Ricardo Aler Mur

CLASIFICADORES KNN-I

En esta clase se habla del aprendizaje de modelos de clasificación y regresión basados en instancias o ejemplares. En concreto:

- Se define la clasificación y regresión basada en los vecinos más cercanos (KNN = k-nearest neighbours)
- Se explican algunas de las ventajas (no es necesario construir un modelo explícito) e inconvenientes de KNN, entre ellas, la lentitud en la clasificación cuando hay muchos datos, la dependencia de la función de distancia, la influencia nociva de atributos irrelevantes, la maldición de la dimensionalidad y la gran influencia del ruido en los datos.
- A continuación se muestra como se pueden solventar algunas de esas deficiencias:
    - Cómo seleccionar el parámetro K para combatir el ruido en los datos
    - Cómo seleccionar las instancias o datos más representativos para evitar la lentitud en la clasificación. Se explican las técnicas de edición (para eliminar datos engañosos/ruido) y las de condensación (para eliminar datos supérfluos), así como técnicas híbridas

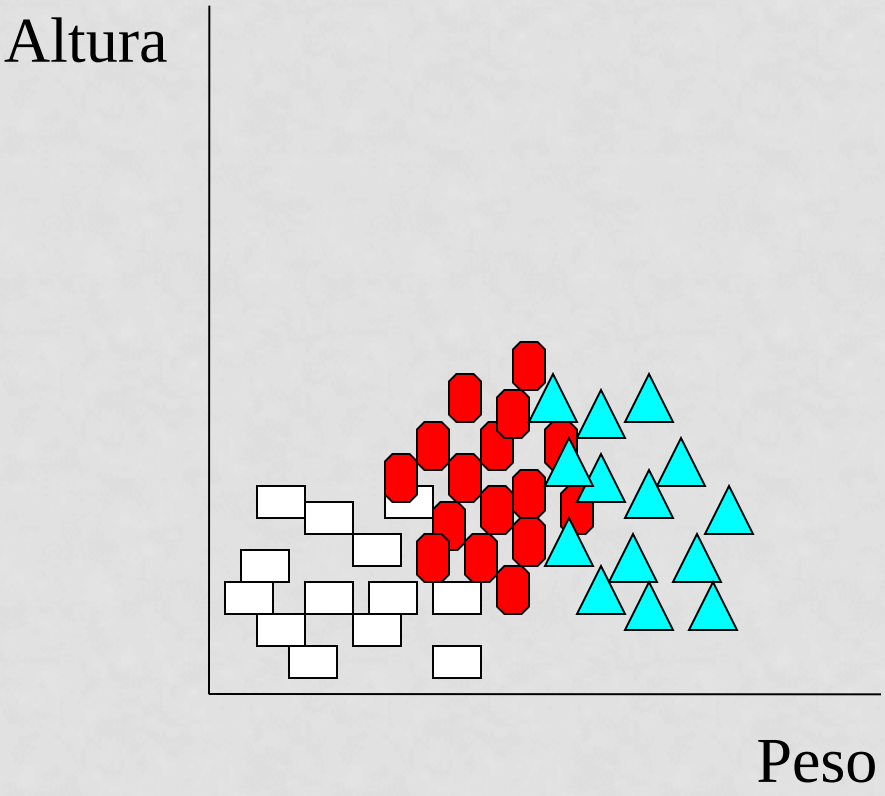
    - Por último, se explican los algoritmos RT1 / RT2 y RT3 que realizan edición y condensación de manera avanzada
- Se termina la clase introduciendo otra clase de algoritmos – los basados en prototipos - que clasifican basándose en la idea de vecindad, pero que eliminan el problema de la lentitud en la clasificación al utilizar como modelo un conjunto de prototipos en lugar del conjunto de datos completo. En concreto, se explica un algoritmo de posicionamiento de prototipos denominado Learning Vector Quantization o LVQ.

Fases del análisis de datos

- Recopilación de los datos (tabla datos x atributos)
- Preproceso:
  - De los datos
  - **De los atributos:**
    - **Selección de atributos:**
      - **Ranking**
      - **Subset selection: CFS y WRAPPER**
    - Transformación / Generación de atributos:
      - No supervisada: **PCA, random projections, autoencoders**
      - Supervisada: **mediante redes de neuronas**
- Generación del modelo:
  - **Clasificación: árboles de decisión**
  - **Regresión:**
    - **Modelos lineales**
    - **Árboles de modelos**
- Evaluación: **validación cruzada, matriz de confusión**
- Despliegue y uso del modelo

KNN: VECINO(S) MAS CERCANO(S)

K NEAREST NEIGHBORS (KNN)



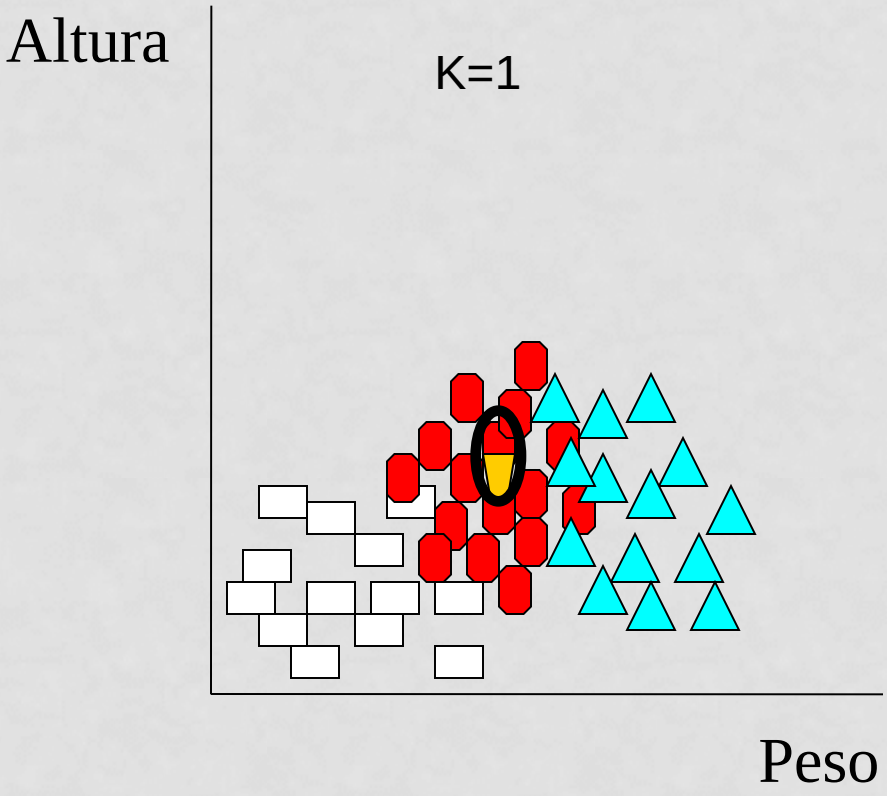
Niño

Adulto

Mayor

Se guardan todos los ejemplos

K NEAREST NEIGHBORS (KNN)



Niño

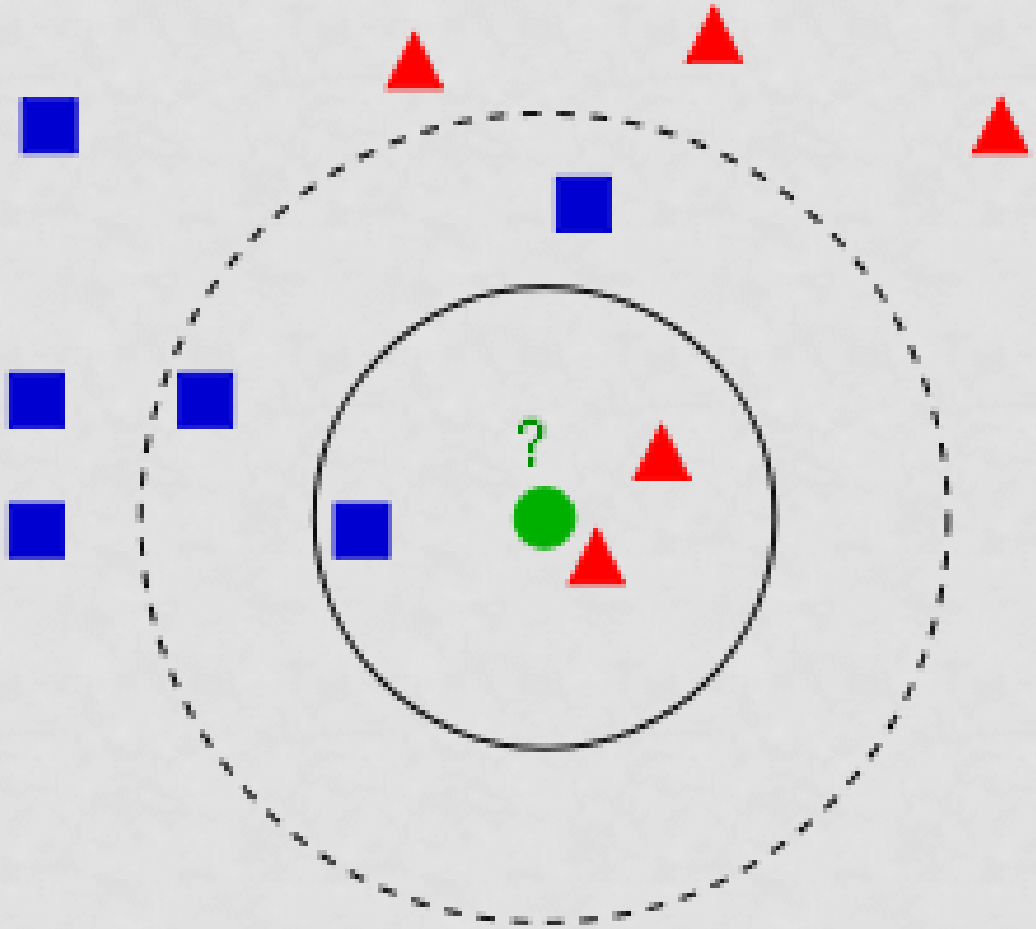
Adulto

Mayor

Se guardan todos los ejemplos

K NEAREST NEIGHBORS (KNN)

K=3



CARACTERISTICAS KNN

- KNN: Clasifica nuevas instancias como la clase mayoritaria de entre los k vecinos mas cercanos de entre los datos de entrenamiento
- KNN es un algoritmo perezoso (lazy)
  - Durante el entrenamiento, sólo guarda las instancias, no construye ningún modelo (a diferencia de, por ejemplo, los árboles de decisión)
  - La clasificación se hace cuando llega la instancia de test
- Es no paramétrico (no hace suposiciones sobre la distribución que siguen los datos, a diferencia de por ejemplo, un modelo lineal)
  - El mejor modelo de los datos son los propios datos

CARACTERISTICAS KNN

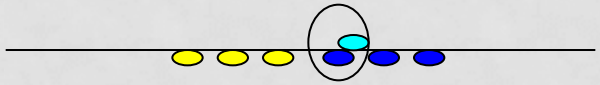
- KNN es local: asume que la clase de un dato depende sólo de los k vecinos mas cercanos (no se construye un modelo global)
- No necesita adaptación para mas de dos clases (a diferencia de clasificadores lineales)
- Si se quiere hace regresión, calcular la media de los k vecinos
- Es sencillo

