DIPLOMADO EN BIG DATA PARA LA TOMA DE DECISIONES

Introducción al Modelamiento Estadístico

Parte 2: Métodos de Clasificación

2.1.- Introducción

Introducción

- Las técnicas de clasificación son formas de análisis de datos que pretenden generar modelos que describen un atributo discreto de interés.

- Estos modelos reciben el nombre de *clasificadores* y "predicen" clases discretas no ordenables.

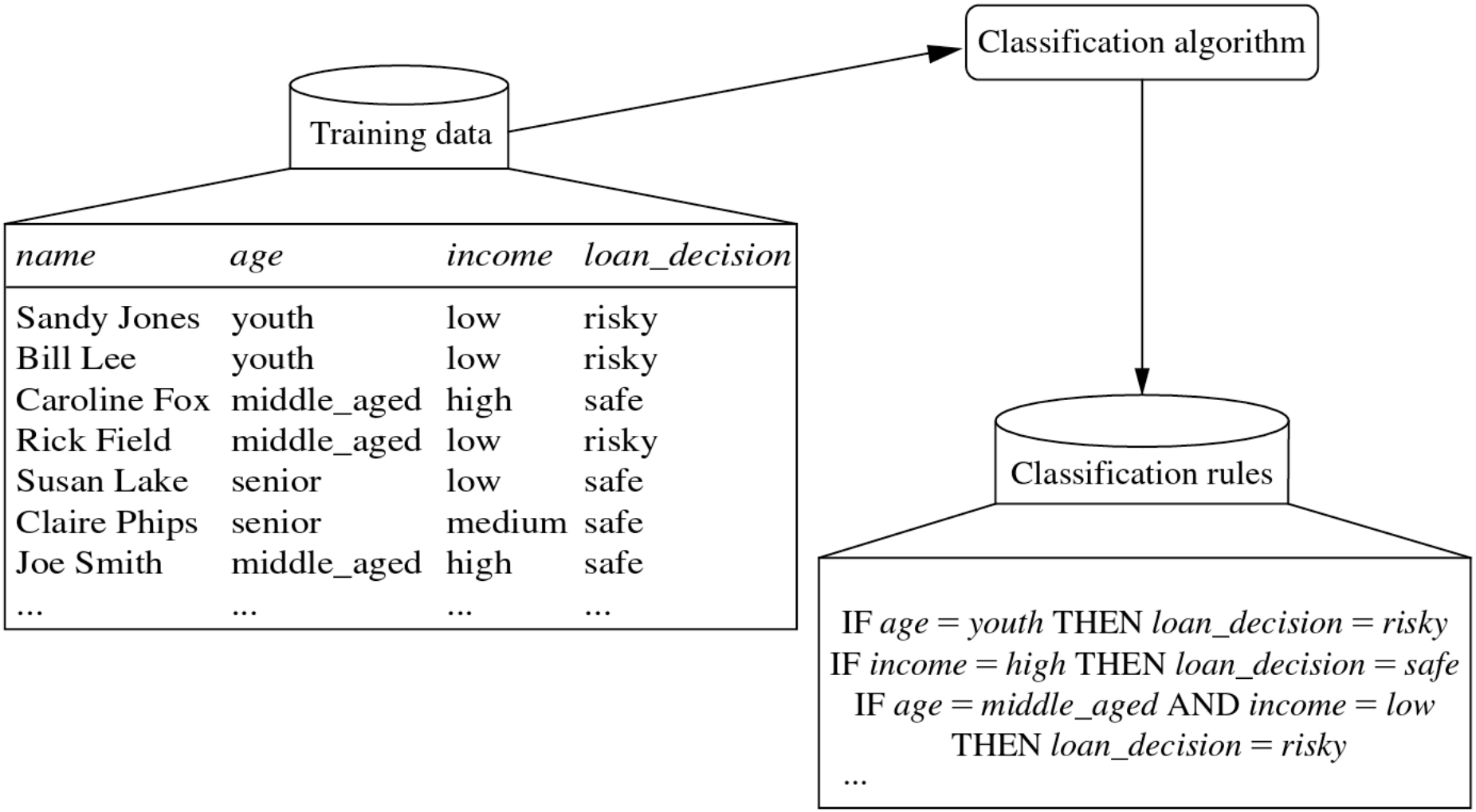
- Ejemplos de aplicaciones:
  - categorización de textos (ej., spam).
  - detección de fraudes (ej. firmas).
  - visión de máquinas (ej., detección de rostros).
  - segmentación de mercado (ej., tipos de clientes que responden a una promoción).
  - bioinformática (ej., proteinas de acuerdo a su función).

Introducción

El proceso de clasificación consta de dos pasos:

1) Paso de aprendizaje: se construye el modelo basado en datos recopilados previamente (trainning data).

Introdución



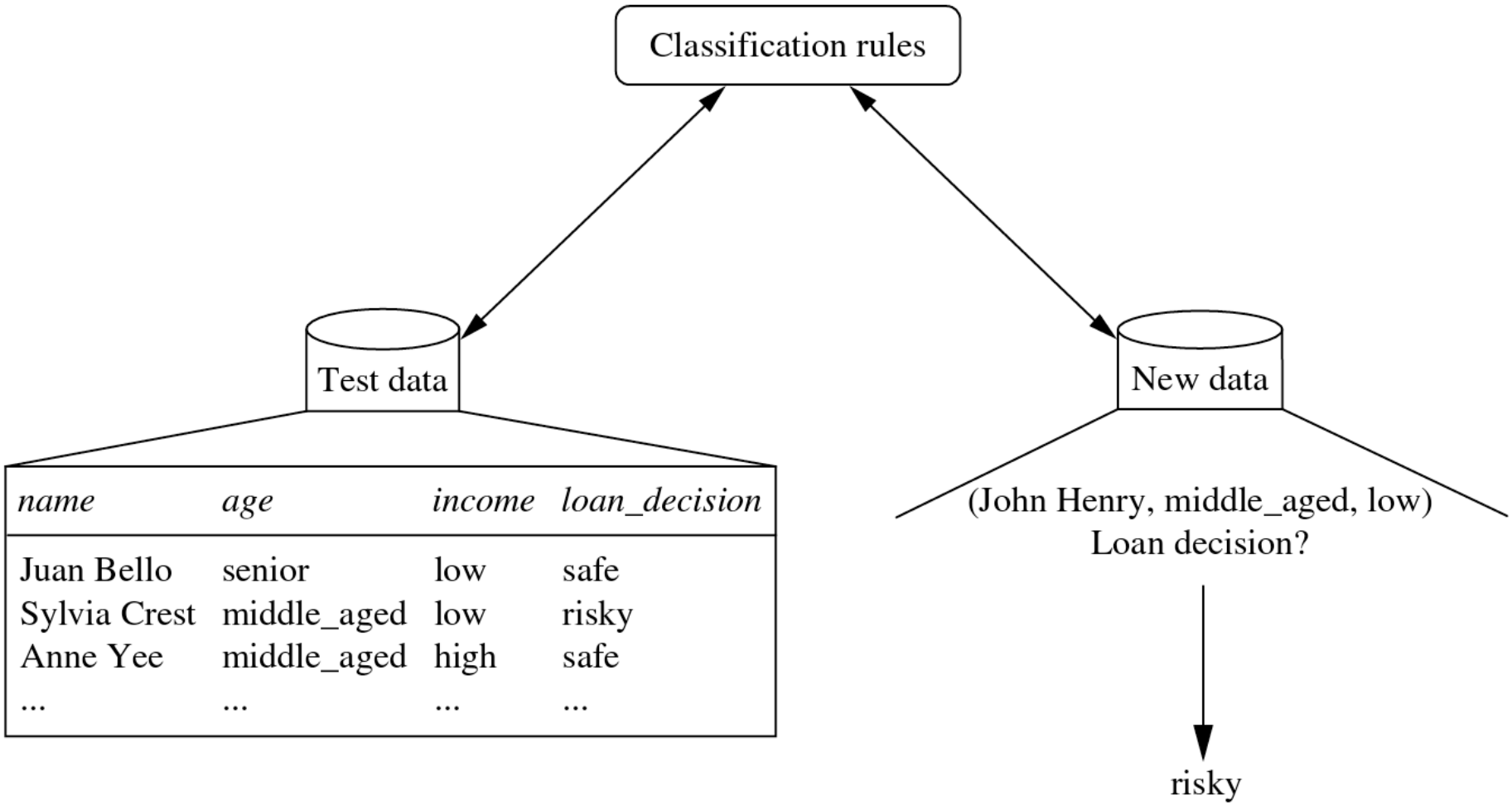
name	*age*	*income*	*loan_decision*
Sandy Jones	youth	low	risky
Bill Lee	youth	low	risky
Caroline Fox	middle_aged	high	safe
Rick Field	middle_aged	low	risky
Susan Lake	senior	low	safe
Claire Phips	senior	medium	safe
Joe Smith	middle_aged	high	safe
...	...	...	...

Introducción

El proceso de clasificación consta de dos pasos:

2) Paso de validación y clasificación: se determina la precisión del clasificador, y si es aceptable, se usa para predecir las clases de un conjunto de datos.

Introdución



name	*age*	*income*	*loan_decision*
Juan Bello	senior	low	safe
Sylvia Crest	middle_aged	low	risky
Anne Yee	middle_aged	high	safe
...	...	...	...

