
- Sea una instancia arbitraria x descrita por el vector de características $(a_1(x), a_2(x), \ldots, a_n(x))$ donde $a_r(x)$ denota el valor del r-ésimo atributo de la instancia x.
- La distancia entre dos instancias x_i y x_j se define como

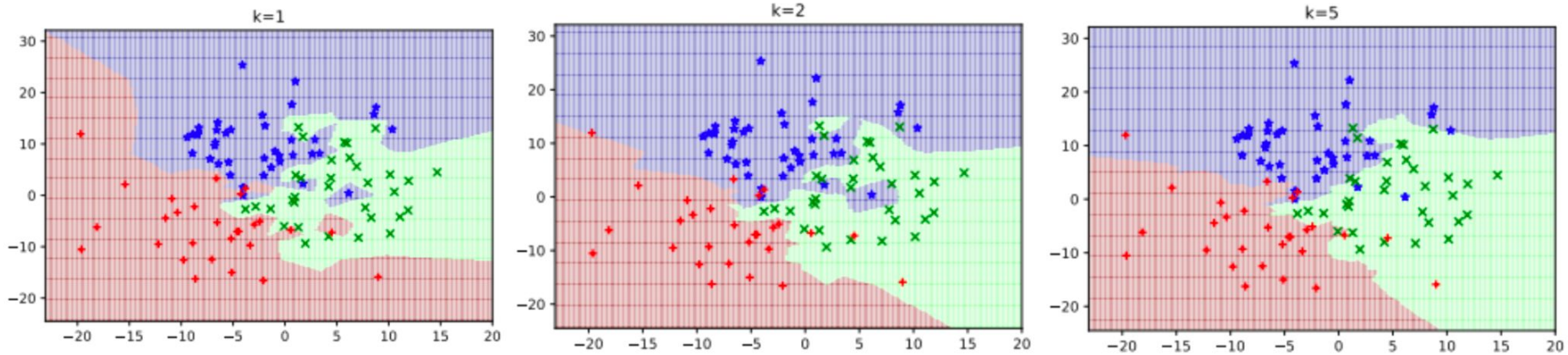
$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} \left(a_r(x_i) - a_{r(x_j)}\right)^2}$$

