UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria **Informàtica**

etsinf

Tema 8. Clasificación basada en distancias

Percepción (PER)

Curso 2017/2018

Departamento de Sistemas Informáticos y Computación

Índice

Índice

Introducción

- ***Clasificadores basados en distancias***: basados en la similitud entre la muestras a clasificar, y las *muestras* etiquetadas o ***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 $X = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\}$
  - Clasificación: dado $\mathbf{y}$, clasificar según las distancias $d(\mathbf{y}, \mathbf{x}_i)$, $1 \leq i \leq n$
- ***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***, $d : E \times E \to \mathbb{R}$, que cumple $(\forall p, q, r \in E)$:
  - $d(p, q) \geq 0; \quad d(p, q) = 0 \Leftrightarrow p = q$ *(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***

Espacios métricos y pseudométricos

- Los espacios métricos y pseudométricos son mas "primitivos" que los *espacios vectoriales*

- Todo espacio vectorial con *producto escalar* se convierte en métrico mediante la definición de la ***distancia euclídea***:

$$d(\boldsymbol{p}, \boldsymbol{q}) = \|\boldsymbol{p} - \boldsymbol{q}\| = \sqrt{(\boldsymbol{p} - \boldsymbol{q})^t (\boldsymbol{p} - \boldsymbol{q})} =$$

$$\sqrt{(p_1 - q_1)^2 + \cdots + (p_D - q_D)^2}$$

Distancias usuales

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

- Familia Lp: $d_p(\boldsymbol{a}, \boldsymbol{b}) = \left(\sum_1^D |a_i - b_i|^p \right)^{\frac{1}{p}}$

$L1$	$L2$ (Euclídea)	$L_\infty / L0$
$\sum_1^D \lvert(a_i - b_i)\rvert$	$\left(\sum_1^D (a_i - b_i)^2\right)^{\frac{1}{2}}$	$\text{máx}_{1 \leq i \leq D} \lvert(a_i - b_i)\rvert$

- Distancia Euclídea Ponderada: $d(\boldsymbol{a}, \boldsymbol{b}) = \left(\sum_1^D w_i \cdot (a_i - b_i)^2 \right)^{\frac{1}{2}}$

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

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

Índice

Clasificación por el vecino más cercano

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

Vecino más cercano (Nearest Neighbour, NN):

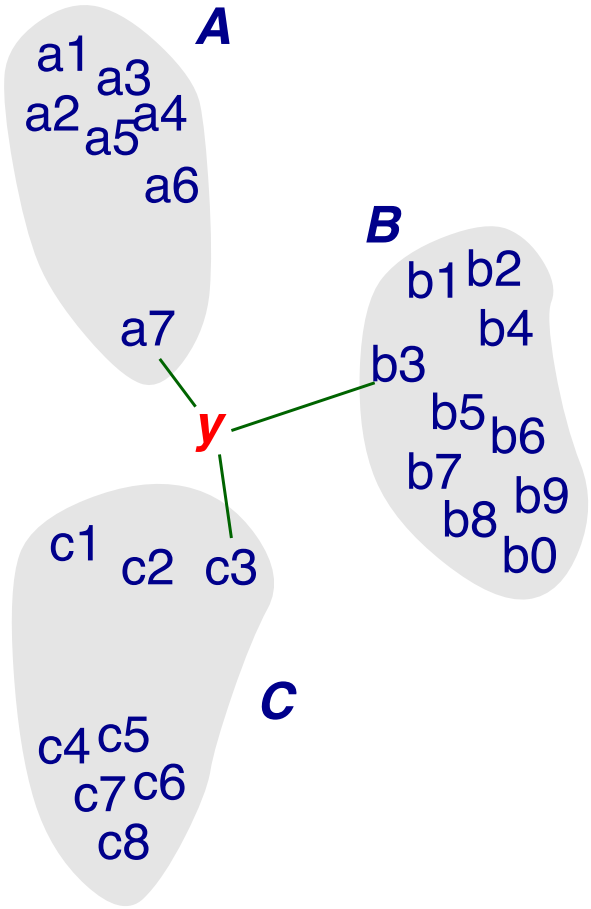
Sea X_c el *conjunto de prototipos* de la clase c:

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

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***:

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

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

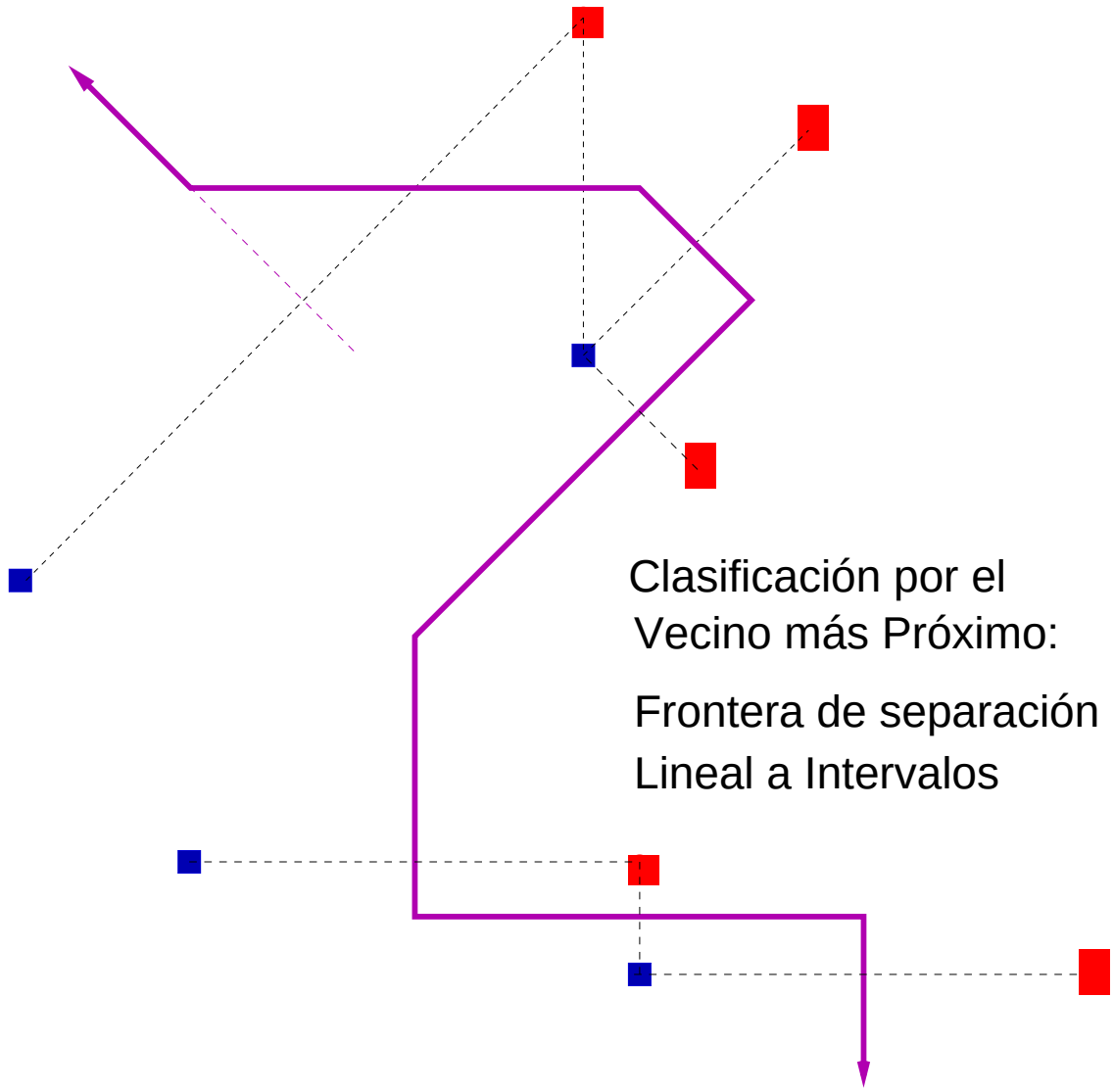
$$\min_{\mathbf{x}\in X_i}(d(\mathbf{y},\mathbf{x})) \;=\; \min_{\mathbf{x}\in X_j}(d(\mathbf{y},\mathbf{x})) \equiv \min_{\mathbf{x}\in X_i}(\|\mathbf{y}-\mathbf{x}\|^2) \;=\; \min_{\mathbf{x}\in X_j}(\|\mathbf{y}-\mathbf{x}\|^2) \equiv$$

