



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica



Tema 4. Clasificación basada en distancias

Percepción (PER)

Curso 2021/2022

Departamento de Sistemas Informáticos y Computación

Índice

- 1 Introducción: espacio métrico y distancias ▷ 3
- 2 Vecino más cercano ▷ 7
- 3 k -vecinos más cercanos ▷ 13
- 4 Relación con la probabilidad a posteriori ▷ 17
- 5 Optimizaciones: aprendizaje de distancias, edición y condensado ▷ 22

Índice

- 1 *Introducción: espacio métrico y distancias* ▷ 3
- 2 Vecino más cercano ▷ 7
- 3 k -vecinos más cercanos ▷ 13
- 4 Relación con la probabilidad a posteriori ▷ 17
- 5 Optimizaciones: aprendizaje de distancias, edición y condensado ▷ 22

Introducción

- **Clasificadores basados en distancias**: basados en la similitud entre la muestras a clasificar y las *muestras* etiquetadas (**prototipos**) dados
- En clasificación un prototipo representa el objeto \mathbf{x} y su clase c
- Los prototipos son almacenados para la clasificación de muestras sin etiquetar mediante el cálculo de distancias
- Formalmente:
 - Prototipos: conjunto de pares $X = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_N, c_N)\}$
 - Clasificación: dado \mathbf{y} , clasificar según las distancias $d(\mathbf{y}, \mathbf{x}_n)$, $1 \leq n \leq N$
- **Espacio de representación E** : debe ser *Métrico* o *Pseudométrico*
 - E puede ser un espacio no vectorial ($E \neq \mathbb{R}^D$)
- **Medida de disimilitud o distancia d** : función que dada la representación de dos objetos $(x, x') \in E \times E$ devuelve un valor de *disimilitud*, que debería correlarse con la disimilitud de dichos objetos en la realidad

Espacios métricos y pseudométricos

Espacio métrico:

- Par (E, d)
- E : espacio de representación
- d : función *métrica* o ***distancia***, $d : E \times E \rightarrow \mathbb{R}$, que cumple $(\forall p, q, r \in E)$:
 - $d(p, q) \geq 0$; $d(p, q) = 0 \Leftrightarrow p = q$ *(Positiva o nula)*
 - $d(p, q) = d(q, p)$ *(Simétrica)*
 - $d(p, q) + d(q, r) \geq d(p, r)$ *(Desigualdad Triangular)*

Espacio pseudométrico:

- Versión “simplificada”
- No requiere *simetría* ni *desigualdad triangular*
- $d(\cdot, \cdot)$ se denomina ***medida de disimilitud***

Distancias usuales

Representaciones vectoriales D -dimensionales ($E = \mathbb{R}^D$)

- Familia L_p : $d_p(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^D |a_i - b_i|^p \right)^{\frac{1}{p}}$

$L1$	$L2$ (Euclídea)	$L_\infty / L0$
$\sum_{i=1}^D (a_i - b_i) $	$\left(\sum_{i=1}^D (a_i - b_i)^2 \right)^{\frac{1}{2}}$	$\max_{1 \leq i \leq D} (a_i - b_i) $

- Distancia Euclídea Ponderada: $d(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^D w_i \cdot (a_i - b_i)^2 \right)^{\frac{1}{2}}$

Representaciones estructurales por cadenas de primitivas ($E \subseteq \Sigma^$)*

- Distancia (ponderada) de edición: $d(a, b) = \text{mínima talla (o peso) de una secuencia de operaciones de edición que transforma } a \text{ en } b$

Índice

- 1 Introducción: espacio métrico y distancias ▷ 3
- 2 *Vecino más cercano* ▷ 7
- 3 k -vecinos más cercanos ▷ 13
- 4 Relación con la probabilidad a posteriori ▷ 17
- 5 Optimizaciones: aprendizaje de distancias, edición y condensado ▷ 22

Clasificación por el vecino más cercano

Sea $d : E \times E \rightarrow \mathbb{R}$ una métrica y sea y un punto de E

Vecino más cercano (Nearest Neighbour, NN):

