

Metaheurísticas

Seminario 2. Problema para las prácticas: Aprendizaje de Pesos en Características

Problema del Aprendizaje de Pesos en Características (APC)

- Definición del Problema de Clasificación. Clasificador k-NN.
- Definición y Representación del Problema del Aprendizaje de Pesos en Características.
- Solución Greedy.
- Búsquedas por Trayectorias Simples.
- Bases de Datos a Utilizar.

Definición del Problema de Clasificación

Concepto de aprendizaje supervisado en clasificación

- Se conocen las clases existentes en el problema
- Se conoce la clase concreta a la que pertenece cada objeto de un conjunto de datos

Existen una gran cantidad de técnicas para el aprendizaje supervisado de Sistemas de Clasificación:

- Técnicas estadísticas: k vecinos más cercanos, discriminadores bayesianos, etc...
- Árboles de clasificación, Sistemas basados en reglas, Redes Neuronales, Máquinas de Soporte Vectorial, ...

Definición del Problema de Clasificación

Disponemos de una muestra de objetos ya clasificados w_1, \dots, w_n , representados en función de sus valores en una serie de atributos:

w_i tiene asociado el vector $(x_1(w_i), \dots, x_n(w_i))$

Cada objeto pertenece a una de las clases existentes $\{C_1, \dots, C_M\}$

OBJETIVO: Obtener un sistema que permita clasificar dichos objetos de modo automático

$$\begin{aligned} w_1 &= (x_1(w_1), \dots, x_n(w_1)) \rightarrow C_{i_1} \\ &\dots \\ w_k &= (x_1(w_k), \dots, x_n(w_k)) \rightarrow C_{i_k} \end{aligned} \quad i_j \in \{1, \dots, M\}, i \in \{1, \dots, k\}$$

Definición del Problema de Clasificación

Ejemplo: Diseño de un Clasificador para la flor del Iris

- *Problema simple muy conocido: clasificación de lirios*
- *Tres clases de lirios: setosa, versicolor y virgínica*
- *Cuatro atributos: longitud y anchura de pétalo y sépalo, respectivamente*
- *150 ejemplos, 50 de cada clase*
- *Disponible en <http://www.ics.uci.edu/~mlearn/MLRepository.html>*



setosa



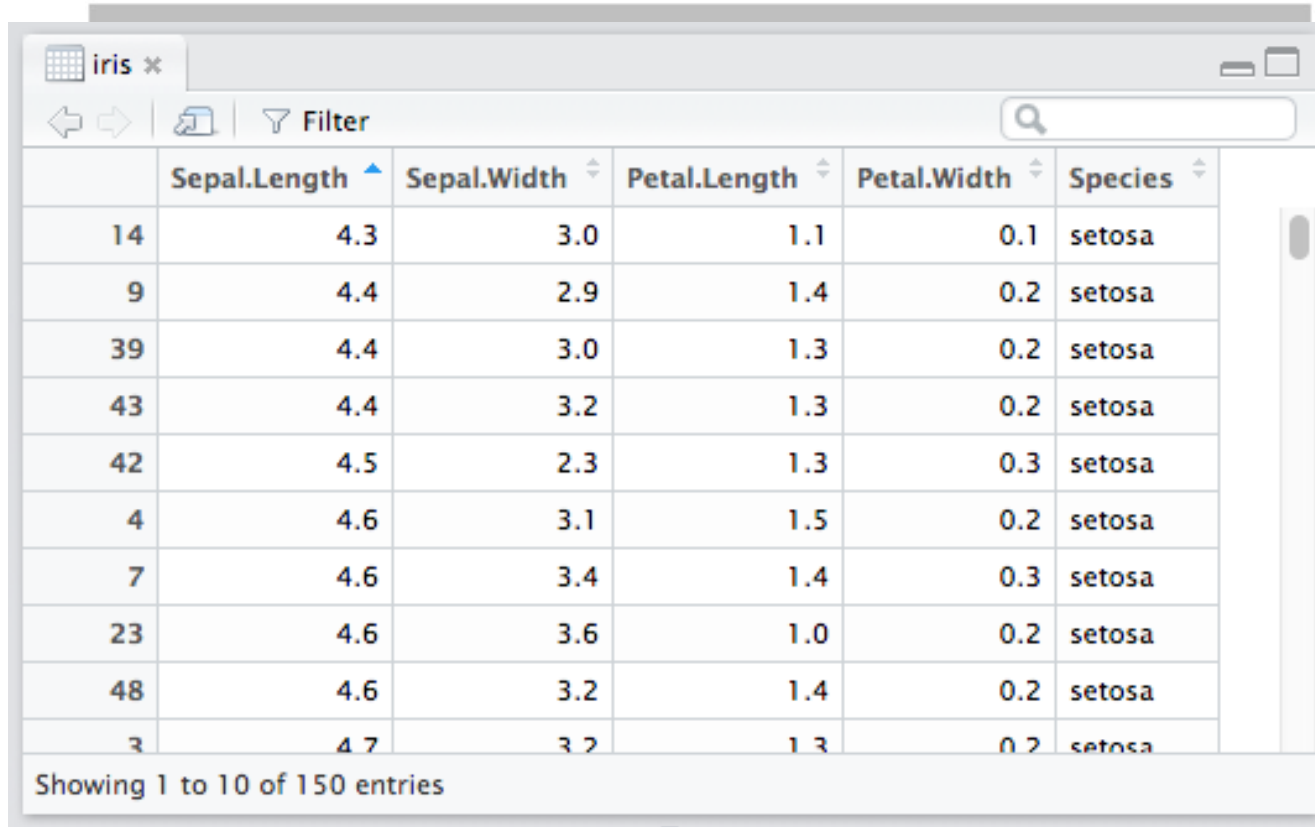
versicolor



virgínica

Echemos un vistazo a los datos de Iris

- Es un datasets muy sencillo, sólo 4 atributos.
- A las personas se nos hace difícil visualizar más de dos atributos.



