Modelos computacionales supervisados

Francisco J. Ribadas-Pena

28 de abril de 2021

Grupo CoLe – 2021

Contenidos

Aprendizaje de árboles de decisión

Aprendizaje por analogía: k-NN

Support Vector Machines (SMV)

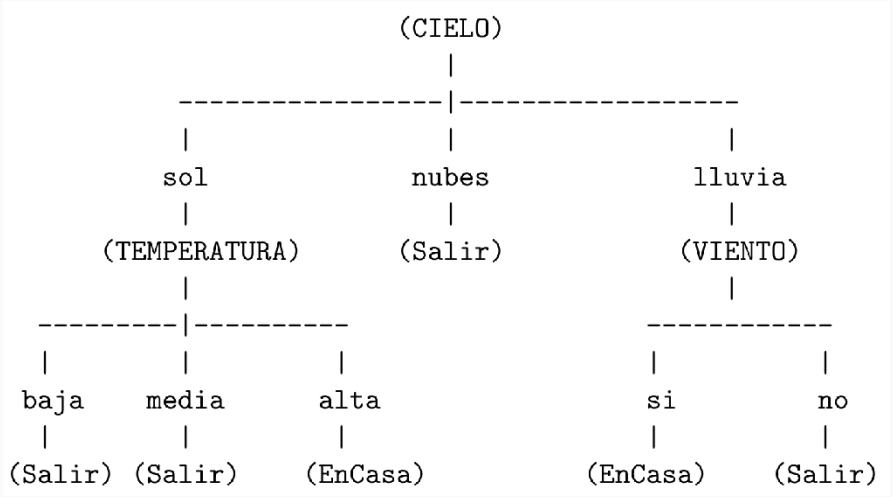
Ensemble learning

Aprendizaje de árboles de decisión

Árboles de decisión

Representación en forma de árbol de un clasificador de ejemplos en base a sus atributos (en principio, atributos y clases discretos)

- **hojas:** asignan clases
- **nodo internos:** nodos de decisión
  - especifican un test sobre un atributo
  - descendientes se corresponden con los diferentes valores del atributo



https://scikit-learn.org/stable/modules/tree.html

Árboles de decisión (II)

OBJETIVO: Encontrar el árbol más pequeño consistente con ejemplos.

- Define un sesgo (preferencia) en la búsqueda de hipótesis
- Sigue principio de la *Navaja de Occam*:

  "Ante distintas hipótesis consistentes con los datos, dar preferencia a la más simple."

  - Árboles grandes $\Rightarrow$ simplemente "memorizan" ejemplos
  - Árboles pequeños $\Rightarrow$ mayor capacidad de generalización/predicción
- Costoso encontrar el más pequeño $\Rightarrow$ uso de heurísticas (soluciones aproximadas)

Potencia expresiva equivalente a lógica proposicional

- Posibilidad de convertir árboles en reglas

Construcción de árboles de decisión

IDEA BASE: Seguir aproximación descendente.

- Clasificar/agrupar primero por atributos que mejor organicen los ejemplos.
- Cada conjunto resultante es un nuevo problema (más simple)
- Aplicar esta misma idea recursivamente

```
funcion ArbolDecicision(ejemplos, atributos): arbol
    SI todos los ejemplos en una Categoria
       DEVOLVER Hoja con esa Categoria
    SINO SI NO hay más atributos
       DEVOLVER Hoja con Categoría más comun en los ejemplos
    SINO
       Seleccionar MEJOR ATRIBUTO (A)
       Etiquetar arbol con A
       PARA CADA Posible Valor de A (a_i)
          tomar ejemplos con ese valor (ejemplos_i)
          /* CREAR DESCENDIENTE  */
          llamada recurs. ArbolDecision(ejemplos_i, atributos - {A})
          enlazar con nodo A, etiquetando el enlace como "A=a_i"
      FIN PARA
   FIN SI
```

Construcción de árboles de decisión (II)

Heurística de selección del MEJOR ATRIBUTO

Algoritmo ID3 (Quinlan 79): Selección de atributos basada en la Teoría de la Información (Shanon y Weaver, 49)

Entropia: (cantidad de información) Mide la *impureza* (desorden) de un conjunto de datos D

- Cantidad de información (en bits) necesaria para identificar una clase C_i en D
- Inversamente proporcional a la frecuencia (probabilidad) de ocurrencia de dicha clase C_i.

$$\text{Se mide en bits} \Rightarrow \text{cálculo} = \boxed{-log_2(frecuencia(C_i))}$$

- Para k clases $(C_1, C_2, \ldots, C_k)$, la entropía del conjunto D es suma ponderada de las entropías de cada clase

$$\boxed{entropia(D) = \sum_{i=1}^{k} frecuencia(C_i) \times (-log_2(frecuencia(C_i)))}$$

Construcción de árboles de decisión (III)

Ganancia de información: Reducción de la entropía después de clasificar D en base a un atributo A

- Diferencia entre la entropía de los subconjutos resultantes de clasificar D de acuerdo los valores de un atributo A y la entropía de D sin clasificar.
- Siendo $a_1, a_2, ..., a_k$ los posibles valores de A y D_i los subconjuntos de D asociado a cada valor a_i:

$$ganancia(D, A) = entropia(D) - \sum_{i=1}^{k} \frac{|D_k|}{|D|} \times entropia(D_k)$$

Para elegir el mejor atributo en cada paso del algoritmo anterior:

- Para cada atributo no utilizado $\rightarrow$ calcular su ganancia sobre el conjunto de ejemplos actual
- Quedarse con el atributo que proporcione mayor ganancia ("separa" los ejemplos en grupos más homogéneos)