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

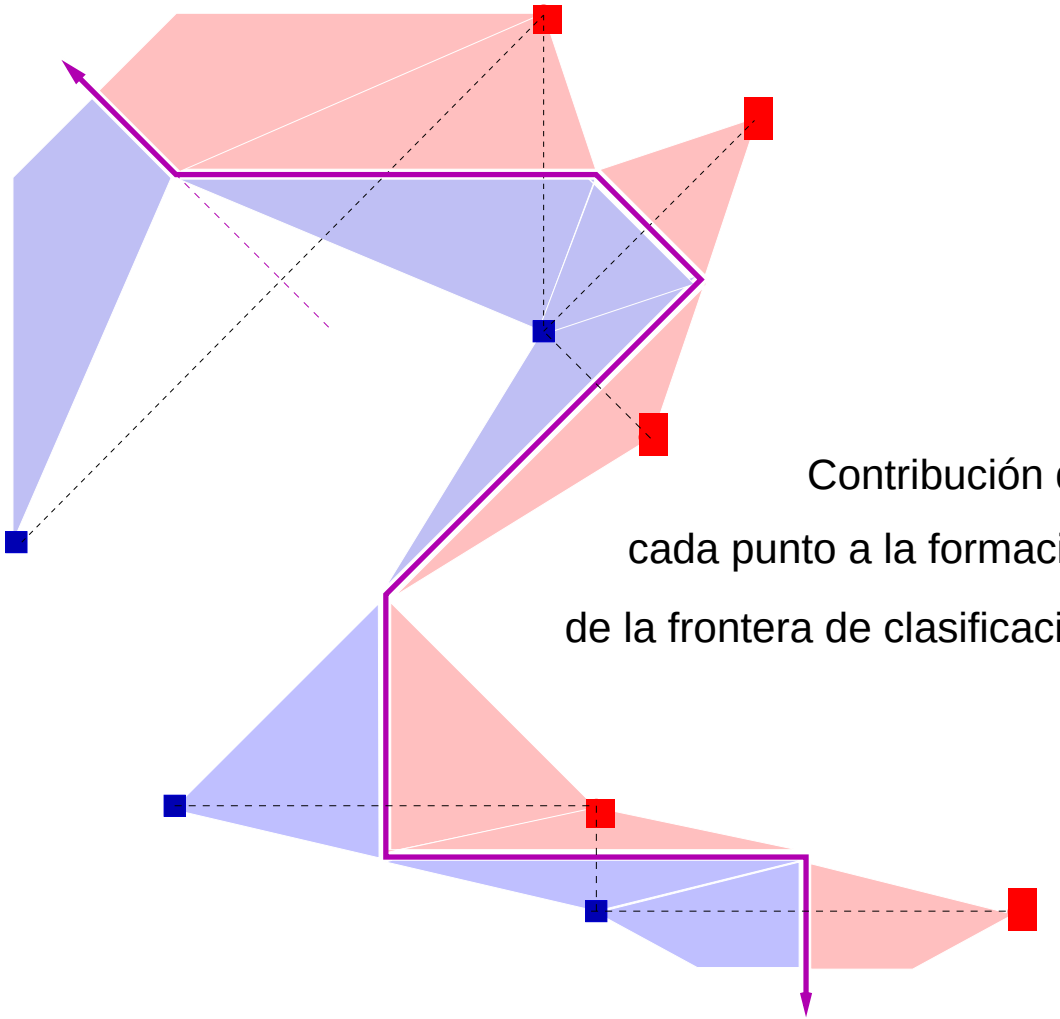
Separación lineal dependiente de los prototipos involucrados: ***Funciones Discriminantes Lineales a Trozos (LT)***