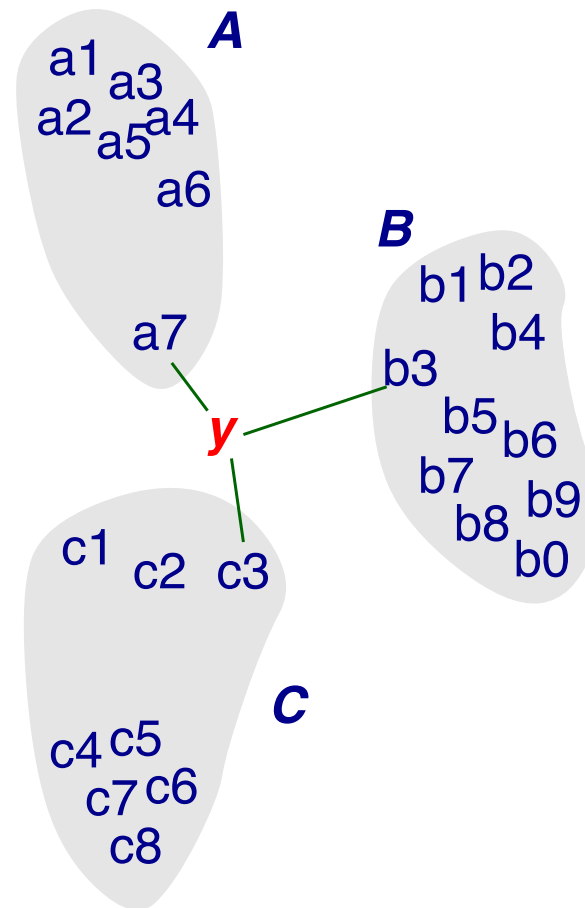
Sea X_c el conjunto de prototipos de la clase c :

$$y \in \hat{c} \Leftrightarrow \exists \mathbf{x} \in X_{\hat{c}} : d(y, \mathbf{x}) \leq d(y, \mathbf{x}') \quad \forall \mathbf{x}' \in X_c, 1 \leq c \leq C, c \neq \hat{c}$$

En caso de empate decidir, entre las empatadas, la clase con mayor número de prototipos o bien aleatoriamente

Clasificación por el vecino más cercano

Regla NN



$$NN(y)=a7$$

$$\hat{c} = A$$

Fronteras de decisión

Sea E espacio vectorial con *métrica euclídea*:

$$E = \mathbb{R}^D; \quad d(\mathbf{y}, \mathbf{x}) = \sqrt{(\mathbf{y} - \mathbf{x})^t (\mathbf{y} - \mathbf{x})}$$

La *frontera de decisión* entre las clases i y j son los puntos *equidistantes* al prototipo más cercano de cada clase:

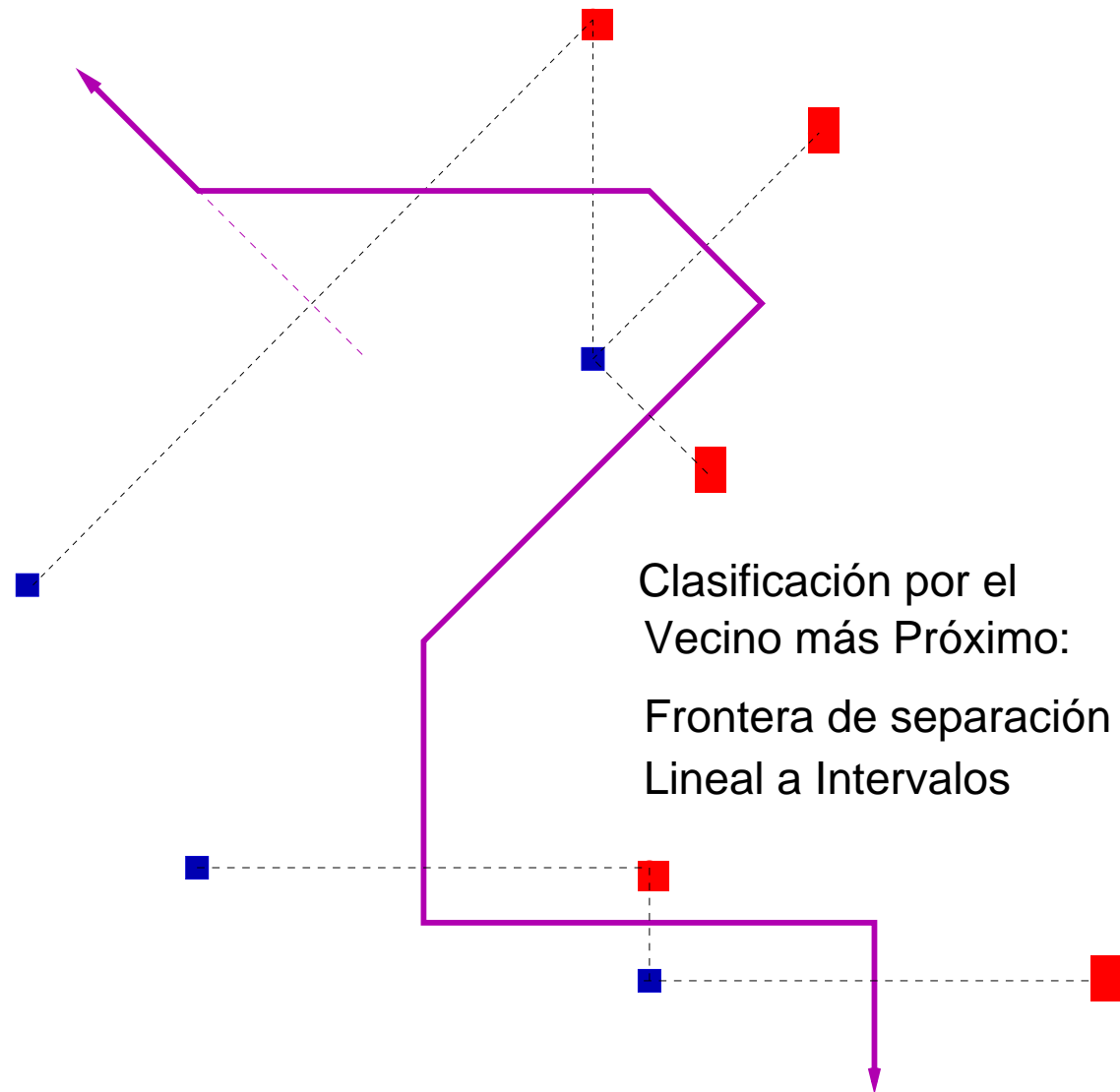
$$\min_{\mathbf{x} \in X_i} (d(\mathbf{y}, \mathbf{x})) = \min_{\mathbf{x} \in X_j} (d(\mathbf{y}, \mathbf{x})) \equiv \min_{\mathbf{x} \in X_i} (\|\mathbf{y} - \mathbf{x}\|^2) = \min_{\mathbf{x} \in X_j} (\|\mathbf{y} - \mathbf{x}\|^2) \equiv$$