Ejemplo

Cielo	Temp	Viento	Acción
Sol	Media	Si	Salir
Sol	Alta	Si	En Casa
Sol	Alta	No	En Casa
Sol	Media	No	Salir
Sol	Baja	No	Salir
Nubes	Media	Si	Salir
Nubes	Alta	No	Salir
Nubes	Baja	Si	Salir
Nubes	Alta	No	Salir
Lluvia	Media	Si	En Casa
Lluvia	Baja	Si	En Casa
Lluvia	Media	No	Salir
Lluvia	Baja	No	Salir
Lluvia	Media	No	Salir

Entropía de los datos (2 clases: Salir/EnCasa)

$$entr(D) = \frac{6}{14}\left(-\log_2 \frac{6}{14}\right) + \frac{8}{14}\left(-\log_2 \frac{8}{14}\right) = 0{,}98 \text{ bits}$$

Ejemplo (II)

1. Clasificación por **Cielo** (tres valores: Sol, Nubes, Lluvia)

$$entr(D_{\text{sol}}) = \tfrac{3}{5}\left(-\log_2 \tfrac{3}{5}\right) + \tfrac{2}{5}\left(-\log_2 \tfrac{2}{5}\right) = 0{,}97$$
$$entr(D_{\text{subes}}) = \tfrac{4}{4}\left(-\log_2 \tfrac{4}{4}\right) + \tfrac{0}{4}\left(-\log_2 \tfrac{0}{4}\right) = 0 \text{ (ya clasificado)}$$
$$entr(D_{\text{lluvia}}) = \tfrac{2}{5}\left(-\log_2 \tfrac{2}{5}\right) + \tfrac{3}{5}\left(-\log_2 \tfrac{3}{5}\right) = 0{,}97$$

$$gan(D, Cielo) = entr(D) - \left(\tfrac{5}{14} entr(D_{\text{sol}}) + \tfrac{4}{14} entr(D_{\text{nubes}}) + \tfrac{5}{14} entr(D_{\text{lluvia}})\right) = 0{,}98 - 0{,}69 = 0{,}29 \text{ bits}$$

2. Clasificación por **Temperatura** (tres valores: Baja, Media, Alta)

$$entr(D_{\text{baja}}) = \tfrac{3}{4}\left(-\log_2 \tfrac{3}{4}\right) + \tfrac{1}{4}\left(-\log_2 \tfrac{1}{4}\right) = 0{,}81$$
$$entr(D_{\text{media}}) = \tfrac{5}{6}\left(-\log_2 \tfrac{5}{6}\right) + \tfrac{1}{6}\left(-\log_2 \tfrac{1}{6}\right) = 0{,}65$$
$$entr(D_{\text{alta}}) = \tfrac{2}{4}\left(-\log_2 \tfrac{2}{4}\right) + \tfrac{2}{4}\left(-\log_2 \tfrac{2}{4}\right) = 1$$

$$gan(D, Temp) = entr(D) - \left(\tfrac{4}{14} entr(D_{\text{baja}}) + \tfrac{6}{14} entr(D_{\text{media}}) + \tfrac{4}{14} entr(D_{\text{alta}})\right) = 0{,}98 - 0{,}79 = 0{,}19 \text{ bits}$$

Ejemplo (III)

3. Clasificación por **Viento** (dos valores: Si, No)

$$entr(D_{\text{si}}) = \tfrac{3}{6}\left(-\log_2 \tfrac{3}{6}\right) + \tfrac{3}{6}\left(-\log_2 \tfrac{3}{6}\right) = 1$$
$$entr(D_{\text{no}}) = \tfrac{7}{8}\left(-\log_2 \tfrac{7}{8}\right) + \tfrac{1}{8}\left(-\log_2 \tfrac{1}{8}\right) = 0{,}54$$

$$gan(D, Viento) = entropia(D) - \left(\tfrac{6}{14}\, entr(D_{\text{si}}) + \tfrac{8}{14}\, entr(D_{\text{no}})\right) = 0{,}98 - 0{,}74 = 0{,}24 \text{ bits}$$

Atributo Mayor Ganancia: **Cielo** $\Rightarrow$ Clasificar por Cielo $\Rightarrow$ Resultan tres subproblemas

Cielo = Sol

Temp	**Viento**	**Acción**
Media	Si	Salir
Alta	Si	EnCasa
Alta	No	EnCasa
Media	No	Salir
Baja	No	Salir

Siguiente clasificación: **Temp**

Cielo = Nubes

Temp	**Viento**	**Acción**
Media	Si	Salir
Alta	No	Salir
Baja	Si	Salir
Alta	No	Salir

Ya clasificado (etiqueta = **Salir**)