2.2.- MÉTODOS

2.2.1.- Árboles de Decisión

Árboles de Decisión: Definición

Definición

Un árbol de decisión es un diagrama de flujo, donde:

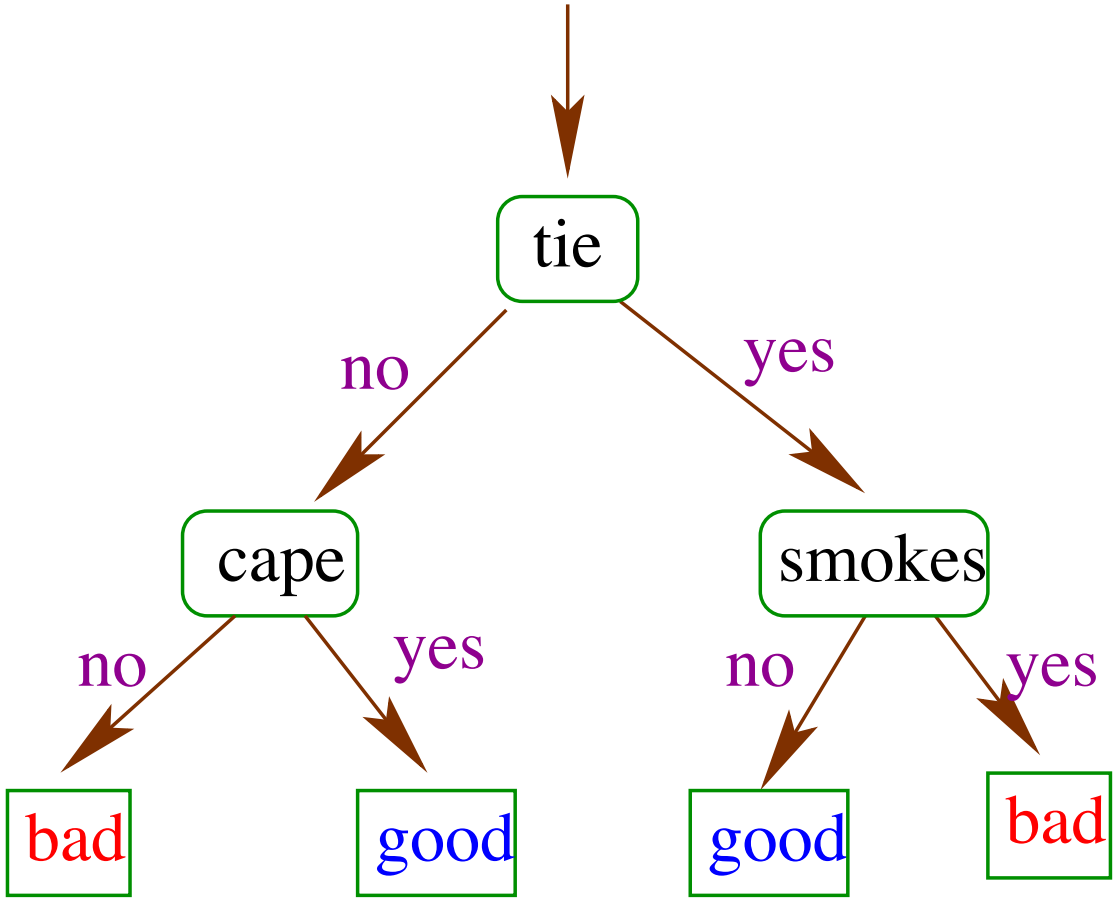
- *cada nodo interno representa una "pregunta" sobre un atributo,*
- *cada rama representa una respuesta a esa pregunta y*
- *cada hoja o nodo terminal representa la clase de clasificación.*

Árboles de Decisión: Ejemplo

Identificación de personas de acuerdo a su apariencia.

	sex	mask	cape	tie	ears	smokes	class
			training data				
batman	male	yes	yes	no	yes	no	Good
robin	male	yes	yes	no	no	no	Good
alfred	male	no	no	yes	no	no	Good
penguin	male	no	no	yes	no	yes	Bad
catwoman	female	yes	no	no	yes	no	Bad
joker	male	no	no	no	no	no	Bad
			test data				
batgirl	female	yes	yes	no	yes	no	??
riddler	male	yes	no	no	no	no	??

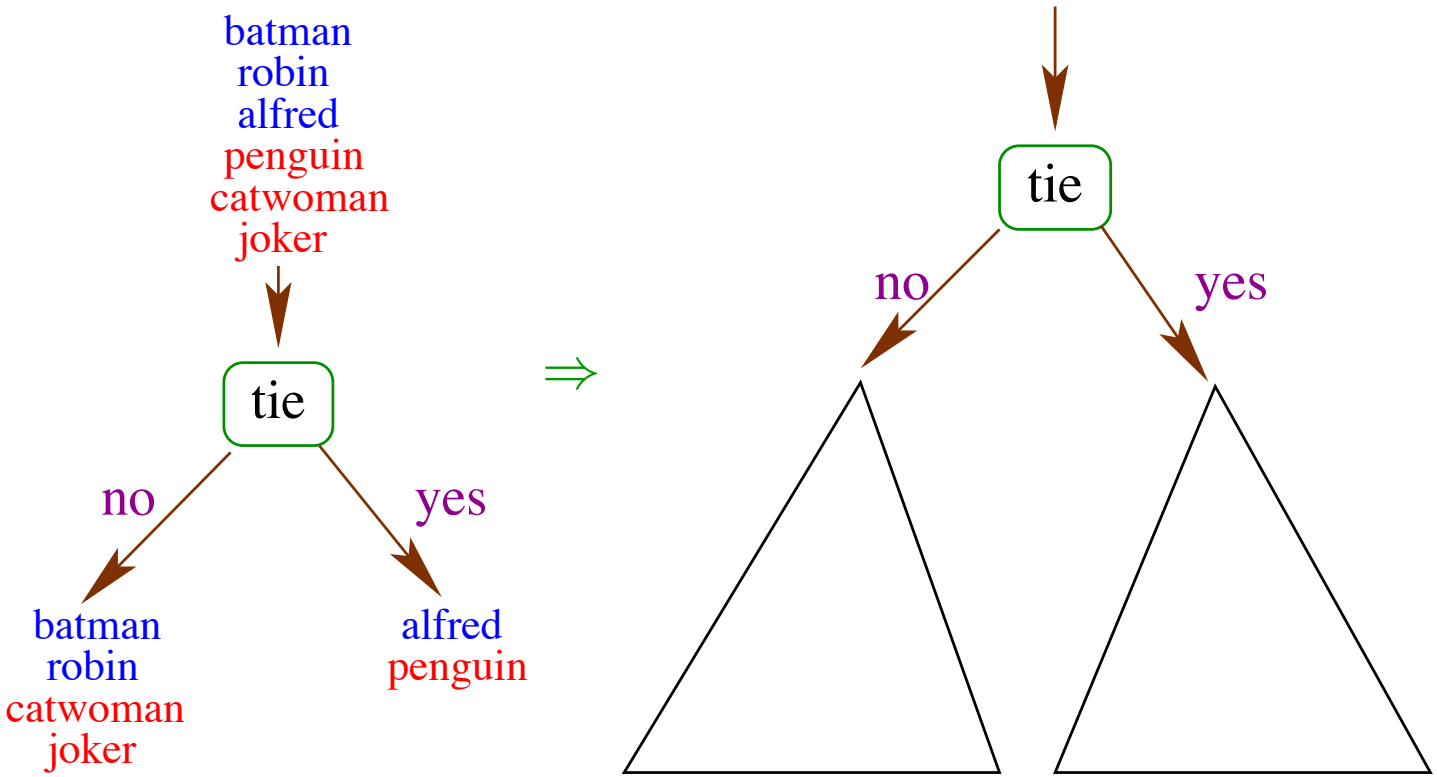
Árboles de Decisión: Ejemplo



Árboles de Decisión: Algoritmo Básico

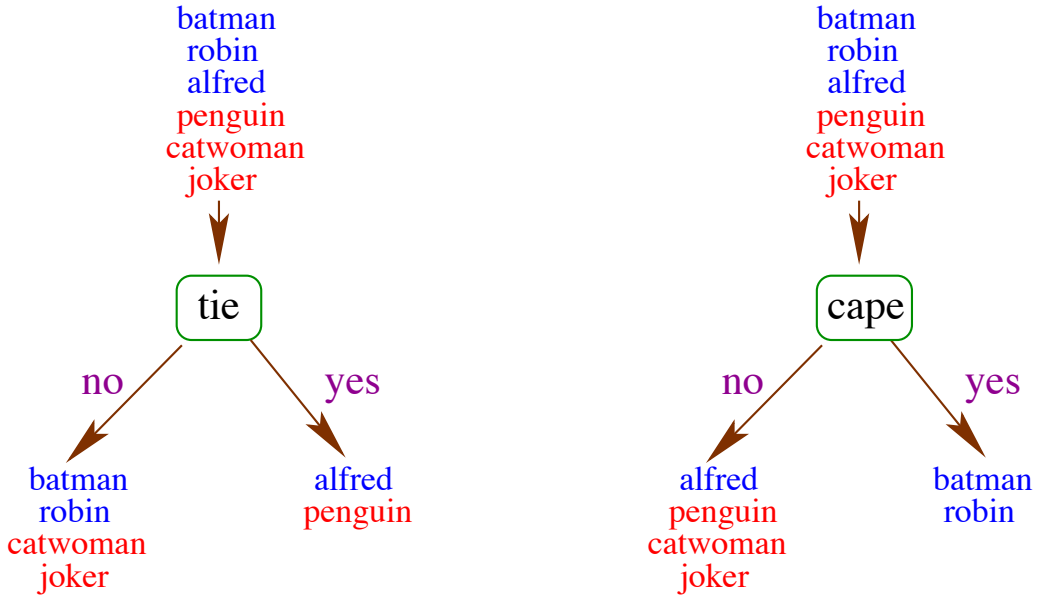
- ¿Cómo se construyen los árboles de decisión?
  - Elección de la regla en base a la cuál particionar los datos.
  - Dividir los datos en subconjuntos disjuntos utilizando la regla de partición.
  - Repetir en forma recursiva para cada subconjunto.
  - Parar cuando las hojas son (casi) “puras”.