$$\min_{\mathbf{x} \in X_i} (-2\mathbf{x}^t \mathbf{y} + \mathbf{x}^t \mathbf{x}) = \min_{\mathbf{x} \in X_j} (-2\mathbf{x}^t \mathbf{y} + \mathbf{x}^t \mathbf{x})$$

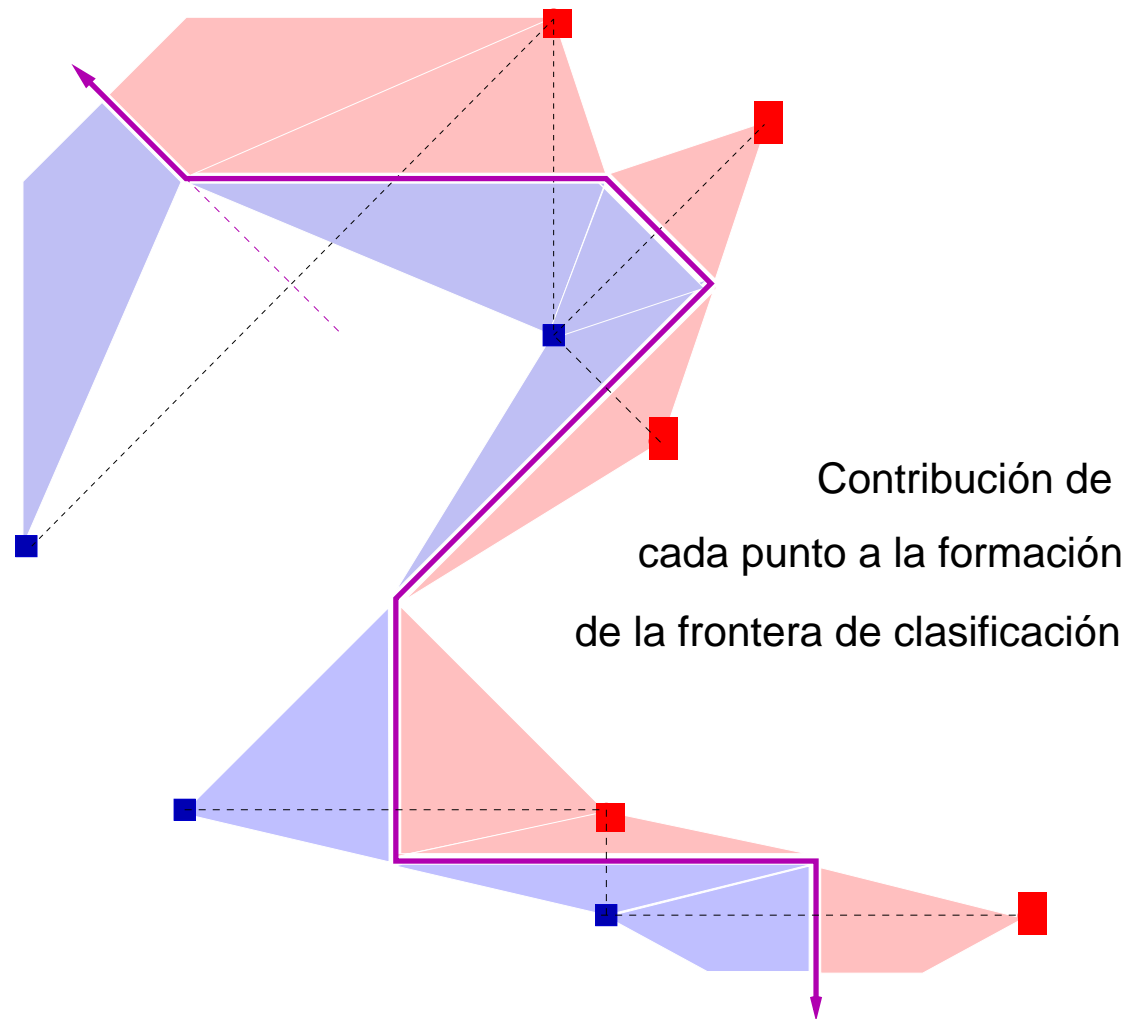
Separación lineal dependiente de los prototipos involucrados: *Funciones Discriminantes Lineales a Trozos (LT)*