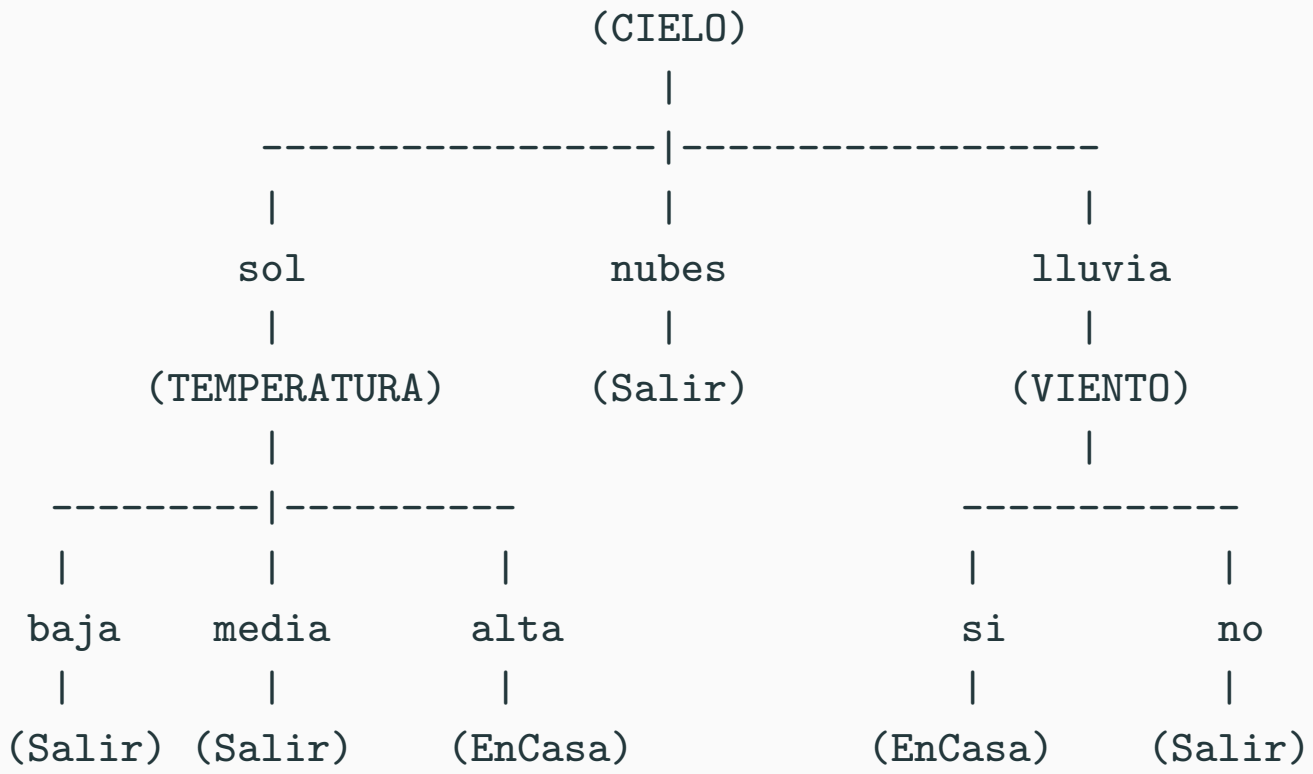
Cielo = Lluvia

Temp	**Viento**	**Acción**
Media	Si	EnCasa
Baja	Si	EnCasa
Media	No	Salir
Baja	No	Salir
Media	No	Salir

Siguiente clasificación: **Viento**

Ejemplo (IV)

Árbol final



Limitaciones y mejoras

Sobreadaptación y ruido

Un árbol que clasifique los datos de entrenamiento perfectamente, no asegura que proporcione la mejor generalización.

- Si hay ruido (ejemplos erróneos), árbol se ajusta para cubrir esos casos $\Rightarrow$ crece
- Decisiones en niveles inferiores soportadas por un conjunto de datos demasiado pequeño (no representativo)

Solución: Eliminar algunos nodos de niveles inferiores (podar).

- **pre-poda:** (*prepruning*) detener crecimiento de nodos durante la construcción
- **post-poda:** (*postpruning*) eliminar nodos después de construido el árbol

Limitaciones y mejoras (II)

Heurísticas alternativas

Selección de atributos basada sólo en ganancia de información favorece atributos con muchos valores diferentes.

- Dividen datos en conjunto muy pequeños $\Rightarrow$ datos más homogéneos (baja entropía)
- Situar atributos con muchos valores en nodos superiores hacen el árbol menos general (menor poder predictivo)

Solución: Ganancia ponderada. (Algoritmo C4.5, Quinlan)

- Normalizar la ganancia dividiendo por la entropía respecto a los valores del atributo, no respecto a las clases
- Penaliza ganancia de atributos con muchos valores $\Rightarrow$ asegura una división realmente informativa (útil)

Limitaciones y mejoras (III)

Otras limitaciones

- Soporte atributos continuos $\rightarrow$ *discretizar* valores agrupándolos en rangos con mayor ganancia de información
- Valores de atributos desconocidos $\rightarrow$ "replicación" ponderada de los ejemplos asignando los posibles valores del atributo ausente

Aprendizaje por analogía: k-NN

k Nearest Neighbors

También: aprendizaje basado en instancias (*Instance Based Learning, IBL)*, aprendizaje *vago* (*"lazzy"*)

Idea Base: Almacenar los ejemplos de entrenamiento.

- No se construye representación explícita de la hipótesis aprendida
  - Se almacenan los ejemplos de entrenamiento (todos o un subconjunto seleccionado)
  - La generalización tiene lugar al clasificar un ejemplo nuevo
- Clasificación de nuevos ejemplos se obtiene midiendo su *"similaridad"* con los ejemplos memorizados
  - la salida se construye en base a los ejemplos más parecidos
  - normalmente se omiten ejemplos no cercanos (pero pueden tenerse en cuenta)
  - requiere definir medida de similaridad entre ejemplos

`https://scikit-learn.org/stable/modules/neighbors.html`

Medidas de similaridad

Aspecto crucial en métodos IBL: definición de funciones de similaridad entre ejemplos adecuadas.