Árboles de Decisión: Algoritmo Básico



Árboles de Decisión: Algoritmo Básico

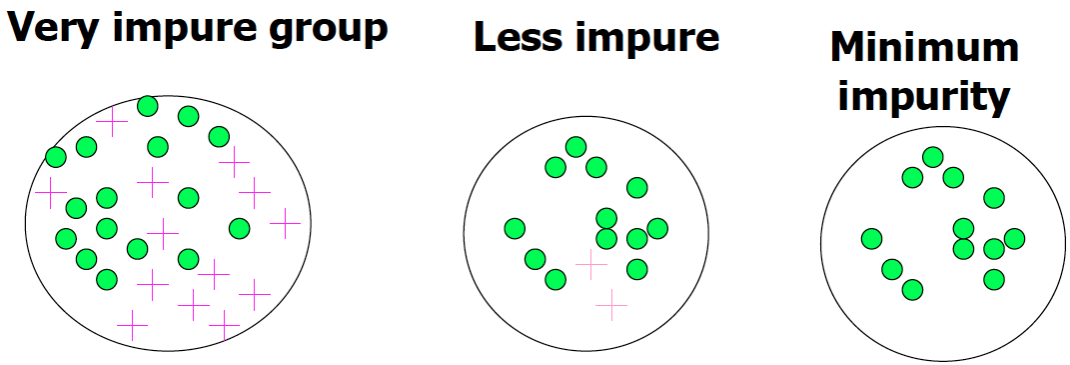
- ¿Cómo se escoge la mejor regla de partición?



Árboles de Decisión: Algoritmo Básico

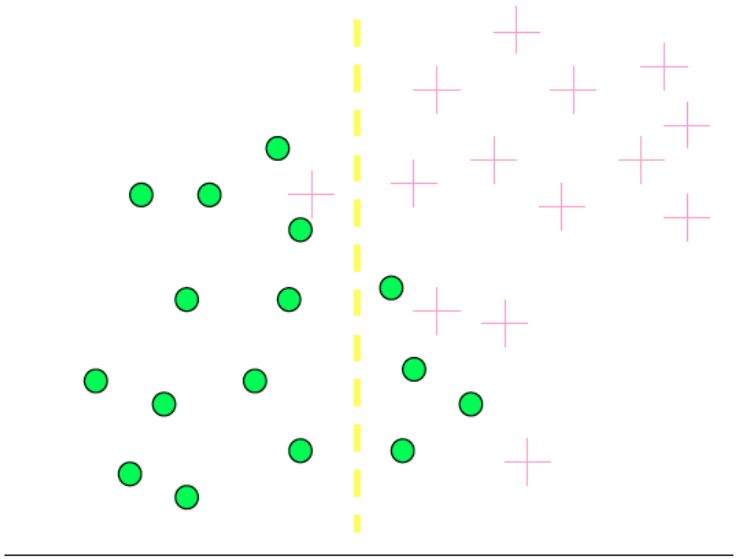
- Una opción es escoger la regla que permita el mayor aumento en la “pureza” de los grupos.

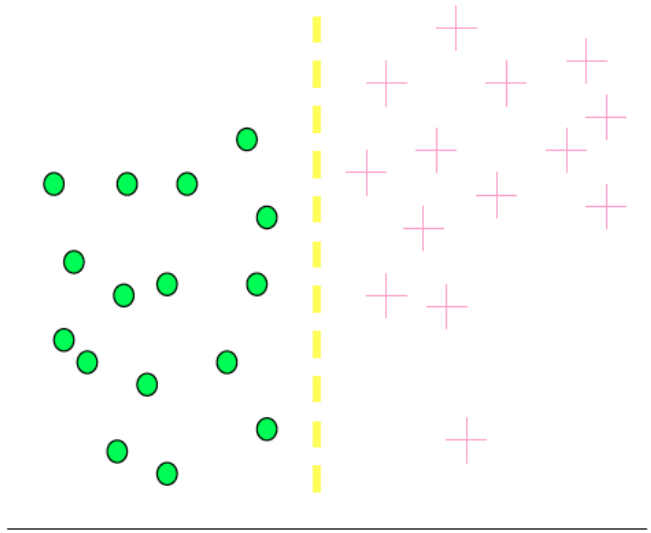
- Entonces necesitamos una medida que cuantifique el nivel de ‘impureza” en un grupo:



Árboles de Decisión: Algoritmo Básico

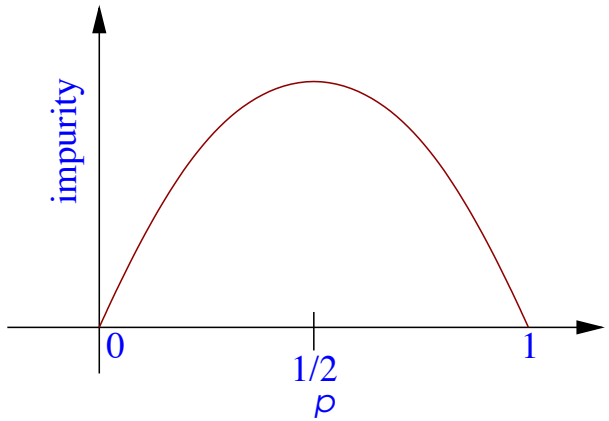
¿Qué atributo es más informativo?





Árboles de Decisión: Algoritmo Básico

- Nos gustaría que la función que cuantifique el nivel de impureza se vea así (p = fracción de casos positivos):



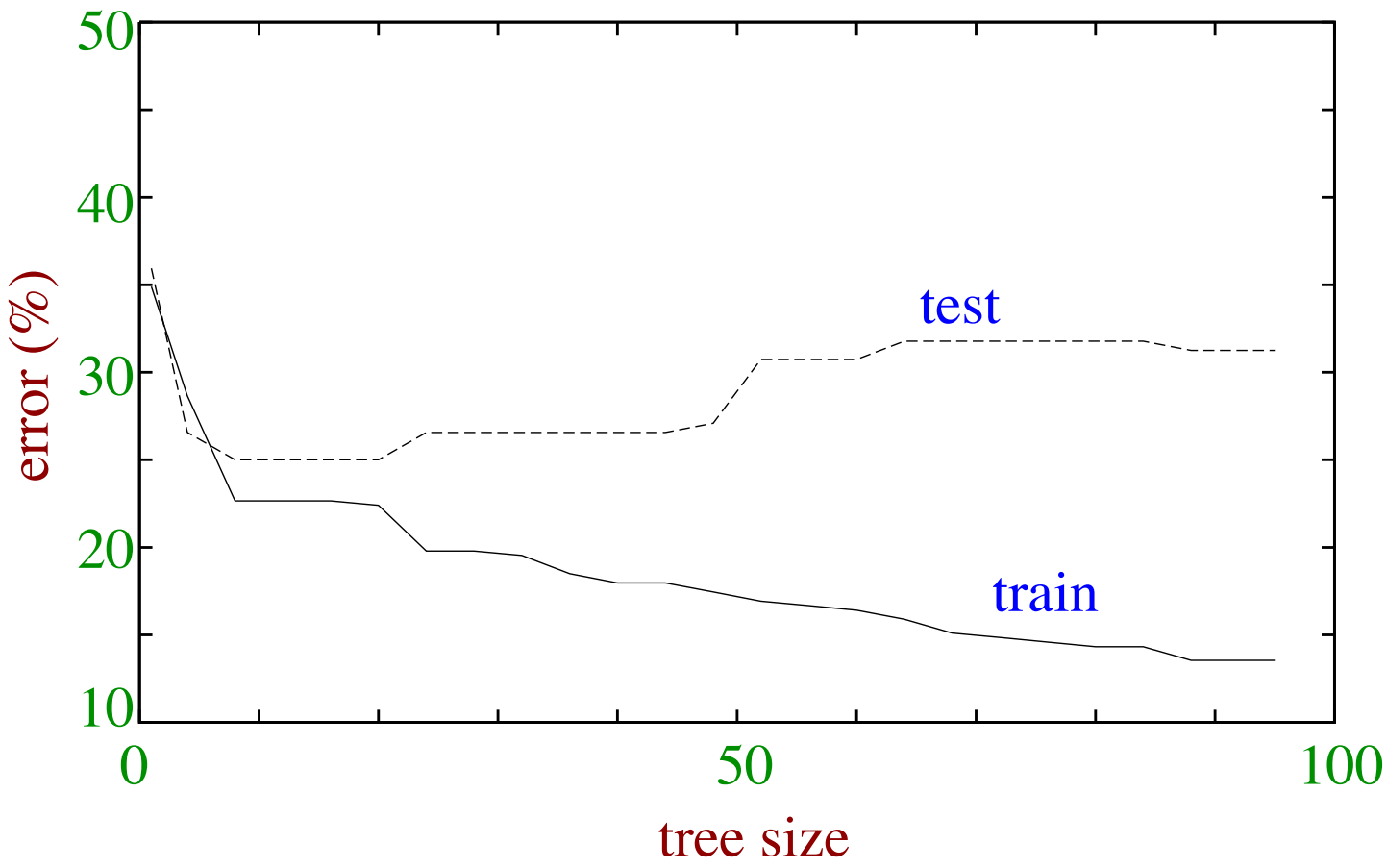
- Medidas de impureza utilizadas usualmente:
  - Entropia: $-p\log(p) - (1-p)\log(1-p)$.
  - Índice de Gini: $p(1-p)$.

Árboles de Decisión: Tipos de Errores y Poda

- **Error de entrenamiento:** proporción de los datos de entrenamiento incorrectamente clasificados.
- **Error de prueba:** proporción de los datos de prueba clasificados de forma incorrecta.

Árboles de Decisión: Tipos de Errores y Poda

- Tamaño del árbol vs tipo de error:



Árboles de Decisión: Tipos de Errores y Poda

- Los árboles deben se suficientemente grandes para ajustar los datos de entrenamiento.
- Sin embargo, árboles muy “complejos” pueden “sobreajustar” los datos de entrenamiento (capturando ruidos en los datos o patrones espurios) y clasificar mal registros futuros.