Problemas / limitaciones KNN

- Muy sensible a los atributos irrelevantes y la maldición de la dimensionalidad
- Muy sensible al ruido
- Lento, si hay muchos datos de entrenamiento (O(n*d) en almacenamiento y en tiempo)
- Depende de que la función de distancia sea la adecuada
  - Normalmente es la Euclidea:
    - En 2D: $d(\mathbf{x_i},\mathbf{x_j})^2 = (x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2$
      - $d(\mathbf{x_i},\mathbf{x_j}) = \sqrt{((x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2)}$
    - En dD: $d(\mathbf{x_i},\mathbf{x_j})^2 = (x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2 + \ldots + (x_{id}-x_{jd})^2$

Atributos irrelevantes

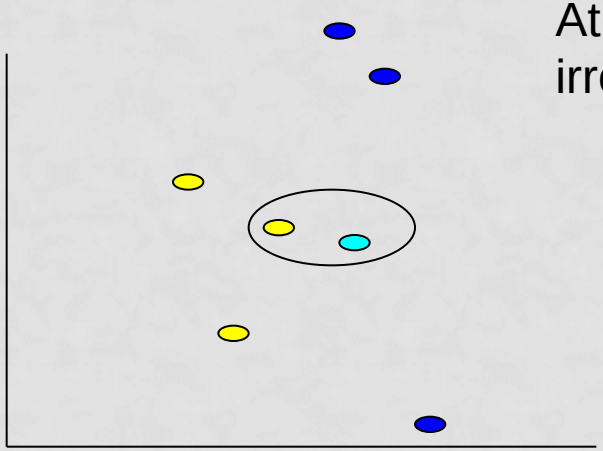
0 atributos irrelevantes



Con el atributo relevante, se clasifica bien

Atributo irrelevante

Atributo irrelevante



Con el atributo irrelevante, se clasifica mal (las distancias cambian)

1 atributo irrelevante

Atributo relevante

Vecino mas cercano k=1

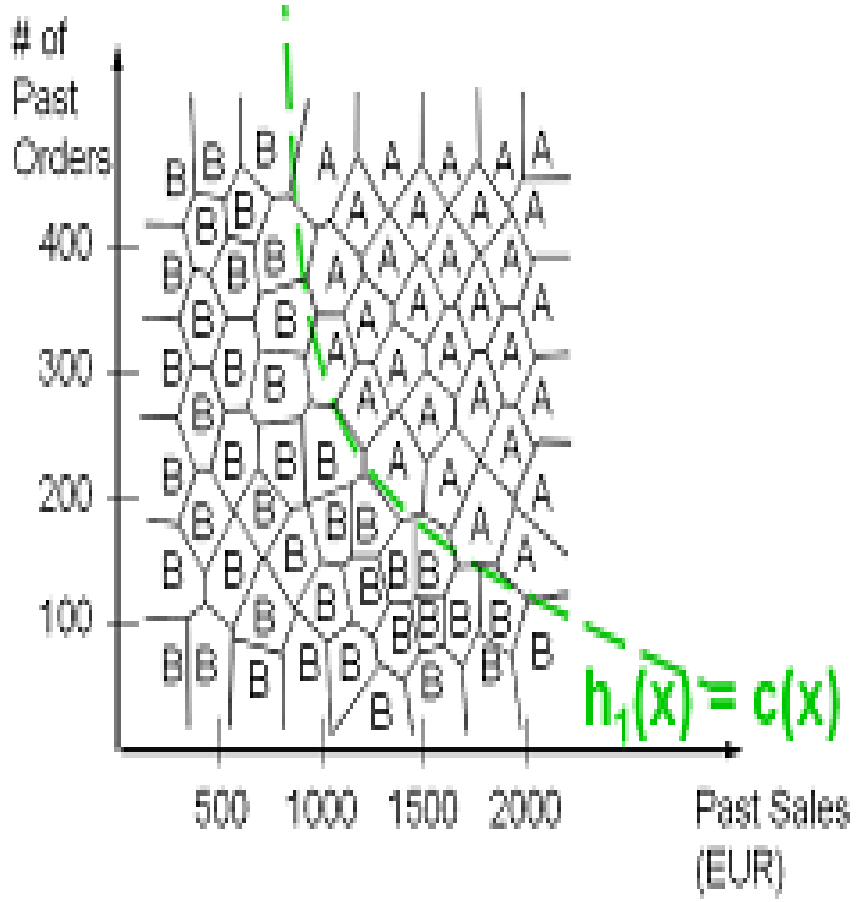
La maldición de la dimensionalidad

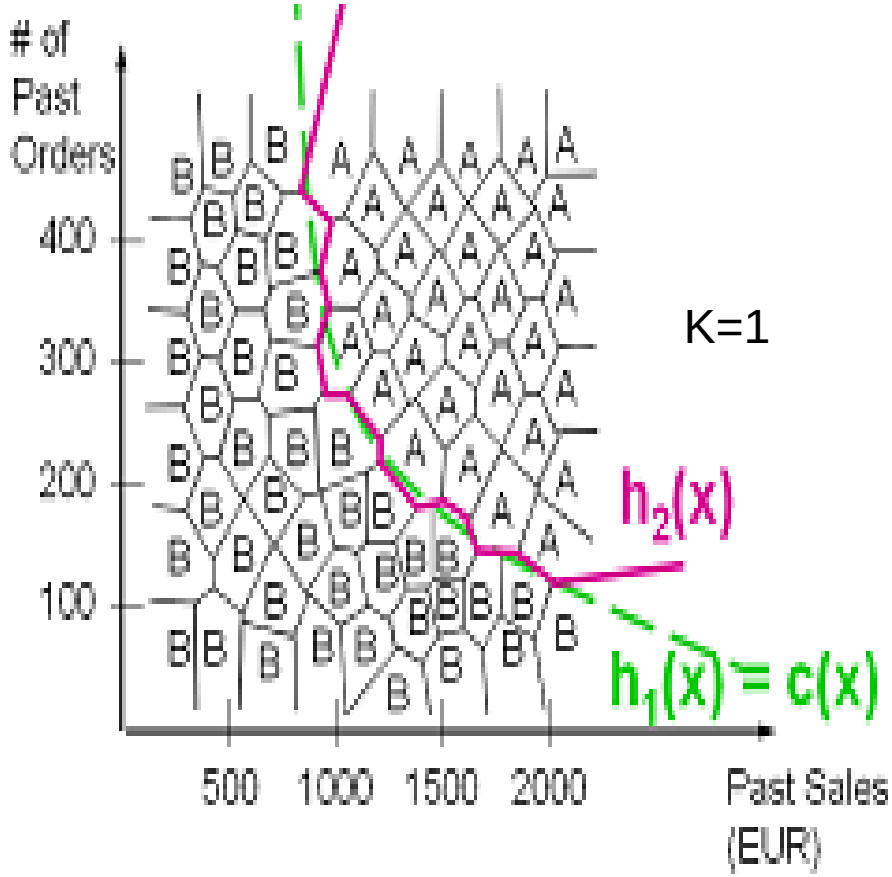
- Le afecta mucho, dado que KNN no crea un modelo global sino que utiliza los datos para representar las fronteras de separación
- Recordemos que el número de datos necesario para representar la superficie de una esfera en d dimensiones crece exponencialmente con d:
  - $O(r^d)$

Normalización

- Si los atributos son nominales, usar distancia de Hamming:
  - Si el atributo e es nominal, en lugar del componente $(x_{ie}-x_{je})^2$ se usa $\delta(x_{ie}, x_{je})$: 0 si $x_{ie}=x_{je}$ y 1 en caso contrario
- Es necesario normalizar los atributos para que atributos con mucho rango no tengan mas peso que los demás. Normalizaciones:
  - Estandarización: $x'_{1j}= (x_{1j}-\mu_j)/\sigma_j$
  - Normalización: $x'_{1j}= (x_{1j}-min_j)/(max_j-min_i)$
  - Es como usar una distancia ponderada:
    - $d(\mathbf{x_i},\mathbf{x_j}) = m_1*(x_{i1}-x_{j1})^2 + m_2*(x_{i2}-x_{j2})^2 + m_3*(x_{i3}-x_{j3})^2 + \text{---} + m_d*(x_{id}-x_{jd})^2$