iris x

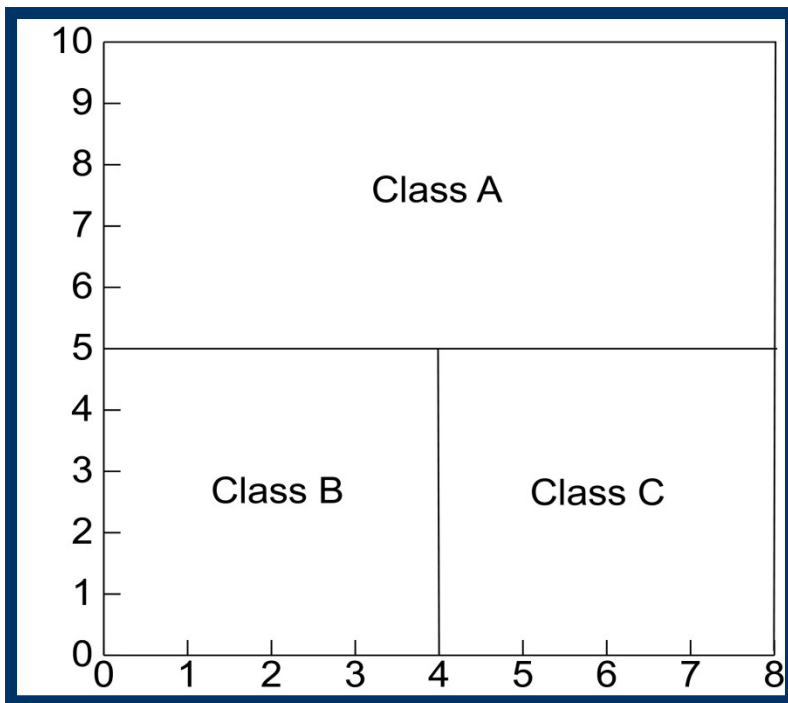
← → | 📄 | 🗑️ Filter 🔍

	Sepal.Length ▲	Sepal.Width ▼	Petal.Length ▼	Petal.Width ▼	Species ▼
14	4.3	3.0	1.1	0.1	setosa
9	4.4	2.9	1.4	0.2	setosa
39	4.4	3.0	1.3	0.2	setosa
43	4.4	3.2	1.3	0.2	setosa
42	4.5	2.3	1.3	0.3	setosa
4	4.6	3.1	1.5	0.2	setosa
7	4.6	3.4	1.4	0.3	setosa
23	4.6	3.6	1.0	0.2	setosa
48	4.6	3.2	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa

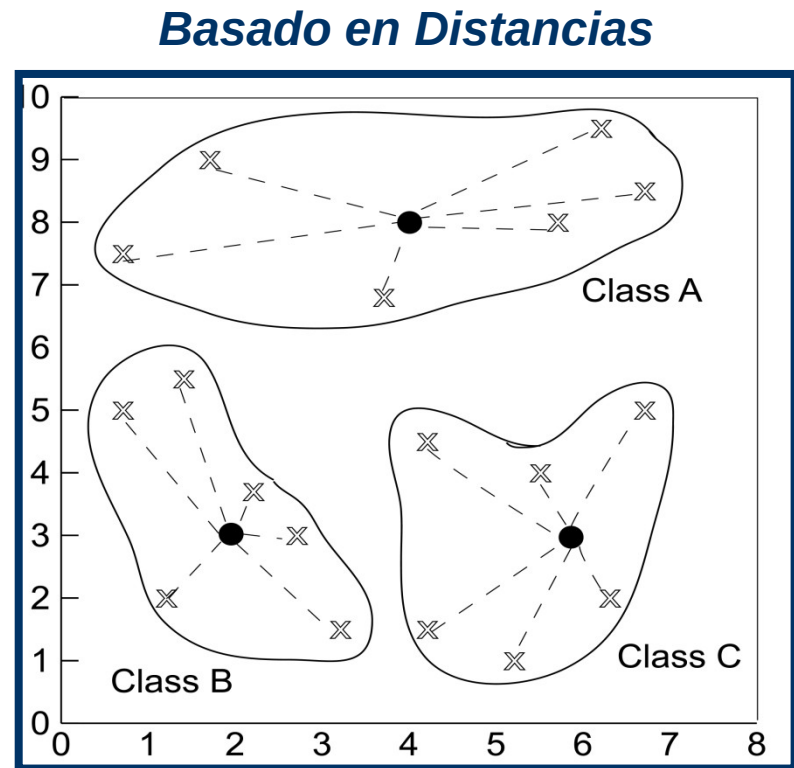
Showing 1 to 10 of 150 entries

Definición del Problema de Clasificación

Ejemplos de clasificación sobre clases definidas: Basada en particiones y en distancias



Basado en Particiones



Basado en Distancias

Definición del Problema de Clasificación

Para diseñar un clasificador, son necesarias dos tareas:
Aprendizaje y Validación

El conjunto de ejemplos se divide en dos subconjuntos:

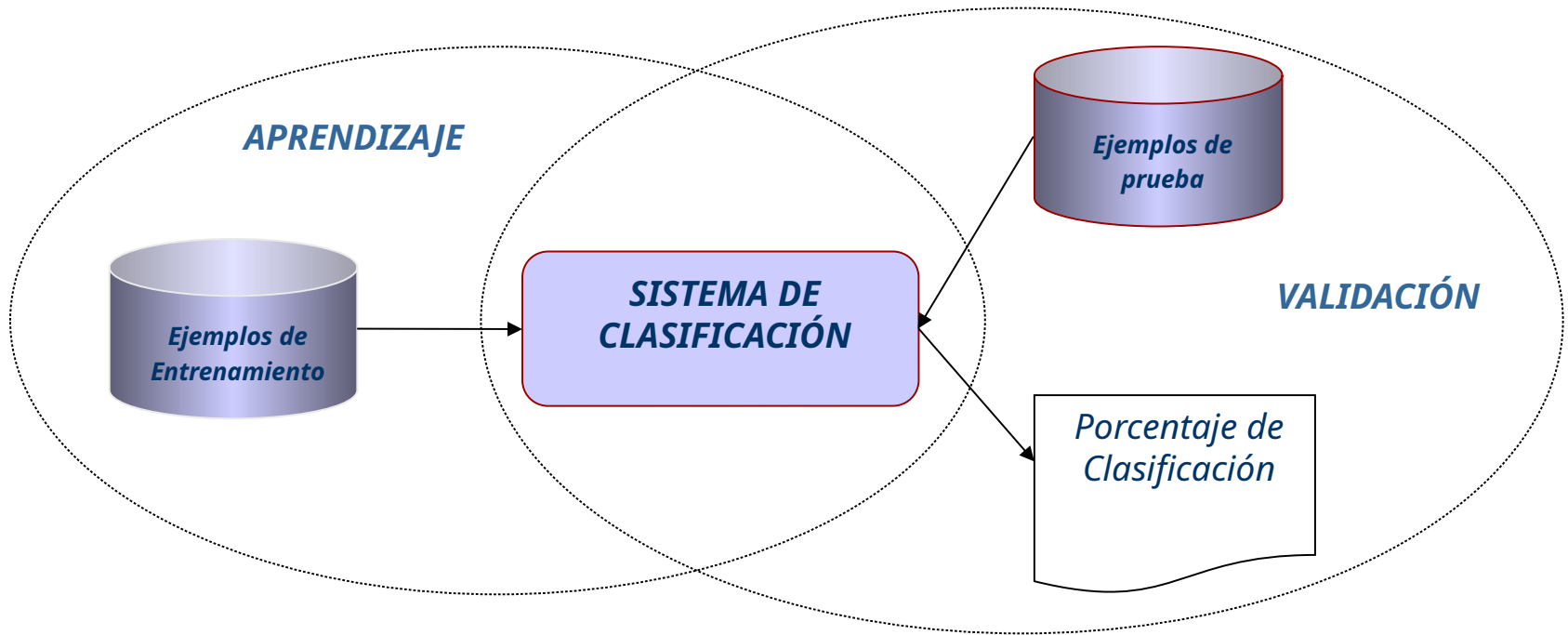
- **Entrenamiento:** Utilizado para aprender el clasificador
- **Prueba:** Se usa para validarlo. Se calcula el porcentaje de clasificación sobre los ejemplos de este conjunto (desconocidos en la tarea de aprendizaje) para conocer su poder de generalización