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



Contribución de

cada punto a la formación

de la frontera de clasificación

Índice

Clasificación por los k-vecinos más cercanos

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

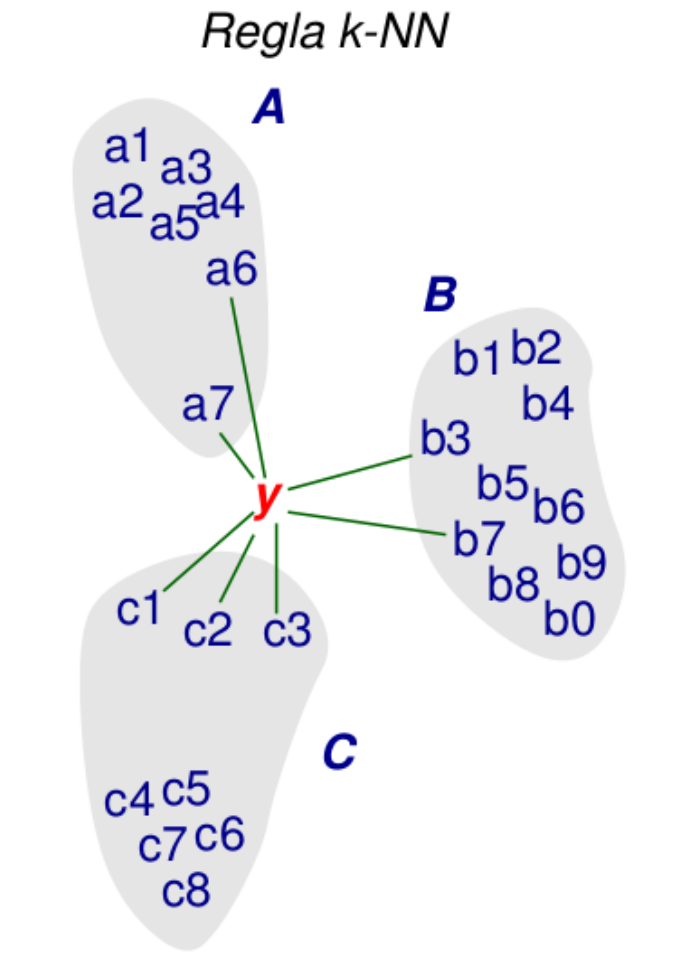
k-vecinos más cercanos (k-NN):

- 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 $\mathbf{y}$

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

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

Clasificación por los k-vecinos más cercanos



7-NN(y)={c1,c2,c3,a7,a6,b3,b7}

$\hat{c} = C$

Fronteras de decisión

Sea E espacio vectorial con ***métrica euclídea***:

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

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

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*:

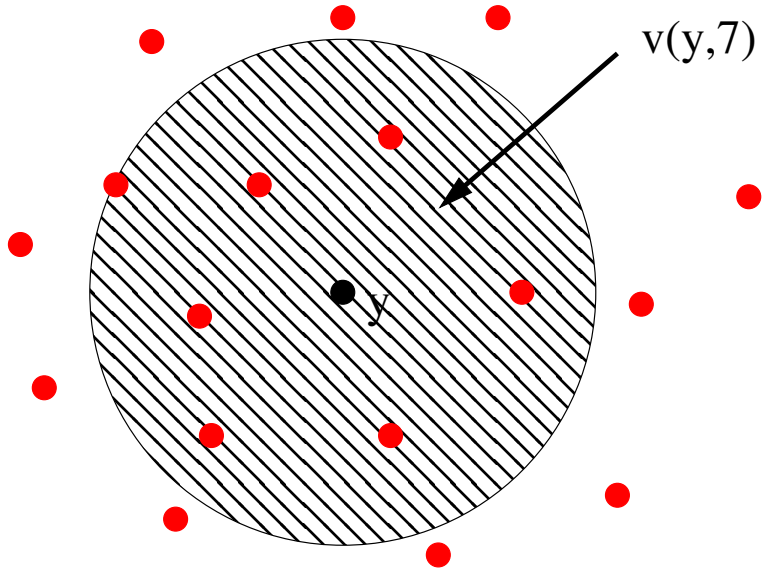
$$P(c \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid c)\ P(c)}{\sum_{c'=1}^{C} p(\mathbf{y} \mid c')\ P(c')}$$

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

k-NN y probabilidad a posteriori

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

$v(\mathbf{y}, k)$: volumen de la menor hiperesfera centrada en $\mathbf{y}$ que contiene a los k vecinos más próximos de $\mathbf{y}$ (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: $\hat{p}(\mathbf{y} \mid c) = \frac{k_c/n_c}{v(\mathbf{y},k)}$
- *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: $\hat{c} = \text{argmax}_c\ \hat{P}(c \mid \mathbf{y}) = \text{argmax}_c\ \frac{k_c}{k} = \text{argmax}_c\ k_c$