- depende del problema y del tipo de atributos que se manejen
- en general, ejemplos podrán representarse en un espacio $n-$dimensional ($n =$ nº atributos)

Atributos continuos: uso de la *distancia euclídea*

- Similaridad entre ejemplos x e y = distancia entre sus vectores de atributos

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (a_i(x) - a_i(y))^2}$$

Medidas de similaridad (II)

Atributos no continuos:
- convertirlos en números (si posible)
- uso de métricas discretas (dist. *Hamming*)

- Distancia *Hamming*: x e y ejemplos con n atributos discretos

$$d_{Hamming}(x,y) = \sum_{i=1}^{n} \delta\left(a_i(x), a_i(y)\right) \quad \text{con} \quad \delta(a,b) = \begin{cases} 0 & \text{si } a = b \\ 1 & \text{si } a \neq b \end{cases}$$

Otras opciones:

- Medidas de similaridad no numéricas basadas únicamente en valores simbólicos/discretos
- Para valores simbólicos estructurados (ontologías, redes semánticas) posible definir similaridades numéricas (no $\{0,1\}$) entre los valores de los atributos
- Métricas específicas dependientes del dominio.
  - No necesitan cumplir las propiedades matemáticas de las distancias.
  - *Ejemplo:* Modelo vectorial de recuperación de información
    - Coseno del ángulo entre los vectores que representan documento y consulta
    - Máxima similaridad (idénticos): 1 Mínima similaridad: 0

Clasificación k vecinos más próximos (k-NN)

Procedimiento de clasificación

Dado un ejemplo nuevo: $y = [a_1(y), \ldots, a_n(y)]$

1. Calcular la distancia entre el nuevo ejemplo, y, y cada ejemplo de entrenamiento memorizado, x_j.
2. La clasificación del nuevo ejemplo se obtiene de las clasificaciones de los k ejemplos memorizados más similares (k-vecinos más próximos)
3. El valor estimado $\tilde{f}(y)$ será la clase más común (*moda*) de esos k ejemplos más similares)
   - Votación simple: gana la clase que aparece más veces.
   - Valores $k > 1$ aumentan la tolerancia a ruido (errores en conj. de entrenamiento)

Uso $k-$NN en aproximación de funciones (regresión)

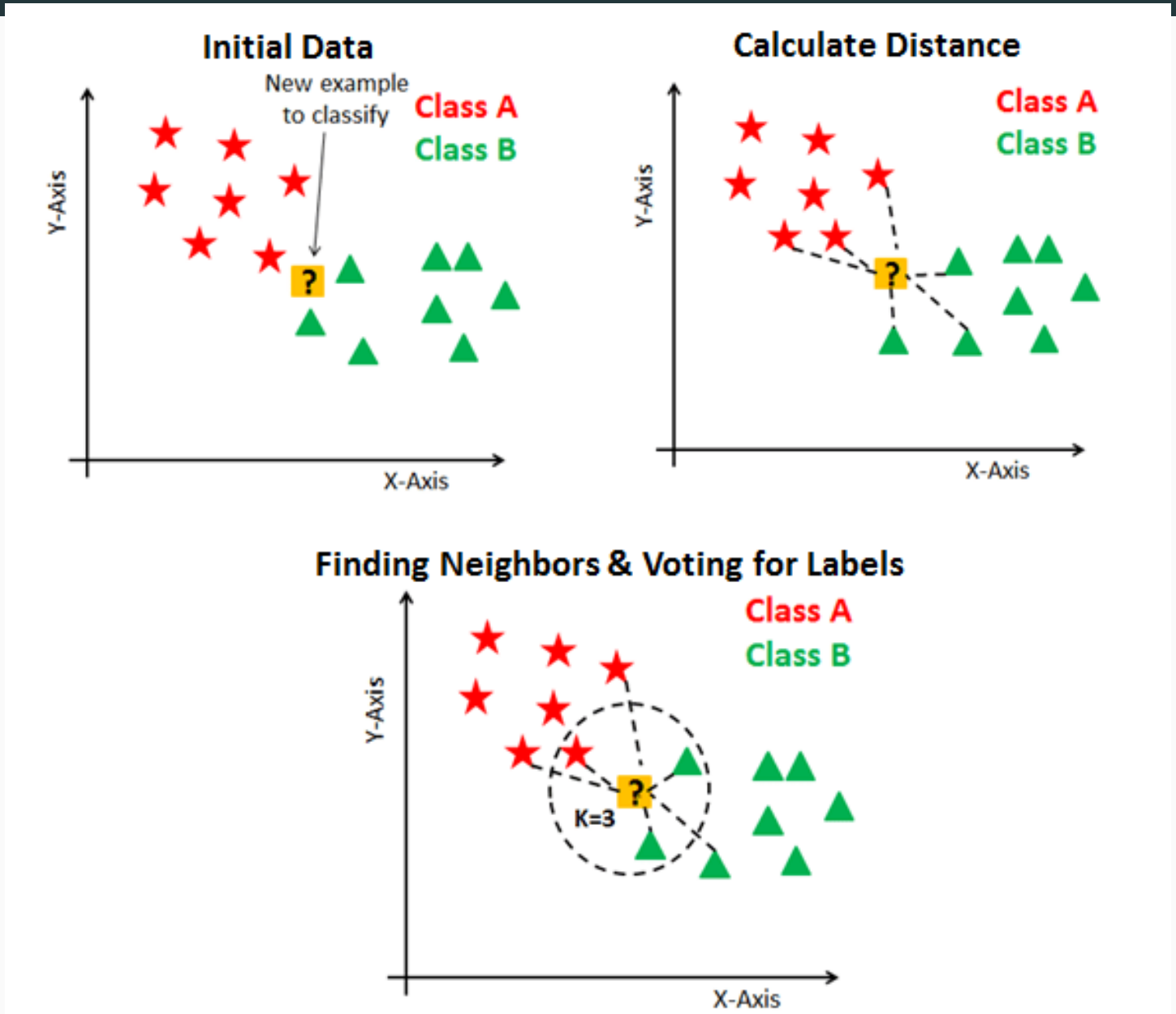
- Salida deseada, $f(y)$, es un número real
- Su estimación, $\tilde{f}(y)$, se obtiene como la **media** de las salidas de los $k-$ejemplos más similares