Diagrama de Voronoi

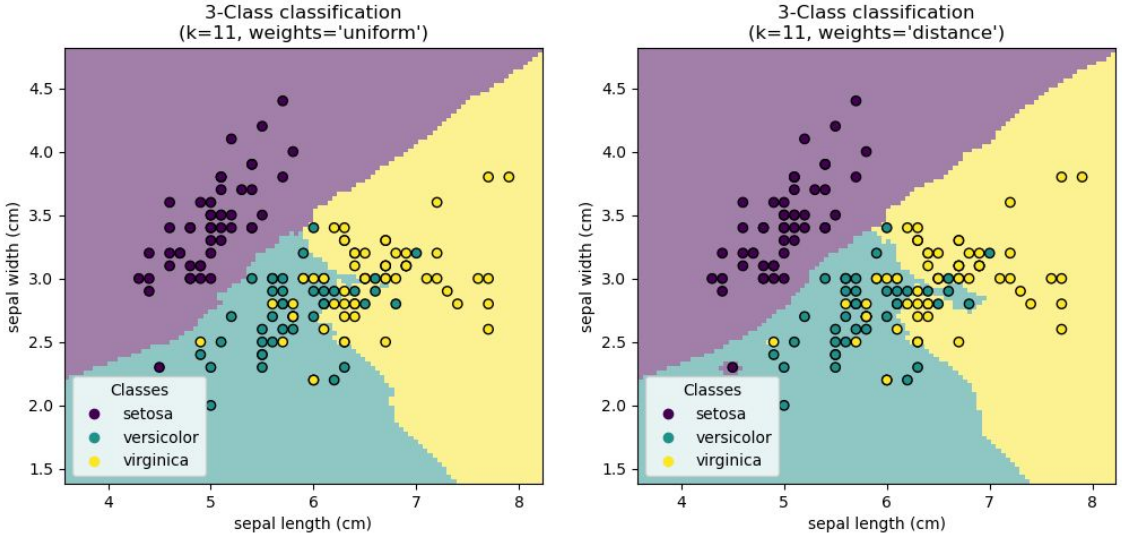




Límites de decisión inducidos por KNN



Límites de decisión inducidos por KNN

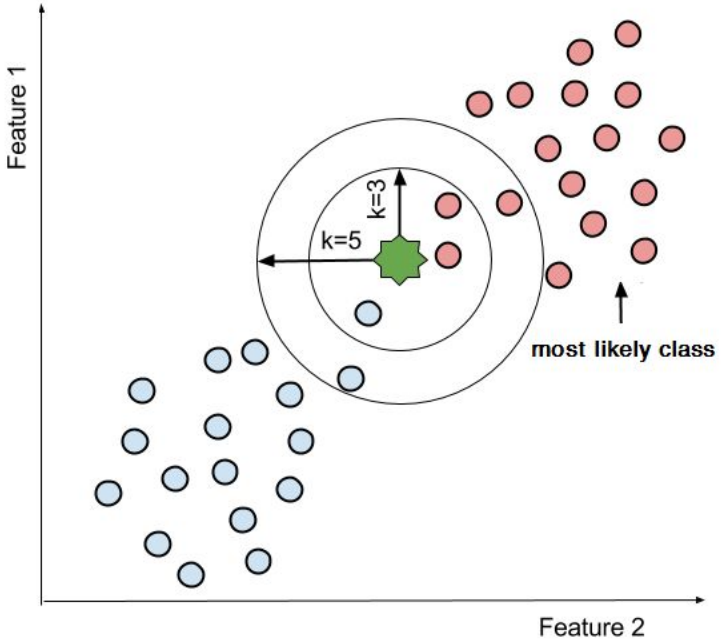


Comportamiento en el límite

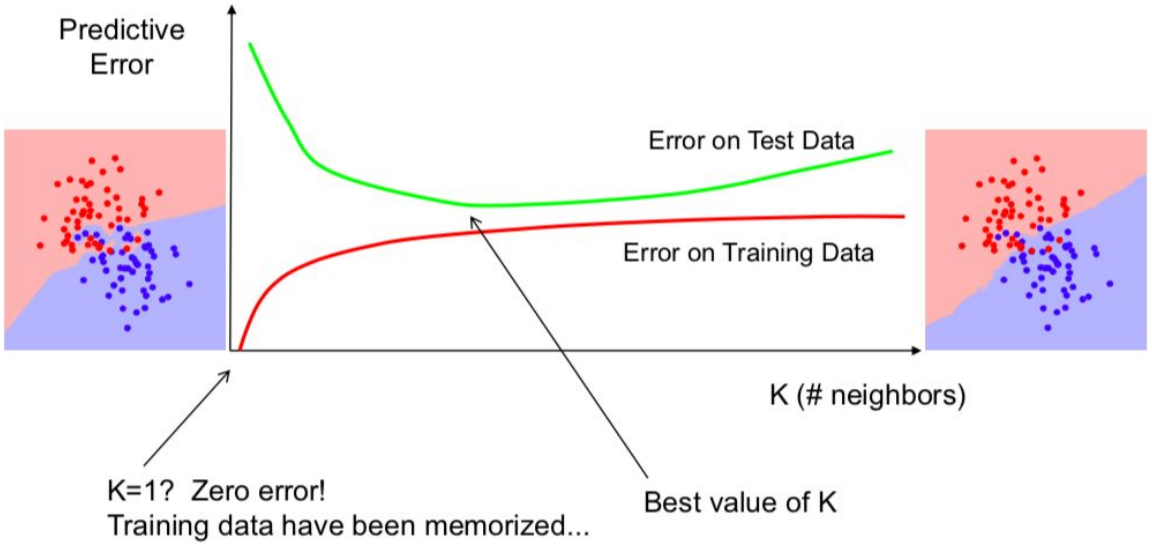
Considere que $p(x)$ define la probabilidad de que la instancia x sea etiquetada como 1 (positiva) frente a 0 (negativa).

- Vecino más cercano:
  - Como el número de ejemplos de entrenamiento $\rightarrow \infty$, se aproxima al algoritmo de Gibbs: con probabilidad $p(x)$ predice 1, si no 0.
- k-vecino más cercano:
  - A medida que el número de ejemplos de entrenamiento $\rightarrow \infty$ y k se hace grande, se aproxima al óptimo de Bayes: si $p(x) > 0.5$ predice 1, si no 0.

Elegir el valor de *k* (límite)



Elegir el valor de k (límite)

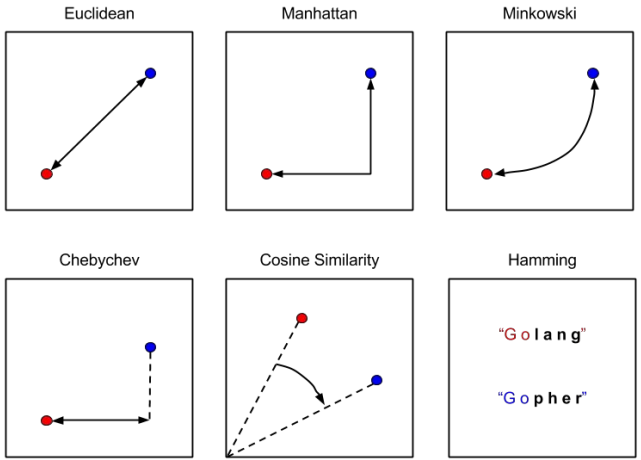


Algoritmo base KNN

Algorithm K-Nearest Neighbors (KNN)

Input: Training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, Test instance x_{test}, Number of neighbors k
Output: Predicted class for x_{test}
for each instance x_i in D **do**
    Compute the distance $d(x_i, x_{\text{test}}) = \text{distance\_function}(x_i, x_{\text{test}})$
    Store the distance and the corresponding class label y_i
end for
Sort the distances in ascending order
Select the top k instances with the smallest distances
Determine the majority class among the k-nearest neighbors:
    Count the occurrences of each class label among the k neighbors
    Assign the class with the highest count as the predicted class
return Predicted class for x_{test}

¡La elección de la medida de distancia afecta!



Distance-Weighted kNN

- Podría quererse ponderar más a los vecinos cercanos...

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

donde

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

y $d(x_q, x_i)$ es la distancia entre w_q y x_i.

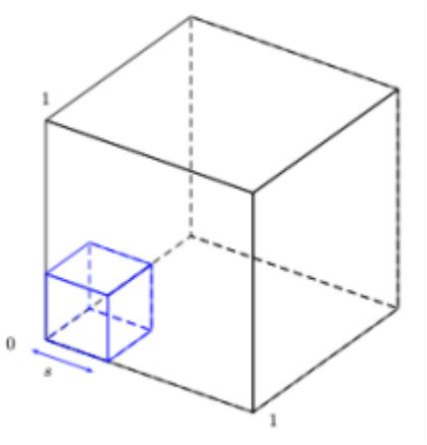
- Tener en cuenta que ahora tiene sentido utilizar todos los ejemplos de entrenamiento en lugar de sólo *k* (método de Shepard).

Maldición de la dimensionalidad

- El principal problema estadístico de los clasificadores *k*NN es que no funcionan bien con entradas de alta dimensionalidad, debido a la maldición de la dimensionalidad.
- Imagine instancias descritas por 20 atributos, pero sólo 2 son relevantes para la función objetivo.
- *Curse of dimensionality:* el vecino más cercano se equivoca fácilmente cuando *X* es de alta dimensionalidad.

