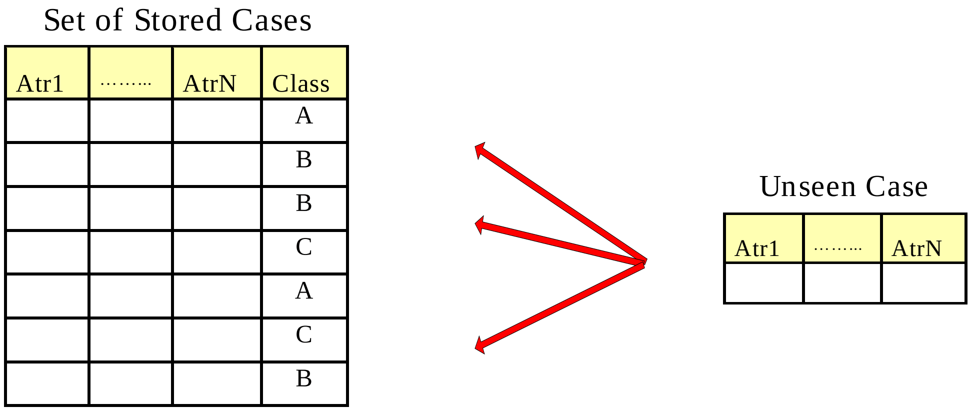
ALGORITMO DE LOS *K* VECINOS MÁS CERCANOS

Modelos basados en instancias

- Están basados en el aprendizaje por analogía
- Se encuadran en el paradigma perezoso de aprendizaje (lazy learning): El trabajo se retrasa todo lo posible
  - No se construye ningún modelo, el modelo es la propia BD o conjunto de entrenamiento
  - Se trabaja cuando llega un nuevo caso a predecir: Se buscan los casos más parecidos y la predicción se construye en función de la salida de que dichos casos



- Los algoritmos más conocidos están basados en la regla del vecino más próximo

Problemas de clasificación

Algoritmos: Vecino más cercano (1-NN)

- Entrenamiento: almacenar el conjunto de entrenamiento
  - CE = $\{e_1, \ldots, e_P\}$ → P es el número de ejemplos
- Clasificación de un nuevo ejemplo e'

1. c_{min} = clase(e_1)
2. d_{min} = d(e_1, e')
3. Para $i=2$ hasta P hacer
   1. $d=d(e_i,e')$
   2. Si ($d<d_{min}$) Entonces

      c_{min} = clase (e_i)

      $d_{min} = d$
4. Devolver c_{min} como clasificación de e'

Vecino más cercano: distancias

- $d(e_i, e')$ es una función que mide la distancia entre 2 ejemplos (el ejemplo i-ésimo, e_i, y e')
- En el caso de variables numéricas
  - Distancia Euclidea $d_e(e_i, e') = \sqrt{\sum_{j=1}^{N}(e_i^j - e'^j)^2}$
  - Distancia de Manhattan $d_m(e_i, e') = \sum_{j=1}^{N}\left|e_i^j - e'^j\right|$
  - Distancia de Minkowski $d^r(e_i, e') = \left(\sum_{j=1}^{N}\left|e_i^j - e'^j\right|^r\right)^{1/r}$
    - Como se puede observar $d^1 = d_m$ y $d^2 = d_e$
- Ejemplo (distancia Euclidea)
  - $e_1 = \{2.3, 3.7, A\}$ y $e_2 = \{2. 4.4, B\}$

$$d_e(e_1, e_2) = \sqrt{(2.3 - 2)^2 + (3.7 - 4.4)^2} = 0.7616$$

Vecino más cercano: distancias

- Problemas
  - Algunos atributos pueden dominar la decisión
    - Normalización de los valores de cada atributo j en [0, 1]

$$e_k^j = \frac{e_k^j - min^j}{max^j - min^j}, con\ k = \{1, \dots, P\}$$

  - Tratamiento de valores perdidos (e_1^j =? y/o e_2^j =?)
    - Entonces la distancia asociada al j-ésimo atributo es la máxima, es decir, 1
  - Todos los atributos tienen la misma importancia
    - Asignar pesos a los atributos de forma que se pondere su importancia dentro del contexto

$$d_e(e_i, e') = \sqrt{\sum_{j=1}^{N} w^j * (e_i^j - e'^j)^2}$$

    - La suma de los pesos, w^j, debe ser 1 (100%)