Clasificación k vecinos más próximos (k-NN) (II)

Extensión: k-vecinos ponderado

- Dar mayor relevancia a los vecinos más cercanos al nuevo ejemplo
- Se asocia un peso al "voto" de cada ejemplo
- Normalmente se ponderá la contribución de cada vecino en base a su distancia al ejemplo
  - mayor peso a los ejemplos más cercanos
  - uso de la inversa del cuadrado de la distancia ($w_i = \frac{1}{d(y,x_i)^2}$)
- Métodos globales: usan *todos* los ejemplos de entrenamiento
- Métodos locales: usan sólo los k más cercanos

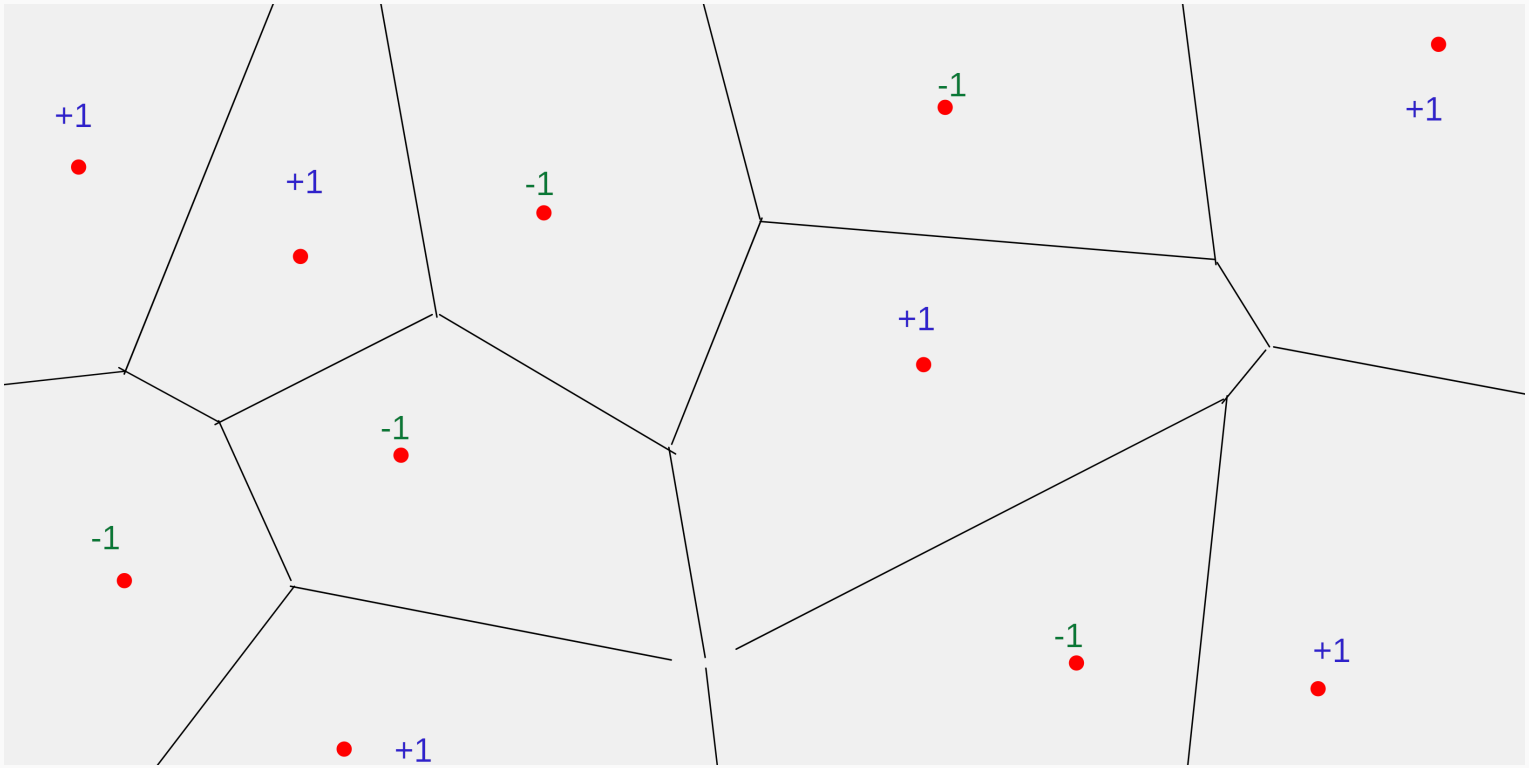
Clasificación k vecinos más próximos (k-NN) (III)



https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Clasificación k vecinos más próximos (k-NN) (IV)

Diagrama de Voronoi: regiones de decisión para K=1



Ventajas e inconvenientes k-NN

Ventajas

- Entrenamiento rápido (sólo memorizar ejemplos (todos o parte))
- Capacidad para aprender hipótesis muy complejas (superficies de decisioón complejas)
- Tolerancia al ruido en los ejemplos de entrenamiento

Inconvenientes:

- Lentitud en clasificacioón
- Muy sensible a la presencia de atributos irrelevantes

k-NN uútil si:

- ejemplos son "mapeables" como puntos en $\mathbb{R}$
- num. no demasiado elevado de atributos
- grandes cantidades de ejemplos de entrenamiento disponibles