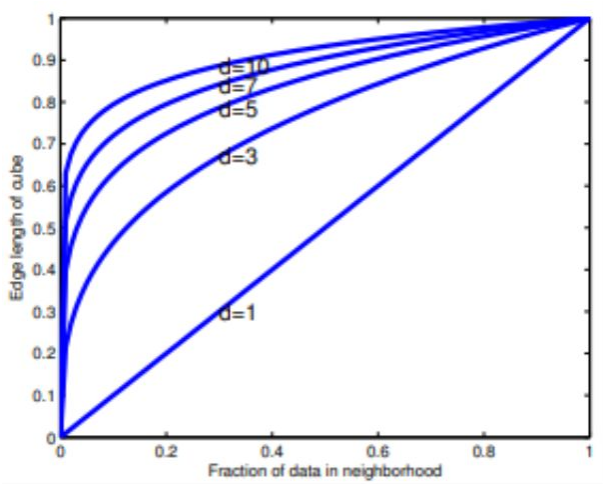
Longitud de borde esperada

- Suponga que se aplica un clasificador *k*NN a datos en los que las entradas se distribuyen uniformemente en un cubo unitario D–dimensional.
- Ahora, suponga que se estima la densidad de etiquetas de clase alrededor de un punto de prueba $\boldsymbol{x}$ haciendo crecer un hipercubo alrededor de $\boldsymbol{x}$ hasta que contenga una fracción deseada p de los puntos de datos.



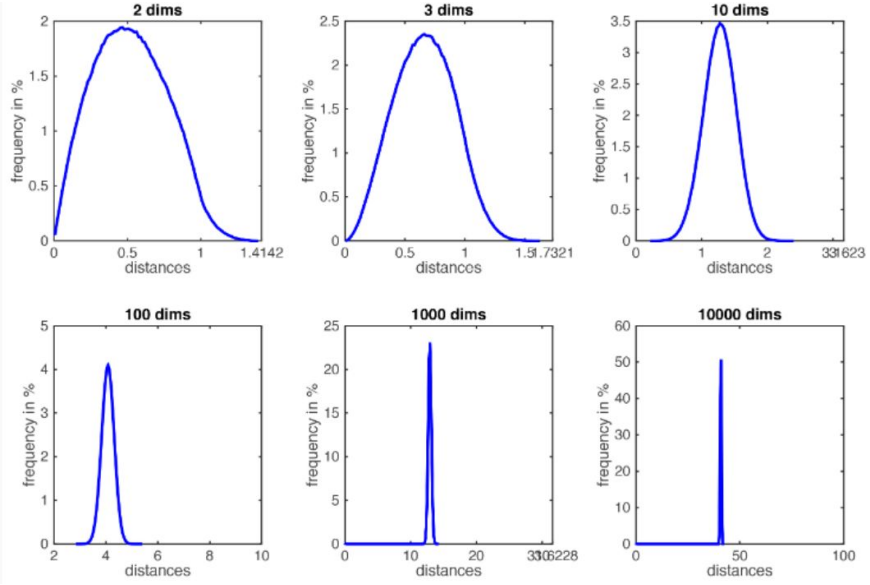
Longitud de borde esperada

- La longitud de borde esperada de este cubo será $e_D(s) = p^{1/D}$.
- Si D = 10, y se quiere basar la estimación en el 10% de los datos, se tiene $e_{10}(0.1) = 0.8$, por lo que se necesita extender el cubo un 80% a lo largo de cada dimensión alrededor de $\boldsymbol{x}$.



Longitud de borde esperada

- Otra forma de verlo es generando aleatoriamente puntos en el cubo unitario D–dimensional en varias dimensiones y calculando las distancias.



Reducción de memoria y tiempo

Se han propuesto varias técnicas heurísticas de poda para eliminar puntos o coordenadas que no afectan a los límites de decisión.

Un primer enfoque:

- Estirar el eje *j* mediante el peso z_j, donde $z_1, \ldots, z_n$ se elige para minimizar el error de predicción.
- Utilizar validación cruzada para elegir automáticamente los pesos $z_1, \ldots, z_n$.
- Tener en cuenta que establecer z_j a cero elimina esta dimensión por completo.

Otros enfoques: reducción de dimensionalidad (e.g. PCA).

Reducción de memoria y tiempo

- En términos de tiempo de ejecución, el reto es encontrar los K vecinos más cercanos en menos de $O(n)$ tiempo, donde N es el tamaño del conjunto de entrenamiento.
- Encontrar los vecinos más cercanos exactos es computacionalmente intratable cuando la dimensionalidad del espacio supera las 10 dimensiones, por lo que la mayoría de los métodos se centran en encontrar los vecinos más cercanos aproximados.
- Hay dos clases principales de técnicas, basadas en la partición del espacio en regiones (e.g. regresión ponderada local) o en el uso de hashing.

Regresión ponderada local

- Tener en cuenta que kNN forma una aproximación local a *f* para cada punto de consulta x_q.
- Por qué no formar una aproximación explícita *^f(x)* para la región que rodea x_q.
  - Ajustar una función lineal a los *k* vecinos más cercanos.
  - Ajuste cuadrático, ...
  - Produce una aproximación “por partes" de *f*.

Regresión ponderada local

- Varias opciones para minimizar el error:
  - Error cuadrático sobre *k* vecinos más cercanos.

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k \ \ nearest \ \ nbrs \ \ of \ \ x_q} (f(x) - \hat{f}(x))^2$$

  - Error cuadrático ponderado por distancia sobre todos los vecinos.

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 \ K(d(x_q, x))$$

Redes de funciones de base radial

- Aproximación global a la función objetivo, en términos de combinación lineal de aproximaciones locales.
- Se utiliza, e.g., para la clasificación de imágenes.
- Otro tipo de red neuronal.
- Estrechamente relacionada con la regresión ponderada por distancia, pero ansiosa (*eager)* en lugar de perezosa (*lazy)*.

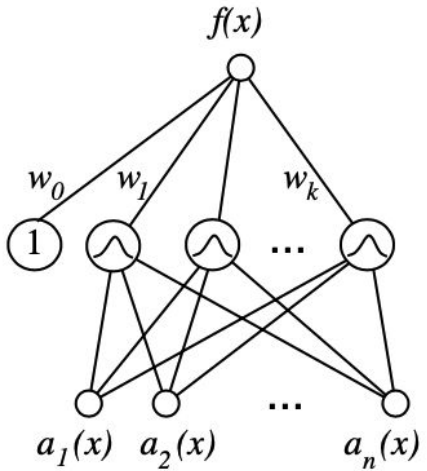
Redes de funciones de base radial

donde $a_i(x)$ son los atributos que describen la instancia x, y

$$f(x) = w_0 + \sum_{u=1}^{k} w_u \; K_u(d(x_u, x))$$

una opción común para $K_u(d(x_u, x))$ es

$$K_u(d(x_u, x)) = e^{-\frac{1}{2\sigma_u^2} d^2(x_u, x)}$$



Entrenamiento de redes de base radial

Q1: Qué x_u usar para cada función kernel $K_u(d(x_u, x))$.

- Dispersión uniforme en todo el espacio de instancias, o
- Utilizar instancias de entrenamiento (refleja la distribución de instancias).

Q2: Cómo entrenar los pesos (supongamos que K_u es gaussiano).

- Primero elegir la varianza (y quizás la media) para cada K_u.
  - e.g., usar EM.
- A continuación, mantener K_u fijo y entrenar la capa de salida lineal.
  - Métodos eficientes para ajustar la función lineal.