Existen fronteras de decisión LT que no pueden obtenerse mediante ningún clasificador NN

Fronteras de decisión asociadas al clasificador NN



Fronteras de decisión asociadas al clasificador NN



Índice

- 1 Introducción: espacio métrico y distancias ▷ 3
- 2 Vecino más cercano ▷ 7
- 3 *k-vecinos más cercanos* ▷ 13
- 4 Relación con la probabilidad a posteriori ▷ 17
- 5 Optimizaciones: aprendizaje de distancias, edición y condensado ▷ 22

Clasificación por los k -vecinos más cercanos

Sea $d : E \times E \rightarrow \mathbb{R}$ una métrica y sea y un punto de E

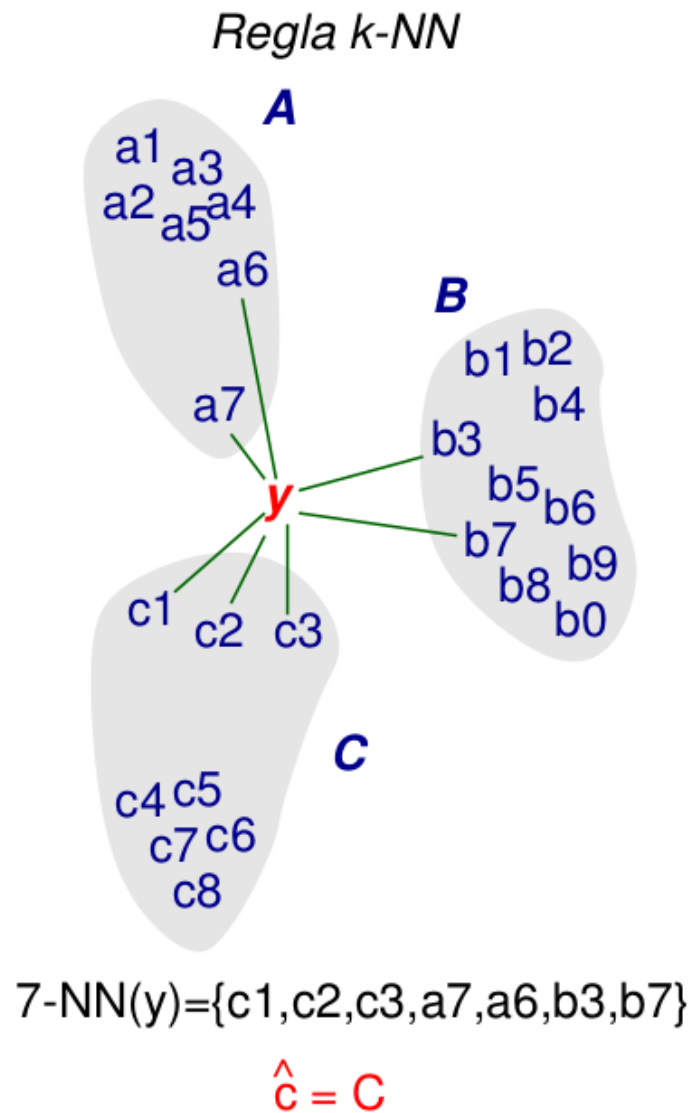
k -vecinos más cercanos (k -NN):

- Sea X_c el conjunto de prototipos representante de la clase c
- Sea X^k el conjunto de los $k \in \mathbb{N}^+$ prototipos más próximos a y

$$y \in \hat{c} \Leftrightarrow |X^k \cap X_{\hat{c}}| \geq |X^k \cap X_c| \quad 1 \leq c \leq C, c \neq \hat{c}$$

- En caso de empate, decidir entre las clases empatadas según 1-NN (1-NN equivale a NN)