TESELACIÓN DE VORONOI



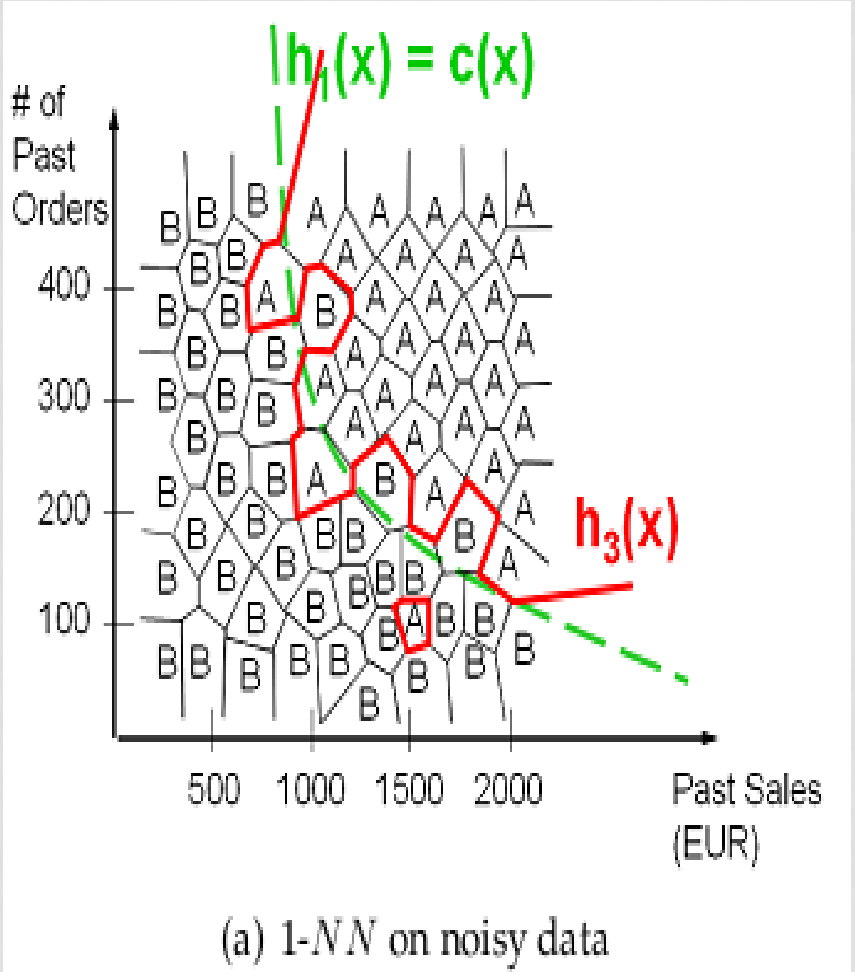


K=1

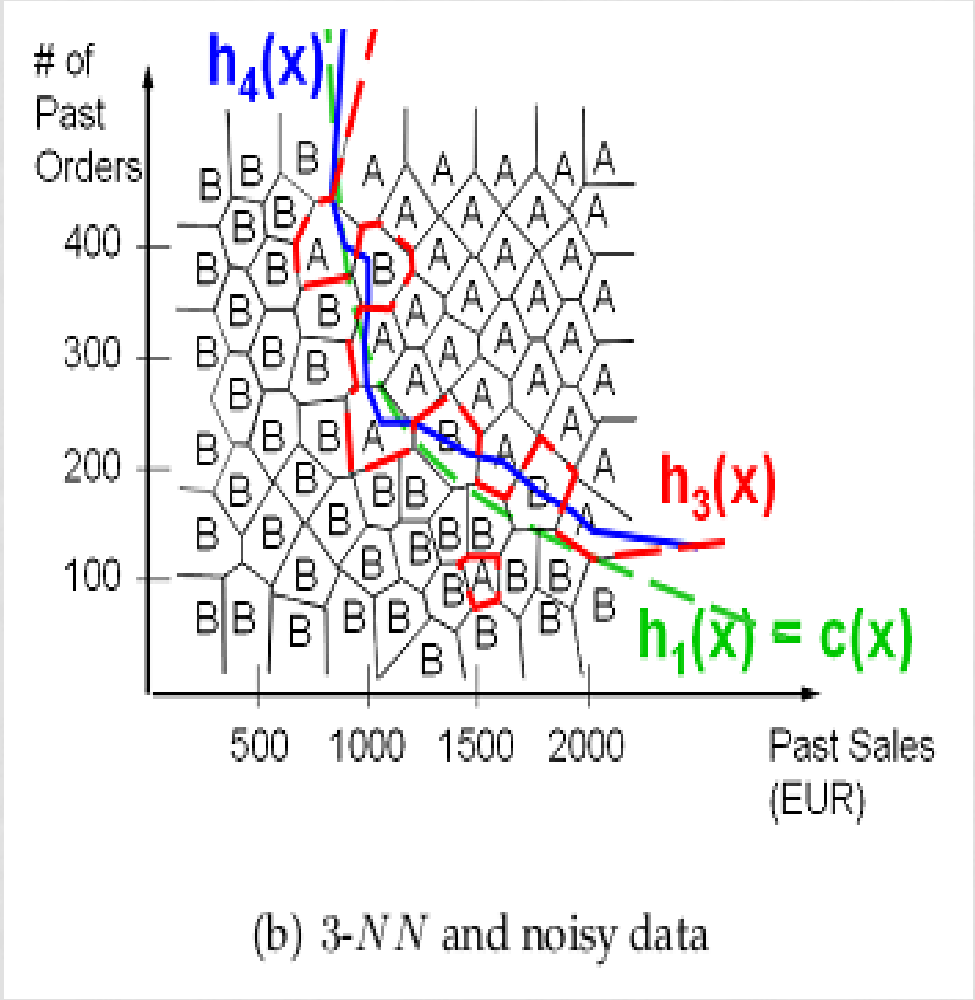
¿PORQUÉ USAR K VECINOS?

- Con k=1, las instancias que son ruido (o solape entre clases) tienen mucha influencia
- Con k>1, se consideran mas vecinos y el ruido pierde influencia (es como hacer una promediado)
- Si k es muy alto, se pierde la idea de localidad
  - ¿En que se convierte KNN si k=número de datos?
- Para evitar que los vecinos lejanos tengan mucha influencia, se puede hacer que cada vecino vote de manera inversamente proporcional a la distancia 1/d
- Si hay dos clases, usar k impar para deshacer empates
- Se puede elegir k mediante validación cruzada

¿PORQUÉ USAR K VECINOS?



(a) 1-NN on noisy data



(b) 3-NN and noisy data

METODOLOGÍA ANÁLISIS DE DATOS

- Recopilación de los datos (tabla datos x atributos)
- Preproceso:
  - **De los datos: Normalización**
  - **De los atributos:**
    - **Selección de atributos:**
      - **Ranking**
      - **Subset selection: CFS y WRAPPER**
    - Transformación / Generación de atributos:
      - No supervisada: **PCA, random projections, autoencoders**
      - Supervisada: **mediante redes de neuronas**
- **GENERACIÓN DE MODELOS / AJUSTE DE PARÁMETROS / SELECCIÓN DE MODELO**
  - **Clasificación: árboles de decisión, reglas, KNN**
  - **Regresión: modelos lineales, árboles de modelos, KNN**
- Evaluación: **validación cruzada, matriz de confusión**
- Despliegue y uso del modelo

SELECCIÓN DE K

- A la elección de un modelo para un problema concreto se le denomina selección del modelo (model selection)
- Se puede seleccionar la familia o tipo del modelo:
  - Separador lineal
  - Árboles de decisión
  - KNN
  - …
- Si se ha seleccionado una familia de modelos, hay que dar valor a ciertos parámetros
  - Árboles de decisión: confidence factor
  - KNN: K
  - …

SELECCIÓN DE K

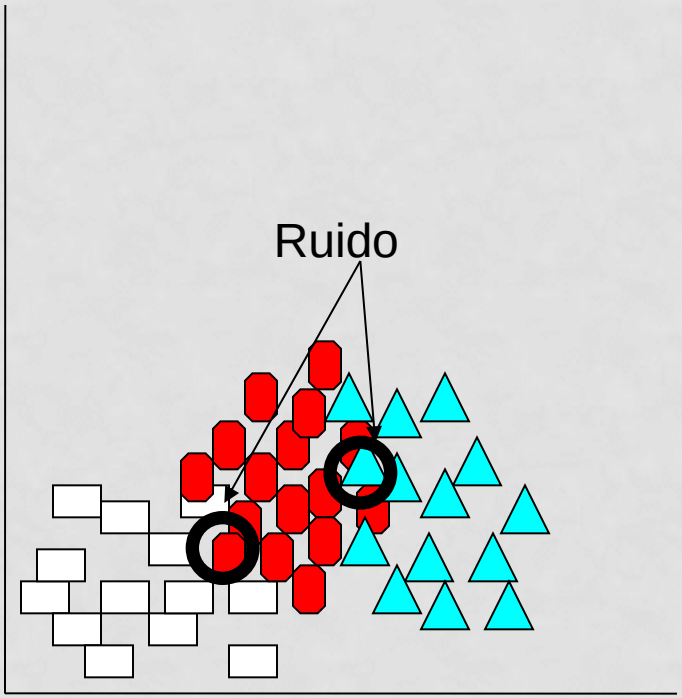
- Se le puede dar valor a k mediante validación cruzada de los datos de entrenamiento (si es demasiado costoso en tiempo, se puede utilizar un solo conjunto de validación)
- No confundir con la validación cruzada que se hace para evaluar el modelo

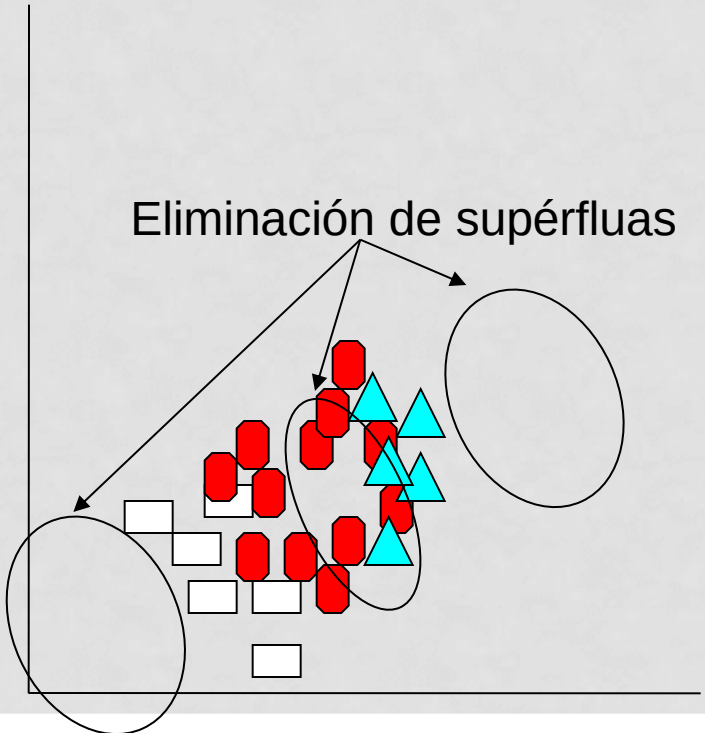
Problemas / limitaciones KNN

- Lento, porque tiene que guardar todos los datos de entrenamiento
  - Solución: eliminación de instancias supérfluas
- Muy sensible al ruido
  - Solución: eliminación de instancias con ruido
- Depende de que la función de distancia sea la adecuada
  - Solución (parcial): aprendizaje de la función distancia
- Muy sensible a los atributos irrelevantes y la maldición de la dimensionalidad:
  - Solución: selección de atributos

SELECCIÓN DE INSTANCIAS

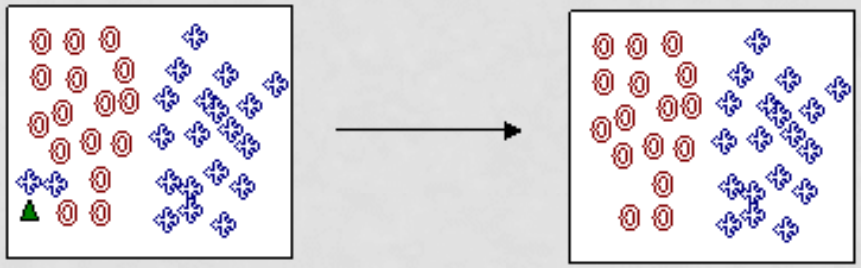
- Hay instancias supérfluas: no son necesarias para clasificar. Si las borramos, se decrementará el tiempo de clasificación
- Hay instancias que son ruido (o solape entre clases): confunden al clasificador. Si las borramos, mejorará el porcentaje de aciertos esperado