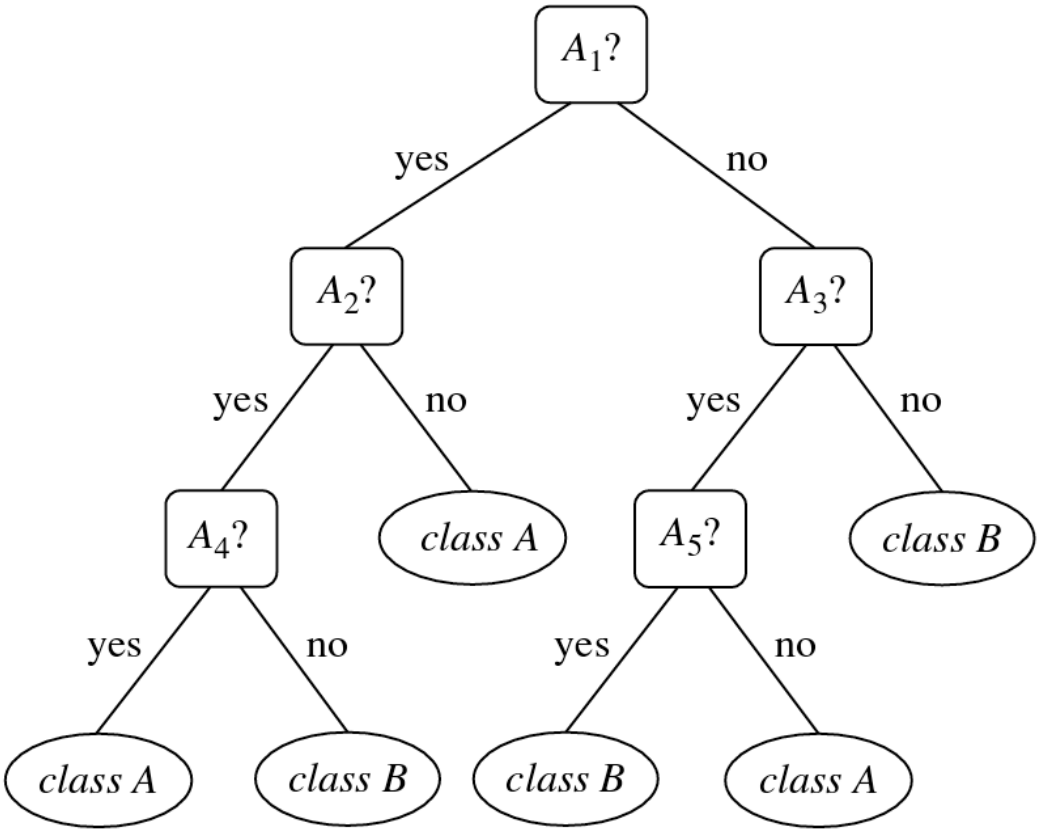
Árboles de Decisión: Tipos de Errores y Poda

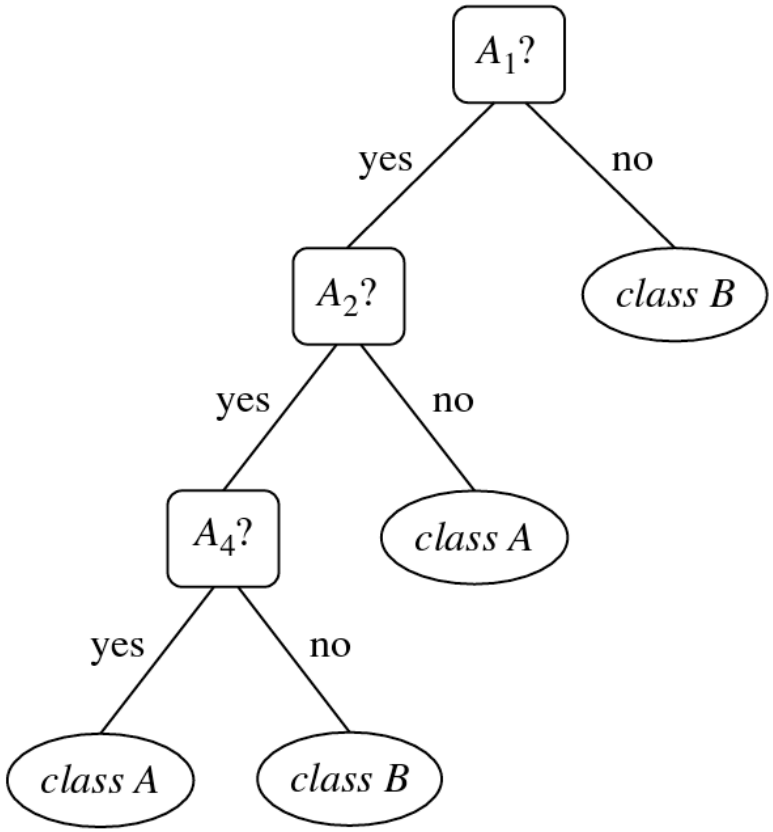
- Los métodos de poda, pretenden solucionar el problema de sobreajuste.
- Un árbol podado tiende a ser más pequeño y menos complejo, por lo que es más comprensible.
- Existen 2 métodos de poda: pre-poda (controlan crecimiento del árbol antes de ajuste perfectamente los datos) y post-poda (permite ajuste perfecto y luego se poda).
- En la práctica se usan más las técnicas de post-poda, porque es difcil estimar con precisión cuando se debe detener el crecimiento del árbol.

Árboles de Decisión: Tipos de Errores y Poda

- Los métodos de poda, pretenden solucionar el problema de sobreajuste.
- Un árbol podado tiende a ser más pequeño y menos complejo, por lo que es más comprensible.
- Existen 2 métodos de poda: pre-poda (controlan crecimiento del árbol antes de ajuste perfectamente los datos) y post-poda (permite ajuste perfecto y luego se poda).
- En la práctica se usan más las técnicas de post-poda, porque es difícil estimar con precisión cuando se debe detener el crecimiento del árbol.

Árboles de Decisión: Tipos de Errores y Poda





Árboles de Decisión: Ejemplo con Datos

- Se cuenta con información de 146 pacientes con cáncer de próstata en estado C.
- Médicamente es interesante estudiar si la enfermedad reaparece luego de la cirugía de remoción de la próstata y el intervalo de tiempo hasta que eso ocurra (si es que ocurre).
- Los datos se encuentran bajo el nombre de `stagec` de la librería `rpart`.
- Se aplican las funciones `rpart`, `printcp` y `prune` de la librería `rpart`.
- Lectura complementaria para detalles en archivo adjunto (ComplementaryReading2.pdf).

Árboles de Decisión: Código R

```
library(rpart)
data(stagec)
head(stagec)
names(stagec)

progstat <- factor(stagec$pgstat, levels=0:1,
                                  labels=c("No", "Prog"))

cfit <- rpart(progstat ~ age + eet + g2 + grade +
                    gleason + ploidy,
                    data=stagec, method='class')
print(cfit)
plot(cfit)
text(cfit)
printcp(cfit)
```

Árboles de Decisión: Código R

```
pruned.tree <- prune(cfit,cp=0.06)

plot(pruned.tree)

text(pruned.tree)
```

2.2.2.- Bayes Ingenuo

Bayes Ingenuo

- Predicen las probabilidades de pertenencia a clases.
- Están basados en el teorema de Bayes.
- Los clasificadores Bayesianos simples tienen rendimiento comparable con Árboles de Decisión.
- Han demostrado tener alta precisión y velocidad al aplicarse a grandes bases de datos.
- Asumen que el efecto del valor de un atributo en una clase dada es independiente de los valores de los otros atributos $\Longrightarrow$ `naive`.

Bayes Ingenuo: Teorema de Bayes

- Thomas Bayes: clérigo que trabajó en probabilidad y teoría de decisión durante el siglo XVIII.
- Sea $\boldsymbol{X}$ los atributos de una unidad, que en términos Bayesianos es considerada "evidencia", descrita por n atributos.
- Sea H alguna hipótesis sobre la clase de pertenencia de la unidad.
- Interesa determinar la probabilidad de H dado $\boldsymbol{X}$, $P(H \mid \boldsymbol{X})$.
- Se busca determinar la probabilidad de que la unidad $\boldsymbol{X}$ pertenezca a la clase C, dado que conocemos la descripción de $\boldsymbol{X}$.
- $P(H \mid \boldsymbol{X})$: probabilidad posterior o *a posteriori*.
- $P(H)$: probabilidad previa o *a priori*.

Bayes Ingenuo: Teorema de Bayes

- $P(H)$, $P(\boldsymbol{X} \mid H)$ y $P(\boldsymbol{X})$ pueden estimarse desde los datos.
- El teorema de Bayes nos entrega una forma de estimar la probabilidad posterior $P(H \mid \boldsymbol{X})$ en base a ellas.

Teorema de Bayes

$$P(H \mid \boldsymbol{X}) = \frac{P(\boldsymbol{X} \mid H)P(H)}{P(\boldsymbol{X})}$$

Bayes Ingenuo (naive Bayes): Algoritmo

- Suponga se quiere clasificar en una de m clases, $C_1, C_2, \ldots, C_m$, a una unidad representada por el vector de atributos

$$\boldsymbol{X} = (x_1, x_2, \ldots, x_n).$$

- El clasificador Bayesiano predice que la unidad pertenece a la clase C_i si

$$P(C_i \mid \boldsymbol{X}) > P(C_j \mid \boldsymbol{X}),$$

para $1 \leq j \leq m, j \neq i$.

- La clase C_i que maximiza $P(C_i \mid \boldsymbol{X})$ se llama máxima hipótesis posterior.

- Por el teorema de Bayes, $P(C_i \mid \boldsymbol{X}) = \frac{P(\boldsymbol{X}|C_i)P(C_i)}{P(\boldsymbol{X})}$.