Para mayor seguridad, se suele hacer varias particiones entrenamiento-prueba

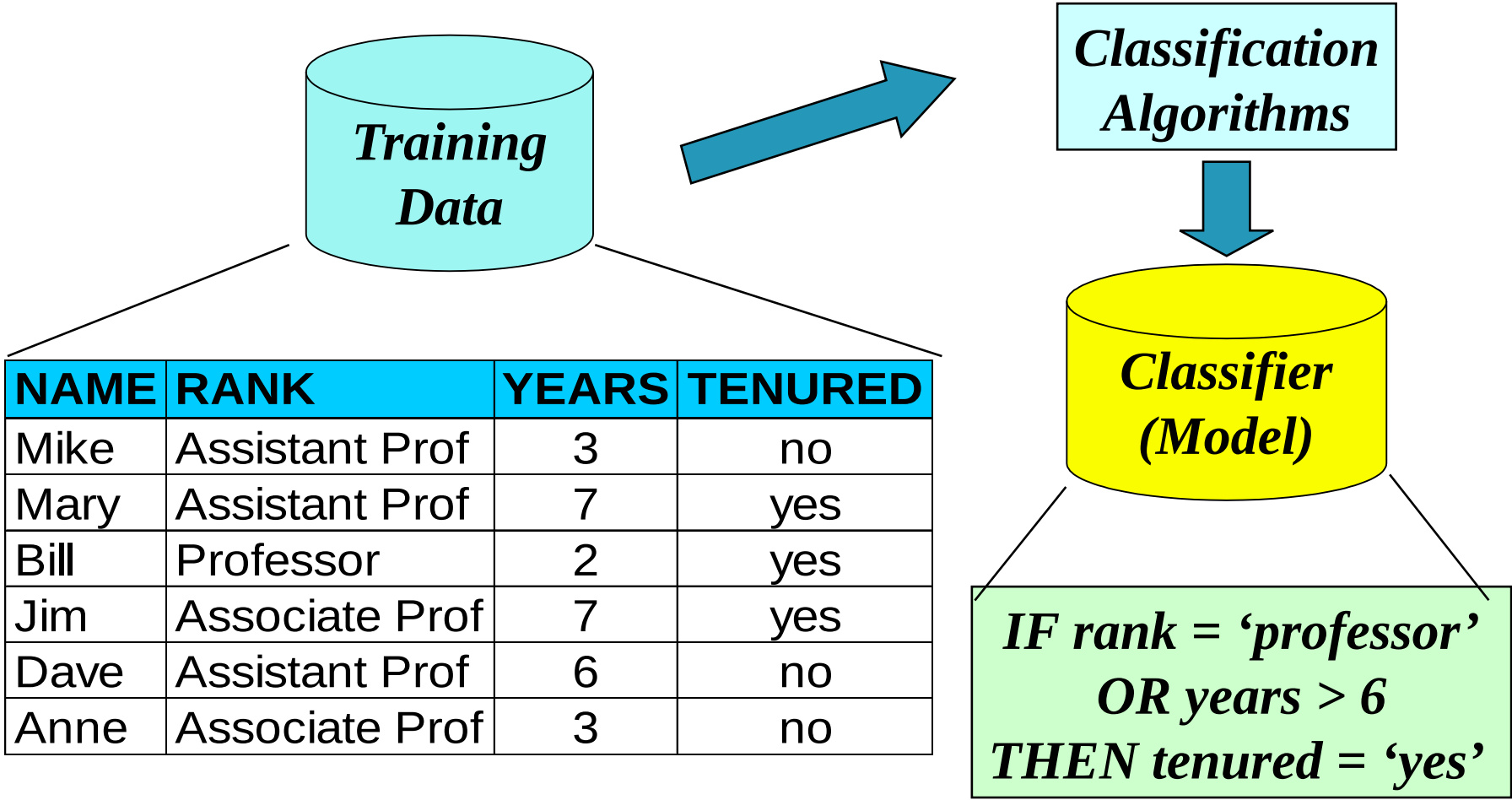
Para cada una, se diseña un clasificador distinto usando los ejemplos de entrenamiento y se valida con los de prueba

Definición del Problema de Clasificación

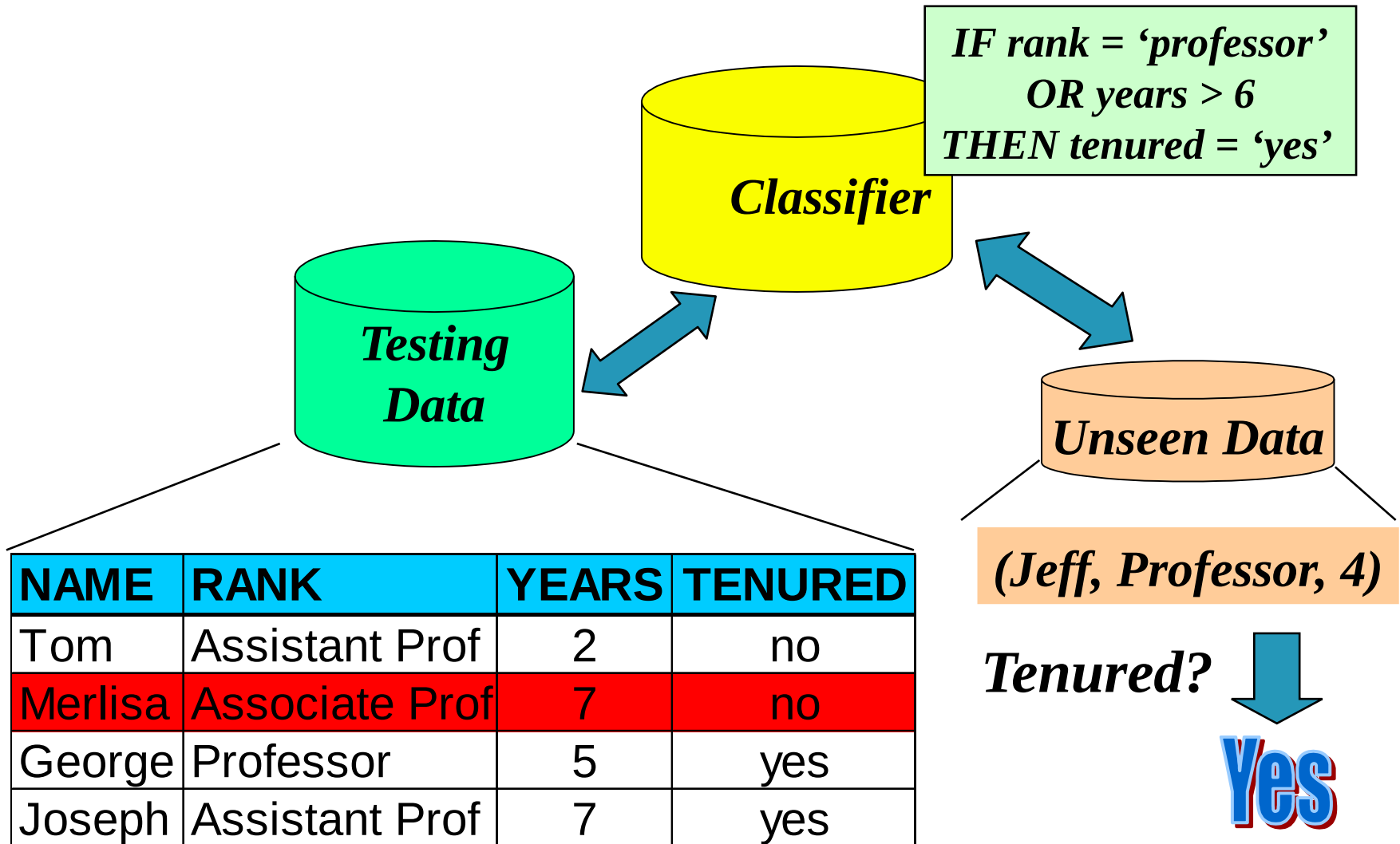
Esquema de aprendizaje en clasificación



Definición del Problema de Clasificación



Definición del Problema de Clasificación

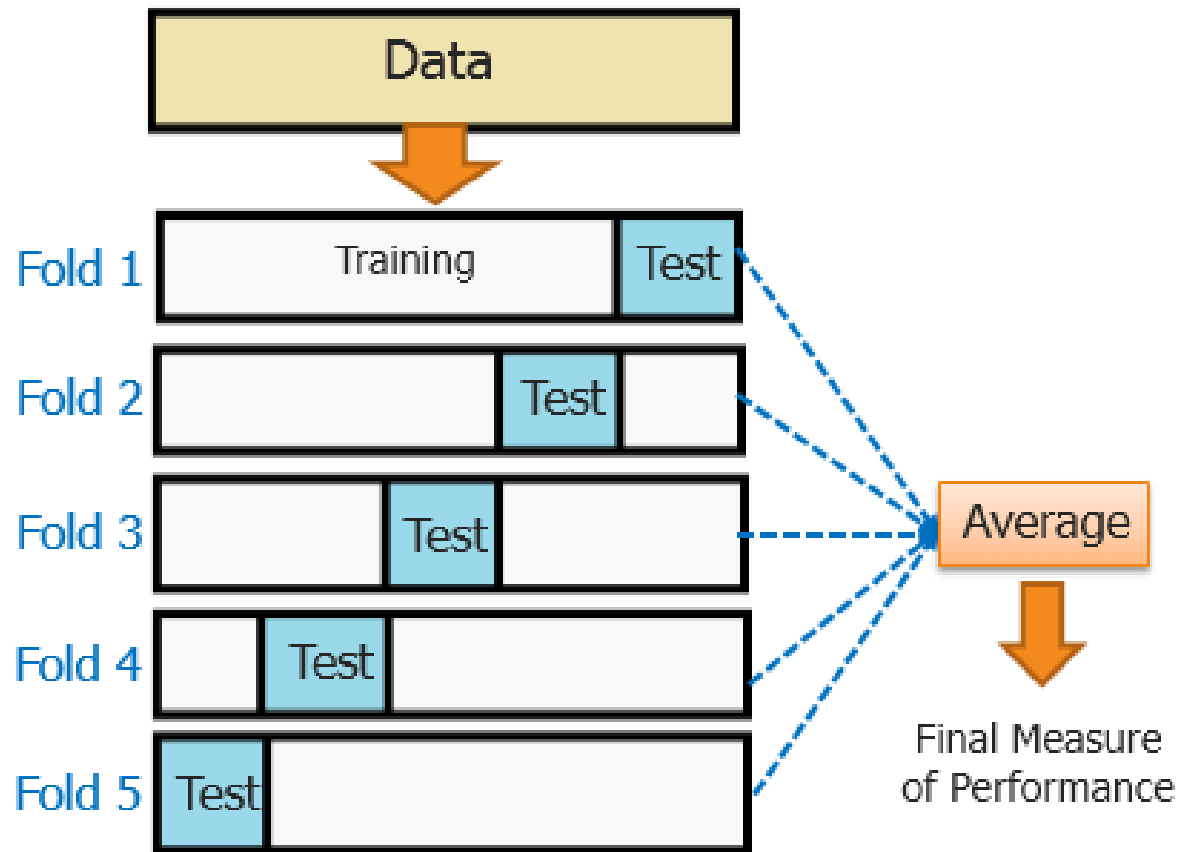


Definición del Problema de Clasificación

Usaremos la técnica de validación cruzada **5-fold cross validation**:

- El conjunto de datos ya está dividido en 5 **particiones disjuntas** al 20% **desordenadas**.
- Aprenderemos un clasificador utilizando el 80% de los datos disponibles (4 particiones de las 5) y validaremos con el 20% restante (la partición restante)
- Así obtendremos un total de 5 valores de porcentaje de clasificación en el conjunto de prueba, uno para cada partición empleada como conjunto de validación
- La calidad del método de clasificación se medirá con un único valor, correspondiente a la media de los 5 porcentajes de clasificación del conjunto de prueba

Definición del Problema de Clasificación



Clasificador k-NN

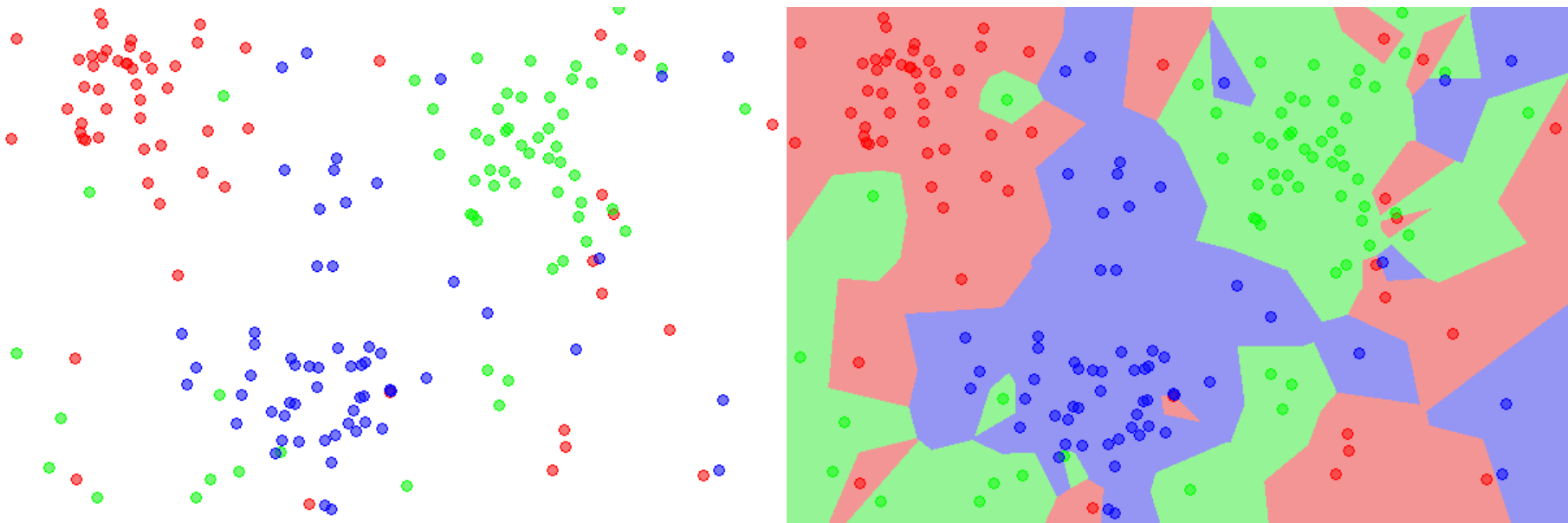
El k-NN (k vecinos más cercanos) es uno de los clasificadores más utilizados por su simplicidad

- i. El proceso de aprendizaje de este clasificador consiste en almacenar una tabla con los ejemplos disponibles, junto a la clase asociada a cada uno de ellos
- ii. Dado un nuevo ejemplo a clasificar, se calcula su distancia (usaremos la euclídea) a los n ejemplos existentes en la tabla y se escogen los k más cercanos
- iii. El nuevo ejemplo se clasifica según la clase mayoritaria de esos k ejemplos más cercanos
- iv. El caso más simple es cuando $k = 1$ (1-NN)

Clasificador k-NN

Imagen derecha: Puntos, su posición (x,y) indican sus características, y el color indica su categoría.

Imagen inferior Izquierda: Clasificación mediante el 1-NN.

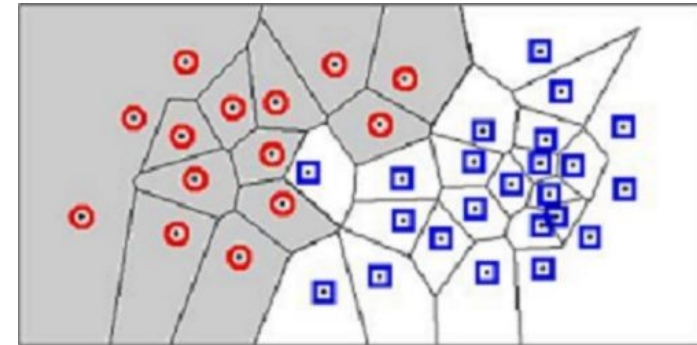


Clasificador k-NN

Regla del vecino más próximo o Nearest neighbour (1-NN)

Si tenemos m ejemplos $\{e_1, \dots, e_m\}$ en nuestro conjunto de datos, para clasificar un nuevo ejemplo e' se hará lo siguiente:

- 1) $c_{min} = \text{clase}(e_1)$
- 2) $d_{min} = d(e_1, e')$
- 3) **Para $i=2$ hasta m hacer**
 - 4) $d = d(e_i, e')$
 - 5) **Si $(d < d_{min})$**
 - 6) **Entonces $c_{min} = \text{clase}(e_i)$, $d_{min} = d$**
- 7) **Devolver c_{min} como clasificación de e'**



$d(\cdot, \cdot)$ es una función de distancia

En el caso de *variables nominales o categóricas* se utiliza la distancia de Hamming:

$$d_h(a, b) = \begin{cases} 0, & \text{si } a = b \\ 1, & \text{si } a \neq b \end{cases}$$

Clasificador k-NN

Distancias para las variables numéricas

Las variables numéricas se suelen normalizar al intervalo [0,1]

Si e_j^i es el valor de la variable j en e_i , es decir

$e_i = (e_i^1, \dots, e_i^n)$ entonces algunas de las distancias más utilizadas son:

Euclídea:
$$d_e(e_1, e_2) = \sqrt{\sum_{i=1}^n (e_1^i - e_2^i)^2}$$

Manhattan:
$$d_m(e_1, e_2) = \sum_{i=1}^n |e_1^i - e_2^i|$$

Minkowski:
$$d_m^k(e_1, e_2) = \left(\sum_{i=1}^n |e_1^i - e_2^i|^k \right)^{1/k}$$

Como se puede observar, $d_m^1 = d_m$ y $d_m^2 = d_e$

Clasificador k-NN

Por tanto, la distancia entre dos ejemplos e_1 y e_2 , utilizando p.e. d_e para las variables numéricas sería

$$d_e(e_1, e_2) = \sqrt{\sum_i (e_1^i - e_2^i)^2 + \sum_j d_h(e_1^j, e_2^j)}$$

siendo i el índice que se utiliza para recorrer las variables numéricas y j el índice que se utiliza para recorrer las variables nominales o categóricas.

Nosotros usaremos la distancia Euclídea para variables numéricas y la distancia de Hamming para variables nominales o categóricas.