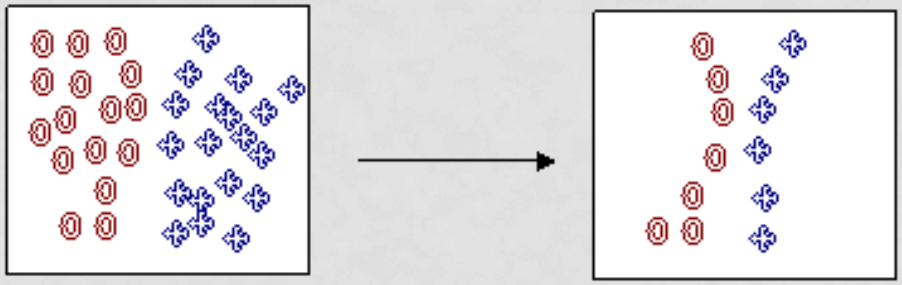


SELECCIÓN DE INSTANCIAS

Engañosas



Supérfluas

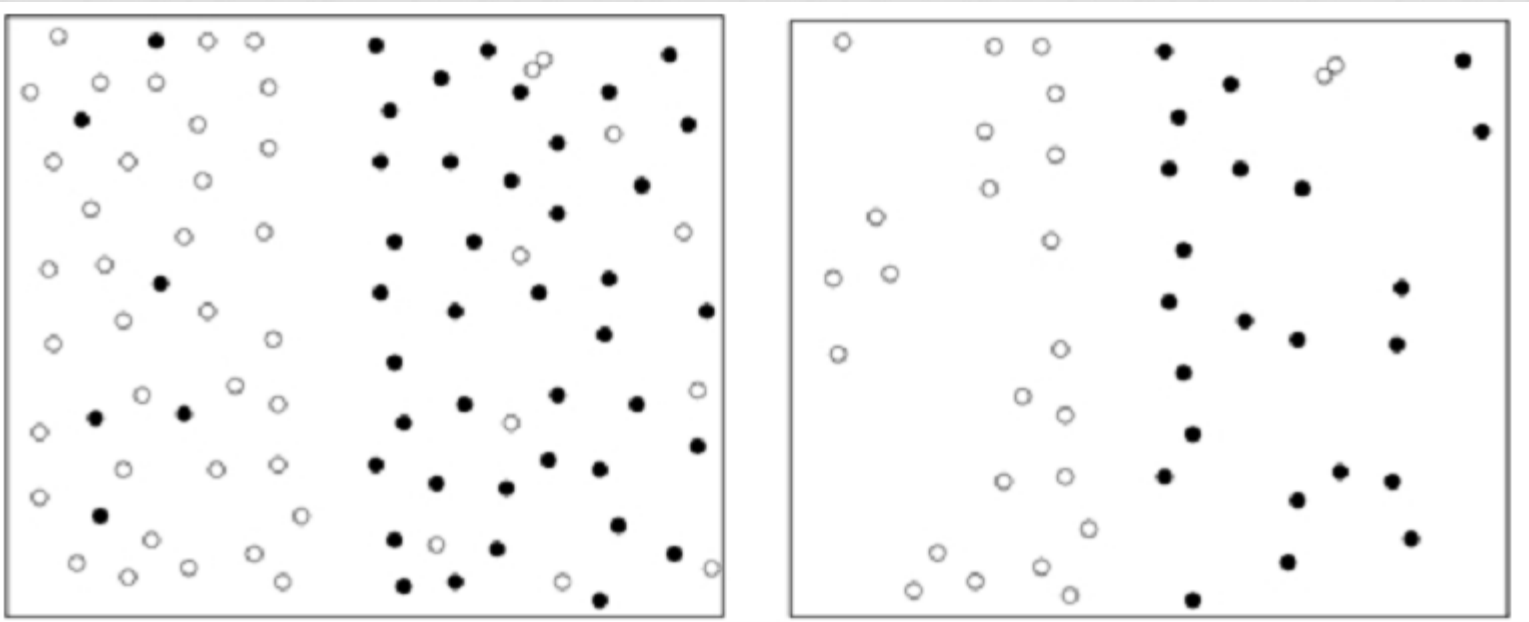


DOS TIPOS DE TÉCNICAS

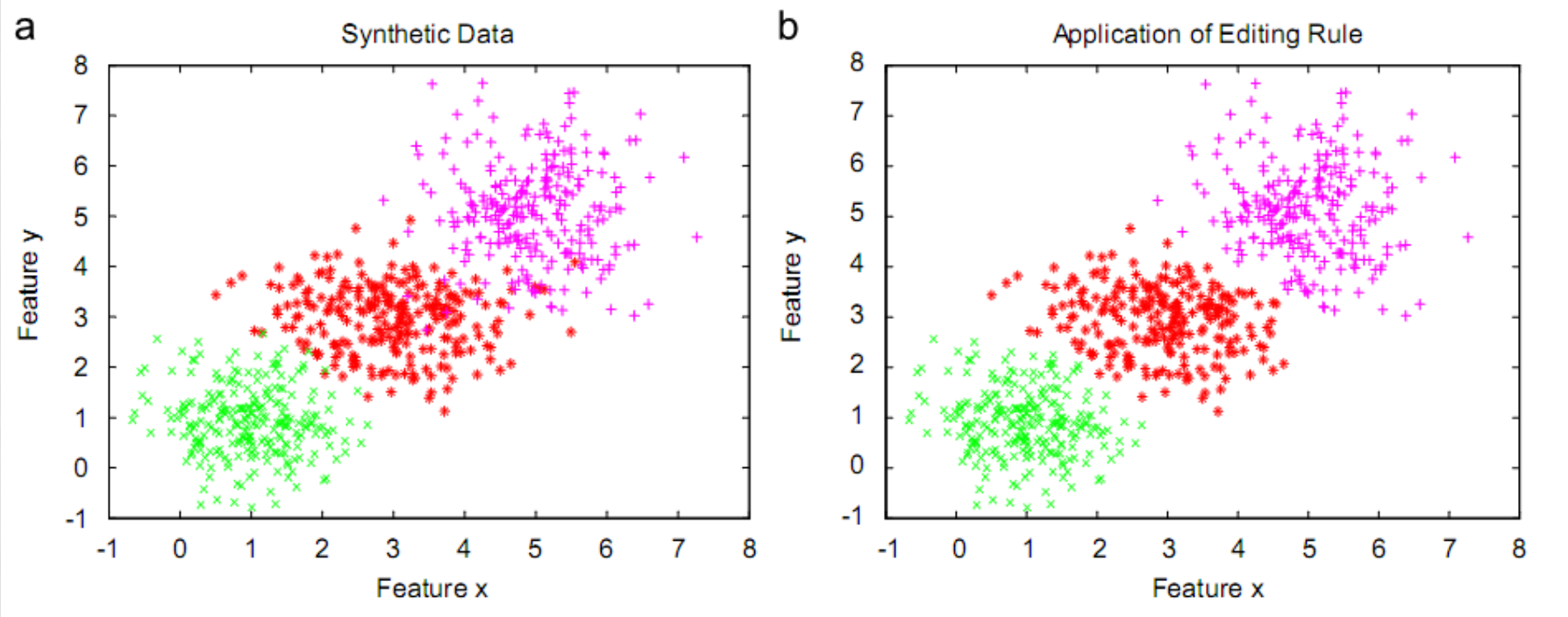
- ***Editing***: eliminar instancias engañosas (ruido)
  - Típicamente se eliminarán sólo unas pocas
- ***Condensación***: eliminar instancias supérfluas:
  - Se pueden llegar a eliminar muchas, las del interior, manteniendo las de la frontera

Wilson editing

- Wilson editing: elimina instancia $\mathbf{x}_i$ si es clasificada incorrectamente por sus k vecinos:
  - Excepciones en el interior de una clase
  - Algunos puntos en la frontera (suaviza las fronteras)
- Repeated Wilson editing: repite Wilson editing hasta que no se pueda aplicar mas

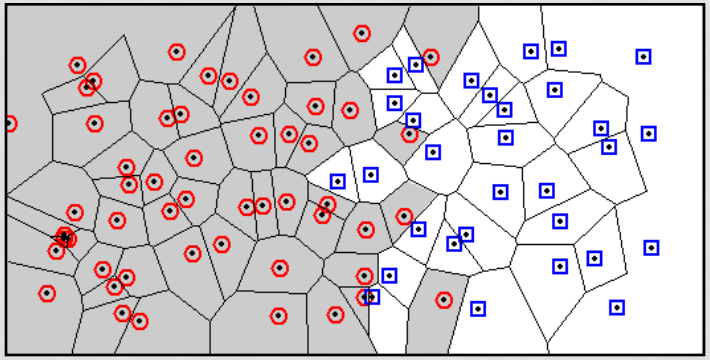


Wilson editing: ejemplo 2

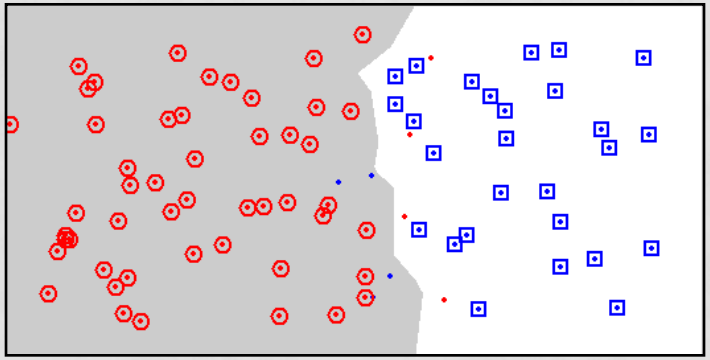


Wilson editing: ejemplo 3

Overlapping classes



Original data



Wilson editing with k=7

Wilson editing

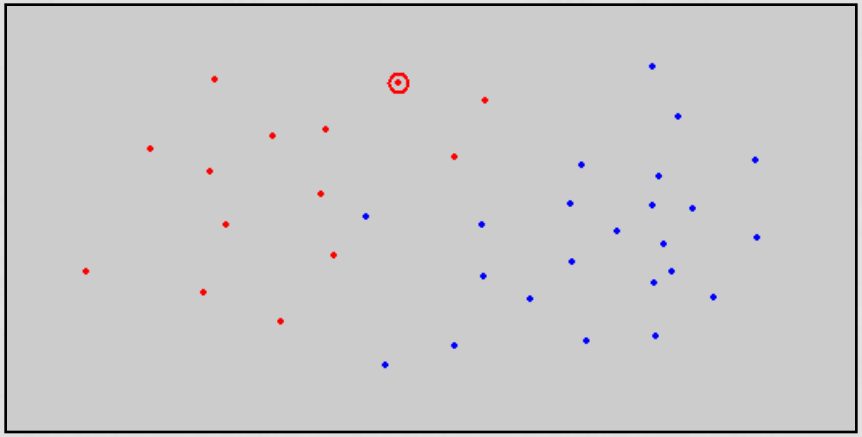
- No quita demasiados datos (es decir, no hay gran mejora en eficiencia)
- Funciona bien si no hay demasiado ruido.
  - Si hay mucho ruido, las instancias con ruido clasificarán bien a otras instancias con ruido

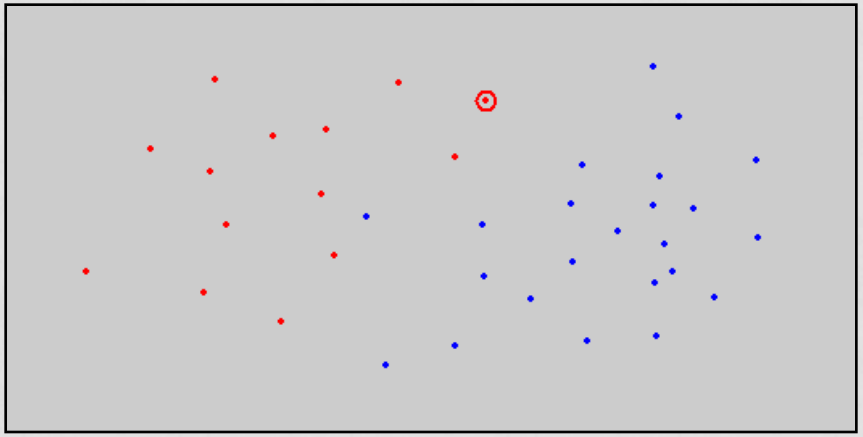
Condensed Nearest Neighbor (CNN)

- Intenta reducir el número de instancias, eliminando las supérfluas
- Va recorriendo las instancias, y si esa instancia ya está bien clasificada con las que ya hay guardadas, no la guarda. Sólo guarda aquellas que no se clasifican bien con las ya existentes (y por tanto, es crítica, es necesaria)