Vecino más cercano: distancias

- En el caso de variables discretas
  - Asignar valores a las categorías. Aplicar las medidas de distancia numéricas
  - Distancia de Hamming (overlap distance)

$$d_H(e_i, e') = \sum_{j=1}^{N} D\left(e_i^j, e'^j\right) \; con \; D\left(e_i^j, e'^j\right) = \begin{cases} 0 & si \quad e_i^j = e'^j \\ 1 & si \quad e_i^j \neq e'^j \end{cases}$$

  - Value Difference Metric
    - Considera la distribución de los valores x e y: implica que tiene en cuenta si los valores ayudan a resolver el problema o no

$$vdm_j(x, y) = \sum_{k=1}^{M} \left| \frac{N_{j,x,k}}{N_{j,x}} - \frac{N_{j,y,k}}{N_{j,y}} \right|^q$$

      - $N_{j,x,k}$: número de ejemplos de la clase k con valor x en el atributo j
      - $N_{j,x}$: número de ejemplos con valor x en el atributo j
      - q: constante (normalmente 1 o 2)
      - M: número de clases

Vecino más cercano: distancias

- Ejemplo de Value Difference Metric

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Class	Marital Status		
	Single	Married	Divorced
Yes	2	0	1
No	2	4	1

d(Single,Married) = | 2/4 – 0/4 | + | 2/4 – 4/4 | = 1

d(Single,Divorced) = | 2/4 – 1/2 | + | 2/4 – 1/2 | = 0

d(Married,Divorced) = | 0/4 – 1/2 | + | 4/4 – 1/2 | = 1

Class	Refund	
	Yes	No
Yes	0	3
No	3	4

d(Refund=Yes,Refund=No) = | 0/3 – 3/7 | + | 3/3 – 4/7 | = 6/7

Vecino más cercano: distancias

- Heterogeneous Euclidean-Overlap Metric (HEOM)

$$HEOM(e_i, e') = \sqrt{\sum_{j=1}^{N} d_j(e_i^j, e'^j)^2}$$

$$d_j(e_i^j, e'^j) = \begin{cases} 1 & si \quad e_i^j \ o \ e'^j \ son\ valores\ perdidos \\ d_H(e_i^j, e'^j) & si \quad atributo\ j\ es\ categórico \\ \dfrac{\left|e_i^j - e'^j\right|}{max^j - min^j} & si \quad atributo\ j\ es\ numérico \end{cases}$$

Vecino más cercano: distancias

- Heterogeneous Value Difference Metric (HVDM)

$$HVDM(e_i, e') = \sqrt{\sum_{j=1}^{N} d_j(e_i^j, e'^j)^2}$$

$$d_j(e_i^j, e'^j) = \begin{cases} 1 & si \quad e_i^j \ o \ e'^j \ son \ valores \ perdidos \\ \sqrt{\sum_{k=1}^{M} \left| \frac{N_{j,x,k}}{N_{j,x}} - \frac{N_{j,y,k}}{N_{j,y}} \right|^2} & si \quad atributo \ j \ es \ categórico \\ \frac{\left| e_i^j - e'^j \right|}{4 * desvStd^j} & si \quad atributo \ j \ es \ numérico \end{cases}$$

Algoritmos: Vecino más cercano (1-NN)