Clasificación por los k -vecinos más cercanos



Fronteras de decisión

Sea E espacio vectorial con *métrica euclídea*:

$$E = \mathbb{R}^D; \quad d(\mathbf{y}, \mathbf{x}) = \sqrt{(\mathbf{y} - \mathbf{x})^t (\mathbf{y} - \mathbf{x})}$$

La *frontera de decisión* para *k-NN* vendrá dada por los puntos que empatan a número máximo de vecinos de dos clases distintas:

$$|X^k \cap X_i| = |X^k \cap X_j|$$

Separación dada por *Funciones Discriminantes Lineales a Trozos*

Toda frontera de decisión LT puede obtenerse mediante FDs *k-NN*

Índice

- 1 Introducción: espacio métrico y distancias ▷ 3
- 2 Vecino más cercano ▷ 7
- 3 k -vecinos más cercanos ▷ 13
- 4 *Relación con la probabilidad a posteriori* ▷ 17
- 5 Optimizaciones: aprendizaje de distancias, edición y condensado ▷ 22

k -NN y probabilidad a posteriori

Planteamiento habitual de estimación de $P(c \mid \mathbf{y})$: por fórmula de Bayes

- A partir de los datos de entrenamiento, estimar:
 - *Probabilidades a priori* de las clases $P(c)$, $1 \leq c \leq C$
 - *Probabilidad condicional* de cada clase $p(\mathbf{y}|c)$, $\mathbf{y} \in E$, $1 \leq c \leq C$
- Clasificar por máxima *probabilidad a posteriori* según la *regla de Bayes*:

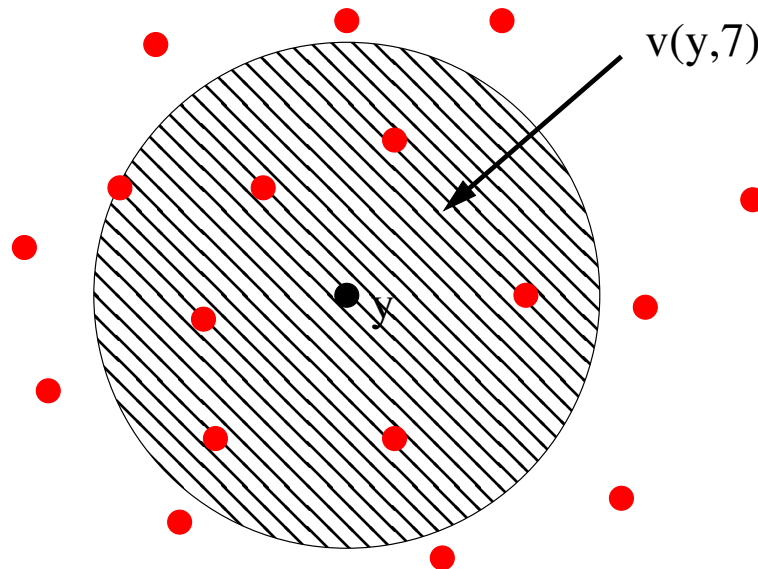
$$P(c \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid c) P(c)}{\sum_{c'=1}^C p(\mathbf{y} \mid c') P(c')}$$

Posibilidad alternativa: *estimar directamente $P(c \mid \mathbf{y})$ a partir de los datos mediante k -NN*

k -NN y probabilidad a posteriori

Dado y un punto en el que queremos estimar $P(c \mid y)$, sean:

$v(y, k)$: volumen de la menor hiperesfera centrada en y que contiene a los k vecinos más próximos de y (de cualquier clase)



k_c : número de prototipos de la clase c de entre los k -NN, $\sum_{c=1}^C k_c = k$

N_c : número de prototipos de la clase c , $\sum_{c=1}^C N_c = N$ (todos los prototipos)

k -NN y probabilidad a posteriori

Estimadores:

- Probabilidades *a priori*: $\hat{P}(c) = \frac{N_c}{N}$, $1 \leq c \leq C$
- *Masa de probabilidad* de la clase c en la hiperesfera de volumen $v(\mathbf{y}, k)$ centrada en \mathbf{y} : k_c/N_c
- Con $v(\mathbf{y}, k)$ infinitesimal, la *condicional* es: $\hat{p}(\mathbf{y} \mid c) = \frac{k_c/N_c}{v(\mathbf{y}, k)} = \frac{k_c}{N_c v(\mathbf{y}, k)}$
- *Probabilidad a posteriori* por regla de Bayes:

$$\hat{P}(c \mid \mathbf{y}) = \frac{\frac{k_c}{N_c v(\mathbf{y}, k)} \frac{N_c}{N}}{\sum_{c'=1}^C \frac{k_{c'}}{N_{c'} v(\mathbf{y}, k)} \frac{N_{c'}}{N}} = \frac{k_c}{k}$$