CNN

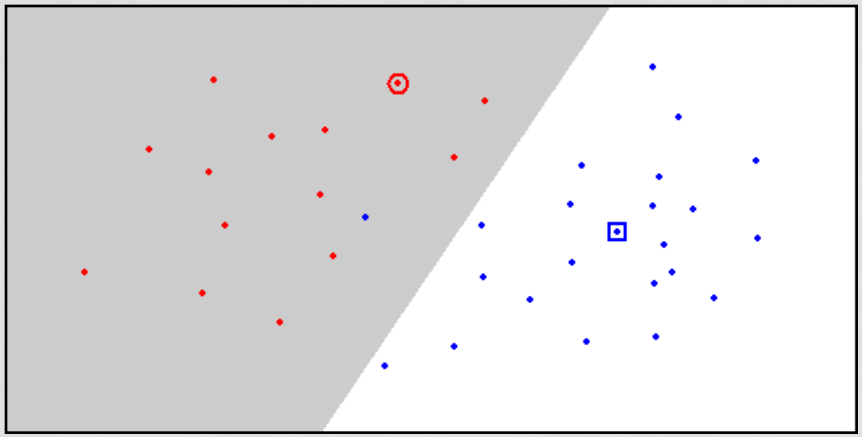
1. Inicializa *store* con $\mathbf{x}_1$
2. Elige un $\mathbf{x}_i$ fuera de *store* mal clasificado según *store*. Muévelo a *store*
3. Repite 2 hasta que no se muevan mas instancias a *store*
4. Para clasificar con KNN, usar *store* en lugar de los datos originales

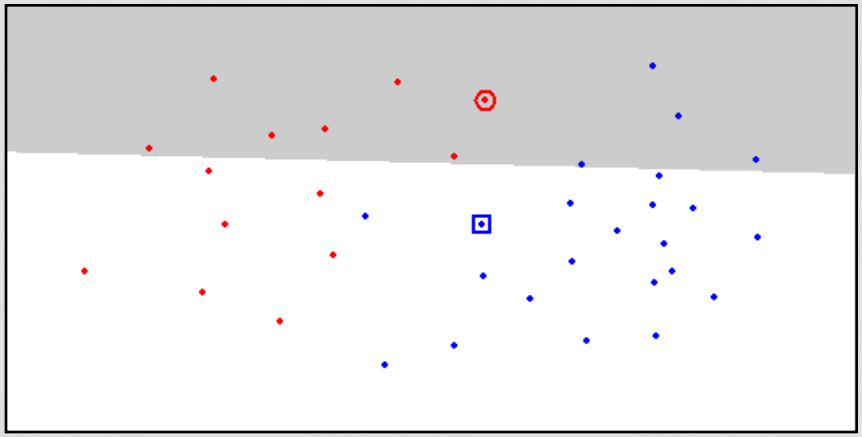




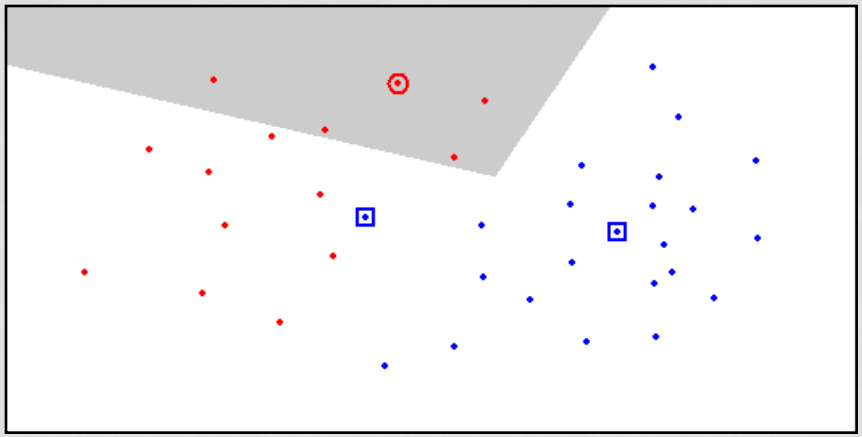
CNN

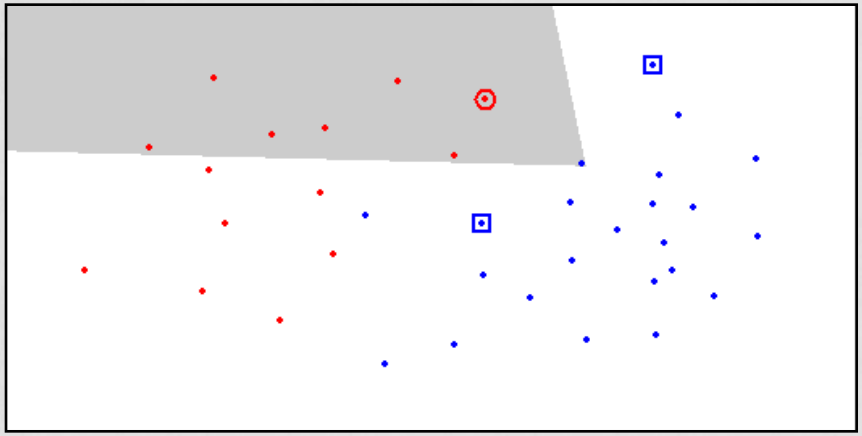
CNN depende mucho del orden en el que se toman las instancias