Bayes Ingenuo (naive Bayes): Algoritmo

- Por el teorema de Bayes, $P(C_i \mid \boldsymbol{X}) = \frac{P(\boldsymbol{X}|C_i)P(C_i)}{P(\boldsymbol{X})} \propto P(\boldsymbol{X} \mid C_i)P(C_i)$.
- Si no se conocen las probabilidades a priori de las clases, comúnmen se asumen iguales $P(C_1) = P(C_2) = \ldots = P(C_m)$.
- Para el simplificar el cálculo de $P(\boldsymbol{X} \mid C_i)$, se toma el supuesto "ingenuo" de independencia condicional dentro de las clases.
- Los atributos son condicionalmente independientes entre sí, dada la clase de la unidad, $P(\boldsymbol{X} \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$.

Bayes Ingenuo (naive Bayes): Algoritmo

- Para atributos discretos $P(x_k \mid C_i)$ es estimada por a proporción de unidades de la clase C_i que toman el valor x_k para ese atributo en la muestra de entrenamiento.

- Para atributos continuos se asume una distribución normal y se calcula $P(x_k \mid C_i) = \phi(x_k, \mu_{C_i}, \sigma^2_{C_i})$, donde $\phi(\cdot a, b)$ es la densidad de la distribución normal con media a y varianza b, y μ_{C_i} y $\sigma^2_{C_i}$ corresponde a la media y varianza, respectivamente, del atributo para las unidades de la clase C_i en la muestra de entrenamiento.

- La corrección de Laplace se utliza cuando existen ceros.

Bayes Ingenuo (naive Bayes): Ejemplo

- El conjunto de datos `HouseVotes84` incluidos en la libería `mlbench` contiene los registros de votación del Congreso de Estados Unidos en el año 1984.
- El conjunto de datos tiene la votación de 435 representantes para 16 temas claves discutidos en el congreso. El conjunto de datos también incluye la afiliación política del representante (demócrata o republicano).
- El objetivo del análisis es predecir la afiliación política (`Class`) del representante, en base a sus votaciones.
- Se usa función `naiveBayes` de la librería `e1071`.

Bayes Ingenuo: Ejemplo

```
library(e1071)
data(HouseVotes84,package="mlbench")
names(HouseVotes84)
head(HouseVotes84)

model <- naiveBayes(Class~.,data=HouseVotes84)
predict(model, HouseVotes84[1:10,])
predict(model, HouseVotes84[1:10,],type="raw")

pred <- predict(model, HouseVotes84)
table(pred, HouseVotes84$Class)

### Usando correcci\'on de Laplace ###
model2 <- naiveBayes(Class~.,data=HouseVotes84,laplace=3)
pred2 <- predict(model2,HouseVotes84[,-1])
table(pred2, HouseVotes84$Class)
```

2.2.3.- Método de los k-Vecinos Más Cercanos (k-NN)

k-NN

Idea con un ejemplo:

Cuando se clasifica un email com spam o no, una estrategia es mirar los emails existentes que son similares al que está entrando, y ver si son spam o no. Ésta es la filosofía detrás de este algoritmo.

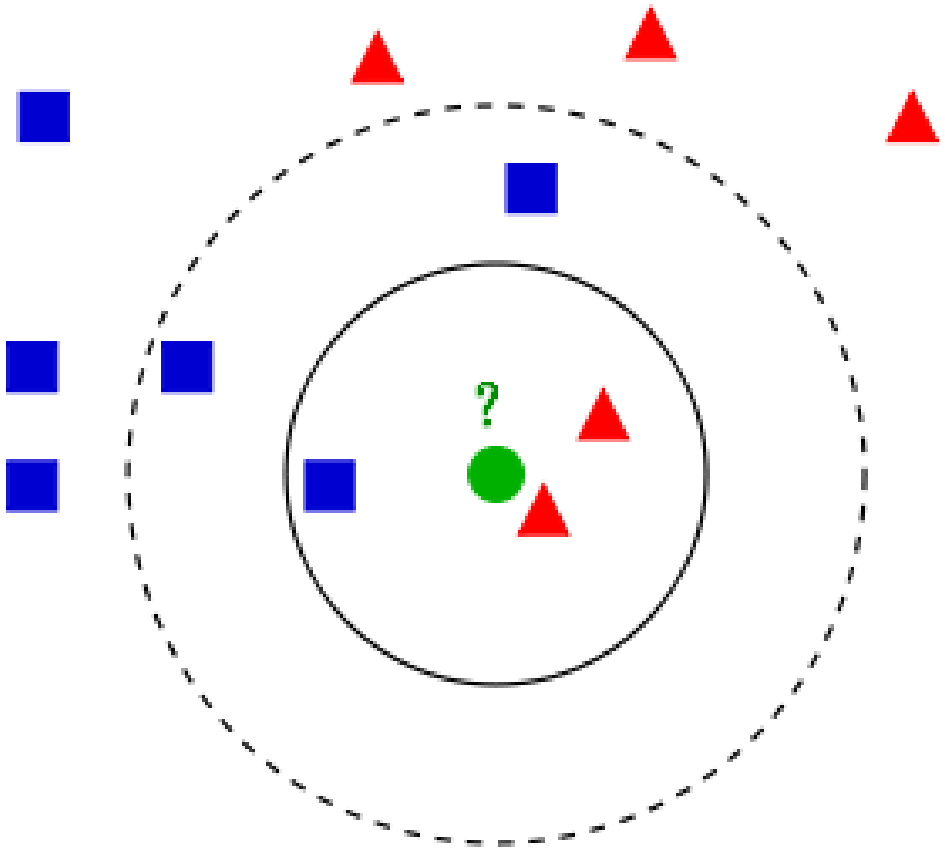
k-NN

- Fue descrito inicialmente en los años 50.
- El método es bastante intenso cuando se tienen bases de datos grandes.
- Ganó popularidad en los 60's cuando aumentó la potencia computacional.
- Es muy utilizado en el área de reconocimiento de patrones.

k-NN

- Estos clasificadores están basados en el aprendizaje por analogía, es decir, comparando una unidad de validación con unidades de entrenamiento que son similares a ella.
- Las unidades de entrenamiento son descritas por n atributos.
- Cada unidad representa un punto en el espacio n-dimensional.
- Cuando se tiene una unidad desconocida, el clasificador de k-NN busca el espacio de patrones para las k unidades de entrenamiento que están más cercanas a la unidad desconocida.
- Estas k unidades de entrenamiento son los k vecinos más cercanos de la unidad desconocida.

k-NN



k-NN: Definición de “cercanía”

- Se define en términos de una métrica, como la distancia Euclidiana. Si se tienen las unidades $\boldsymbol{X}_1 = (x_{11}, x_{12}, \ldots, x_{1n})$ y $\boldsymbol{X}_2 = (x_{21}, x_{22}, \ldots, x_{2n})$,

$$dist(\boldsymbol{X}_1, \boldsymbol{X}_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}$$

- Es conveniente normalizar los valores antes de calcular las distancias para prevenir que prevalezcan atributos con valores de mayor magnitud (e.j., ingreso v/s variable binaria).

k-NN

- Los clasificadores k-NN, clasifican la unidad desconocida de acuerdo a la clase más común entre los k vecinos más cercanos a ella.
- Si $k = 1$, la unidad se clasifica de acuerdo a la clase de la unidad de entrenamiento que esté más cercana.
- Este procedimiento también puede utilizarse para realizar predicciones numéricas, caso en el que el clasificador entrega el valor promedio de los valores correspondientes de los k vecinos cercanos a la unidad de validación.

¿Qué ocurre si los atributos nominales?

En estos casos podemos utilizar medias de "distancia" apropiadas, por ejemplo, asignando un 1 si son diferentes y 0 si no.

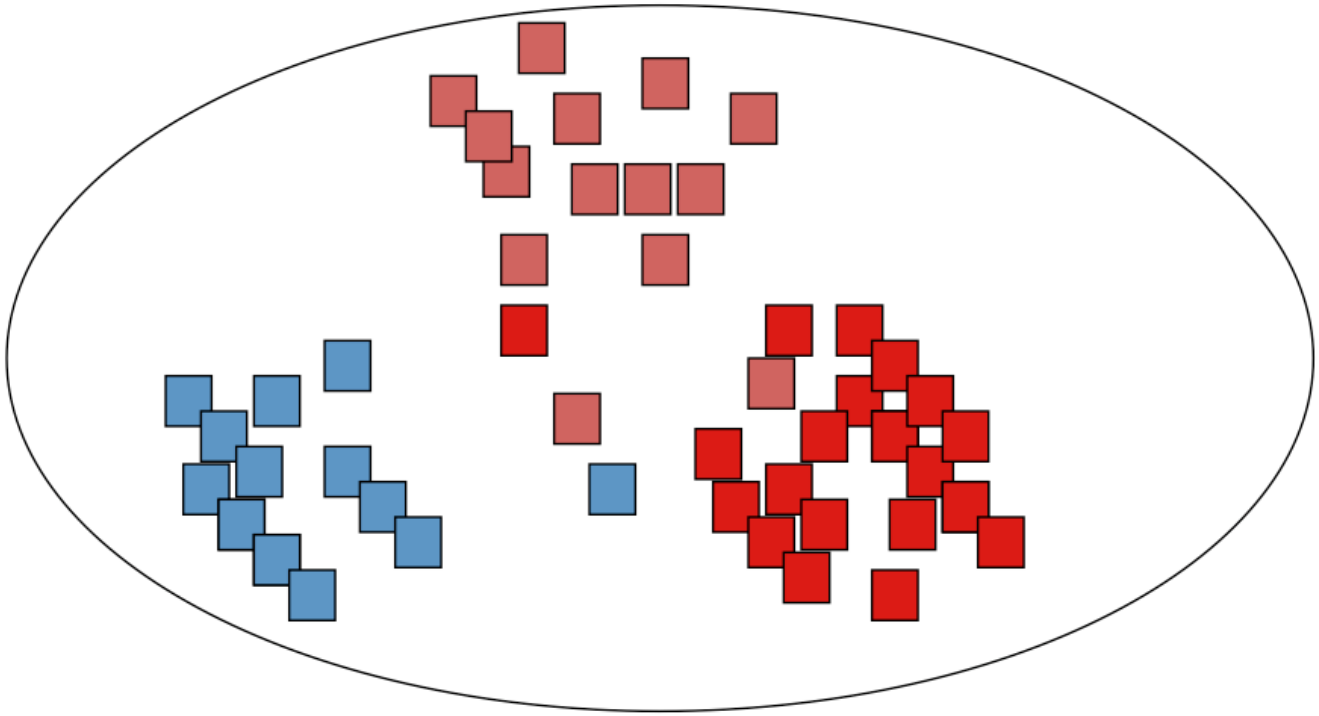
¿Qué ocurre si hay valores faltantes?