- Ejemplo algoritmo 1-NN
- Dado el siguiente conjunto de entrenamiento (CE) con 4 ejemplos, 3 atributos y 2 clases {+, -}
  - $e_1 = \{0.4, 0.8, 0.2, +\}$
  - $e_2 = \{0.2, 0.7, 0.9, +\}$
  - $e_3 = \{0.9, 0.8, 0.9, -\}$
  - $e_4 = \{0.8, 0.1, 0.0, -\}$
- Sea $\mathrm{e}' = \{0.7, 0.2, 0.1\}$ el ejemplo a clasificar
- Calculamos la distancia Euclidea del ejemplo e' con todos los ejemplos del CE

$$d_e(e_1, e') = \sqrt{(0.4-0.7)^2 + (0.8-0.2)^2 + (0.2-0.1)^2} = 0.678$$
$$d_e(e_2, e') = \sqrt{(0.2-0.7)^2 + (0.7-0.2)^2 + (0.9-0.1)^2} = 1.068$$
$$d_e(e_3, e') = \sqrt{(0.9-0.7)^2 + (0.8-0.2)^2 + (0.9-0.1)^2} = 1.020$$
$$d_e(e_4, e') = \sqrt{(0.8-0.7)^2 + (0.1-0.2)^2 + (0.0-0.1)^2} = 0.173$$

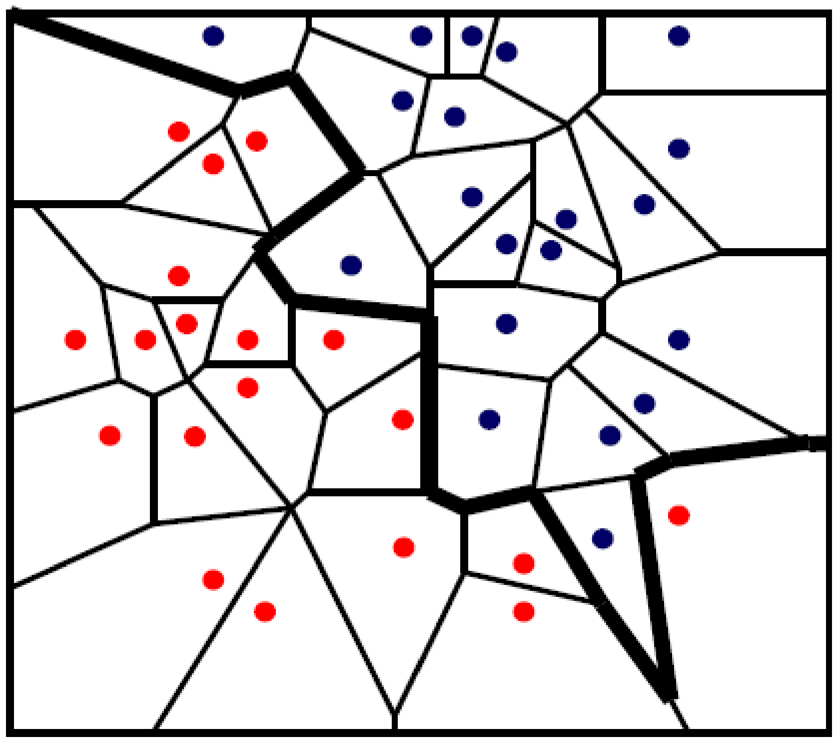
- Por tanto el ejemplo se clasificará en la clase – por ser la clase de e_4

Algoritmos: Vecino más cercano (1-NN)

- Problema
  - Si queremos obtener el rendimiento de 1NN con el propio conjunto de entrenamiento
    - Al clasificar el ejemplo i-ésimo, e_i, la distancia con él mismo es 0 por lo que siempre será el vecino más cercano (100% de acierto)
      - Leave-one-out

