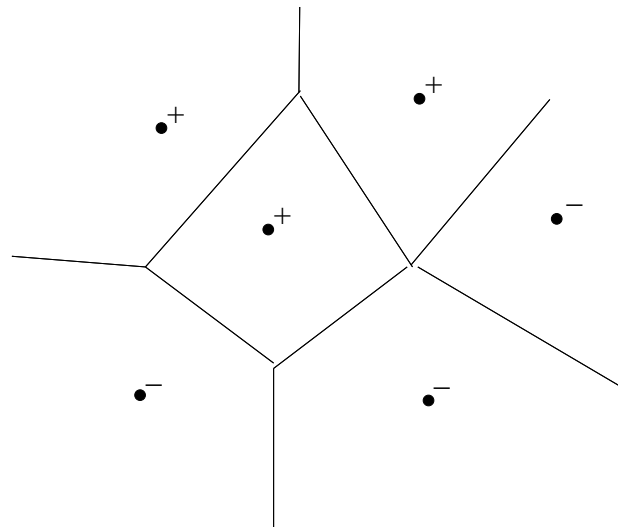


K-nearest neighbours

- Mecanismo de aprendizaje perezoso (*lazy learning*)
- No hay un modelo global asociado a los conceptos a aprender
- Las predicciones se realizan basandose en los ejemplos mas parecidos al que hay que predecir
- El coste del aprendizaje es 0, todo el coste pasa al cálculo de la predicción
- Debemos definir una función de distancia que mida la similaridad entre ejemplos (p. ej: euclídea)

K-nearest neighbours - Espacio de hipótesis (1 vecino)



K-nearest neighbours - Algoritmo

- Entrenamiento: Guardar todos los ejemplos
- Predicción:
 - Sean x_1, \dots, x_k los k ejemplos mas parecidos al que hay que predecir x_{nueva}
 - $clase(x_{nueva}) = combinación_clases(x_1, \dots, x_k)$
- Los parámetros del algoritmo son el número k de ejemplos a usar y la forma de combinar las decisiones de los k ejemplos

K-nearest neighbours - Variantes

- Existen diferentes posibilidades para calcular la clase a partir de los k ejemplos vecinos
 - Voto por mayoría
 - Voto ponderado por distancia
 - * Inversa de la distancia
 - * Inversa del cuadrado de la distancia
 - * Funciones de kernel (función gaussiana, tricubo, ...)
- Ponderación de los ejemplos en función de los aciertos/fallos previos
- Ponderación de los atributos

K-nearest neighbours - Ventajas

- El coste del aprendizaje es nulo
- No necesitamos hacer ninguna suposición sobre los conceptos a aprender
- Podemos aprender conceptos complejos usando funciones sencillas como aproximaciones locales
- Podemos extender el mecanismo para predecir un valor continuo (regresión)
- Es muy tolerante al ruido en los ejemplos

K-nearest neighbours - Inconvenientes

- El coste de encontrar los k mejores vecinos es grande (estructuras especializadas kd-trees)
- No hay un mecanismo para decidir el valor óptimo para k (depende de cada conjunto de datos)
- Su rendimiento baja si el número de descriptores crece
- Su interpretabilidad es nula (no hay una descripción de los conceptos aprendidos)