Es decir, si el valor de un atributo A falta en la unidad $\boldsymbol{X}_1$, $\boldsymbol{X}_2$ o en ambas.

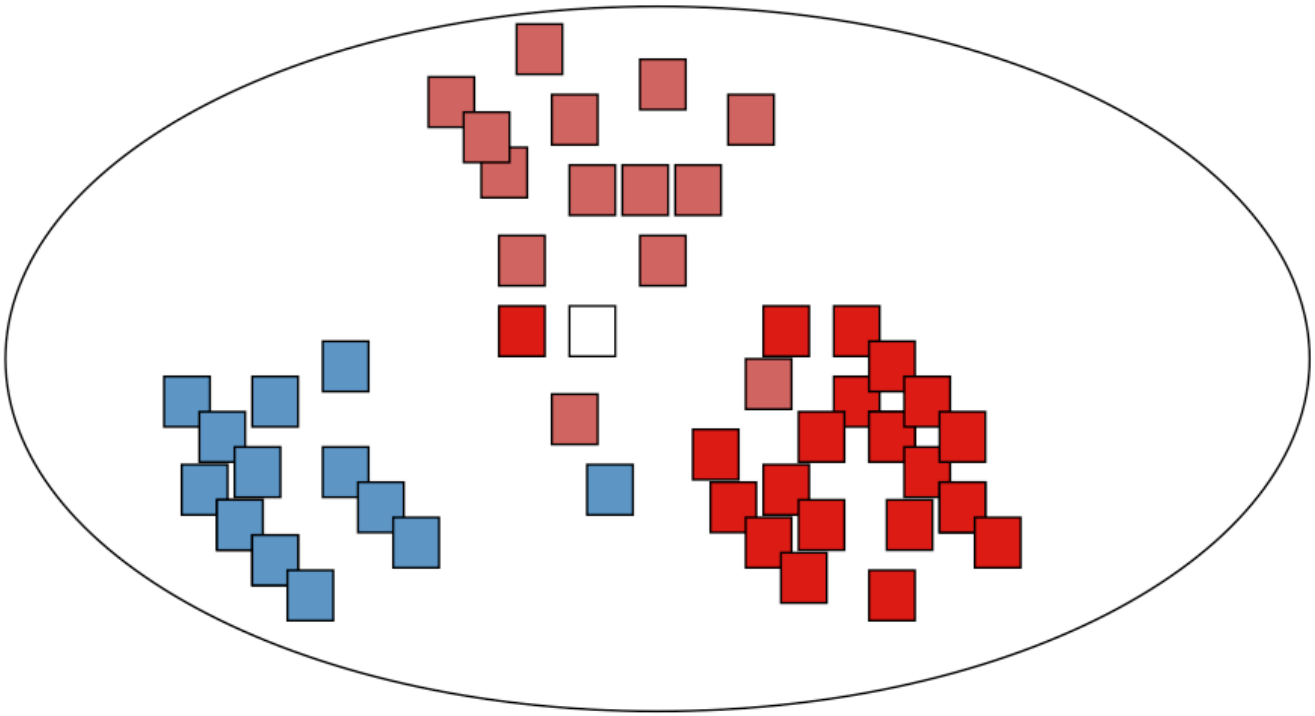
En general, la implementación asume la mayor diferencia posible.

- *Nominales: si falta uno o ambos valores, se toma el valor 1.*
- *Numérica: si faltan ambos valores, se toma el valor 1. Si falta uno solo y el otro está normalizado, se toma la diferencia como el máximo entre* $1 - v'$ *o* v'*, donde* v' *es el valor normalizado.*

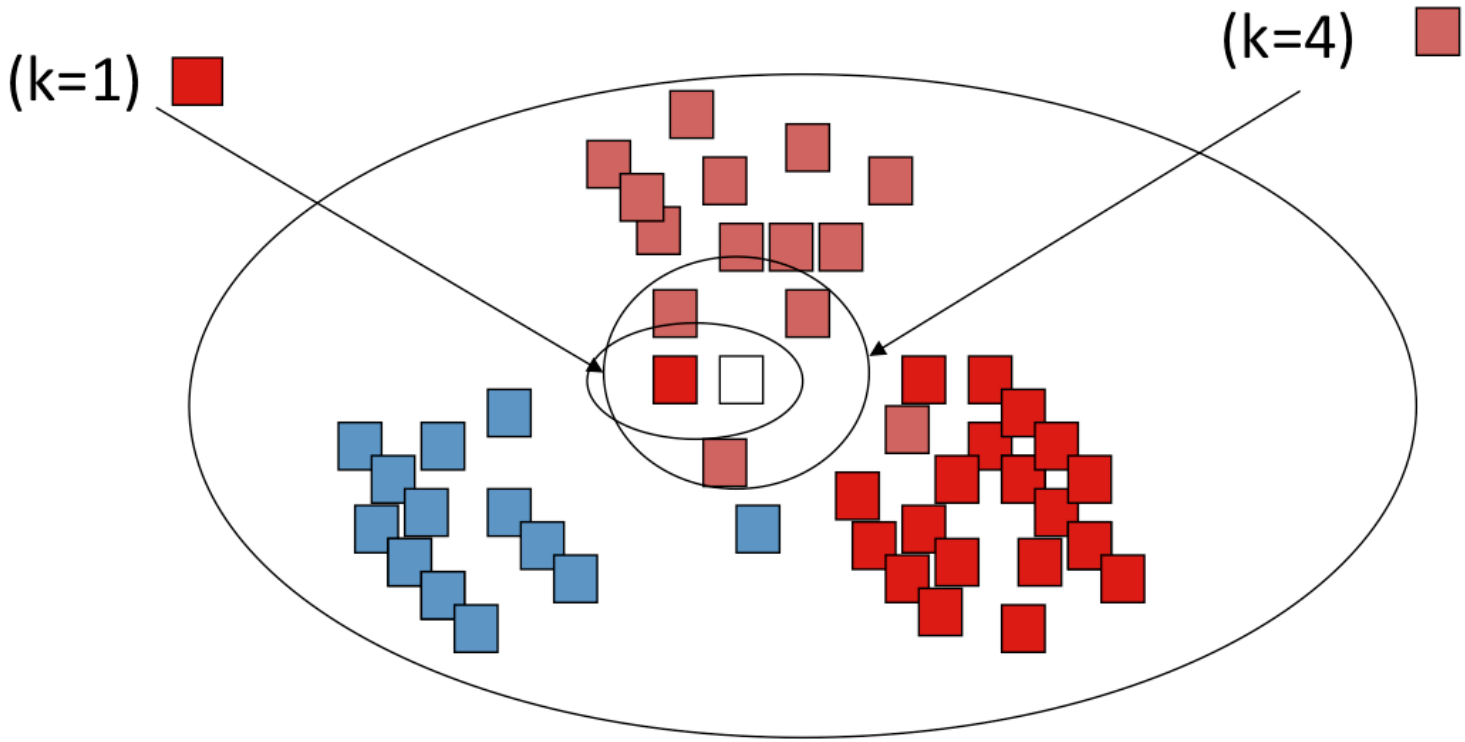
k-NN



k-NN



k-NN



k-NN: ¿Cómo determinar el valor de k?

Experimentalmente:

- Se parte con $k = 1$ y datos de validación para estimar la tasa de error del clasificador.
- Se repite el proceso aumentando k en 1 vecino más.
- Se selecciona el valor de k que entrega la tasa de error mínima.

k-NN: Ejemplo

- Base de datos `iris`: contiene datos de 50 muestras de cada una de 3 clases de flores de iris.
- Información sobre:
  - largo del sépalo en cm.
  - ancho del sépalo en cm.
  - largo del pétalo en cm.
  - ancho del pétalo en cm.
  - tipo: setosa, versicolour, virginica.
- Uso de función `knn` de librería `class`.

k-NN: Ejemplo

```
data("iris")
library(class)
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])

cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
lazy<-knn(train, test, cl, k = 3, prob=TRUE)
print(lazy)
attributes(.Last.value)
```

k-NN: Ejemplo

```
library(kknn)
data(iris)
m <- dim(iris)[1]
val <- sample(1:m, size = round(m/3), replace = FALSE,
              prob = rep(1/m, m))
iris.learn <- iris[-val,]
iris.valid <- iris[val,]
iris.kknn <- kknn(Species~., iris.learn, iris.valid, distance = 1,
                  kernel = "triangular")
summary(iris.kknn)
fit <- fitted(iris.kknn)
table(iris.valid$Species, fit)
pcol <- as.character(as.numeric(iris.valid$Species))
pairs(iris.valid[1:4], pch = pcol, col = c("green3", "red")
      [(iris.valid$Species != fit)+1])
```