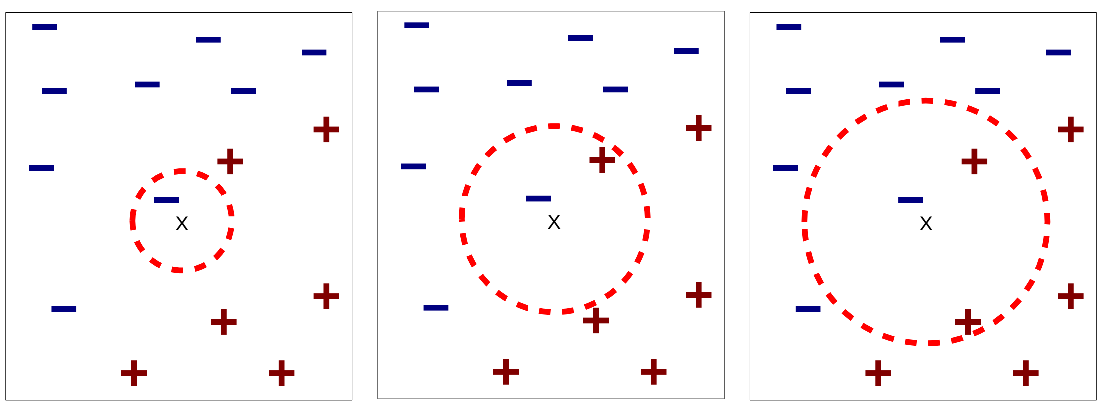
Algoritmos: Vecino más cercano (1-NN)

- Permite abordar problemas tanto binarios como multi-clase
- Frontera de decisión no lineal: diagrama de Voronoi
  - Ejemplo: problema de dos clases solucionado con 1-NN



Algoritmos: K vecinos más cercanos (K-NN)

- La extensión de la regla del vecino más cercano es considerar los k vecinos más cercanos del conjunto de entrenamiento, CE, al ejemplo a clasificar
- Funcionamiento: Dado el ejemplo a clasificar e′
  1. Seleccionar los k ejemplos con $EK = \{e_1, \dots, e_k\}$ tal que no existe ningún ejemplo e* fuera de EK con $d(e^*, e') < d(e_i, e'), e_i \in EK\ y\ e^* \in CE$
  2. Devolver la clase que más se repite en el conjunto EK (voto por mayoría) $\{clase(e_1), \dots, clase(e_k)\}$
- Ejemplo

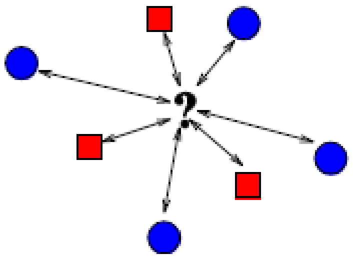


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Algoritmos: K vecinos más cercanos (K-NN)

- Problemas
  - Varias clases obtienen el mismo número de votos (empates)
    - Desempate entre clases
      - Aleatorio
      - Clase con la mínima suma de distancias
      - Clase más frecuente en el conjunto de entrenamiento

  - Se trata de forma igual a todos los vecinos



      - Voto con pesos en función de la distancia

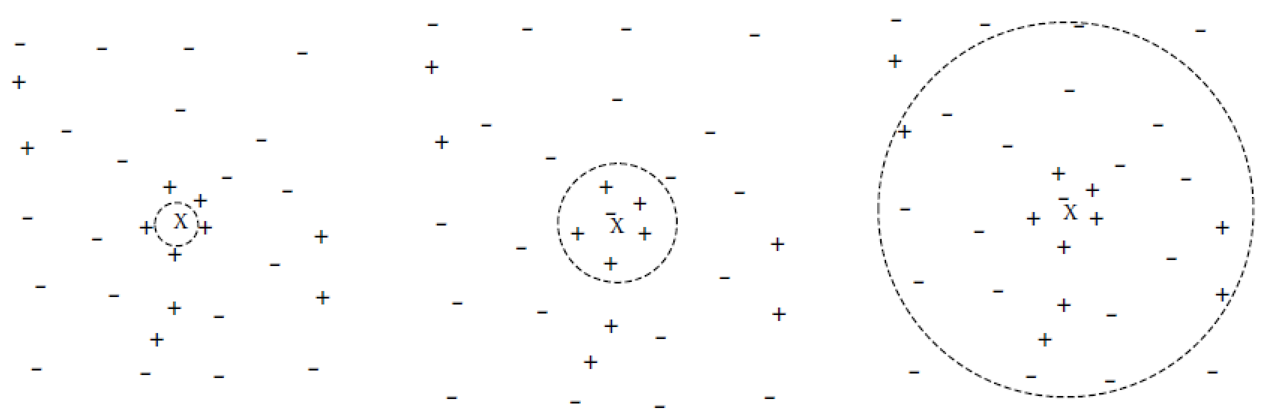
$$clase_z = \underset{v \in C}{argmax} \sum_{y \in EK} \left(\frac{1}{d(e_y, e')^2} * (v = clase(e_y)) \right)$$