Razonamiento basado en casos

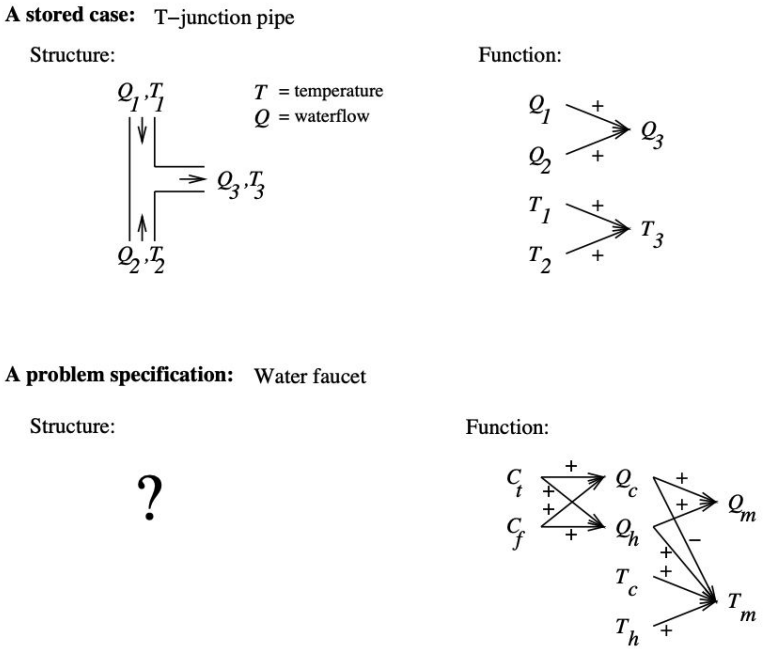
- Se puede aplicar el aprendizaje basado en instancias incluso cuando $X \neq \Re^n$.
- Se requiere una métrica de "distancia" diferente.
- El razonamiento basado en casos es el aprendizaje basado en instancias aplicado a instancias con descripciones lógicas simbólicas.

Razonamiento basado en casos en CADET

CADET: 75 ejemplos almacenados de dispositivos mecánicos:

- cada ejemplo de entrenamiento: {función cualitativa, estructura mecánica},
- nueva consulta: función deseada,
- valor objetivo: estructura mecánica para esta función.

Métrica de distancia: coincidencia de descripciones de funciones cualitativas.



Lazy and eager learning

Lazy: espera a la consulta antes de generalizar.

- Vecino más cercano, razonamiento basado en casos.

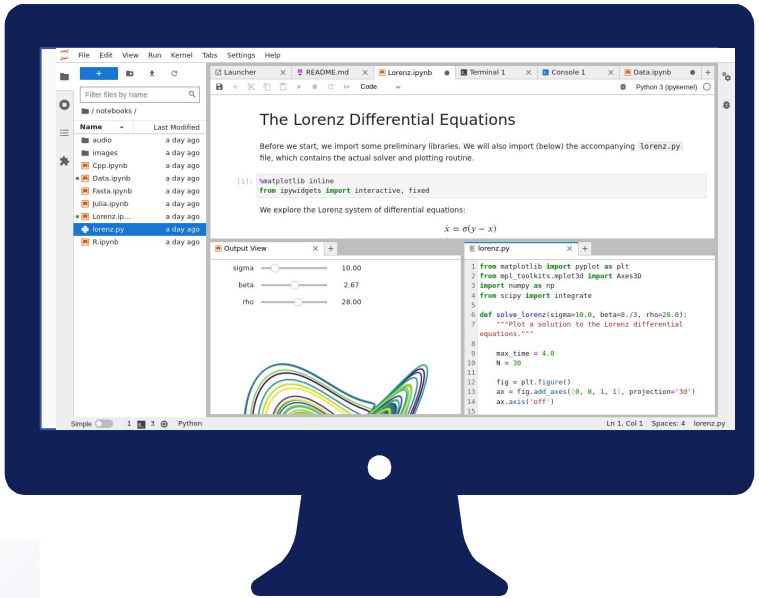
Eager: generaliza antes de ver la consulta.

- Redes de funciones de base radial, ID3, retropropagación, Naive Bayes, ...

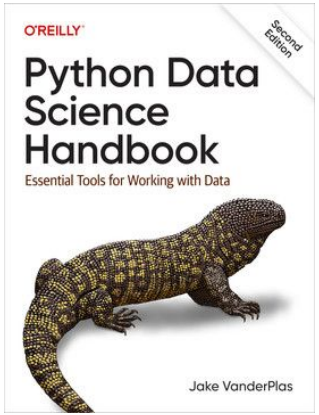
¿Importa?

- El aprendiz ansioso debe crear una aproximación global.
- El aprendiz perezoso puede crear muchas aproximaciones locales.
- Si usan la misma H, los perezosos pueden representar funciones más complejas.

(Go to live notebook)



Extra Libro



GitHub

?

Gracias!

¿Alguna pregunta?

hussein@cicese.mx

https://sites.google.com/view/husseinlopeznava

SCAN ME