Mejoras k-NN

Selección de atributos relevantes

- Métricas estándar asignan misma importancia a cada atributo de los ejemplos (problema de la dimensionalidad)
- Con pocos atributos realmente relevantes $\Rightarrow$ atributos irrelevantes pueden desviar la la clasificación
- **Conclusión:** conveniencia de métodos de *feature selection* y/o ponderación de atributos

Mejoras k-NN (II)

Selección de ejemplos a almacenar

- Almacenar sólo un subconjunto de los ejemplos de entrenamiento disponibles
  - Reducción coste computacional durante fase de clasificación
  - Posiblilidad de aumentar la capacidad de abstracción
- Modelo de *mejores ejemplos*
  - Asumen que existen "prototipos" para cada clase (representado por un conjunto de ejemplo de esa clase)
  - **Idea base:** almacenar sólo los ejemplos prototipo de cada clase
- Modelo de *selección de ejemplos*
  - No asume la existencia de prototipos
  - Almacena sólo los ejemplos "potencialmente útiles"
  - Aprendizaje incorpora un proceso incremental de selección de ejemplos
    - **métodos de crecimiento:** añade ejemplos cuyos k-vecinos no los predicen correctamente
    - **métodos de decrecimiento:** elimina ejemplos, manteniendo sólo los que son mal clasificados por sus vecinos

Razonamiento Basado en Casos

Case Based Reasoning (CBR)

- Aproximación alternativa para el desarrollo de sistemas inteligentes
- Funcionamiento similar al del aprendizaje por analogía, utilizando ejemplos (casos) basados en descripciones simbólicas complejas
- **Idea básica**

  Un sistema CBR resuelve problemas nuevos mediante la adaptación de soluciones previas utilizadas en la resolución de problemas similares

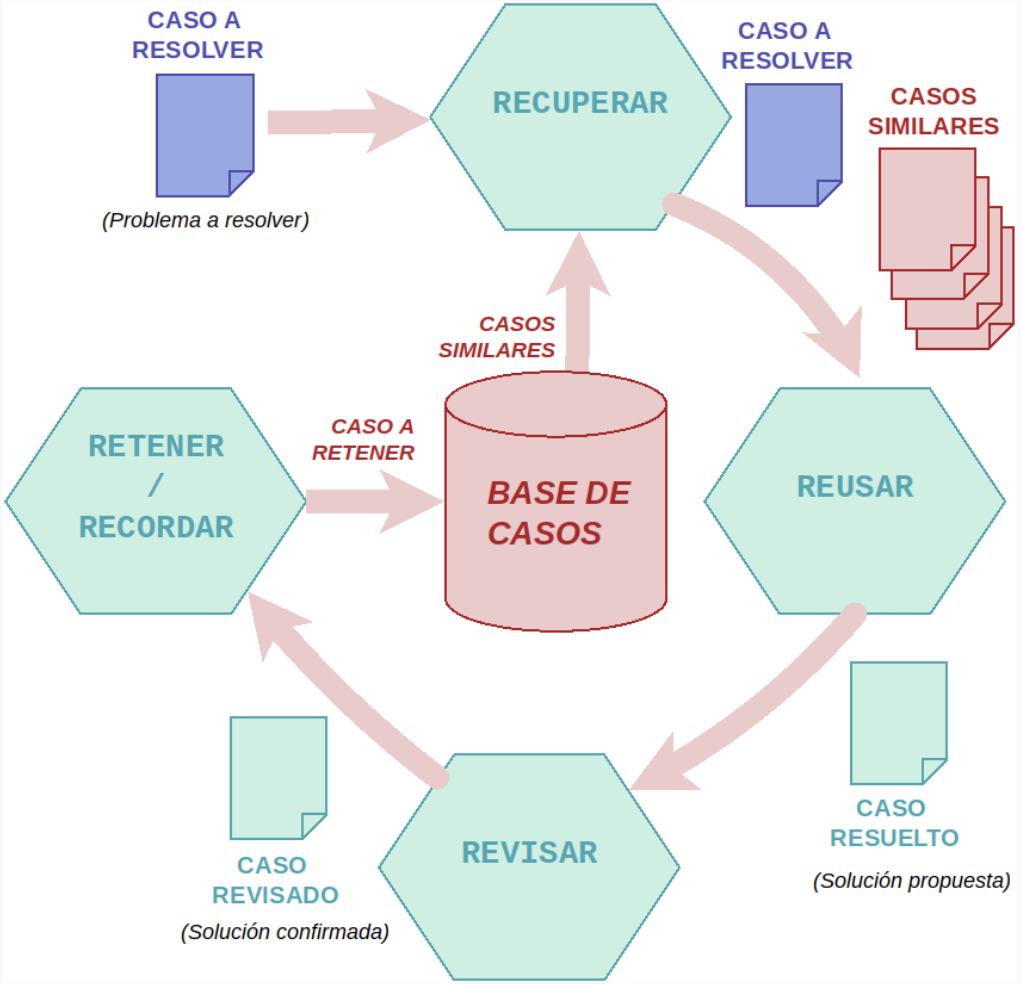
Funcionamiento

- Casos = "Piezas" de conocimiento reutilizable aplicable en determinados contextos
- Representación de casos:
  1. Descripción de la situación/problema
  2. Descripción de la solución
  3. Resultado (éxito o fracaso en el problema real)

Facilidad para incorporar nuevo conocimiento

Razonamiento Basado en Casos (II)

Ciclo sistemas CBR (etapas)



Support Vector Machines (SMV)

Support Vector Machines (SMV)

Ver documento `ml_svm.pdf`

Ensemble learning

Ensemble learning

Objetivo: Mejorar rendimiento del aprendizaje combinando varios modelos/hipótesis

- Espacio de hipótesis/modelos suele ser muy amplio
- Rendimiento de las hipótesis generadas para unos mismos datos puede ser muy dispar