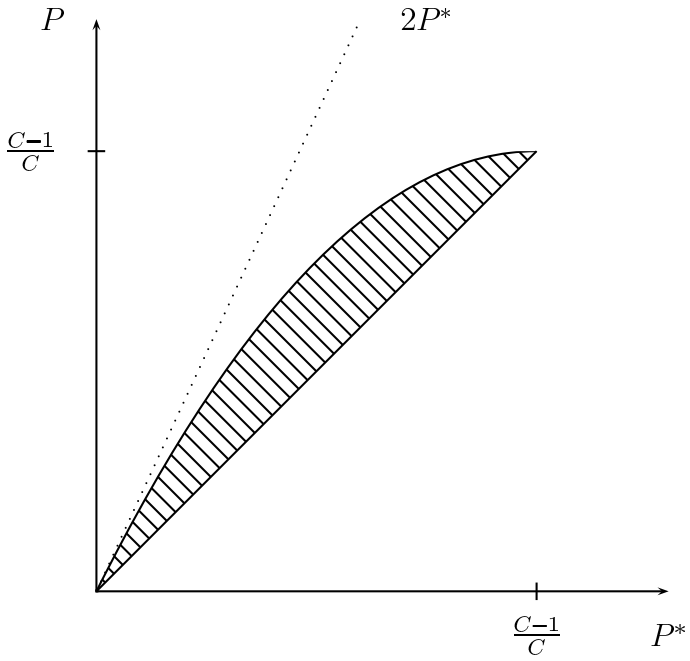
Es decir, clasificar $\mathbf{y}$ en la clase ***a la que pertenezcan la mayoría de sus k vecinos más próximos***

Probabilidad de error de los clasificadores NN y k-NN

Sea P^* el error de Bayes

Cuando el número de prototipos es ilimitado ($n \to \infty$), 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:

$$n \to \infty; \qquad k \to \infty; \qquad \frac{k}{n} \to 0$$

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

Índice

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: $\mathbf{w}_i > 0, i = 1, \ldots, D$ (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: desviación típica de la componente i-ésima de la representación vectorial de los datos
- $w_i = \frac{1}{\sigma_i^2}$: inversa de la varianza de la componente i-ésima

Equivale a pre-normalizar los datos dividiendo cada componente por la desviación típica 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***:

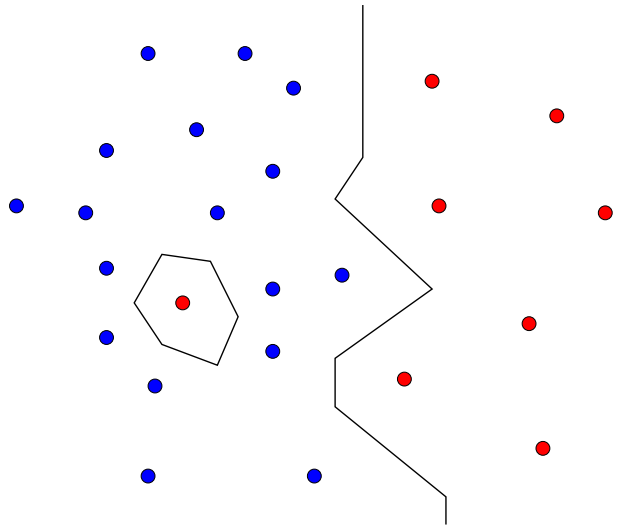
$$d(\mathbf{y}, \mathbf{p}) = \sqrt{\left(\sum_{i=1}^{D} \frac{1}{\sigma^2_{i\mathbf{p}}} (y_i - p_i)^2 \right)}$$

- $\sigma^2_{i\mathbf{p}}$: varianza de la componente i-ésima de los prototipos que son k-NN de $\mathbf{p}$
- $w_i = \frac{1}{\sigma^2_{ip}}$: 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

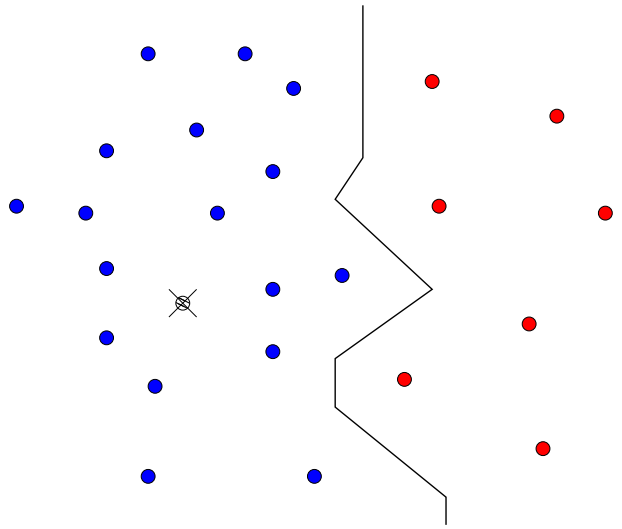


- El punto ruidoso genera *huecos* en las regiones de decisión
- Eliminar prototipos ruidosos da regiones de decisión simplemente conexas (sin *huecos*)

Edición de prototipos

Algoritmo de edición de Wilson

- Clasifica 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 por recorrido para n prototipos: $O(n^2)$
  - Probar n prototipos
  - Para cada uno calcular vecinos más cercanos (n distancias)
- Técnicas para bajar 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:

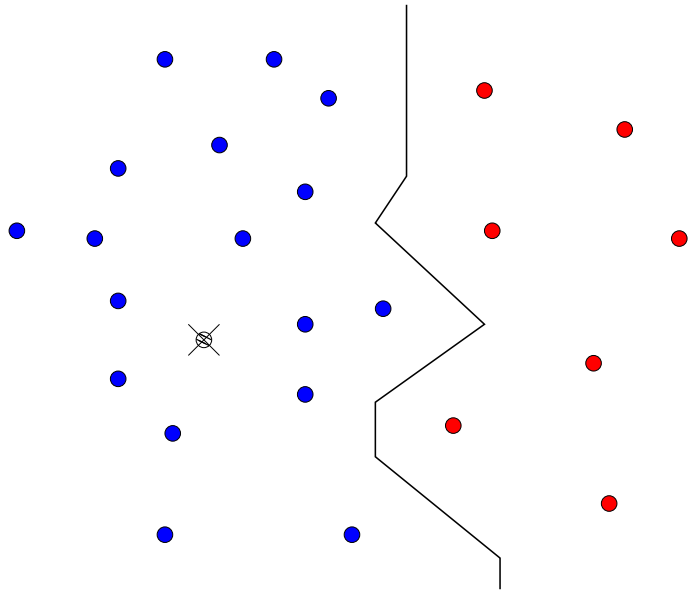
- Entrada: $X = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\}$, k, d
- Salida: $X' = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_m, c_m)\}$
- Algoritmo:

1. error=true;
2. while (error)
3. error=false;
4. for i=1:n
5. $\hat{c} = knn(\mathbf{x}_i, X - \mathbf{x}_i, d, k)$;
6. if $(\hat{c} \neq c_i)$ eliminar($\mathbf{x}_i$); error=true;
7. endfor
8. endwhile

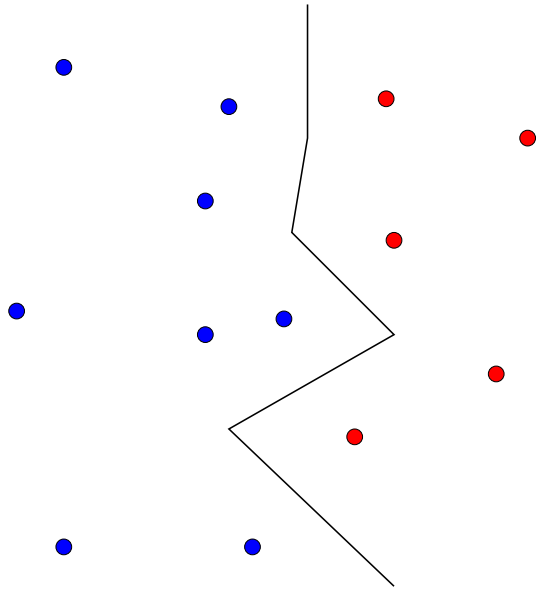
- Entrada: conjunto de prototipos original, valor de k, distancia d a emplear
- Salida: conjunto reducido o igual, $X' \subseteq X$

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

Condensado de prototipos

Algoritmo CNN

- Entrada: $X = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\}$ editado, k, d
- Salida: S
- Algoritmo:

1. // Primera fase
   a) S=G=$\emptyset$
   b) $\{(\mathbf{x}_1, c_1)\}$ →S
   c) for i=2:n
   d) $\quad \hat{c} = knn(\mathbf{x}_i, S, d, k)$;
   e) $\quad$ if $(\hat{c} \neq c_i)$ $\{(\mathbf{x}_i, c_i)\}$ →S
   f) $\quad$ else $\{(\mathbf{x}_i, c_i)\}$ →G
   g) endfor

2. //Segunda fase
   a) error=true
   b) while (G$\neq \emptyset$ && error)
   c) $\quad$ error=false;
   d) $\quad$ forall $(\mathbf{x}, c)$ ∈G
   e) $\quad\quad \hat{c} = knn(\mathbf{x}, S, d, k)$;
   f) $\quad\quad$ if $(\hat{c} \neq c)$
   g) $\quad\quad\quad \{(\mathbf{x}, c)\}$ →S;
   h) $\quad\quad\quad$ G=G-$\{(\mathbf{x}, c)\}$;
   i) $\quad\quad\quad$ error=true;
   j) $\quad\quad$ endif
   k) $\quad$ endforall
   l) endwhile