Algoritmos: K vecinos más cercanos (KNN)

- Entrenamiento: almacenar el conjunto de entrenamiento
  - CE = $\{e_1, \ldots, e_P\}$ → P es el número de ejemplos
- Clasificación de un nuevo ejemplo e'
  1. *Para i=1 hasta P hacer*
     1. *d(i)* = *d*(e_i, e')
  2. *EK* = obtenerEjemplosCercanos(d, K)
  3. Para *i=1* hasta *K* hacer
     1. *valorVoto = calcularVoto(EK(i), d, tipoVoto)*
     2. *Votos(clase(EK(i)) = Votos(clase(EK(i)) + valorVoto*
  4. *Si empates(Votos) then*
     1. *Clase = desempate(clasesEmpatadas, tipoDesempate)*
  5. *Else*
     1. *Clase = arg max(Votos)*
  6. Devolver *Clase* como clasificación de e'

Algoritmos: K vecinos más cercanos (K-NN)

- Influencia del valor de *k*
  - Si es muy pequeño: sensible al ruido
  - Si es muy grande: vecindario puede incluir ejemplos de otras clases
  - k-NN es robusto frente al ruido cuando se utilizan valores de k moderados
    - Obtenerlos mediante validación cruzada



(a) Neighborhood too small.

(b) Neighborhood just right.

(c) Neighborhood too large.

Problemas de regresión

Regresión: Vecino más cercano (1-NN)

- **Entrenamiento**: almacenar el conjunto de entrenamiento
  - CE = $\{e_1, \ldots, e_P\}$ → P es el número de ejemplos
- **Predicción** de un nuevo ejemplo e'

1. v_{min} = valor(e_1)
2. d_{min} = d(e_1, e')
3. Para i=2 hasta P hacer
   1. d=d(e_i,e')
   2. Si (d<d_{min}) Entonces

      v_{min} = valor(e_i)

      d_{min} = d
4. Devolver v_{min} como predicción de e'

Regresión: K vecinos más cercanos (KNN)

- **Entrenamiento**: almacenar el conjunto de entrenamiento
  - $CE = \{e_1, \ldots, e_P\} \rightarrow$ P es el número de ejemplos
- **Predicción** de un nuevo ejemplo e'
  1. *Para i=1 hasta P hacer*
     1. $d(i) = d(e_i, e')$
  2. *EK* = obtenerEjemplosCercanos(d, K)
  3. Para *i=1* hasta *K* hacer
     1. *valor += valor(EK(i))*
  4. Pred = valor/K
  5. Devolver Pred como predicción de e'

Comentarios finales

- Es bastante preciso, puesto que utiliza varias funciones lineales locales para aproximar la función objetivo
  - Clasificación: frontera de decisión no lineal y multi-clase nativa
- Es muy ineficiente en memoria ya que hay que almacenar toda la BD
- Su complejidad temporal (para clasificar un ejemplo) es $O(dn^2)$ siendo $O(d)$ la complejidad de la distancia utilizada
  - Una forma de reducir esta complejidad es mediante el uso de prototipos o selección de ejemplos
- La distancia entre vecinos podría estar dominada por variables irrelevantes
  - Selección previa de características