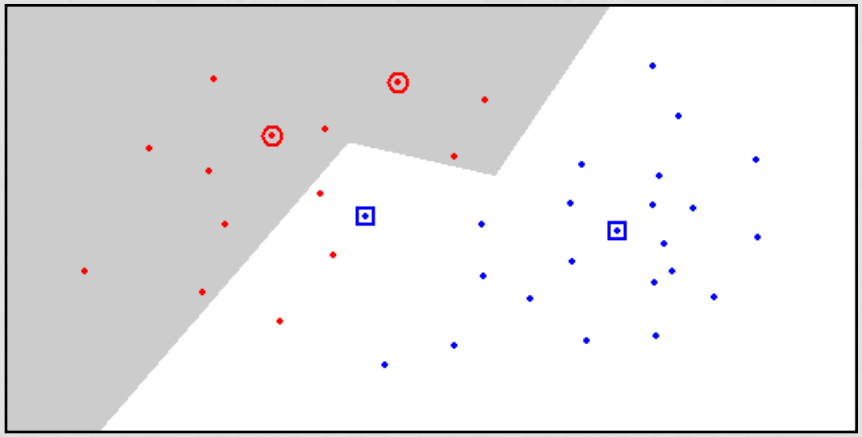


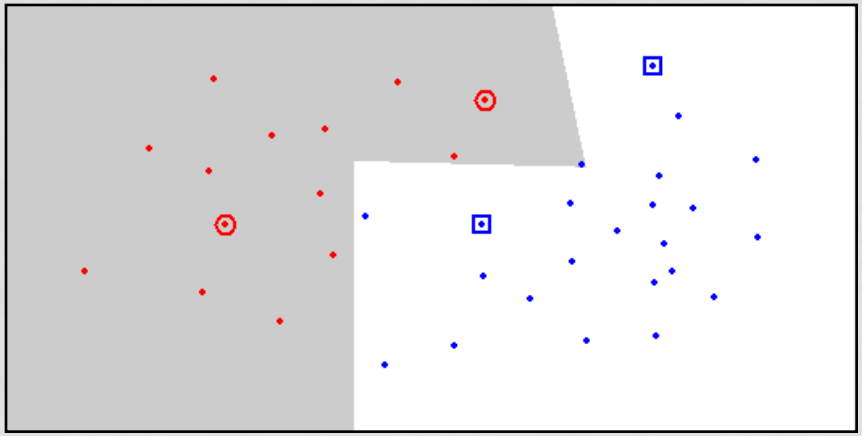
CNN



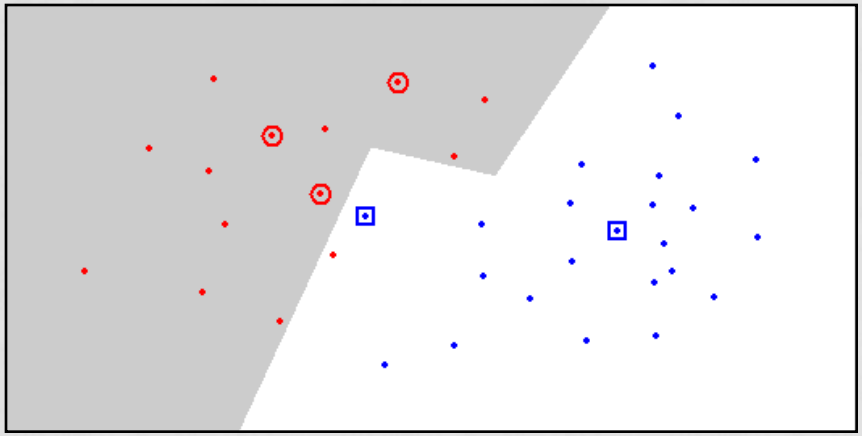


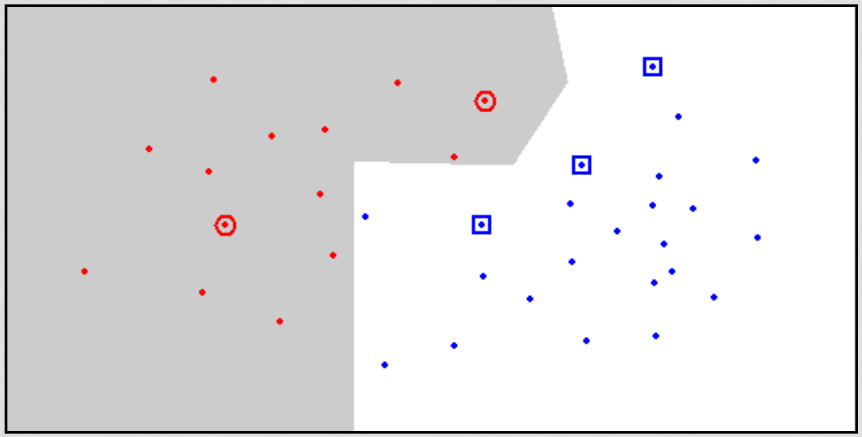
CNN



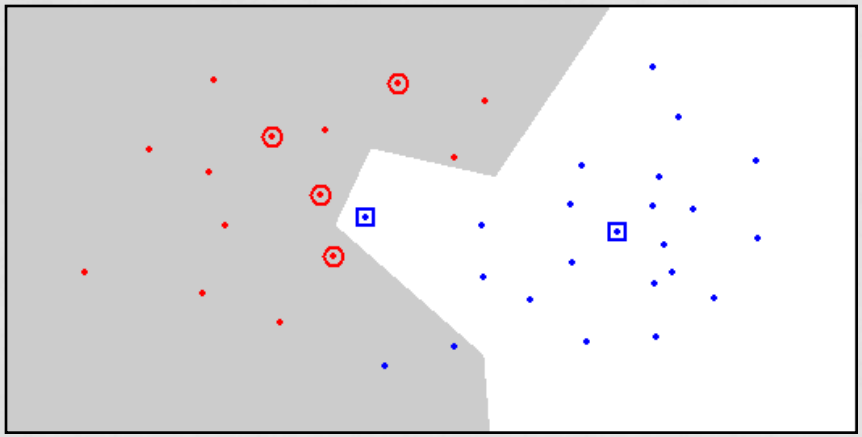


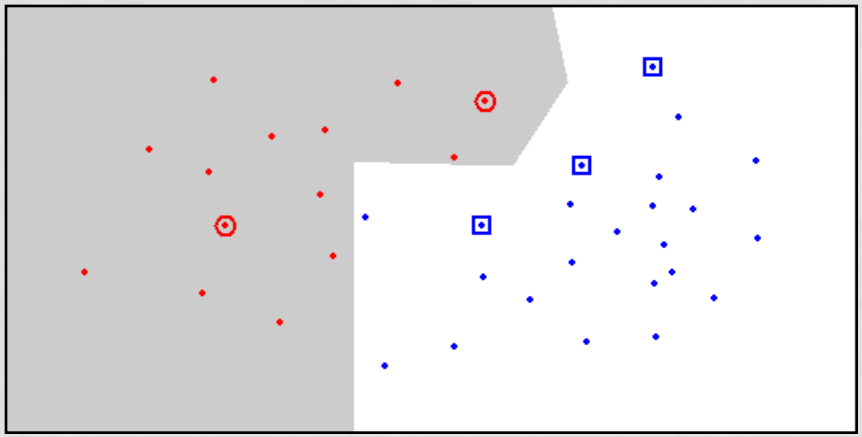
CNN





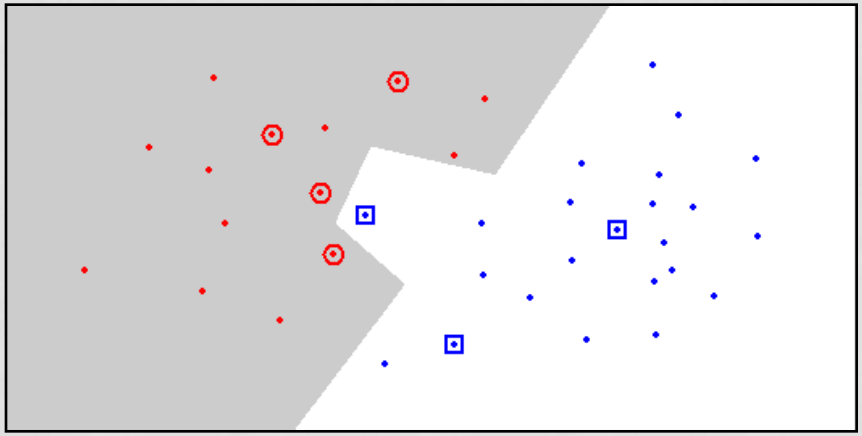
CNN

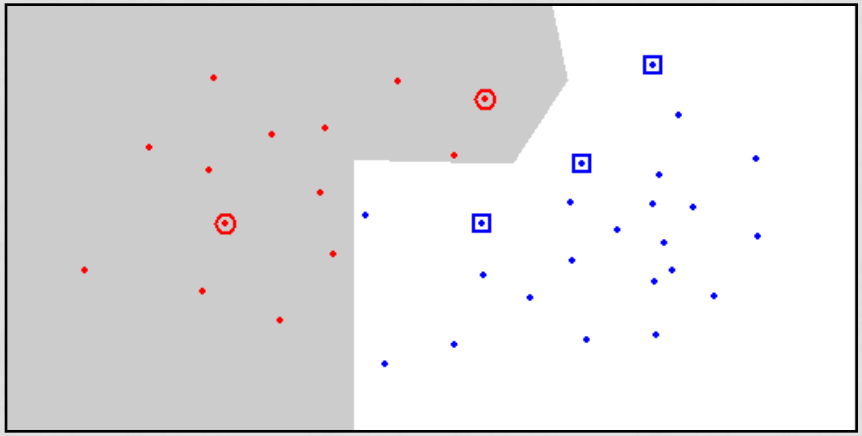




CNN

CNN depende mucho del orden en el que se toman las instancias





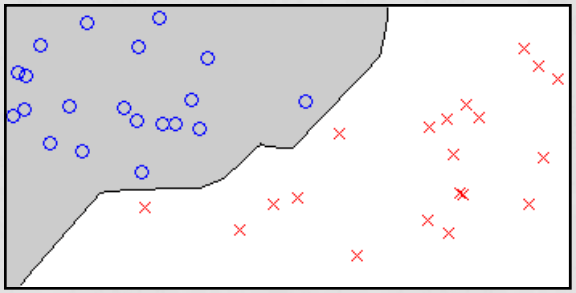
Características de CNN

- +: positivo, -: negativo
- +: elimina todas aquellas instancias no críticas para la clasificación (reduce mucho la necesidad de almacenamiento)
- -: pero tiende a conservar aquellas instancias con ruido (puesto que son mal clasificadas por las instancias en *Store*)
- -: CNN depende mucho del orden en el que se toman las instancias

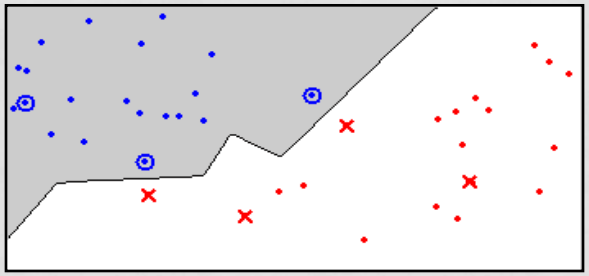
Reduced Nearest Neighbor rule (RNN)

- Es como CNN, pero comienza con todos los datos y va quitando aquellos que, al quitarlos, no hagan que alguna otra instancia pase a estar mal clasificada.
  - Guarda las instancias críticas / necesarias para una clasificación correcta
  - A diferencia de CNN, permite eliminar instancias ruido (puesto que no contribuyen a clasificar correctamente otras instancias, todo lo contrario)

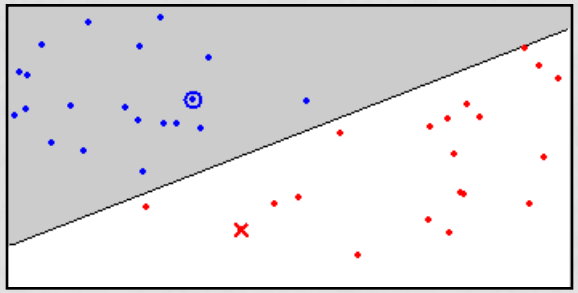
- No hay garantía de que CNN o RNN encuentren el conjunto consistente mínimo



Datos originales



Datos condensados



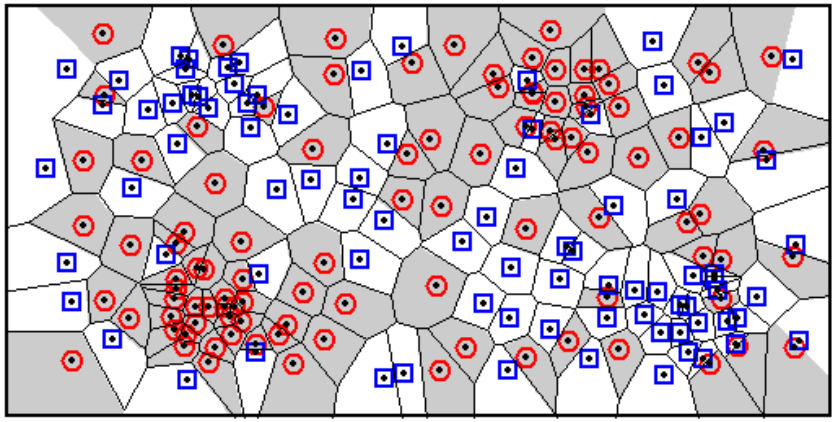
Conjunto consistente mínimo

Edición y condensación

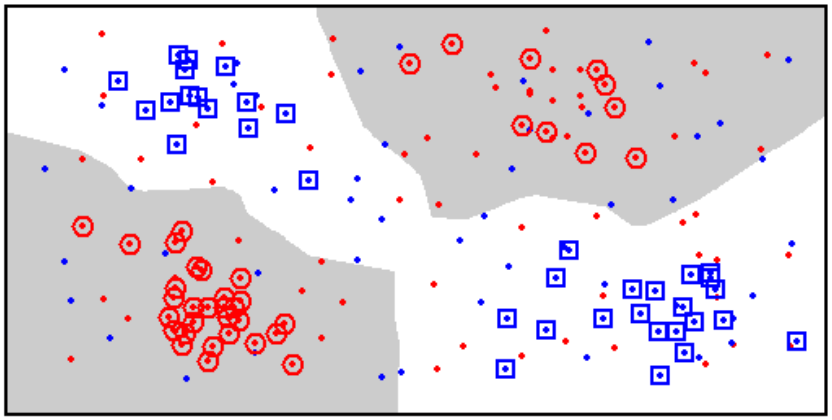
- La edición elimina el ruido y suaviza las fronteras, pero mantiene la mayor parte de los datos (mejora la capacidad de generalización pero no mejora la eficiencia)
- La condensación elimina gran cantidad de datos superfluos, pero mantiene los datos con ruido, puesto que CNN y RSS mantienen aquellos datos clasificados mal por los demás datos (y el ruido tiene siempre esta propiedad)

Híbridos: 1-Editar 2-Condensar

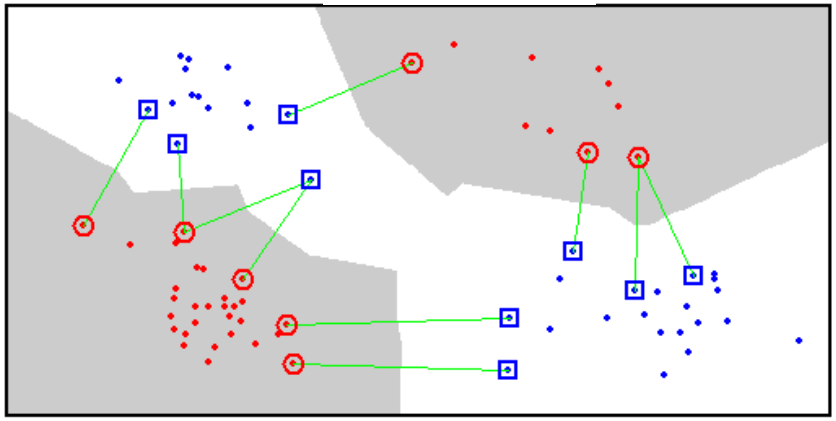
- Primero editar, luego condensar



Original



Edición



Condensación

Algoritmos avanzados

- Editar / Condensar:
  - RT3:
    - D. Randall Wilson, Tony R. Martinez: Instance Pruning Techniques. ICML 1997: 403-411
  - Iterative case filtering (ICF)
    - Henry Brighton, Chris Mellish: Advances in Instance Selection for Instance-Based Learning Algorithms. Data Min. Knowl. Discov. 6(2): 153-172 (2002)