Regla de clasificación: $\hat{c} = \operatorname{argmax}_c \hat{P}(c \mid \mathbf{y}) = \operatorname{argmax}_c \frac{k_c}{k} = \operatorname{argmax}_c k_c$

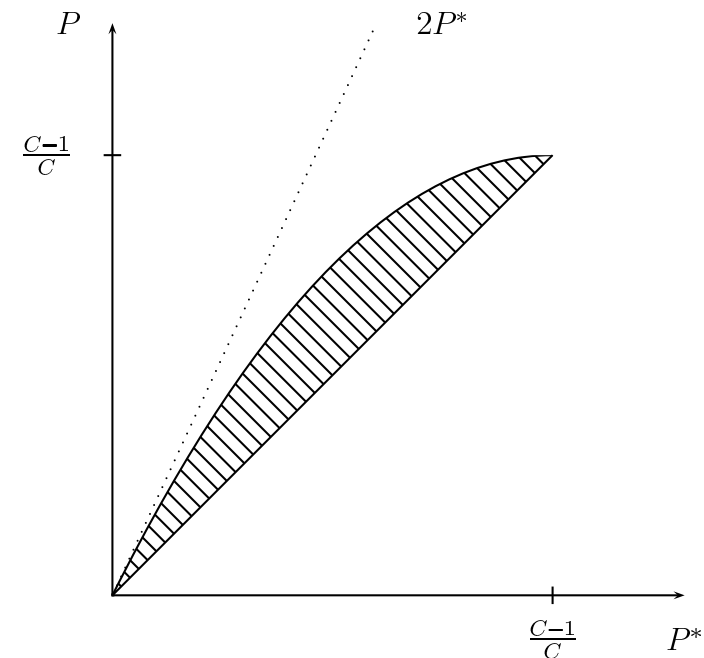
Es decir, clasificar \mathbf{y} en la clase ***a la que pertenezcan la mayoría de sus k vecinos más próximos***

Probabilidad de error de los clasificadores NN y k -NN

Sea P^* el error de Bayes

Cuando el número de prototipos es ilimitado ($N \rightarrow \infty$), el riesgo de **error del clasificador NN** puede acotarse por:

$$P^* \leq P \leq P^* \left(2 - \frac{C}{C-1} P^* \right) \leq 2P^*$$



El riesgo de **error del clasificador k -NN** tiende al **error de Bayes** si:

$$N \rightarrow \infty; \quad k \rightarrow \infty; \quad \frac{k}{N} \rightarrow 0$$

Las dos últimas condiciones se cumplen, por ejemplo, tomando $k = \sqrt{N}$

Índice

- 1 Introducción: espacio métrico y distancias ▷ 3
- 2 Vecino más cercano ▷ 7
- 3 k -vecinos más cercanos ▷ 13
- 4 Relación con la probabilidad a posteriori ▷ 17
- 5 *Optimizaciones: aprendizaje de distancias, edición y condensado* ▷ 22

Optimizaciones de k -NN

- En la clasificación por k -NN, el único parámetro *directo* a establecer es k
- Sin embargo, existen un par de *meta*-parámetros con gran influencia en la calidad de la clasificación:
 - ***Distancia empleada***: puede adecuarse a la distribución de prototipos de cada clase y eliminar efectos de escala en las distintas componentes
 - ***Conjunto de prototipos*** (puede no ser el disponible originalmente)
 - Puede *limpiarse*: fronteras de decisión más simples, mayor generalización
 - Puede *reducirse*: clasificador más compacto en memoria y más rápido en búsqueda

Aprendizaje de distancias

Limitación: aprendizaje de los pesos asociados a la distancia euclídea ponderada:

$$d(\mathbf{y}, \mathbf{p}) = \sqrt{\left(\sum_{i=1}^D w_i \cdot (y_i - p_i)^2 \right)}$$

- $\mathbf{y} \in \mathbb{R}^D$ es el objeto a clasificar
- $\mathbf{p} \in \mathbb{R}^D$ es un prototipo del conjunto de aprendizaje disponible
- $\mathbf{w} \in \mathbb{R}^D$ es el vector de pesos

Restricción: $w_i > 0, i = 1, \dots, D$ (para que la distancia sea una métrica)

Aprendizaje de distancias

Distancia euclídea normalizada o *Mahalanobis-diagonal*:

$$d(\mathbf{y}, \mathbf{p}) = \sqrt{\left(\sum_{i=1}^D \frac{1}{\sigma_i^2} (y_i - p_i)^2 \right)}$$

- σ_i^2 : varianza de la componente i -ésima de la representación vectorial de los prototipos
- $w_i = \frac{1}{\sigma_i^2}$: inversa de la varianza de la componente i -ésima

Equivale a pre-normalizar los datos dividiendo cada componente por su desviación típica σ_i y usar la distancia euclídea sobre los datos normalizados