2.2.4.- Métodos Conjuntos

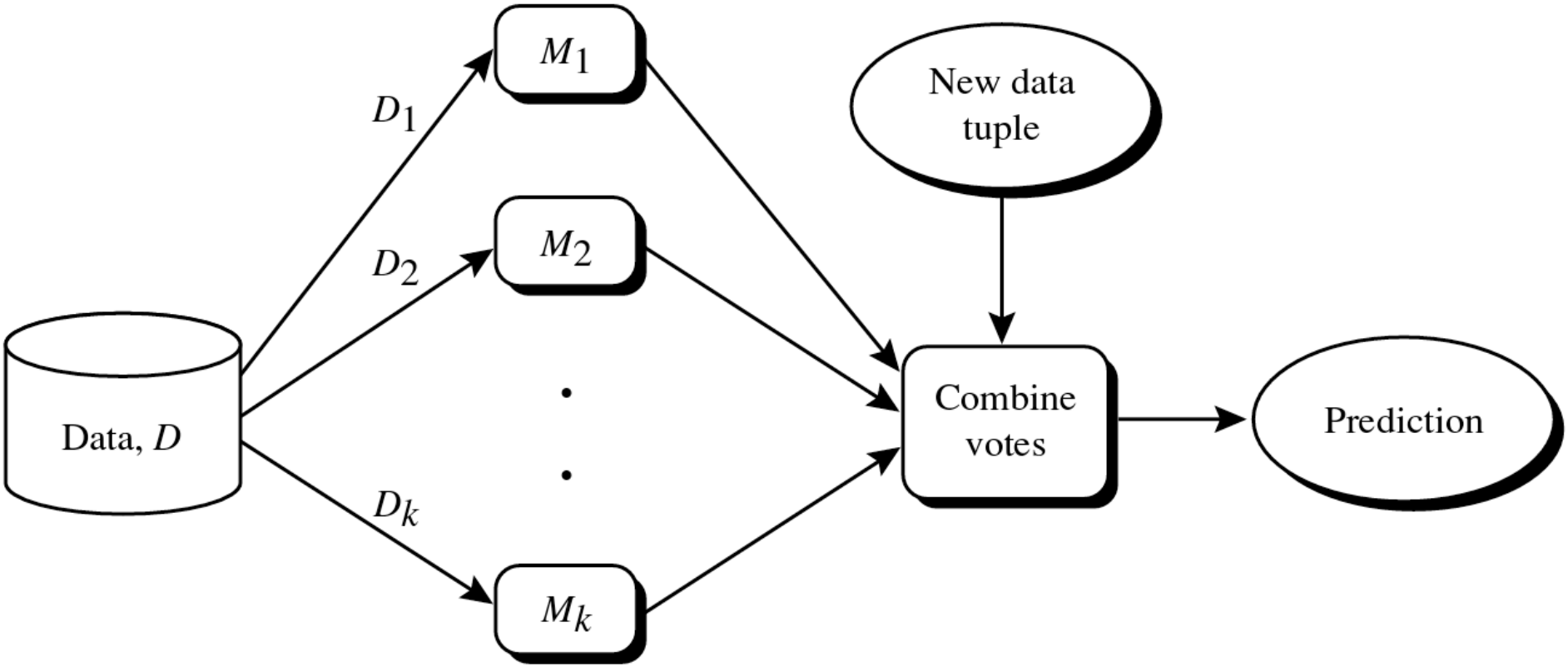
Introducción

- Métodos conjuntos (ensemble methods).
- Modelo compuesto formado por una combinación de clasificadores.
- Los clasificadores individuales votan y la clase predicha se basa en la colección de votos.
- Estos métodos tienden a ser más exactos que los clasificadores que lo componen.

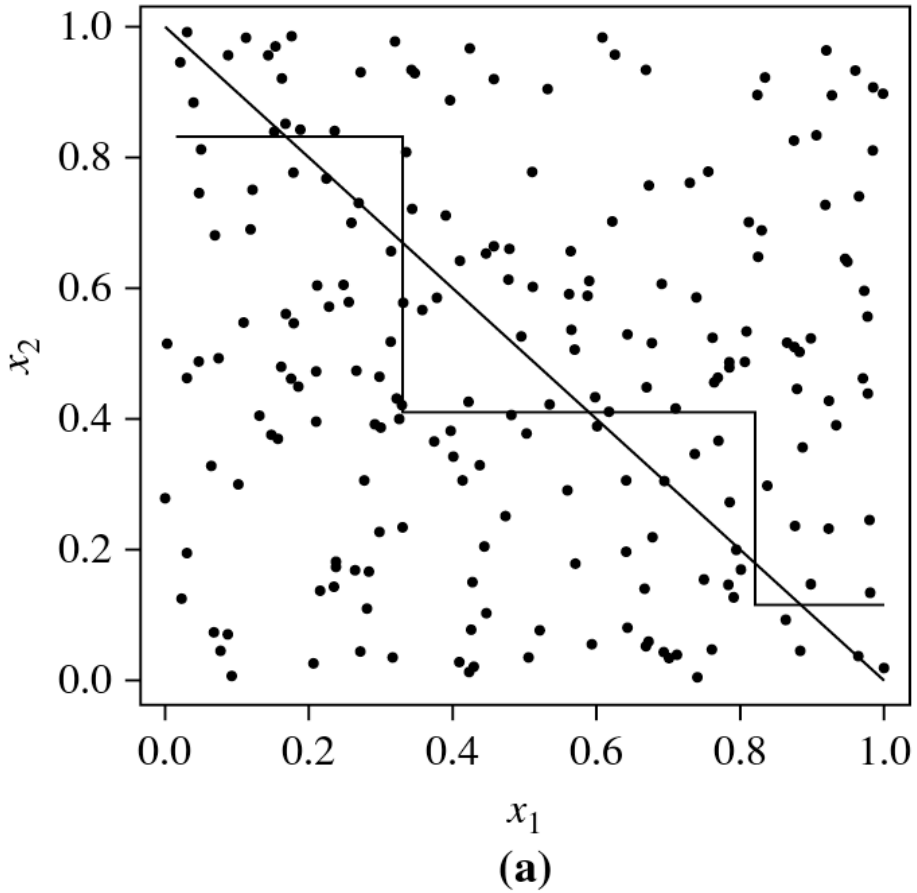
Introducción

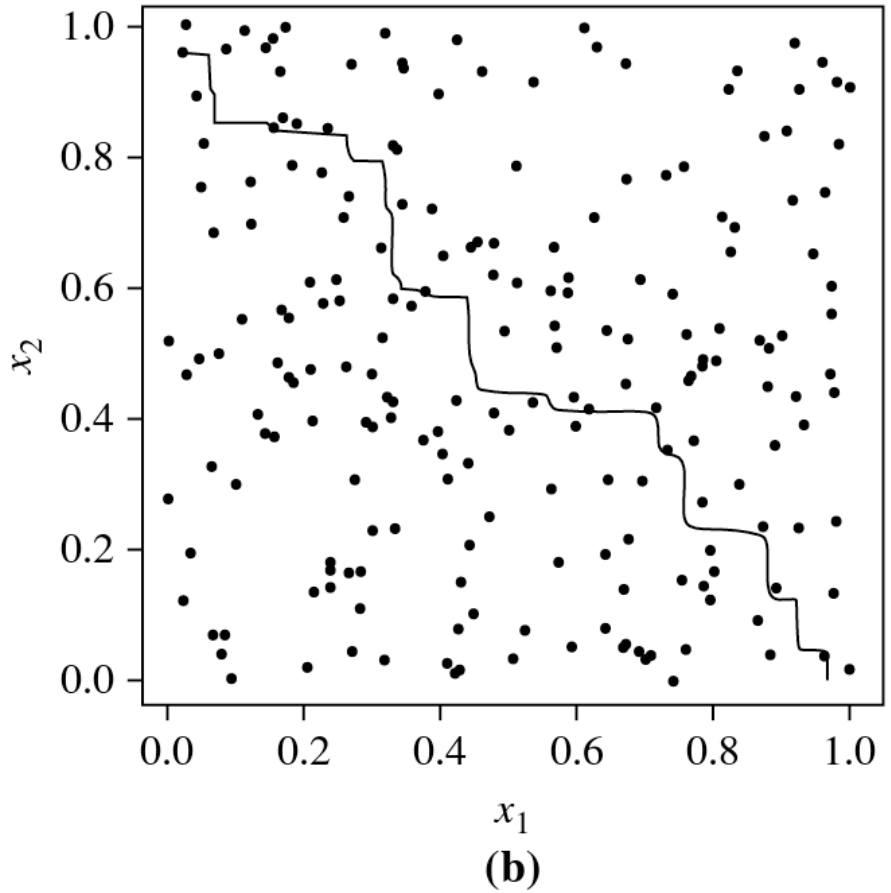
- Los clasificadores conjuntos combinan una serie de k clasificadores, $M_1, M_2, \ldots, M_k$ con el objeto de crear un modelo de clasificación compuesto $M*$.
- Un conjunto de datos D se usa para crear k conjuntos de datos de trabajo $D_1, D_2, \ldots, D_k$, donde D_i $(1 \leq i \leq k-1)$ se usa para generar el clasificador M_i.
- Dado un nuevo registro, cada clasificador entrega una clase predicha.
- El conjunto entrega su predicción basado en los votos de los clasificadores bases.

Introducción



Introducción: (a) indivuales y (b) combinados





2.2.4.1.- BAGGING

Bagging

La idea básica de este método puede ejemplificarse en base a un diagnóstico médico.

- Pedir más de una opinión y quedarse con el diagnóstico que ocurre más frecuentemente.
- El diagnóstico se basa en el voto de la mayoría, donde cada doctor entrega un voto.
- Una votación en base a un grupo más grande de especialistas puede ser más confiable que una basada en un grupo menor.
- `Bagging = bootstrap aggregation.`

Bagging: Algoritmo

Dado un conjunto D, de d unidades, el `bagging` funciona de la siguiente manera:

Paso 1: Para la iteración i ($i = 1, 2, \ldots, k$), se muestrea un conjunto D_i de d unidades con reemplazo de las unidades originales D.
Cada conjunto de datos de trabajo es una muestra bootstrap.

Paso 2: Un modelo M_i es entrenado para cada conjunto de datos de trabajo D_i.

Paso 3: Para clasificar una unidad desconocida $\boldsymbol{X}$, cada M_i entrega su predicción, que cuenta como un voto.

Paso 4: El clasificador conjunto, $M*$, cuenta los votos y asigna a $\boldsymbol{X}$ la clase con mayor votación.

Bagging

- Se puede aplicar también en el caso de predicción de variables continuas, tomando el valor promedio de cada predicción.
- Muchas veces tiene mayor precisión que un clasificador individual derivado de D.
- El aumento de precisión se produce porque el modelo compuesto reduce la varianza de los clasificadores individuales.