RT1 / RT2 / RT3

- Son algoritmos híbridos (condensación + edición)
- Objetivos de RT1 / RT2 / RT3 para mejorar a CNN:
  - Que no dependa del orden (como le ocurría a CNN)
  - Que elimine instancias con ruido
  - Que elimine instancia supérfluas

http://cgm.cs.mcgill.ca/~athens/cs644/Projects/2004/SumedhaAhuja-EdithLaw/hybrid.html

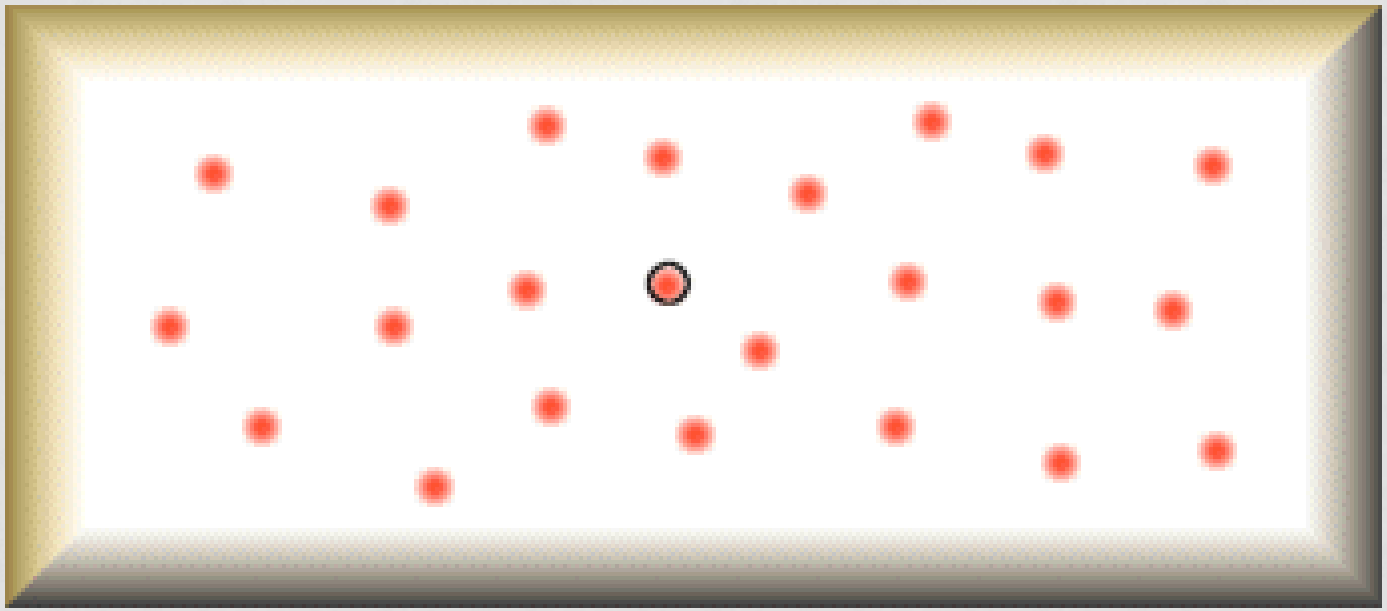
RT1 / RT2 / RT3

- RT1 está inspirado en RNN (que quitaba una instancia si con eso no hacía que otras instancias pasaran a estar mal clasificadas)
- Para cada instancia P, RT1 calcula sus asociados, Un asociado es una instancia que tiene a P como uno de sus k-vecinos.
  - Es decir una instancia asociada a P es una instancia cuya clasificación puede verse afectada si quitamos P
- RT1 quita una instancia P si el número de asociados clasificados correctamente **sin** P es mayor o igual que los clasificados correctamente **con** P

RT1

k=3

La instancia tiene 6 asociados



Algoritmo de RT1

```
1   RT1(Training set T): Instance set S.
2       Let S = T.
3       For each instance P in S:
4           Find P.N_{1..k+1}, the k+1 nearest neighbors of P in S.
5           Add P to each of its neighbors' lists of associates.
6       For each instance P in S:
7           Let with = # of associates of P classified correctly with P as a neighbor.
8           Let without = # of associates of P classified correctly without P.
9           If (without - with) ≥ 0
10              Remove P from S.
11              Remove P from its associates' lists of nearest neighbors, and find
12                  the next nearest neighbor for each of these associates.
13              Remove P from its neighbors' lists of associates.
14          Endif
15      Return S.
```

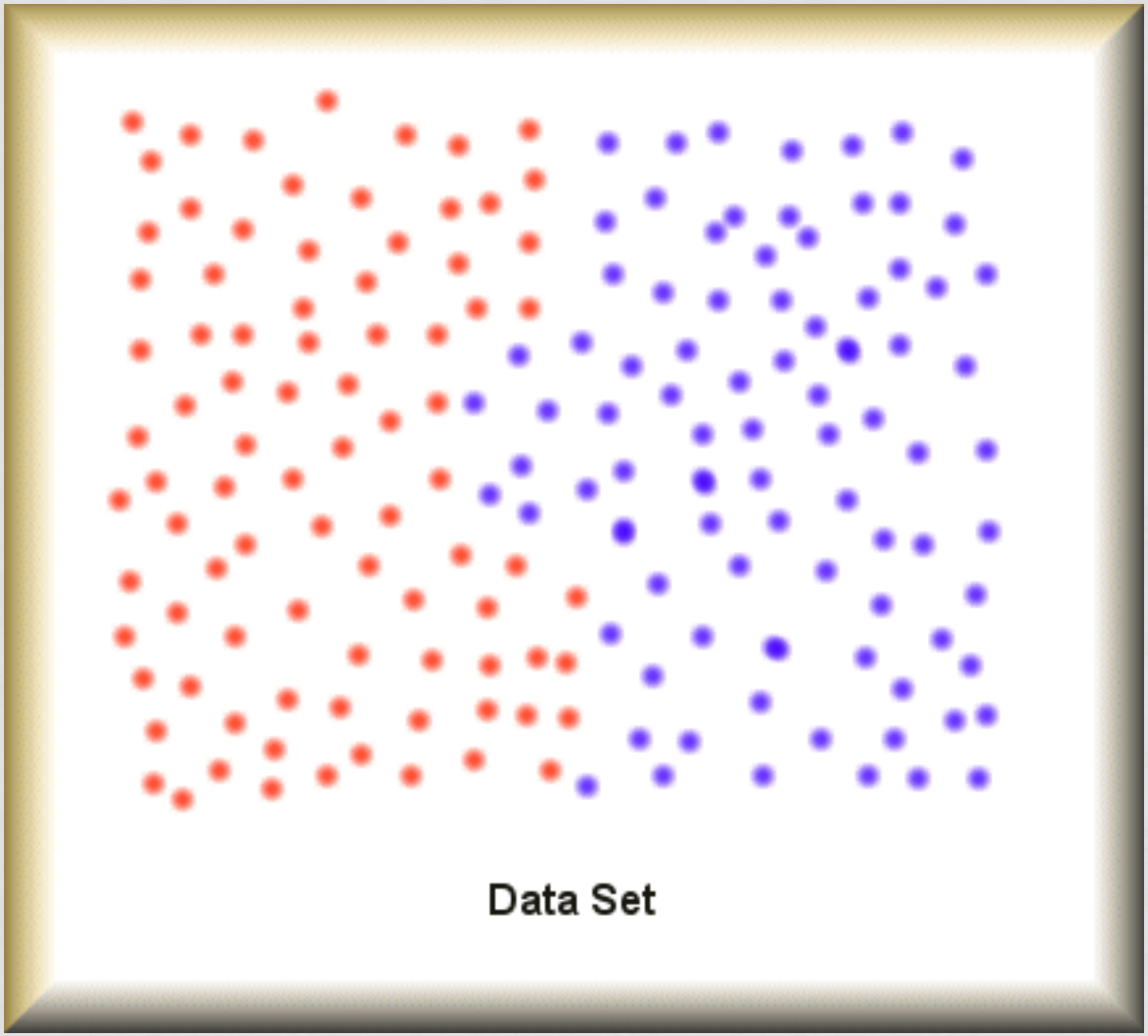
RT1

- RT1 quita instancias con ruido, puesto que si se las quita eso no perjudica a la clasificación de los asociados, todo lo contrario
- RT1 quita instancias supérfluas en el centro de los clusters, porque en esos lugares todas las instancias son de la misma clase, por lo que quitar vecinos no es perjudicial
- RT1 tiende a guardar instancias no-ruido en la frontera, porque si la quitamos, otras instancias pueden pasar a estar mal clasificadas

RT2 / RT3

- RT2 intenta quitar las instancias de los centros de los clusters primero
- Para ello, ordena las instancias por distancia a su vecino mas cercano que pertenezca a una clase distinta
- La idea es considerar antes aquellas instancias lejos de la frontera, pero para ello, hay que eliminar aquellas instancias con ruido en el interior de los clusters.
- Así, RT3 hace una primera pasada para quitar las instancias ruidosas (Wilson editing rule)
- RT3 por tanto tiende a conservar las instancias cerca de la frontera

RT2 / RT3



Algoritmo de RT2

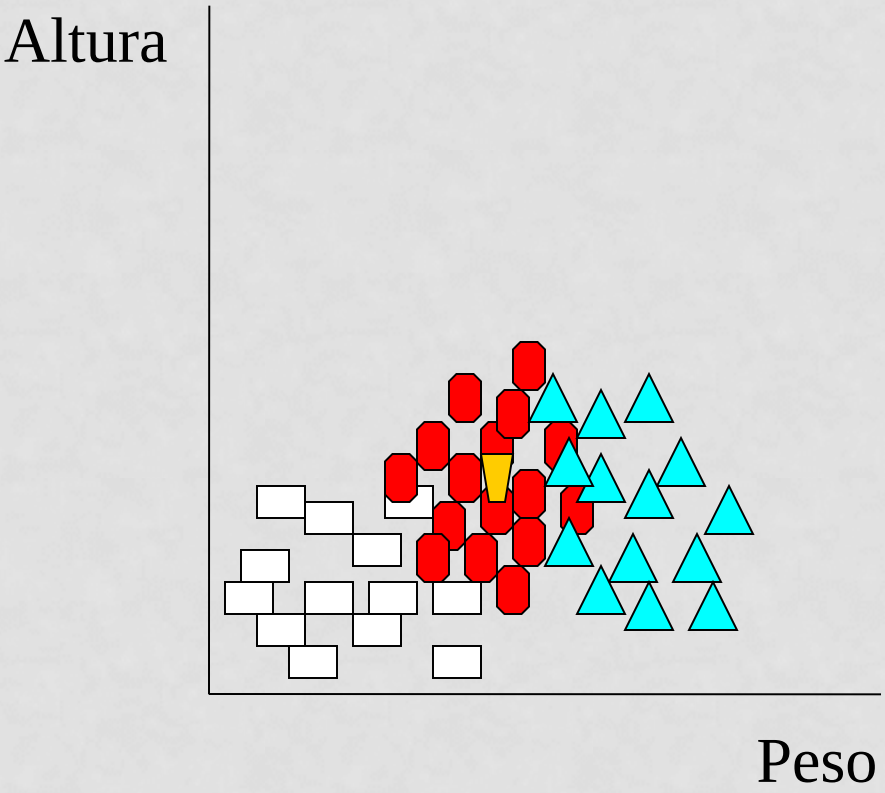
```
1   RT1(Training set T): Instance set S.
2       Let S = T.
3       For each instance P in S:
4           Find P.N_{1..k+1}, the k+1 nearest neighbors of P in S.
5           Add P to each of its neighbors' lists of associates.
6       For each instance P in S:
7           Let with = # of associates of P classified correctly with P as a neighbor.
8           Let without = # of associates of P classified correctly without P.
9           If (without - with) ≥ 0
10              Remove P from S.
11              Remove P from its associates' lists of nearest neighbors, and find
12                  the next nearest neighbor for each of these associates.
13              Remove P from its neighbors' lists of associates.
14          Endif
15      Return S.
```