Este curso todas las variables son numéricas

$$d_e(e_1, e_2) = \sqrt{\sum_{i=1}^n (e_1^i - e_2^i)^2}$$

Clasificador k-NN

Dado el siguiente ejemplo con 4 instancias, 3 atributos y 2 clases:

x_1 : 0.4 0.8 0.2 positiva
x_2 : 0.2 0.7 0.9 positiva
x_3 : 0.9 0.8 0.9 negativa
x_4 : 0.8 0.1 0.0 negativa

Queremos clasificar con 1-NN el ejemplo:
(todos los pesos iguales) x_q : 0.7 0.2 0.1

$$d(x_1, x_q) = \sqrt{(0.4 - 0.7)^2 + (0.8 - 0.2)^2 + (0.2 - 0.1)^2} = 0.678$$

$$d(x_2, x_q) = \sqrt{(0.2 - 0.7)^2 + (0.7 - 0.2)^2 + (0.9 - 0.1)^2} = 1.068$$

$$d(x_3, x_q) = \sqrt{(0.9 - 0.7)^2 + (0.8 - 0.2)^2 + (0.9 - 0.1)^2} = 1.020$$

$$d(x_4, x_q) = \sqrt{(0.8 - 0.7)^2 + (0.1 - 0.2)^2 + (0.0 - 0.1)^2} = 0.173$$

Calculamos la distancia del ejemplo con todos los de la tabla:

Por tanto, el ejemplo se clasificará con respecto a la clase negativa

IMPORTANTE: Los atributos deben estar normalizados en $[0,1]$ para no priorizar unos sobre otros

Clasificador k-NN

Para normalizar los datos, hay que saber el intervalo de dominio de cada uno de los atributos considerando todas las particiones.

Dado un valor x_j perteneciente al atributo j del ejemplo x y sabiendo que el dominio del atributo j es $[Min_j, Max_j]$, el valor normalizado de x_j es:

$$x_j^N = \frac{x_j - Min_j}{Max_j - Min_j}$$

Definición del Problema del Aprendizaje de Pesos en Características

- Un problema que optimiza el rendimiento del clasificador k-NN es el **Aprendizaje de Pesos en Características (APC)**
- APC asigna valores reales a las características, de tal forma que se describe o pondera la relevancia de cada una de ellas al problema del aprendizaje
- APC funciona mejor cuando se asocia a clasificadores sencillos, locales y muy sensibles a la calidad de los datos
- Nosotros usaremos el clasificador 1-NN, por lo que vamos a considerar el vecino más cercano para predecir la clase de cada objeto

Definición del Problema del Aprendizaje de Pesos en Características

Objetivo:

- Ajustar un conjunto de ponderaciones o pesos asociados al conjunto total de características, de tal forma que los clasificadores que se construyan a partir de él sean *mejores*
- Existen distintos criterios para determinar cuándo el clasificador generado es mejor
- Es un problema de **búsqueda con codificación real** en el espacio n -dimensional, para n características

Definición del Problema del Aprendizaje de Pesos en Características

La expresión anterior considera que todas las variables tienen igual importancia

El problema del Aprendizaje de Pesos en Características (APC) asigna pesos a los atributos de forma que se pondere su importancia dentro del contexto:

$$d_e(e_1, e_2) = \sqrt{\sum_i w_i \cdot (e_1^i - e_2^i)^2 + \sum_j w_j \cdot d_h(e_1^j, e_2^j)}$$

Los pesos vienen dados por un vector W , tal que cada w_i ó w_j representa un valor real en $[0, 1]$

Los valores bajos de w_i pueden emplearse para seleccionar características y diseñar un clasificador más simple y preciso

Definición del Problema del Aprendizaje de Pesos en Características

Como W es independiente al tipo de característica asociada (numérica o nominal), utilizaremos a partir de ahora el índice i (w_i) indistintamente al tipo de característica

El tamaño de W será n , debido a que condiciona a todas las características del problema n -dimensional

W también puede verse con un punto n -dimensional en \mathbb{R} acotado en $[0, 1]$

El objetivo consiste en encontrar el mejor W para el problema de clasificación concreto y el clasificador 1-NN

Definición del Problema del Aprendizaje de Pesos en Características

Función de evaluación: combinación de dos criterios, precisión y simplicidad

- Rendimiento promedio de un clasificador 1-NN (**considerando $k=1$ vecino y *leave one out***) aplicando validación sobre el conjunto T de datos: ***tasa_clas***
- ***tasa_clas*** mide el porcentaje de instancias correctamente clasificadas pertenecientes a T (***precisión***):

$$tasa_clas = 100 \cdot \frac{\text{n}^\circ \text{ instancias bien clasificadas de } T}{\text{n}^\circ \text{ instancias en } T}$$

Definición del Problema del Aprendizaje de Pesos en Características

Función de evaluación: combinación de dos criterios, precisión y simplicidad

- Tasa de reducción asociada al número de características utilizadas por el clasificador con respecto al número total de características
n: ***tasa_red***
- *tasa_red* mide el porcentaje de características **descartadas**, aquellas cuyo peso esté cercano a cero en W , con respecto a n (***simplicidad***)
- Consideraremos un umbral de 0.1 en w_i para considerar que se descarta una característica:

$$tasa_red = 100 \cdot \frac{\text{n}^\circ \text{ valores } w_i < 0.1}{\text{n}^\circ \text{ características}}$$

Definición del Problema del Aprendizaje de Pesos en Características

Función de evaluación: combinación de dos criterios, precisión y simplicidad

- Utilizaremos una agregación sencilla que combine ambos objetivos en un único valor. La función objetivo será:

$$F(W) = \alpha \cdot \text{tasaclas}(W) + (1 - \alpha) \cdot \text{tasared}(W)$$

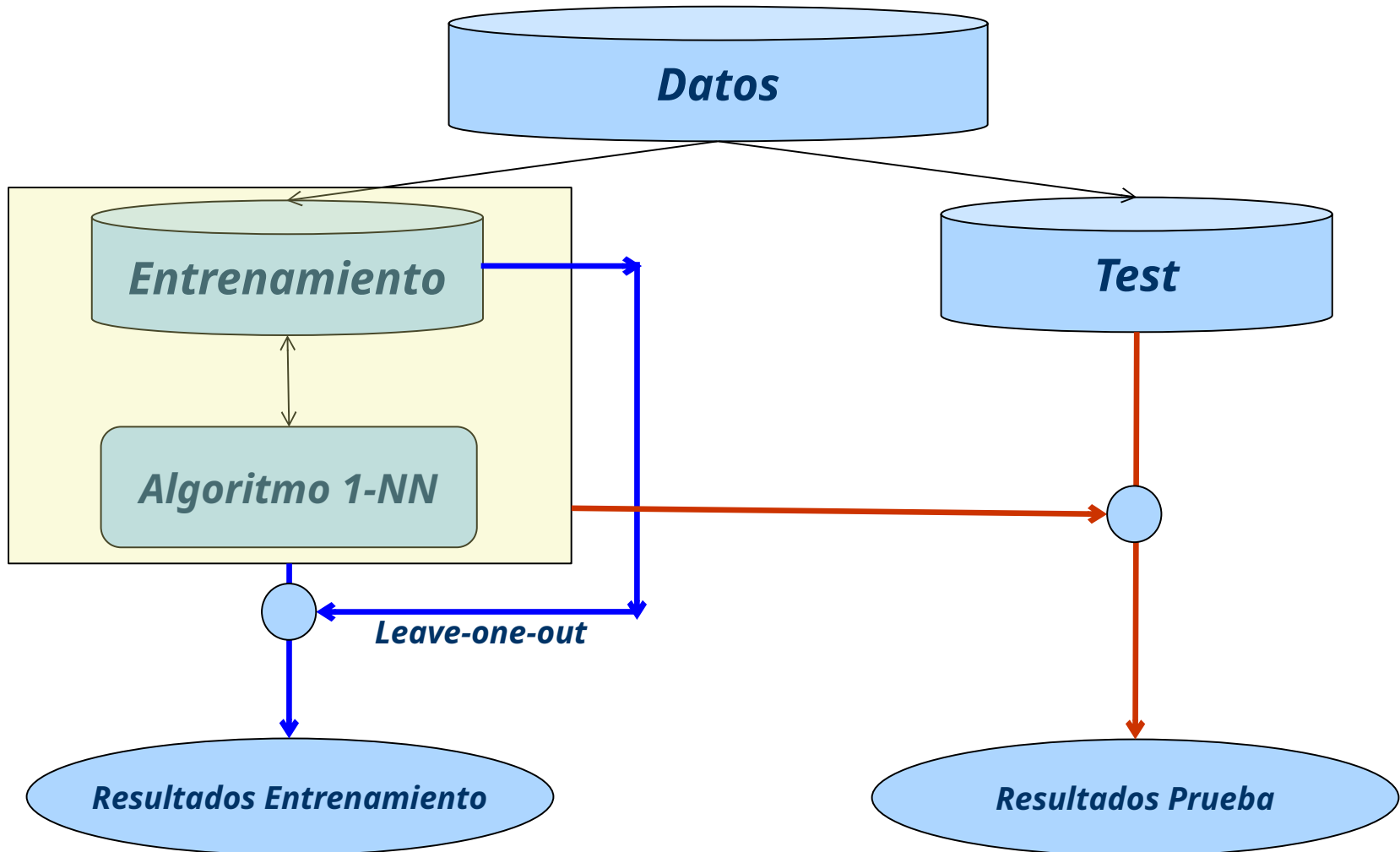
- El valor de α pondera la importancia entre el acierto y la reducción de características de la solución encontrada (el clasificador generado). Usaremos $\alpha = 0.75$, dando más importancia al acierto.
- El objetivo es obtener el conjunto de pesos W que maximiza esta función, es decir, que maximice el acierto del clasificador 1-NN y, a la vez, que considere el menor número de características posible

Definición del Problema del Aprendizaje de Pesos en Características

Cálculo del Porcentaje de Entrenamiento (tasa-clas) en 1-NN: Leave one out

- En el algoritmo 1-NN, no es posible calcular el porcentaje de acierto sobre el conjunto de entrenamiento de un modo directo
- Si intentásemos clasificar un ejemplo del conjunto de entrenamiento directamente con el clasificador 1-NN, el ejemplo más cercano sería siempre él mismo, con lo que se obtendría un 100% de acierto
- Para remediar esto, se debe usar el procedimiento “dejar uno fuera” (“**leave one out**”). Para clasificar cada ejemplo del conjunto de entrenamiento, se busca el ejemplo más cercano **sin considerar a él mismo**
- Por lo demás, se opera igual: cada vez que la clase devuelta coincida con la clase real del ejemplo, se contabiliza un acierto
- El porcentaje final de acierto es el número de aciertos entre el número total de ejemplos

Definición del Problema del Aprendizaje de Pesos en Características



Representación del Problema del Aprendizaje de Pesos en Características

Representación real: Vector real de tamaño n :

$$W = (w_1, w_2, \dots, w_n), \text{ donde } w_i \in [0, 1]$$



Un 1 en la posición w_i indica que la característica en cuestión se considera completamente en el cálculo de la distancia

Un valor menor que 0.1 en la posición w_i indica que la característica no se considera en el cálculo de la distancia

Cualquier otro valor intermedio gradúa el peso asociado a cada característica y pondera su importancia en la clasificación final

Métodos de búsqueda:

- Búsqueda secuencial
 - Método voraz (*greedy*) que parte de un vector de pesos inicializado a 0 que incrementa cada componente en función de la distancia al enemigo más cercano de cada ejemplo, y disminuye cada componente en función de la distancia al amigo más cercano de cada ejemplo.
- Búsqueda probabilística
 - Métodos MonteCarlo y Las Vegas
- Búsqueda con metaheurísticas

Solución *greedy*

Descripción método *RELIEF*:

- Parte de un vector de pesos W inicializado a 0: $w_i = 0$
- En cada paso se modifica W utilizando cada uno de los ejemplos del conjunto de entrenamiento
- Para cada ejemplo e del conjunto de entrenamiento, se busca a su *enemigo* más cercano e_e (ejemplo más cercano con clase diferente) y a su *amigo* más cercano e_a (ejemplo más cercano de la misma clase sin considerar él mismo)
- W se actualiza con la distancia dimensión a dimensión entre e y e_e y entre e y e_a
- Si $w_i < 0 \Rightarrow w_i = 0$. Después, W se normaliza al intervalo $[0, 1]$

Solución *greedy*

Algoritmo método RELIEF:

$W = \{0, 0, \dots, 0\}$

Para cada e_i en T

Buscar el enemigo más cercano de e_i : e_e

Buscar el amigo más cercano de e_i : e_a

$W = W + |e_i - e_e| - |e_i - e_a|$

$w_m = \text{máximo}(W)$

Para cada w_i en W

 Si $w_i < 0$ entonces

$w_i = 0$

 Si no

$w_i = w_i / w_m$

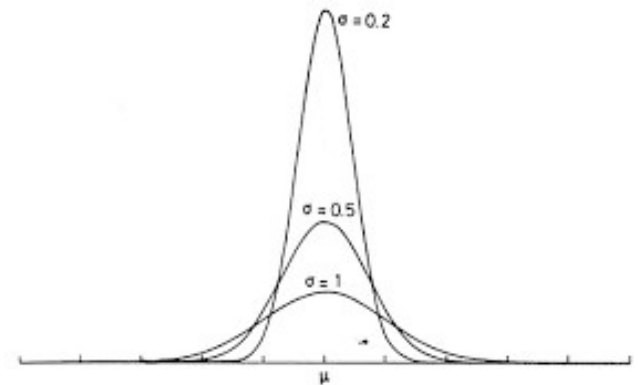
Devolver W

Búsquedas por Trayectorias Simples

- **Representación real:** **Problema de codificación real:** un vector real $W=(w_1, \dots, w_n)$ en el que cada posición i representa una característica y su valor en $[0, 1]$ indica el peso aprendido para cada característica. No tiene restricciones exceptuando el dominio $[0, 1]$ para cada dimensión
- **Operador de vecino por Mutación Normal:** El entorno de una solución W está formado por las soluciones accesibles desde ella a través de un **movimiento** basado en la mutación de una componente z_i , con un radio que depende de δ :

$$Mov(W, \delta) = W' = (w_1, \dots, w_i + z_i, \dots, w_n)$$

$$z_i \sim N_i(0, \delta^2)$$



Búsquedas por Trayectorias Simples

- $Mov(W, \delta)$ verifica las restricciones si después de aplicarlo, truncamos el w_i modificado a $[0, 1]$. Así, si la solución original W es factible siempre genera una solución vecina W' factible.
- El problema del APC **no permite realizar un cálculo factorizado del coste** de forma sencilla.
- El tamaño del entorno es infinito, al ser un problema de codificación real. No es fácil definir una preferencia entre las características para explorar el entorno. Podría considerarse alguna medida de cantidad de información para ello.
- En nuestro caso, en cada paso de la exploración se mutará una componente $i \in \{1, \dots, n\}$ **distinta** sin repetición hasta que haya mejora o se hayan modificado todas. Si se produce mejora, se acepta la solución vecina y se comienza de nuevo el proceso.
- Si ninguna de las n mutaciones produce mejora, se empieza de nuevo el proceso de exploración de vecindario sobre la solución actual.

Búsqueda Local para el APC

- Algoritmo de **búsqueda local del primer mejor**: en cuanto se genera una solución vecina que mejora a la actual, se aplica el movimiento y se pasa a la siguiente iteración
- Se detiene la búsqueda cuando se haya generado un número máximo de vecinos
- No se considera ningún tipo de **factorización** ni ningún mecanismo específico de exploración del entorno más allá de la generación aleatoria de la componente a mutar sin repetición

Bases de Datos a Utilizar

- En la actualidad, hay muchas bases de datos que se utilizan como bancos de prueba (*benchmarks*) para comprobar el rendimiento de los algoritmos de clasificación
- El UCI es un repositorio de bases de datos para aprendizaje automático muy conocido
- Está accesible en la Web en:

<http://www.ics.uci.edu/~mlearn/MLRepository.html>

Bases de Datos a Utilizar

- A partir de este repositorio, se ha desarrollado un formato para definir todas las cualidades de una base de datos en un fichero por partición.
- Se trata del formato ARFF, utilizado en WEKA (<http://www.cs.waikato.ac.nz/ml/weka/>)
- Un fichero ARFF contiene dos partes:
 - Datos de cabecera: Líneas que comienzan por @. Contienen información acerca de los atributos: significado y tipo
 - Datos del problema: Líneas de datos. Son los ejemplos en sí. Cada línea se corresponde con un ejemplo y los valores están separados por comas

Bases de Datos a Utilizar

Ejemplo fichero ARFF:

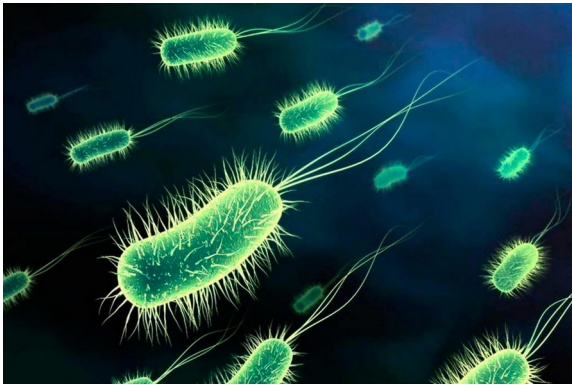
```
@relation iris  
@attribute sepalLength real  
@attribute sepalWidth real  
@attribute petalLength real  
@attribute petalWidth real  
@attribute class {Iris-setosa, Iris-versicolor, Iris-virginica}  
@data  
5.1, 3.5, 1.4, 0.2, Iris-setosa  
4.9, 3.0, 1.4, 0.2, Iris-setosa  
7.0, 3.2, 4.7, 1.4, Iris-versicolor  
6.0, 3.0, 4.8, 1.8, Iris-virginica
```

- 5 Atributos, los 4 primeros de tipo real y el último de tipo nominal
- La clase es el último atributo (atributo de salida) con los posibles valores definidos
- Los datos van a continuación de la directiva @data

Bases de Datos a Utilizar

Trabajaremos con tres bases de datos: Ecoli, Parkinson y Breast-cancer

Ecoli es una base de datos para identificar la posición de proteínas tras obtener métricas mediante una serie de técnicas distintas.



- Consta de 366 ejemplos
- Consta de 8 atributos (clase incluida)
- Consta de 8 clases (citoplasma, membrana interna, periplasma, ...)
- Atributos: Método de McGeoch's, Método de Heijne's...

Bases de Datos a Utilizar

Trabajaremos con tres bases de datos: Ecoli, Parkinson y Breast-cancer

Parkinson contiene datos que se utilizan para distinguir entre la presencia y la ausencia de la enfermedad de Parkinson en una serie de pacientes a partir de medidas biomédicas de la voz.



- Consta de 195 ejemplos
- Consta de 23 atributos (clase incluida)
- Consta de 2 clases (1 Sano, 2 Enfermo), La distribución de ejemplos está desbalanceada (147 enfermos, 48 sanos).
- Atributos: Distintas métricas sonoras (como periodo pitch, Ratio harmónico-ruído).

Fuente:

<https://www.kaggle.com/datasets/vikasukani/parkinsons-disease-data-set>

Bases de Datos a Utilizar

Trabajaremos con tres bases de datos: Ecoli, Parkinson y Breast-cancer

Breast-cancer es una base de datos para identificar la gravedad de cancer de máma a partir de distintos atributos.



- Consta de 569 ejemplos
- Consta de 31 atributos (clase incluida)
- Consta de 2 clase (B Benigno, M Maligno).
- Atributos: Métricas usando 3 imágenes con distinta orientación, cada una con radio, textura, perímetro, área, ... 10 por imágenes, obteniendo 30 en total.

Fuente:

<https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic>