Bagging: Ejemplo

- Función `bagging` de la liberaría `adabag`.

- Ajusta el algoritmo propuesto por Breiman en 1996 usando árboles de clasificación como clasificadores individuales.

Bagging: Ejemplo

```
library(adabag)
data("iris")

### Seleccion de datos de trabajo o entrenamiento.
### Se seleccionan 25 de cada especie.

train <- c(sample(1:50, 25),
           sample(51:100, 25),
           sample(101:150, 25))

### Bagging
iris.bagging <- bagging(Species ~ ., data = iris[train, ], mfinal = 10,
                        control = rpart.control(maxdepth = 1))

iris.bagging <- bagging(Species ~ ., data = iris[train, ], mfinal = 10)
iris.bagging$samples[,1]

### Prediccion
iris.predbagging <- predict.bagging(iris.bagging, newdata=iris[-train,])
iris.predbagging
```

2.2.4.2.- BOOSTING

Boosting - AdaBoost

La lógica del `boosting` es la siguiente:

- Suponga que tienen ciertos síntomas y en vez de consultar a un doctor, visita muchos.
- Suponga que asigna pesos al valor del diagnóstico de cada uno, en base a la precisión de diagnósticos previos que han hecho.
- El diagnóstico final es una combinación ponderada de los diagnósticos.

Boosting - AdaBoost

- En `boosting` se asignan pesos a las unidades de entrenamiento.
- Se entrena iterativamente una serie de k clasificadores.
- Una vez que se ha entrenado el clasificador M_i, se actualizan los pesos de manera que el siguiente clasificador, M_{i+1} ponga más atención a las unidades de entrenamiento que fueron mal clasificadas por M_i.
- El clasificador final, $M*$, combina los votos de cada clasificador individual, donde el peso del voto de cada clasificador es una función de su precisión.

AdaBoost (Adaptative Boosting)

- Suponga que queremos aumentar la precisión de un método de aprendizaje.
- Se tiene un conjunto de datos D, de d unidades clasificadas,

$$(\boldsymbol{X}_1, y_1), (\boldsymbol{X}_2, y_2), \ldots, (\boldsymbol{X}_d, y_d)$$

  donde y_i es la clase de la unidad $\boldsymbol{X}_i$.
- Inicialmente, `AdaBoost` asigna un peso de $1/d$ a cada unidad de D.
- En la iteración i, las unidades de D son muestreadas para formar un conjunto de trabajo de tamaño d, D_i.
- La chance de ser seleccionada de cada unidad está determinada por su peso.

AdaBoost (Adaptative Boosting)

- Se deriva un clasificador M_i y se calcula su error con utilizando D_i como datos de validación.
- Se ajustan los pesos de las unidades de entrenamiento de acuerdo a cómo fueron clasificadas:
  - mal clasificada $\Longrightarrow$ aumenta su peso.
  - bien clasificada $\Longrightarrow$ disminuye su peso.

El peso de cada unidad refleja su dificultad de clasificación.
A mayor peso, más veces ha sido clasificada incorrectamente.

- Estos pesos se utilizan para generar las muestras de entrenamiento para el siguiente clasificador.

AdaBoost (Adaptative Boosting)

- La idea es centrarse más en las unidades con clasificación incorrecta en el paso anterior.
- Algunos clasificadores son mejores para clasificar unidades “difíciles” por lo que se construye una serie de clasificadores que se complementa.

AdaBoost: Algoritmo

Input:

- D: datos con d unidades clasificadas.
- k: número de iteraciones (se genera un clasificador por iteración)

Output: modelo compuesto.

Paso 1: Inicializar el peso de cada unidad en $1/d$.

Para cada iteración:

Paso 2: Muestrear D con reemplazo de acuerdo a los pesos para obtener D_i.

Paso 3: Usar D_i para derivar M_i.

AdaBoost: Algoritmo

Paso 4: Calcular:

$$error(M_i) = \sum_{j=1}^{d} w_j \times err(\boldsymbol{X}_j)$$

donde $err(\boldsymbol{X}_j)$ indica si la unidad $\boldsymbol{X}_j$ fue mal clasificada.

Paso 5: Si $error(M_i) > 0.5$, volver al paso 2.

Paso 6 Para cada unidad en D_i bien clasificada, multiplicar su peso por $\frac{error(M_i)}{(1-error(M_i)}$.

Paso 7 Actualizar los pesos y normalizarlos.

AdaBoost

- `Boosting` asigna un peso a cada voto de los clasificadores, de acuerdo a su rendimiento.
- A menor tasa de error, más exacto, por lo tanto más vale su voto.
- El peso del voto del clasificador M_i es

$$log\left(\frac{1 - error(M_i)}{error(M_i)}\right)$$

- Para cada clase, c, se suman los pesos de cada clasificador que asignó la clase c a $\boldsymbol{X}$.
- "Gana" la clase con la suma mayor y corresponde a la predicción para $\boldsymbol{X}$.

Bagging vs Boosting

- Como `boosting` se focaliza en las unidades mal clasificadas, corre el riesgo de sobreajustar el modelo resultante a esos datos.
- Algunas veces el modelo resultante puede ser menos preciso que un modelo único para los mismos datos.
- `Bagging` es menos susceptible a un sobreajuste del modelo.
- Aunque ambos pueden mejorar significativamente la exactitud del modelo en comparación a un modelo único, `boosting` tiende a lograr mayor precisión.

AdaBoost: Ejemplo

- Función `boosting` de la liberaría `adabag`.

- Implementa los algoritmos `AdaBoost.M1` (Freund and Schapire, 1996) y `SAMME` (Zhu et al., 2009) usando árboles de desición como clasificadores individuales.

Bagging: Ejemplo

```
library(rpart)
data(iris)
iris.adaboost <- boosting(Species~., data=iris, boos=TRUE, mfinal=10)
iris.adaboost <- boosting(Species ~ ., data = iris[train, ], mfinal = 10,
                          control = rpart.control(maxdepth = 1))
iris.adaboost

barplot(iris.adaboost$imp[order(iris.adaboost$imp, decreasing = TRUE)],
        ylim = c(0, 100), main = "Variables Relative Importance",
        col = "lightblue")

table(iris.adaboost$class, iris$Species[train],
      dnn = c("Predicted Class", "Observed Class"))
1 - sum(iris.adaboost$class == iris$Species[train]) /
  length(iris$Species[train])

### Prediccion
iris.predboosting <- predict.boosting(iris.adaboost,
                                      newdata = iris[-train, ])
iris.predboosting
```

2.2.4.3.- Random Forest

Random Forest

- Es un método conjunto en el que los clasificadores individuales corresponden a árboles de decisión.
- Los árboles individuales se generan usando una selección aleatoria de atributos en cada nodo para determinar la división.
- Formalmente, cada árbol depende de los valores de un vector aleatorio bajo muestreo independientemente y con la misma distribución para todos los árboles del bosque.
- Durante la clasificación, cada árbol vota y la clase más popular "gana".
- Los `random forests` pueden construirse usando `bagging` en tándem con selección aleatoria de atributos.

Random Forest: Algoritmo

Input:

- D: datos con d unidades clasificadas.
- k: número de árboles de decisión a considerar.
- F: número de atributos a ser usados para determinar la división en cada nodo, donde $F << A$.

Output: modelo compuesto.

Para cada iteración i: $(i = 1, \ldots, k)$

Paso 1: Muestrear los datos de entrenamiento D_i, de tamaño d con reemplazo desde D.

Paso 2: M_i selecciona, aleatoriamente en cada nodo, F atributos como candidatos.

Paso 3: Se construye el árbol de tamaño máximo (sin poda).

Random Forest

- `Random forests` son comparables en precisión con `AdaBoost`, aunque más robustos frente a errores y outliers.
- El error de un "bosque" converge cuando el número de árboles en él es grande, por lo que no hay problema de sobreajuste.
- Su precisión depende de la fuerza de los árboles individuales y la dependencia que exista entre ellos.
- Es importante mantener la fuerza individual sin aumentar la correlación.

Random Forest

- `Random forests` no son sensibles al número de atributos considerados para cada división.
- El uso de un solo atributo seleccionado aleatoriamente puede entregar buena precisión, la que usualmente es mejor que al usar varios atributos.
- El utilizar muchos menos atributos en cada división, los hacer muy eficientes en grandes bases de datos.

AdaBoost: Ejemplo

- Función `randomForest` de la liberaría `randomForest`.

- IImplementa el algoritmo de `random forests` propuesto por Breima y Cutler para clasificación y regresión.

Random Forest: Ejemplo

```
data("iris")
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
trainData <- iris[ind==1,]
testData <- iris[ind==2,]

library(randomForest)
rf <- randomForest(Species ~ ., data=trainData, ntree=100,
                       proximity=TRUE)
print(rf)

summary(rf)
rf$votes
rf$oob.times
rf$predicted

# Comparar la prediccion con los datos originales, utilizando
# los datos de entrenamiento
table(predict(rf), trainData$Species)
```

Random Forest: Ejemplo

```
# Comparar la prediccion con los datos originales, utilizando
# los datos de validacin
irisPred <- predict(rf, newdata=testData)
table(irisPred, testData$Species)
```

2.2.5.- Otros Métodos

Otros Métodos

- Support vector machines.
- Redes neuronales.
- Métodos Bayesianos noparamétricos.

2.3.- Lecturas Sugeridas

Lecturas Sugeridas

- Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. 1984 Classification and Regression Trees. Wadsworth and Brooks.
- Hastie, T., Tibshirani, R. and Friedman, J. 2001. The Elements of Statistical Learning : Data Mining, Inference, and Prediction. Springer.
- Mller, P., Quintana, F., Jara, A., and Hanson, T. E. 2015. Bayesian Nonparametric Data Analysis, Springer.
- Vapnik, V.N. 1998. Statistical Learning Theory. Wiley.