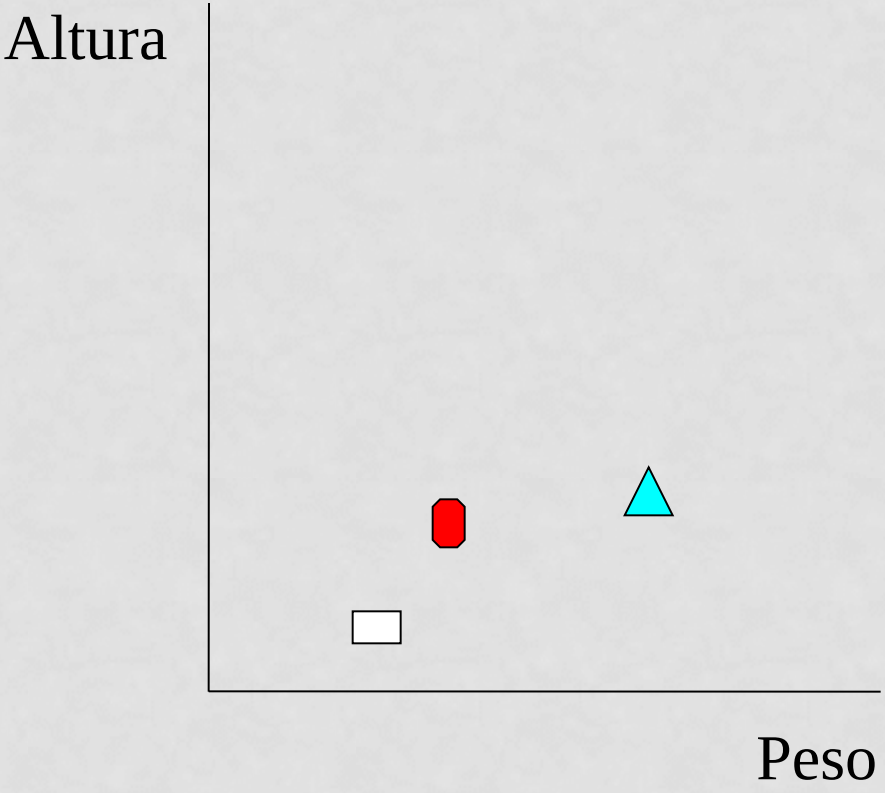
RESULTADOS

Database	*k*NN	(size)	RT1	(size)	RT2	(size)	RT3	(size)	H-IB3	(size)
Anneal	**93.11**	100	**87.85**	9.11	**95.36**	11.42	**93.49**	8.63	**94.98**	7.81
Australian	**84.78**	100	**82.61**	7.67	**84.64**	15.41	**84.35**	5.93	**85.99**	6.48
Breast Cancer WI	**96.28**	100	**94.00**	2.56	**96.14**	5.79	**96.14**	3.58	**95.71**	2.56
Bridges	**66.09**	100	**55.64**	20.86	**59.18**	24.11	**58.27**	18.66	**59.37**	38.67
Crx	**83.62**	100	**81.01**	6.70	**84.93**	14.11	**85.80**	5.46	**83.48**	6.86
Echocardiogram	**94.82**	100	**93.39**	9.01	**85.18**	7.51	**93.39**	9.01	**93.39**	14.85
Flag	**61.34**	100	**58.13**	24.51	**62.34**	32.30	**61.29**	20.45	**51.50**	39.18
Glass	**73.83**	100	**60.30**	26.11	**64.98**	31.52	**65.02**	23.88	**67.77**	32.92
Heart	**81.48**	100	**79.26**	12.96	**81.11**	21.60	**83.33**	13.62	**76.30**	10.33
Heart.Cleveland	**81.19**	100	**77.85**	14.26	**79.87**	20.61	**80.84**	12.76	**74.23**	10.78
Heart.Hungarian	**79.22**	100	**78.92**	11.38	**79.22**	15.98	**79.95**	10.43	**74.83**	8.88
Heart.Long Beach VA	**70.00**	100	**73.00**	11.78	**72.00**	16.33	**73.50**	4.22	**69.50**	11.67
Heart.More	**74.17**	100	**73.20**	11.20	**74.50**	16.98	**76.25**	9.10	**74.75**	13.97
Heart.Swiss	**92.69**	100	**91.15**	2.08	**93.46**	2.89	**93.46**	1.81	**84.62**	4.79
Hepatitis	**80.62**	100	**76.21**	8.67	**82.00**	13.98	**81.87**	7.81	**72.79**	8.03
Horse-Colic	**57.84**	100	**65.09**	10.89	**66.17**	17.98	**71.08**	7.42	**61.82**	17.64
Image.Segmentation	**93.10**	100	**84.76**	10.21	**92.38**	13.76	**92.62**	10.98	**90.24**	14.79
Ionosphere	**84.62**	100	**84.91**	5.67	**88.32**	12.09	**87.75**	7.06	**88.32**	13.61
Iris	**94.00**	100	**89.33**	11.70	**95.33**	16.89	**95.33**	14.81	**92.00**	10.96
				...						
Average	**80.78**	100	**76.93**	14.55	**80.09**	20.16	**80.31**	14.32	**77.41**	16.67

CLASIFICACIÓN BASADA EN PROTOTIPOS

PROTOTIPOS

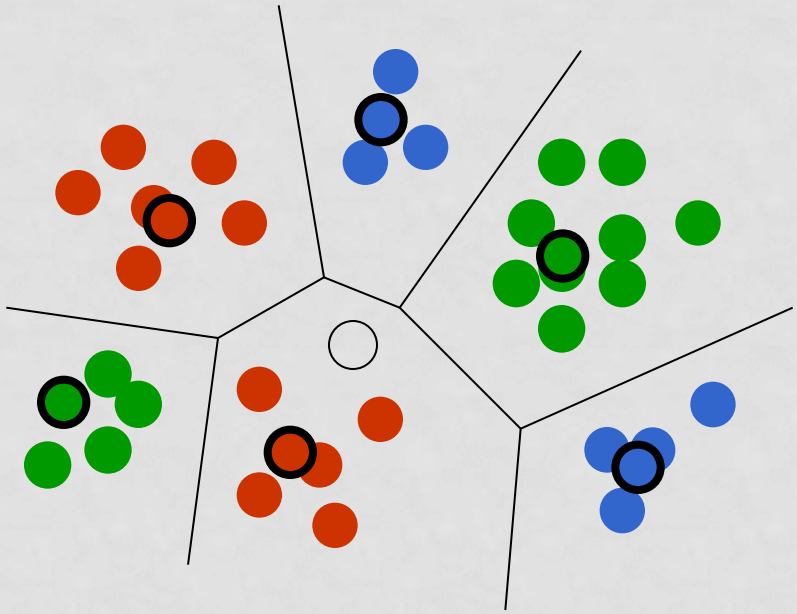




Mejora la eficiencia en espacio (sólo se guardan unos pocos prototipos) y en tiempo (se computan muchas menos distancias cuando llega el dato de test)

Learning vector quantization (LVQ)

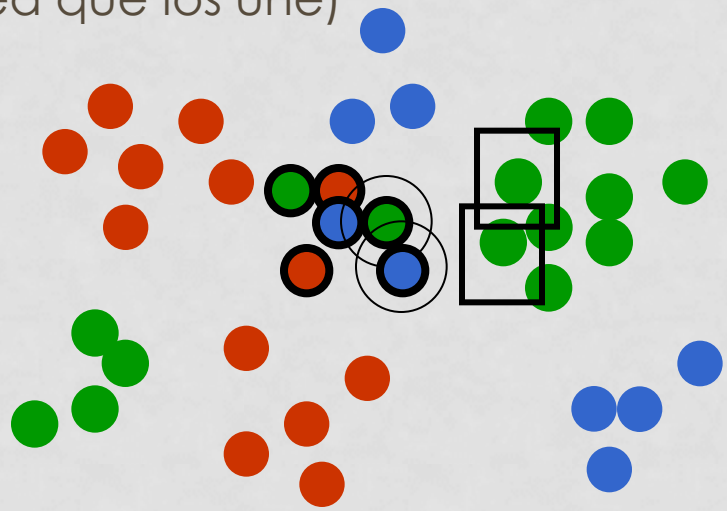
- Cada prototipo tiene una etiqueta
- Se clasifica según la clase del prototipo más cercano (o según sus regiones de Voronoi)



LVQ 1

LVQ I entrenamiento:

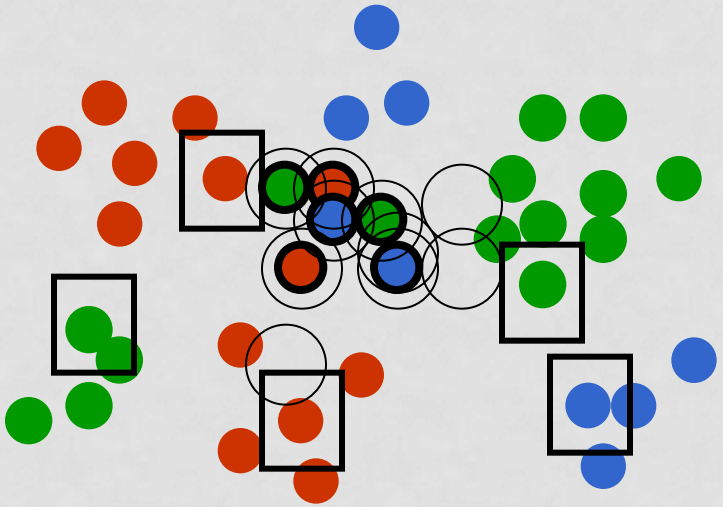
- Inicialización aleatoria
- repetir:
  - presentar instancia de entrenamiento
  - determinar prototipo más cercano
  - Acercarlo o alejarlo de la instancia según la clase (y según la dirección de la línea que los une)



LVQ 2.1

LVQ 2.1 entrenamiento:

- Inicialización aleatoria
- repetir:
  - presentar instancia de entrenamiento
  - Determinar:
    - prototipo más cercano correcto
    - prototipo más cercano incorrecto
  - Acercarlo o alejarlo si está dentro de una ventana



REGLA DE ACTUALIZACIÓN DE LVQ 2.1

$$w_i(t+1) = w_i(t) - \alpha(t)(x - w_i(t)),$$
$$w_j(t+1) = w_j(t) + \alpha(t)(x - w_j(t)),$$

Ventana: que no ocurra que un prototipo está muy cerca y otro muy lejos. Se puede forzar haciendo que el mas cercano esté suficientemente lejos ($d_{cerca} > s * d_{lejos}$) ; $s<1$

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s,$$