Aprendizaje de distancias

Distancia *Mahalanobis-diagonal por clase*:

$$d(\mathbf{y}, \mathbf{p}) = \sqrt{\left(\sum_{i=1}^D \frac{1}{\sigma_{ic}^2} (y_i - p_i)^2 \right)}$$

- c : clase de \mathbf{p}
- σ_{ic}^2 : varianza de la componente i -ésima en clase c
- $w_i = \frac{1}{\sigma_{ic}^2}$: inversa de la varianza de la componente i -ésima considerando sólo prototipos de la clase c

Esta distancia *ya no es una métrica*: pesos diferentes según la clase de \mathbf{p}

Aprendizaje de distancias

Distancia *Mahalanobis-Local*:

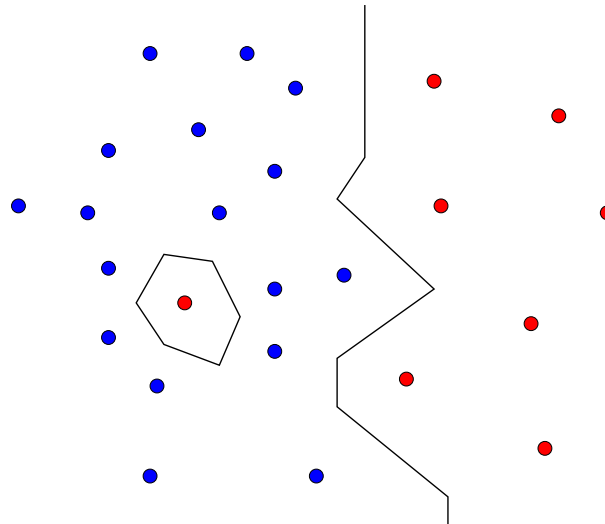
$$d(\mathbf{y}, \mathbf{p}) = \sqrt{\left(\sum_{i=1}^D \frac{1}{\sigma_{i\mathbf{p}}^2} (y_i - p_i)^2 \right)}$$

- $\sigma_{i\mathbf{p}}^2$: varianza de la componente i -ésima de los prototipos que son k -NN de \mathbf{p} de su misma clase
- $w_i = \frac{1}{\sigma_{i\mathbf{p}}^2}$: inversa de la varianza de la componente i -ésima calculada sobre los prototipos k -NN de \mathbf{p} de su misma clase \rightarrow estimación de la varianza local de la clase c

Esta distancia *tampoco es una métrica*: pesos diferentes dependiendo de \mathbf{p}

Edición de prototipos

- Objetivo: eliminar prototipos ruidosos
- Prototipo ruidoso: prototipo de una clase *aislado* dentro de la zona de prototipos de otra clase

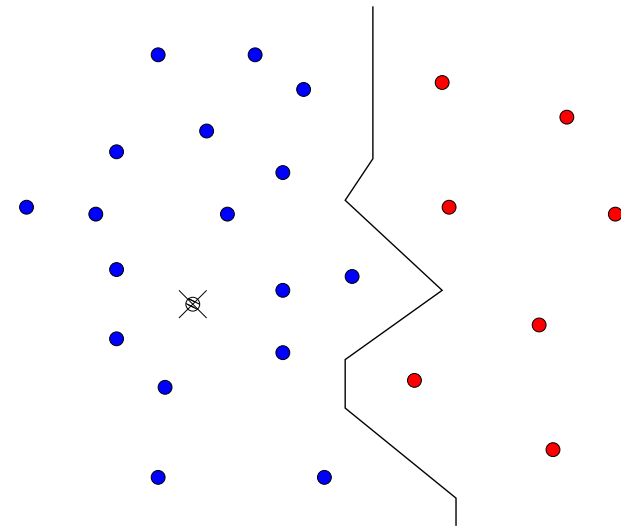


- El punto ruidoso genera *huecos* en las regiones de decisión
- Eliminar prototipos ruidosos da regiones de decisión simplemente conexas (sin *huecos*)

Edición de prototipos

Algoritmo de edición de Wilson

- Clasificación por k -NN (k parámetro) de cada prototipo frente al resto
- Elimina los prototipos cuya clasificación sea diferente de su propia clase
- Finalización: todos los prototipos se clasifican correctamente
- Coste computacional para N prototipos: $O(N^2)$
 - Probar N prototipos
 - Para cada uno calcular vecinos más cercanos (N distancias)
- Técnicas para reducir el coste:
 - Almacenar las distancias ya ordenadas en una matriz en la primera iteración
 - Emplear técnicas de búsqueda rápida
- Usualmente consigue eliminar los prototipos ruidosos
- Crítica principal: resultado dependiente del orden de recorrido



Edición de prototipos

Algoritmo de Wilson:

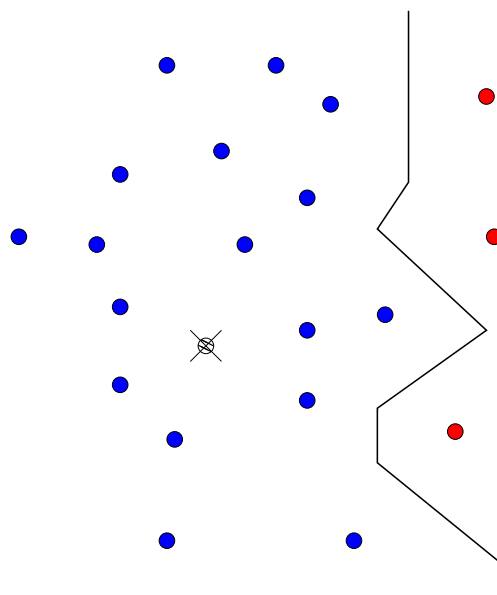
- Entrada: $X = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_N, c_N)\}, k, d$
- Salida: $X' = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_M, c_M)\} \quad M \leq N$
- Algoritmo:
 1. error=true; $X' = X$;
 2. while (error)
 3. error=false;
 4. for n=1:N
 5. $\hat{c} = knn(\mathbf{x}_n, X' - \mathbf{x}_n, d, k)$;
 6. if ($\hat{c} \neq c_n$) $X' = X' - \{\mathbf{x}_n\}$; error=true;
 7. endfor
 8. endwhile

Entrada: conjunto de prototipos original, valor de k , distancia d a emplear

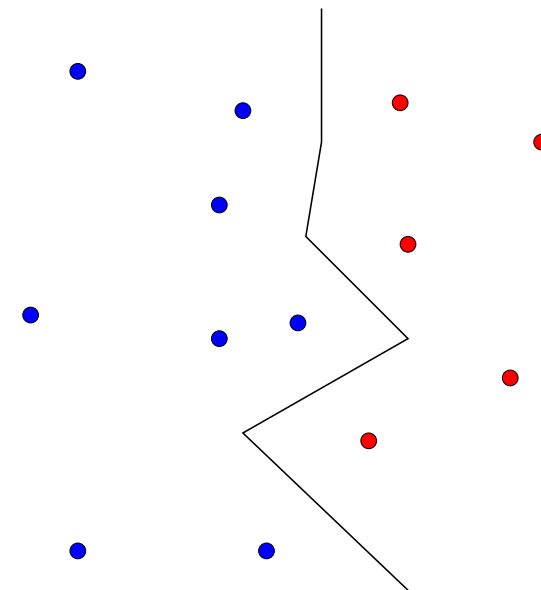
Salida: conjunto reducido o igual, $X' \subseteq X$

Condensado de prototipos

- Objetivo: reducir drásticamente el conjunto de prototipos sin modificar significativamente las fronteras de decisión
- Algunos algoritmos de condensado deben partir del conjunto de prototipos ya editado (sin ruido)



Editado



Condensado

Condensado de prototipos

Algoritmo CNN (Condensed Nearest Neighbor, Hart, 1968):

- Definir dos conjuntos de prototipos:
 - STORE (S): prototipos a retener
 - GARBAGE (G): prototipos a descartar
- Algoritmo en dos fases:
 1. Crear S y G desde los prototipos originales (conjunto X)
 2. Recorrer G hasta que quede vacío o no sufra modificaciones
- Esencialmente:
 - S mantiene los prototipos clasificados incorrectamente
 - G mantiene los prototipos clasificados correctamente
- Crítica principal: resultado dependiente del orden de recorrido

Existen otras propuestas de condensado (Wilson and Martinez, 2000)

Condensado de prototipos

Algoritmo CNN

- Entrada: $X = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_N, c_N)\}$ editado, k, d
- Salida: S
- Algoritmo:

1. // Primera fase

- $S = G = \emptyset$
- $\{(\mathbf{x}_1, c_1)\} \rightarrow S$
- for $n=2:N$
- $\hat{c} = knn(\mathbf{x}_n, S, d, k);$
- if $(\hat{c} \neq c_n)$ $\{(\mathbf{x}_n, c_n)\} \rightarrow S$
- else $\{(\mathbf{x}_n, c_n)\} \rightarrow G$
- endfor

2. // Segunda fase

- error=true
- while $(G \neq \emptyset \ \&\& \text{error})$
- error=false;
- forall $(\mathbf{x}, c) \in G$
- $\hat{c} = knn(\mathbf{x}, S, d, k);$
- if $(\hat{c} \neq c)$
- $\{(\mathbf{x}, c)\} \rightarrow S;$
- $G = G - \{(\mathbf{x}, c)\};$
- error=true;
- endif
- endforall
- endwhile