Idea base: Generar + combinar varias hipótesis para un mismo dataset suele

1. tener mejor rendimiento que hipótesis individuales
2. reducir problemas de ruido y sobreadaptación

Técnicas generales (metaalgoritmos)

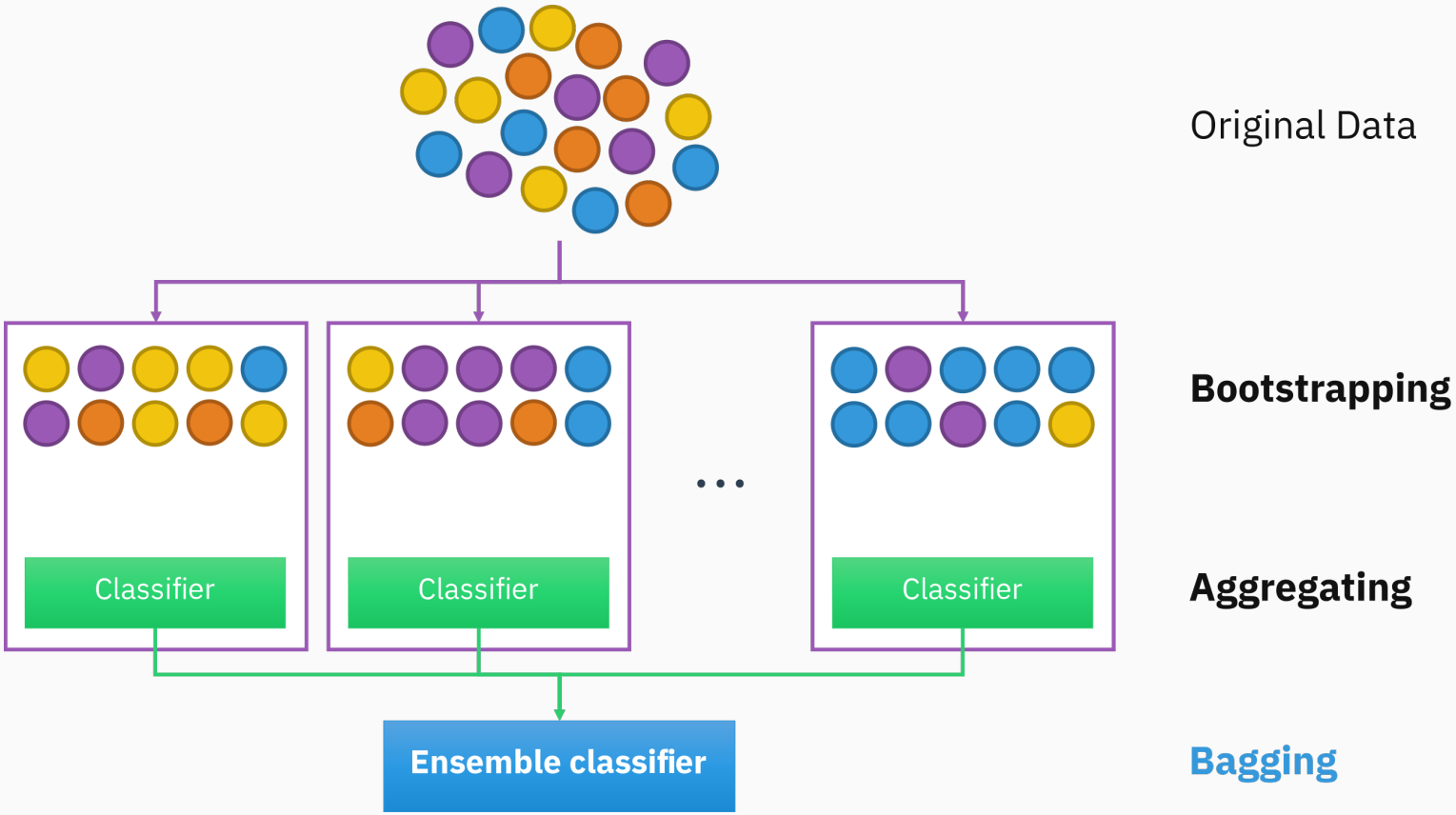
- Bagging
- Boosting
- Stacking

`https://scikit-learn.org/stable/modules/ensemble.html`

Bagging (Bootstrap aggregating)

1. Particiona el conj. de entrenamiento D, de tamaño n, en m nuevo conjuntos, D_i, cada uno de tamaño $n' \leq n$
   - Particiones aleatorias con reemplazo (permite repetición de ejemplos)
2. Con cada nuevo conjunto D_j se entrena una nueva hipótesis
   - Resultado: m hipótesis diferentes entre sí)
3. La salida final para un ejemplo nuevo será la combinación de las m predicciones de esas hipótesis
   - En clasificación: votación simple entre las m clases de salida
   - En regresión: media aritmética de los m valores de salida

Bagging (Bootstrap aggregating) (II)



Fuente: `https://en.wikipedia.org/wiki/Bootstrap_aggregating`

Boosting

Trabaja con conj. de entrenamiento completo, variando el "peso" asignado a ciertos ejemplos para generar secuencialmente m datasets "nuevos"

Objetivo: crear iterativamente una secuencia de m "clasificadores simples" (*weak classifiers*) cuyas salidas se combinarán

En cada iteración:

1. Entrenar una hipótesis con los datos de entrenamiento actuales, D_i
2. Calcular el error de predicción de esa hipótesis sobre ese conj. de entrenamiento (% ejemplos mal clasificados)
3. Generar el dataset de la siguiente iteración, D_i, incrementando el "peso" de los ejemplos mal clasificados de forma proporcional al error cometido al predecir D_i

Boosting (II)

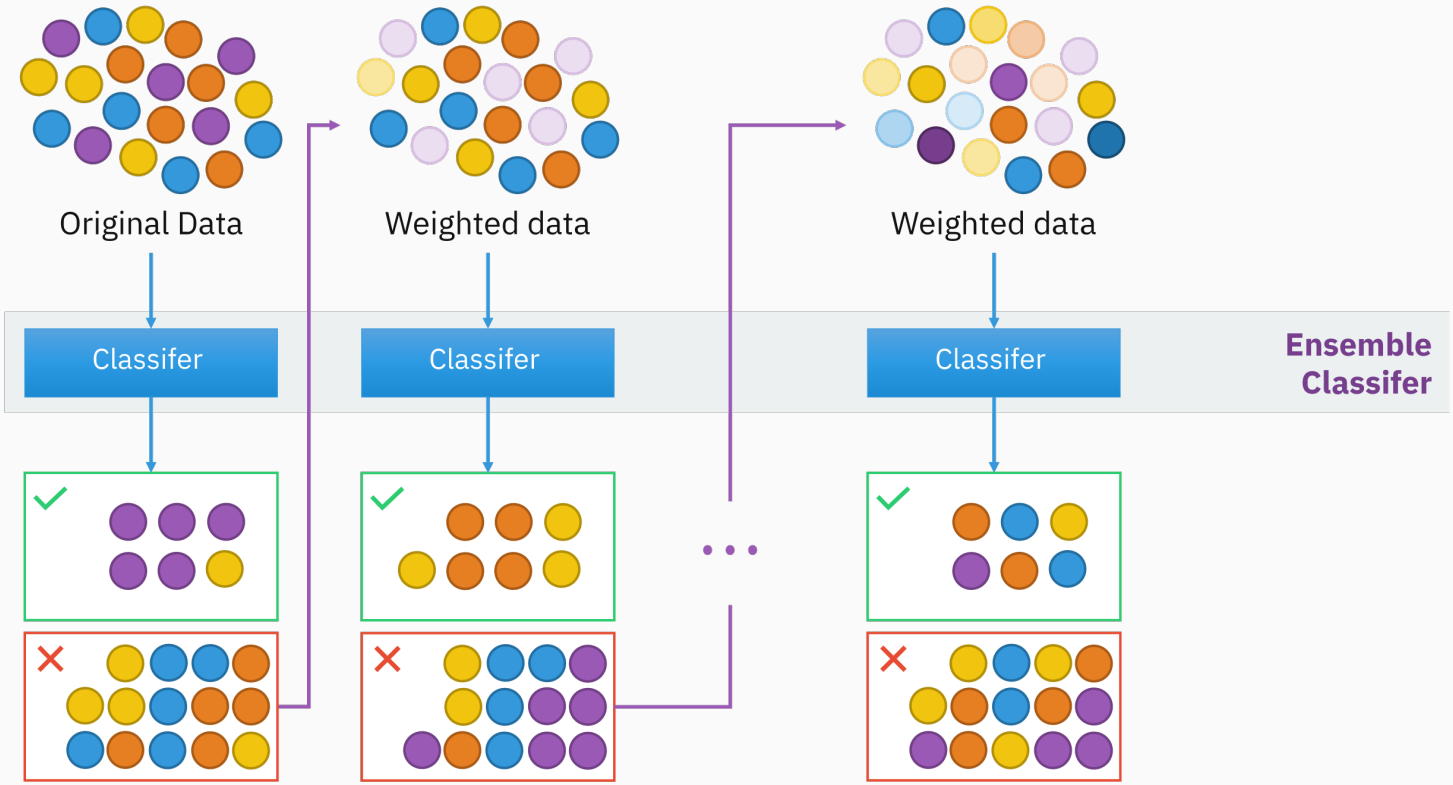
La salida final para un ejemplo nuevo será la combinación ponderada de las m predicciones de esas hipótesis

- En clasificación: votación ponderada entre las m clases de salida
- En regresión: media ponderada de los m valores de salida
- El peso de la predicción de cada una de las m hipotesis es proporcional al error cometido en su correspondiente D_i

Cada iteración aprende una hipótesis "especializada" en corregir los errores de a iteración anterior

Ejemplo: Algoritmo AdaBoost (*Adaptive Boosting*)

Boosting (III)



Fuente: `https://en.wikipedia.org/wiki/Boosting_(machine_learning)`

Stacking (apilado de clasificadores)

Generalización de la aproximación usada en Bagging (también posible con Boosting)

- Intenta mejorar la etapa final de combinación de predicciones de los m clasificadores
- Usa un clasificador final en lugar de un esquema de votación

Pasos

1. Se particiona el conjunto de entrenamiento, D, en m datasets y con cada uno se entrena un clasificador
2. Se crea un dataset con las predicciones de los m clasificadores para el conjunto de entrenamiento original, D
3. Se entrena un clasificador adicional que toma como entrada esas m predicciones y aprende a predecir la clase final

Clasificador final puede a su vez implementarse como una capa adiconal de *stacking*

Random forests

Método *ensemble learning* específico de árboles de decisión

Esquema similar a *bagging*

- Construcción de m árboles de decisión independientes usando m variantes del conj. de entrenamiento
- Opcionalmente, cada una de esas m variantes del conjunto de entrenamiento puede usar un subconjunto específico de los atributos presentes en el dataset original
- Predicción final de nuevo ejemplos mediante votación entre las predicciones de los m árboles de decisión construidos

Mejora rendimiento, pero se pierde interpretabilidad del clasificador

`https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees`