Introducción al Aprendizaje Automático

Alberto Torres Barrán

3 de Enero del 2020

Índice

Índice

Tipos de aprendizaje

- **Supervisado**: dados pares de entrada-salida, el objetivo es inferir su relación
  Ejemplos: regresión linea, regresión logística

- **No supervisado**: dados unos datos de entrada, el objetivo es inferir cierta estructura inherente en los mismos, sin necesidad de especificar las salidas de forma explícita
  Ejemplos: clustering, reducción de dimensión

- Existen otros tipos de tareas en los que el acceso a las salidas está limitado de distintas formas:
  - Aprendizaje activo
  - Aprendizaje semi-supervisado
  - Aprendizaje por refuerzo

Aprendizaje supervisado

- Dado un conjunto de datos, compuesto por observaciones de diferentes variables, llamamos **variable respuesta** a aquella que es objeto del estudio.
- Una vez identificada la respuesta, tenemos dos objetivos principales:
  - Predicción: ser capaz de predecir cual va a ser la respuesta para observaciones futuras.
  - Información: extraer información sobre la relación de la variable de respuesta con el resto.
- A su vez distinguimos dos tipos de problemas:
  - Regresión: la variable respuesta es continua.
  - Clasificación: la variable respuesta es discreta (número finito de categorias).

Índice

Primeros pasos

- Los datos a analizar a menudo provienen de fuentes variadas (redes sociales, sensores, encuestas, ...) y están almacenados en diferentes soportes (ficheros de texto, base de datos, ficheros binarios, *streams*...).
- Lo primero es identificar el problema qué queremos resolver y cuales son las variables que tenemos disponibles y pueden aportar información.
- Ante la duda, no descartar variables/información ni observaciones antes de tiempo.
- Lo segundo es combinar todas esa información y transformarla en una mezcla de variables numéricas (valores continuos) y categóricas (valores discretos).
- El objetivo final del preproceso es organizar esos datos en un formato tabular (filas y columnas).

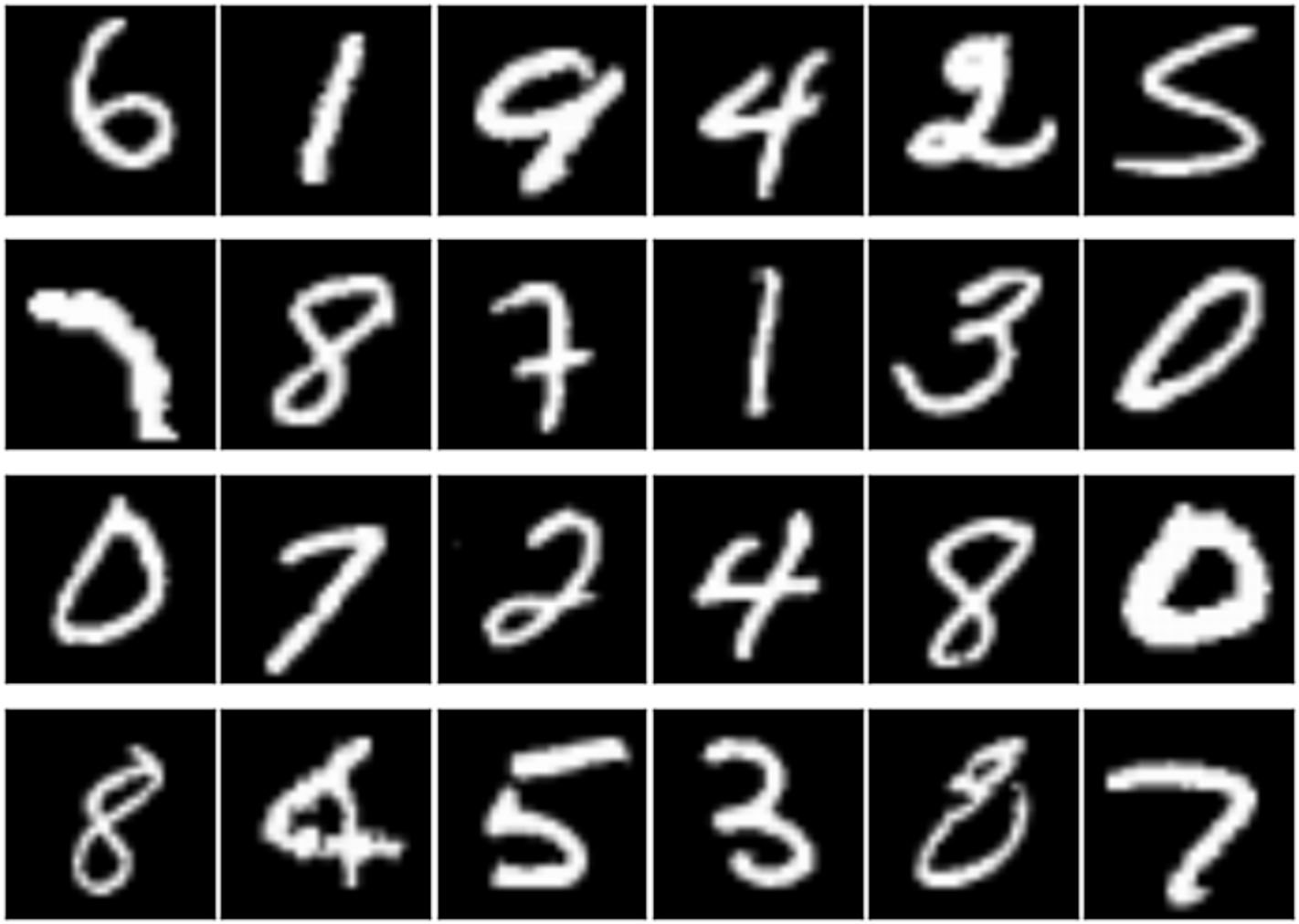
Distintos tipos de información

- En ocasiones no es trivial transformar ciertos tipos de información en variables numéricas y/o categóricas.

- Para estos casos a menudo es necesario un preproceso extra, muy dependiente del problema a resolver y específico del dominio.

- Ejemplos típicos:
  - Texto (tweets, páginas web, documentos): *word2vec*, *bag-of-words*, modelos *n-gram*.
  - Imágenes: valores RGB de los píxeles, intensidad de gris.
  - Audio: transformada de Fourier, coeficientes MFCC.
  - Video: secuencia de *frames*.

Ejemplo: MNIST I



Base de datos MNIST

Ejemplo: MNIST II

Cada dígito:

- se recorta
- se escala para que ocupe exactamente 20×20 píxeles.
- se centra en un recuadro de 28×28 píxeles.

Para transformarlo en datos tabulares:

1. consideramos cada imágen como una matriz de 28 filas y 28 columnas donde se almacenan los valores de 0 (negro) a 255 (blanco) de intensidad de gris.
2. convertimos esa matriz un un vector muy largo de $28 \times 28 = 784$ elementos.
3. apilamos los vectores de los distintos números uno encima de otro.

Missing values

- Es importante distinguir cuando una variable tiene valor 0 ó no conocido.
- Estos valores pueden venir representados por múltiples caracteres (“*”, “-”, campo vacio, etc.).
- Hay que codificarlos de manera especial para tenerlos en cuenta en los análisis.
- En general,
  1. Si tenemos suficientes datos, podemos simplemente ignorar las observaciones en las que falte alguno.
  2. Sino, podemos completar dichas observaciones que faltan con, por ejemplo, la mediana del resto.
- **Ejemplo**: en datos que provienen de un reconocimiento médico varios pacientes no tienen ningún valor en el campo de “Fármacos”. No toman ninguna medicación o el médico no ha registrado la respuesa?

Outliers

- Distinguir si un valor es erróneo o válido pero extremo es muy complicado y dependiente del dominio.
- Existen test estadísticos que permiten identificarlos.
- En general no son perjudiciales, pero si es importante corregir los valores que son **imposibles**.
- **Ejemplo**: en datos provenientes de un reconocimiento médico, aparece un paciente con un IMC de 50.

Normalización

- Las variables numéricas suelen tener rangos muy diversos.
- Ejemplo: salario (1000 – 1000000 €) y edad (0–100).
- Algunos modelos interpretan esta diferencia de escalas como que unas variables son más importantes que otras.
- Existen varias normalizaciones para que estas variables sean comparables:
  - Media 0 varianza 1.
  - Escalar al intervalo $-1, 1$.
  - ...
- En ocasiones normalizar las variables también puede ayudar a que el proceso de aprendizaje sea más rápido.
- Cuidado al analizar los resultados, ya que están en los nuevos rangos.

Variables categóricas

- Muy comunes en distintas fuentes de datos.
- Importante tenerlas en cuenta para la fase de preproceso y análisis previo.
- Muy pocos algoritmos son capaces de tratarlas directamente.
- Por tanto, antes de pasar a la siguiente etapa (modelado) queremos convertirlas en numéricas.
- La transformación donde se asigna a cada uno de sus valores un número entero no suele ser buena idea, ya que crea una relación artificial de orden y falsea las distancias.
- Una forma es utilizar una codificación "*dummy*".

Ejemplo codificación "*dummy*"

Edad	Sexo
34	H
18	M
67	M
21	M
15	H

$\Longrightarrow$

Edad	Es mujer?	Es hombre?
34	0	1
18	1	0
67	1	0
21	1	0
15	0	1

- Finalmente, podemos eliminar una de las dos nuevas variables puesto que están perfectamente correladas.
- En general, para una variable categórica con p valores añadimos $p - 1$ variables nuevas.

Otras codificaciones

- Si en la semántica de la variable hay implícita una relación de orden: Puntuación {baja, media, alta} $\Rightarrow$ {1,2,3}.

- Si hay relación de orden y no queremos falsear las distancias:

Mes	Día	Temp.
Enero	29	22.2
Enero	30	27.8
Enero	31	28.6
Febrero	1	26.1
Febrero	2	25.3

$\Longrightarrow$

Días desde 01/01	Temp.
29	22.2
30	27.8
31	28.6
32	26.1
33	25.3

Factores

- Un factor es un tipo de dato que se utiliza para codificar valores categóricos, por ejemplo: `renta` = {alta, baja, media}.
- Se crean con la función factor:

```
> f <- factor(c("hombre", "mujer", "mujer"))
```

- Se pueden ver los niveles (valores distintos) con la función `levels()`:

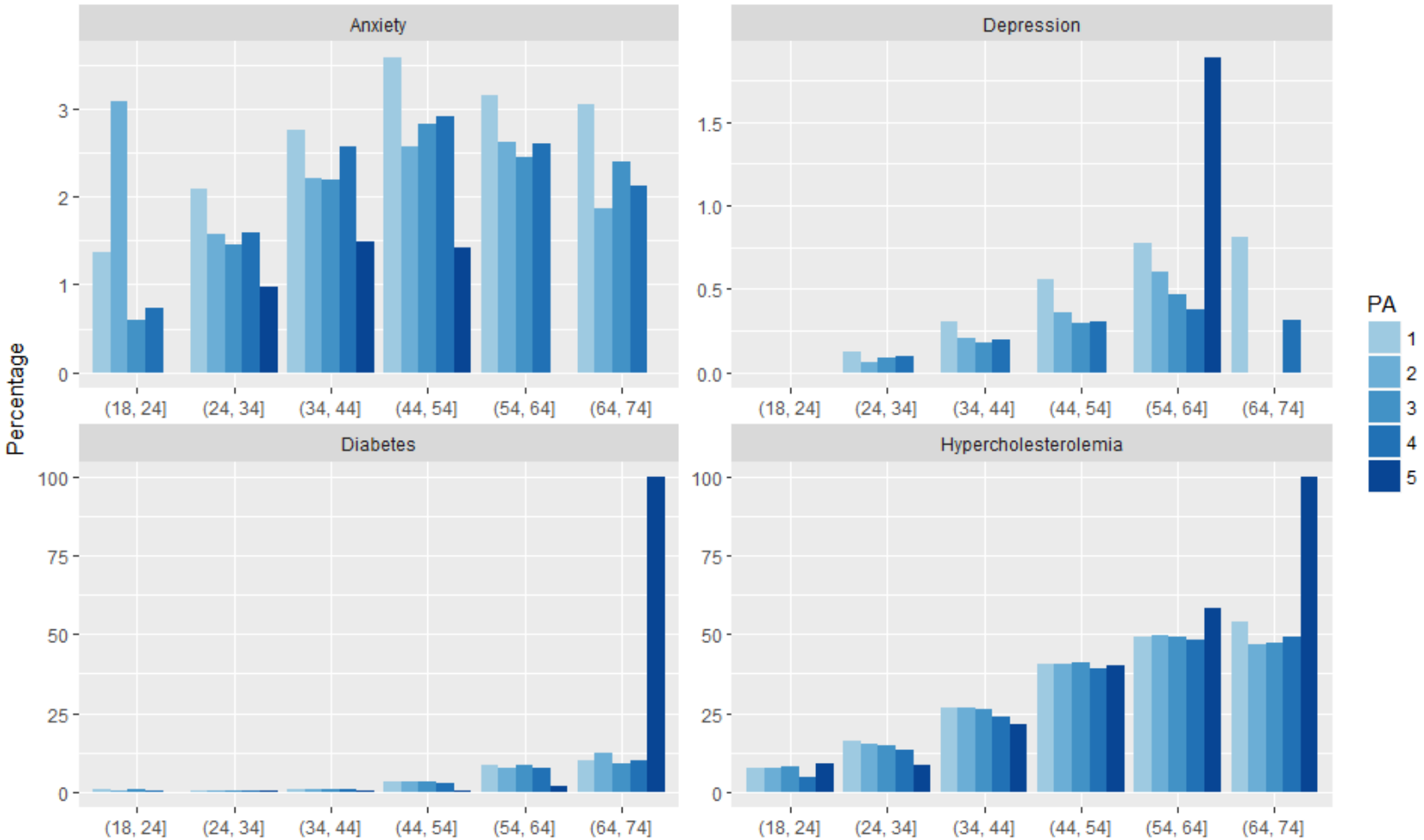
```
> levels(f)
[1] "hombre" "mujer"
```

- Función `relevel()`: reordena los niveles del factor especificado, poniendo el nivel especificado de primero

Visualización

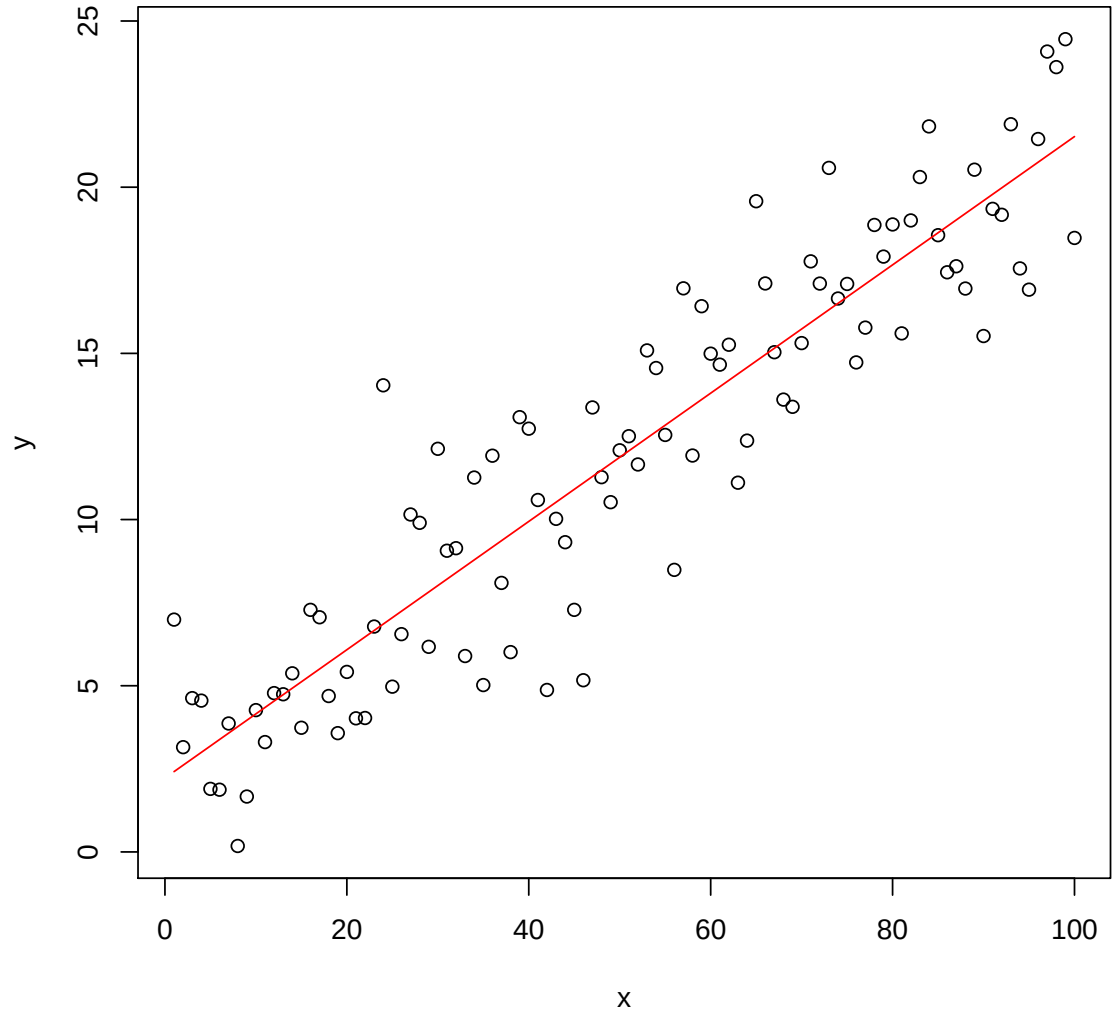
- En ocasiones es útil hacer gráficos de algunas variables para ver que tipo de relación tienen con la respuesta.
- Sin embargo, a medida que los conjuntos de datos son más grandes:
  1. el número de variables puede ser muy grande, incluso del orden de millones.
  2. la relación de las variables de entrada es muy compleja y altamente no lineal.
- En esos casos es muy difícil hacer gráficos que proporcionen información relevante, ya que podemos representar como mucho 2 o 3 dimensiones.
- Los gráficos múltiples (facetas) y algunas transformaciones estadísticas (reducción de dimensionalidad) pueden aliviar el problema.

Ejemplo


Anxiety
Depression
Diabetes
Hypercholesterolemia
Percentage
PA
1
2
3
4
5
(18, 24]
(24, 34]
(34, 44]
(44, 54]
(54, 64]
(64, 74]

Índice

Regresión lineal: ejemplo



Regresión lineal

- La regresión linear asume que la variable de respuesta y depende linealmente de las variables independientes,

$$y = w_0 + x_1 w_1 + x_2 w_2 + x_3 w_3 + \cdots + x_d w_d$$

- Si tenemos n observaciones de cada una de las variables, podemos escribirlo en notación matricial

$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

- El término de bias w_0 se incluye como una columna constante de 1s en $\mathbf{X}$

- El objetivo es estimar los pesos o coeficientes w

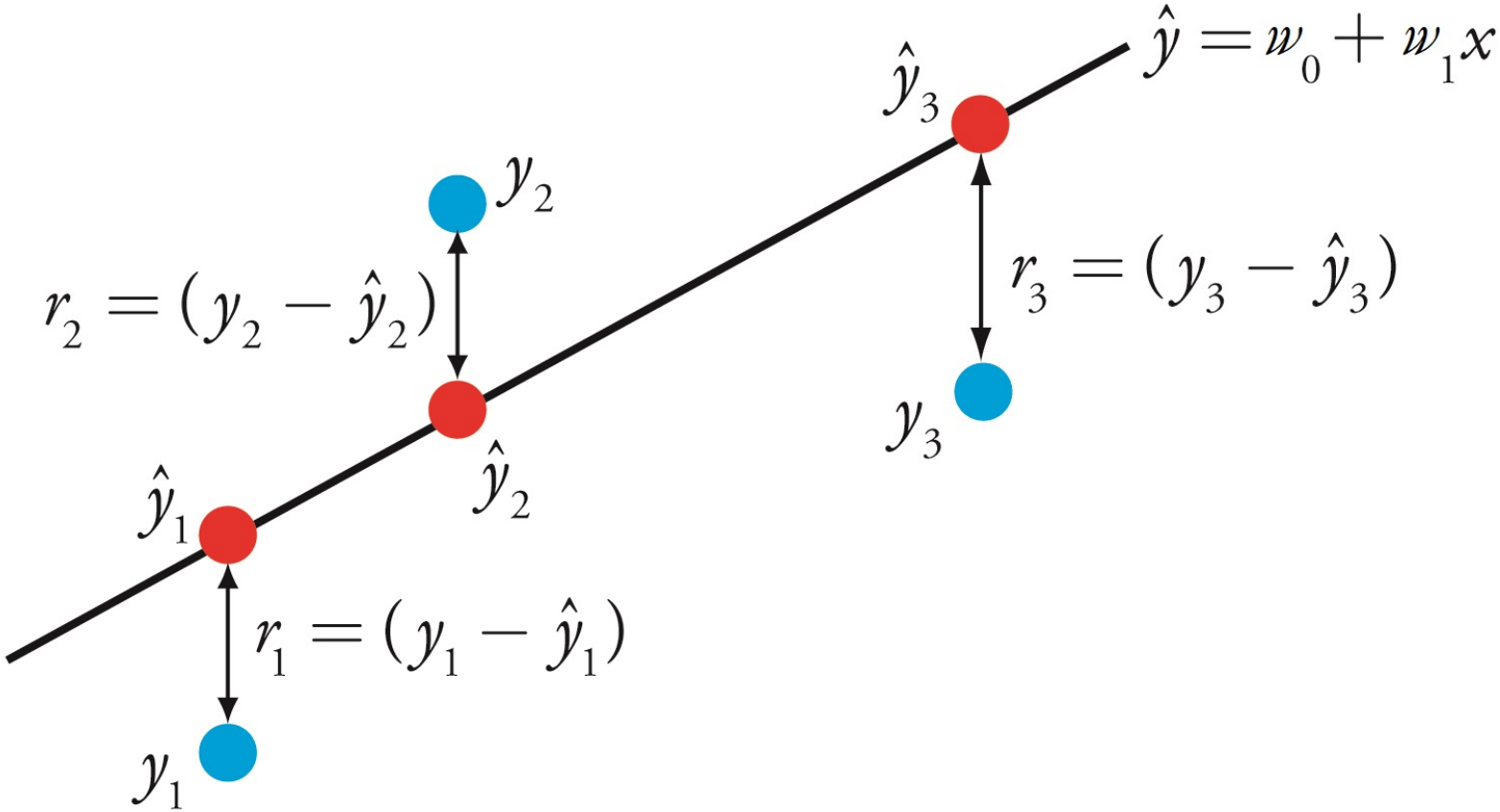
Hipótesis de la regresión lineal

- El método más común para calcular la recta de regresión se conoce como mínimos cuadrados.

- Teóricamente, para que el ajuste esté bien definido se asume que:
  1. la respuesta depende linealmente de las variables
  2. el modelo está especificado correctamente (no faltan variables)
  3. hay menos variables que observaciones
  4. no hay dos variables con correlación perfecta

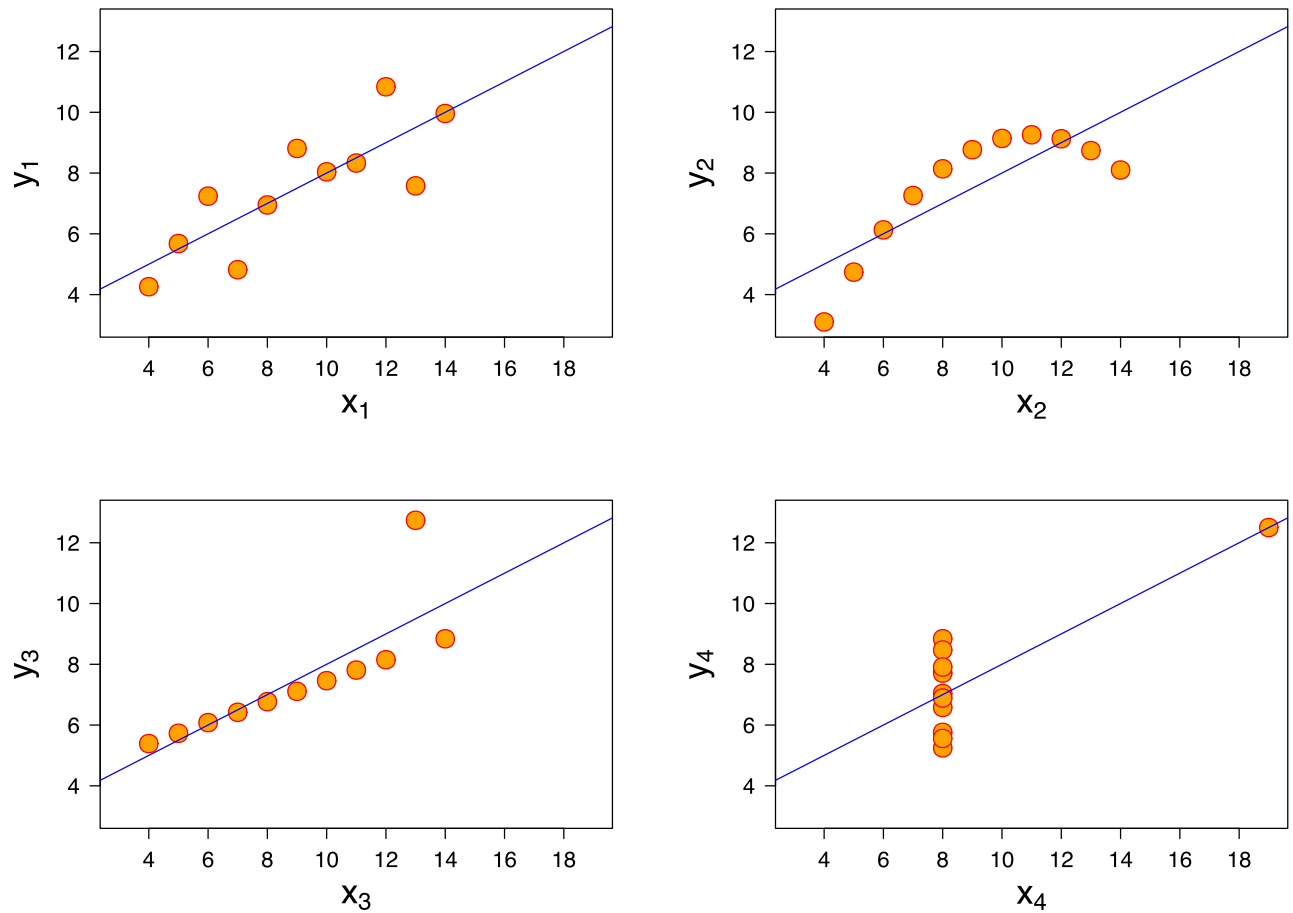
- Es un modelo preditivo, ya que nos permite calcular el valor de y para nuevos valores de x.

Mínimos cuadrados



Cuarteto de Anscombe

La pregunta es, ¿cómo de bien se ajusta la recta a nuestros datos?



Bondad de ajuste

- Históricamente se medía la calidad del modelo ó "bondad del ajuste" con diversos test sobre los residuos:
  - Homocedásticos (varianza constante)
  - Media cero
  - Sin autocorrelación
  - (Distribución normal)

- El módelo puede ser útil a pesar de que las hipótesis de la regresión y los test sobre los residuos no se cumplan.

- A menudo nos interesa únicamente la **capacidad predictiva**.

- "*All models are wrong, but some are useful*" (George Box).

Minimización del riesgo empírico

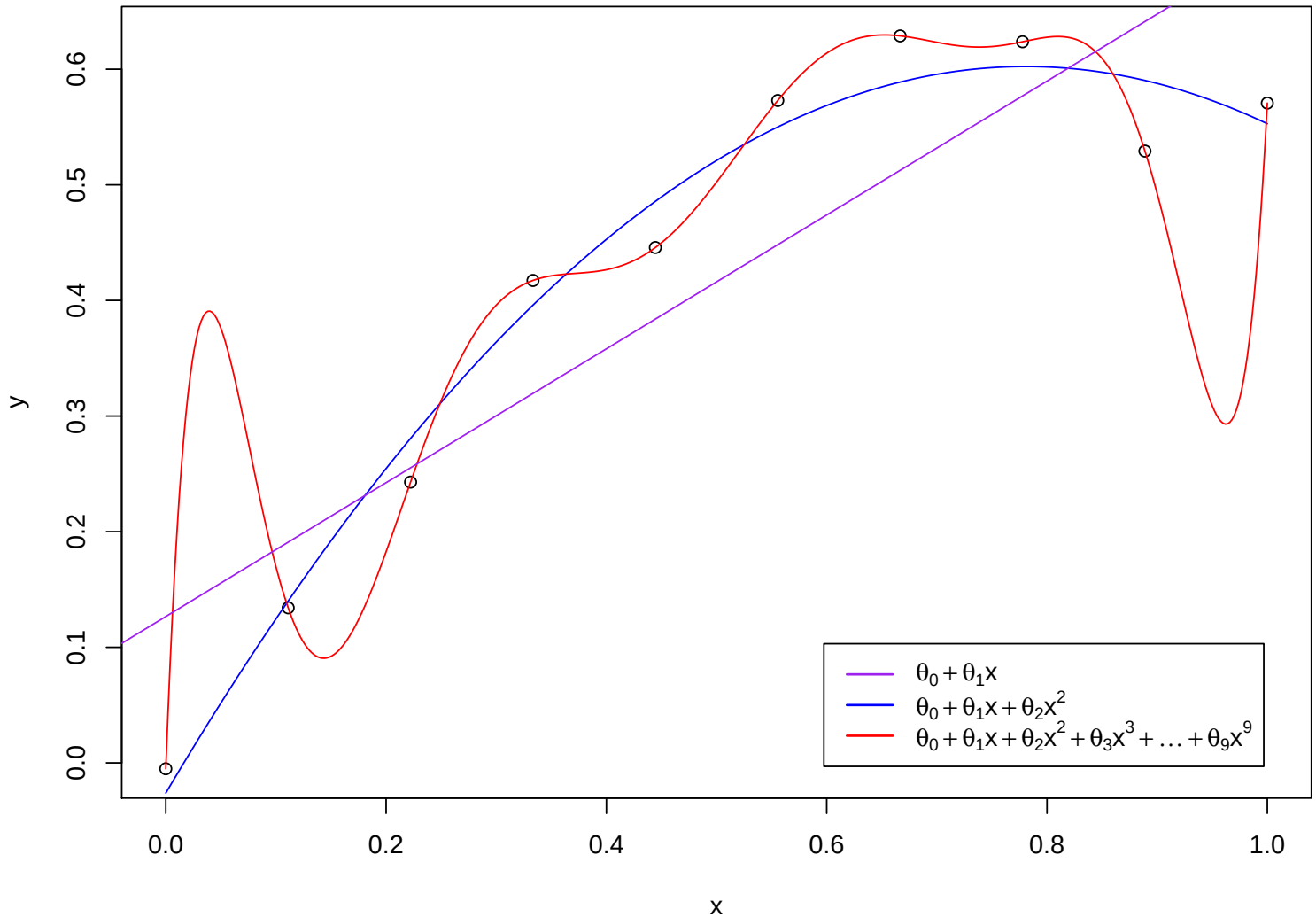
- Necesitamos definir formalmente a que nos referimos con **capacidad predictiva**.
- Dadas unas variables $\mathbf{x}$, queremos encontrar una función $f(\mathbf{x})$ que se parezca lo máximo posible a la respuesta y.
- Para ello, necesitamos una función de pérdida $L(f(\mathbf{x}), y)$ que cuantifique cuando de diferente es nuestra predicción del valor real.
- El problema de aprendizaje consiste por tanto en encontrar la función f que minimiza la pérdida media de todas las observaciones:

$$\hat{f} = \arg\min \frac{1}{n} \sum_{i=1}^{n} L(f(\mathbf{x}_i), y_i)$$

Sobreajuste

- Si resolvemos el problema anterior con el error cuadrático como función de pérdida $L(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$, obtenemos el estimador de mínimos cuadrados.
- Por tanto, ya tenemos una forma de calcular el error de predicción de nuestro modelo original.
- La pregunta ahora es, podemos mejorar el modelo (disminuir su error)?
- Sí, aunque el modelo tiene que ser lineal en las variables, se pueden añadir nuevas variables que sean transformaciones polinómicas.
- De hecho, siempre podemos añadir expansiones polinómicas de forma que el error sea casi 0.

Sobreajuste: ejemplo



Equilibrio sesgo-varianza: intuición

- Si el modelo es muy simple, la solución esta sesgada y no ajusta bien los datos.
- Si el modelo es muy complejo, es muy sensible a pequeños cambios en los datos.
- En general el error calculado sobre las muestras usadas para entrenar el modelo (error de **entrenamiento**) es demasiado optimista.
- El error de entrenamiento se puede hacer arbitrariamente pequeño aumentando la complejidad del modelo.
- Nos interesa el error de **generalización**, es decir el error sobre muestras que el modelo no conoce.

Equilibrio sesgo-variance: definición gráfica

Fuente

Equilibrio sesgo-varianza: formulación

El error teórico de predicción es

$$\text{EP} = \mathbb{E}[(y - \hat{f}(x))^2].$$

Se puede descomponer en

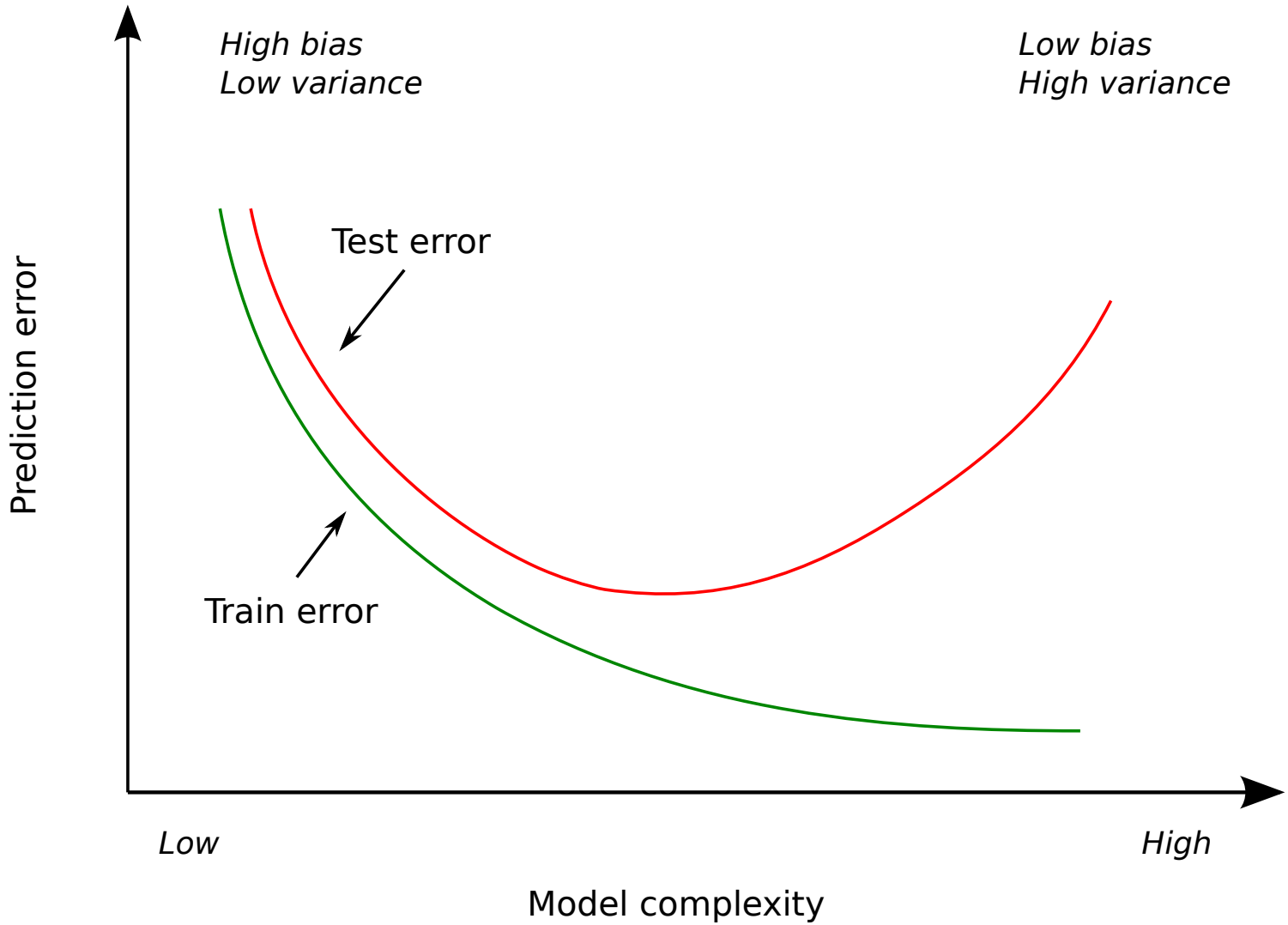
$$\text{EP} = \underbrace{\Big(\mathbb{E}[\hat{f}(x)] - f(x)\Big)^2}_{\text{Sesgo}^2} + \underbrace{E\Big[\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\Big]^2}_{\text{Varianza}} + \underbrace{\sigma^2}_{\text{Ruido}}$$

- Los términos de sesgo y varianza son opuestos: si disminuimos uno el otro aumenta y viceversa.
- El término del ruido es inherente a los datos y no podemos hacer nada.

Conjuntos de entrenamiento y test

- En la práctica, lo primero que hacemos cuando cargamos unos datos es dividirlos aleatoriamente en dos subconjuntos, entrenamiento y test.
- El subconjunto de test nos lo guardamos y no se utiliza **nunca** en la fase de aprendizaje del modelo.
- Una vez construido el modelo, se comprueba su rendimiento en el conjunto de test.
- Este último error es una buena estimación no sesgada de como se va a comportar nuestro modelo con nuevos datos.
- Existe una gran probabilidad de sobreajuste si el error de test es muy alto en comparación con el error de entrenamiento.

Error de predicción en función de la complejidad



Errores de regresión

Dado el valor real de la observación i, y_i y la predicción del modelo $\hat{y}_i$, podemos calcular:

- MAE (Mean absolute error)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

- MSE (Mean squared error)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Modelos lineales en R

- En R un modelo lineal de la variable de respuesta y sobre las variables $x_1, \ldots, x_d$, se define con la fórmula

$$y \sim x_1 + x_2 + \cdots + x_d$$

- Para ajustar un modelo lineal por mínimos cuadrados se usa la función `lm()`, pasando como primer argumento la fórmula anterior.
- Las fórmulas también pueden contener expresiones aritméticas de variables (expansiones polinómicas, logaritmos, etc).
- Ejemplo: modelo de regresión de la variable `mpg` sobre `wt`

```
> fit <- lm(mpg ~ wt, data=mtcars)
```

Fórmulas

- Las fórmulas son objetos especiales de R que respresentan relaciones simbólicas entre variables:
  `respuesta` $\sim$ `variables independientes`
- Se usan en funciones como `aggregate()`, `boxplot()`, y `lm()`.
- Los operadores aritméticos tienen otro significado cuando se usan dentro de las fórmulas. Ejemplos:

```
y ~ u + v + w + u:v + u:w + v:w
y ~ u * v * w - u:v:w
y ~ (u + v + w)^2
```

- Si queremos que tengan su significado habitual tenemos que utilizar el operador `I()`:

```
y ~ u + v + w + I(u*v) + I(u*w) + I(v*w)
```

Ejercicio regresión

Con el conjunto de datos `diamonds`:

- Separarlos aleatoriamente en un 60% de entrenamiento y un 40% de test.
- Ajustar un modelo lineal del precio sobre los quilates.
- Ajustar un modelo cuadrático con las mismas variables.
- Calcular el error cuadrático medio sobre el conjunto de entrenamiento y test, definido como

$$\text{ECM} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

  donde y es la variable respuesta y $\hat{y}$ la predicción del modelo.
- ¿Qué pasa ahora si ajustamos el modelo sobre todas las variables? ¿Disminuye el error?

Regresión logística

- Es un modelo lineal para problemas de clasificación.
- En lugar de una variable continua, la respuesta es ahora una variable discreta con 2 o más valores, que se denominan *clases*.
- En el caso binario, el modelo estima la probabilidad de que cada uno de los ejemplos pertenezca a la clase 0 o 1.
- Finalmente se predice la clase 0 si la probabilidad es menor que 0.5 y la clase 1 en caso contrario.
- Se puede ver como una caso especial del modelo lineal generalizado.

Regresión logística: formulación

- La fórmula de la regresión logística es

$$y = \sigma(w_0 + w_1 x_1 + w_2 x_2 + ... + w_d x_d)$$

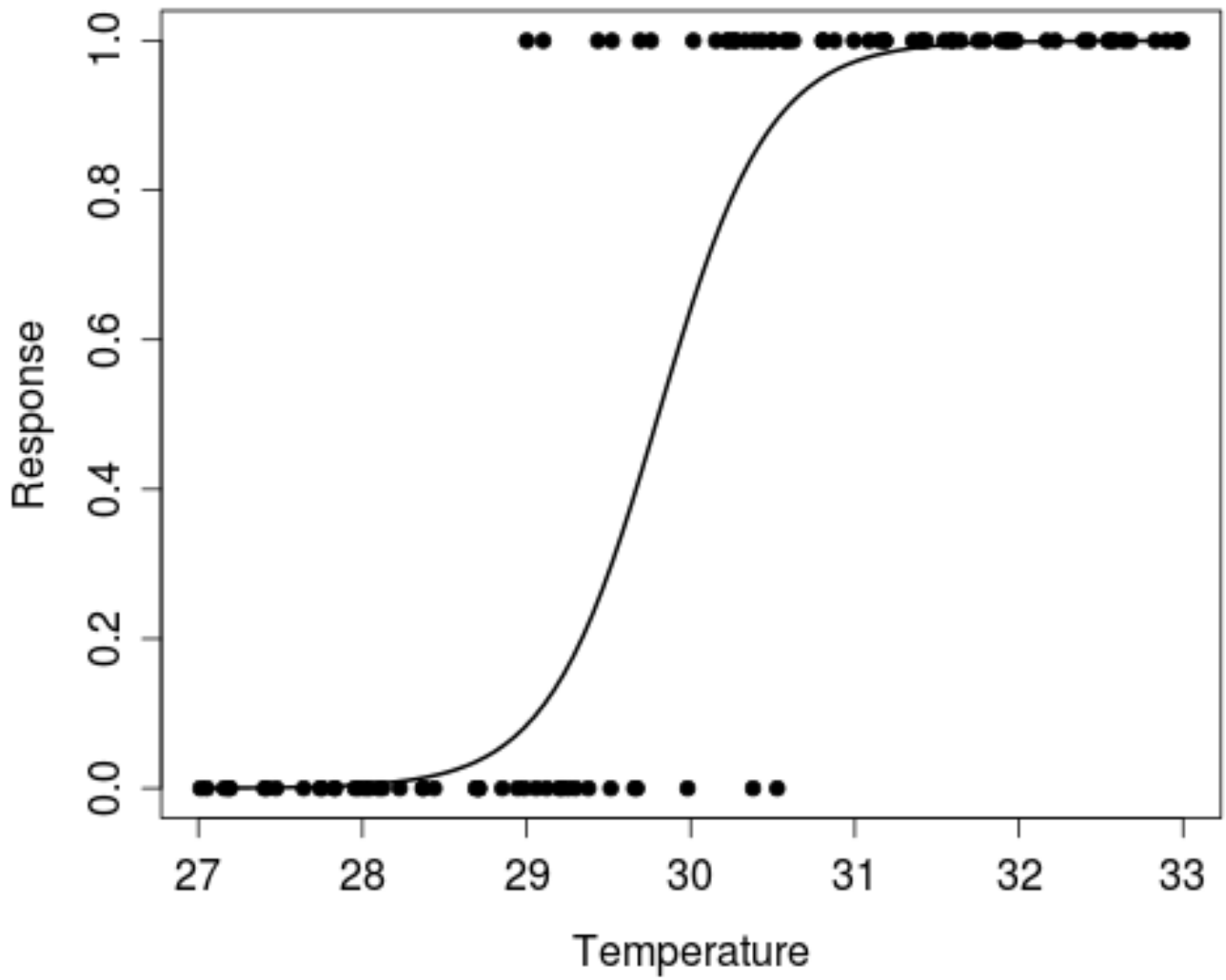
  donde $\sigma(\cdot)$ es la función logística

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

- La función logística siempre tiene como salida un número en el intervalo $(0, 1)$

- Por tanto, interpretamos la salida del modelo como la probabilidad de pertenecer a una clase o a la otra (clasificación binaria)

Regresión logística: ejemplo



Interpretación de coeficientes: odds ratio

- Dado un modelo lineal con una única variable independiente, el odds ratio se define como

$$\text{OR} = \frac{\exp(w_0 + w_1(x+1))}{\exp(w_0 + w_1 x)} = \exp(w_1)$$

- Es decir, como cambian las probabilidades de la salida cuando la variable x aumenta una unidad:
  - Si OR $= 1$ la variable x no tiene ninguna asociación con la salida
  - Si OR > 1 la variable está asociada con una mayor probabilidad de la salida
  - Si OR < 1 la variable está asociada con una menor probabilidad de la salida

Errores de clasificación

Los principales errores de clasificación se pueden calcular a partir de la **matriz de confusión**:

		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Accuracy: $\frac{\text{TP+TN}}{\text{P+N}}$

Sensitivity, recall, TPR: $\frac{\text{TP}}{\text{TP+FN}}$

Specificity, TNR: $\frac{\text{TN}}{\text{TN+FP}}$

Precision, PPV: $\frac{\text{TP}}{\text{TP+FP}}$

F1 score: $2 \times \frac{\text{PPV} \times \text{TPR}}{\text{PPV+TPR}}$

Índice

Modelos lineales generalizados (GLMs)

- Generalización de la regresión lineal que permite distribuciones de errores distintas de la distribución normal.
- Se asume que la media de dicha distribución depende de las variables independientes de la siguiente forma:

$$\mathbb{E}(\mathbf{y}) = \mu = g^{-1}(\mathbf{X}\mathbf{w})$$

  donde $\mathbb{E}(\cdot)$ es el valor esperado y g es la función de enlace
- La función de enlace proporciona la relación entre la media de la distribución y el predictor lineal

Ejemplo: distribución de Bernoulli

- Cuando la distribución de la salida $\mathbf{y}$ es una Bernoulli el modelo se conoce con el nombre de regresión logística

- La función de media es la logística,

$$\mu = g^{-1}(\mathbf{X}\mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{X}\mathbf{w})}$$

- La función de enlace es la inversa de la anterior,

$$\mathbf{X}\mathbf{w} = g(\mu) = \ln\left(\frac{\mu}{1-\mu}\right)$$

- Para cada distribución, hay una función de enlace "canónica" que es la que se usa habitualmente

Ejemplo: distribución de Poisson

- Esta distribución está indicada cuando queremos modelizar una variable de salida entera y no real (por ej. conteos)
- Función de media

$$\mu = \exp(\mathbf{X}\mathbf{w})$$

- Función de enlace

$$\mathbf{X}\mathbf{w} = \ln(\mu)$$

- Otras distribuciones posibles son la Gamma, Exponencial, Multinomial, etc.

GLMs en R

- La función para ajustar modelos lineales generalizados es `glm()`
- Tiene los mismos argumentos principales que `lm()`, pero además tenemos que especificar la distribución de la variables dependiente con el parámetro `family`
- Ejemplo: regresión logística

```
> fit <- glm(Species ~ Petal.Length, data=iris,
                    family=binomial)
```

- Por defecto se usa la función de enlace "canónica", pero esto se puede modificar (ver ayuda)

Regularización

- El estimador de mínimos cuadrados para la regresión lineal es el *mejor* estimador no sesgado, donde *mejor* se refiere al que tiene menor varianza.
- Sin embargo, a menudo se puede reducir esta varianza bastante, en detrimento de introducir un pequeño de sesgo.
- Esto se consigue limitando la complejidad del modelo con un término de **regularización**.
- Un ejemplo muy común es Ridge Regression, que añade una regularización l_2 a mínimos cuadrados:

$$\min_{\mathbf{w}} \; ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + \lambda ||\mathbf{w}||_2^2$$

Lasso

- Similar a Ridge Regression, Lasso añade un término de regularización l_1:

$$\min_{\mathbf{w}} \ ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + \lambda||\mathbf{w}||_1$$

donde

$$||\mathbf{w}||_1 = \sum_{i=1}^{d} |w_i|$$

- El valor absoluto promueve que muchos coeficientes sean 0 después de ajustar el modelo
- Dichos coeficientes no tienen por tanto efecto en la salida
- Se podría considerar que son variables "poco importantes"

Elastic Net

- Combina las regularizaciones de Ridge y Lasso:

$$\min_{\mathbf{w}} \ ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + \lambda_1||\mathbf{w}||_1 + \lambda_2||\mathbf{w}||_2^2$$

- Otra forma de escribirlo:

$$\min_{\mathbf{w}} \ ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + \lambda(\alpha||\mathbf{w}||_1 + (1-\alpha)||\mathbf{w}||_2^2)$$

- Para $\alpha = 0$ recuperamos Ridge Regression y para $\alpha = 1$ el Lasso
- Mantiene la dispersión en los coeficientes del Lasso pero en general se obtienen mejores modelos en términos de error
- Problema: tenemos que decidir el valor de dos hiper-parámetros (α y λ)

Paquete `glmnet` y `glmnetUtils`

- Implementa GLMs con regularización Lasso y Elastic Net
- Muy eficiente (escrito en Fortran)
- Se pueden ajustar no solo regresiones lineales con regularización, sino también regresiones logísticas, regresiones de Poisson, etc.
- Implementa también un mecanismo para seleccionar automáticamente el parámetro λ (pero no α) usando validación cruzada
- `glmnet` no tiene interfaz para fórmulas, pero `glmnetUtils` incorpora una

Ejercicio titanic I

Vamos a intentar predecir la supervivencia de las víctimas del Titanic a partir de las siguientes variables:

- `survival`: Supervivencia (0 = No; 1 = Si)
- `pclass`: Clase de pasajero (1, 2, 3)
- `name`: Nombre
- `sex`: Sexo
- `age`: Edad
- `sibsp`: Número de hermanos/esposos/as a bordo.
- `parch`: Número de padres/hijos a bordo
- `ticket`: Número de ticket
- `fare`: Coste del billete
- `cabin`: Cabina
- `embarked`: Puerto de embarque

Ejercicio titanic II

1. Cargar el fichero `titanic.csv` en R.
2. Ver cuantos valores *missing* tiene cada variable con `summary`.
3. Eliminar la variable `Cabin` (¿por qué?).
4. Eliminar también las variables `PassengerId`, `Name` y `Ticket` (¿por qué?).
5. Eliminar las filas que contengan algún `NA` (función `na.omit`).
6. Convertir la variable `Survived` a un factor.
7. Dividir datos en 80% `entrenamiento` y 20% `test`, aleatoriamente.
8. Ajustar un modelo de regresión logística estándar y otro con regularización (función `glm` y paquete `glmnet`).

Vecinos próximos

- Es uno de los algoritmos más sencillos de clasificación.
- Dado un conjunto de datos, simplemente clasifica nuevas observaciones como la clase más frecuente entre las k observaciones más cercanas.
- La métrica de distancia más común para calcular la cercanía de las observaciones es la distancia Euclídea:

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

- El valor de k es un parámetro del modelo, y puede tener mucha influencia en el rendimiento.

Ventajas y desventajas

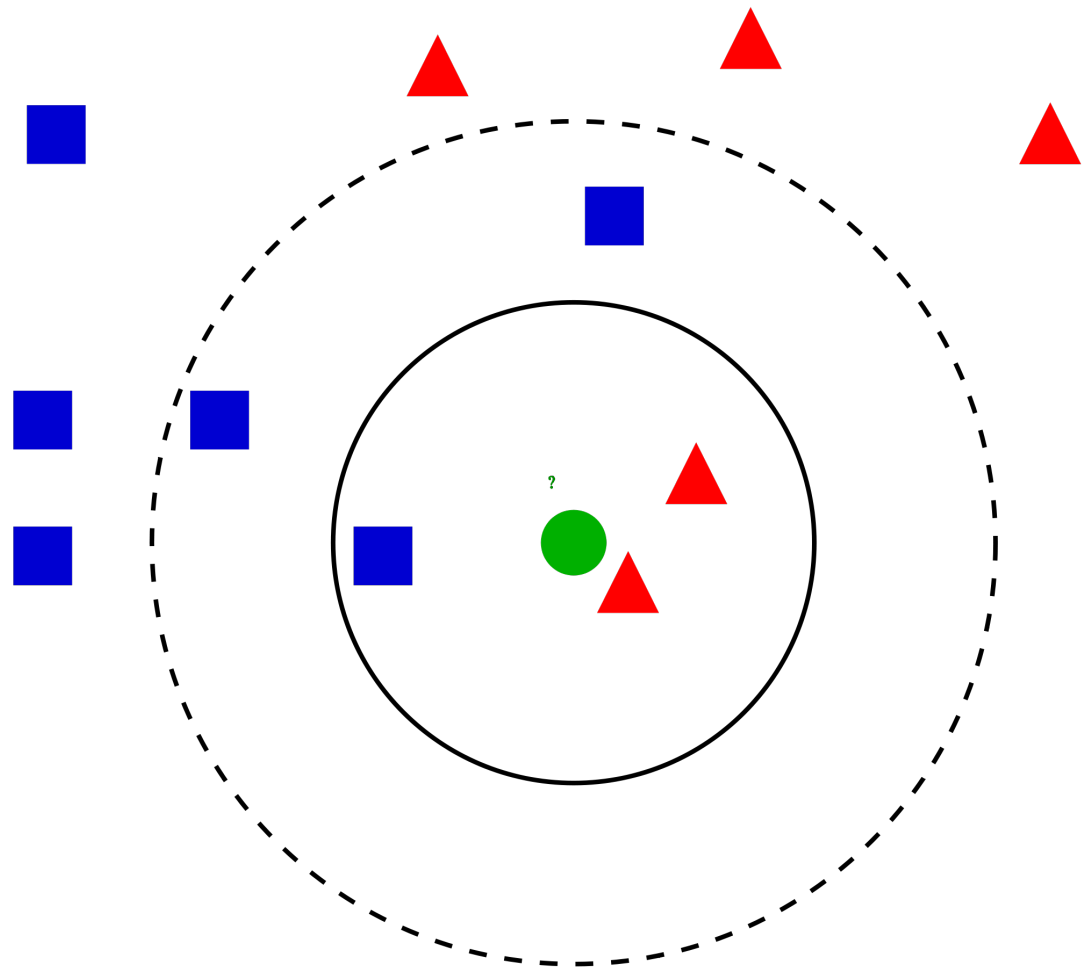
Ventajas

- Es un modelo muy simple, de hecho no se construye ningún modelo explicitamente.
- Robusto a ruido.
- Efectivo si hay muchas muestras de entrenamiento.

Desventajas

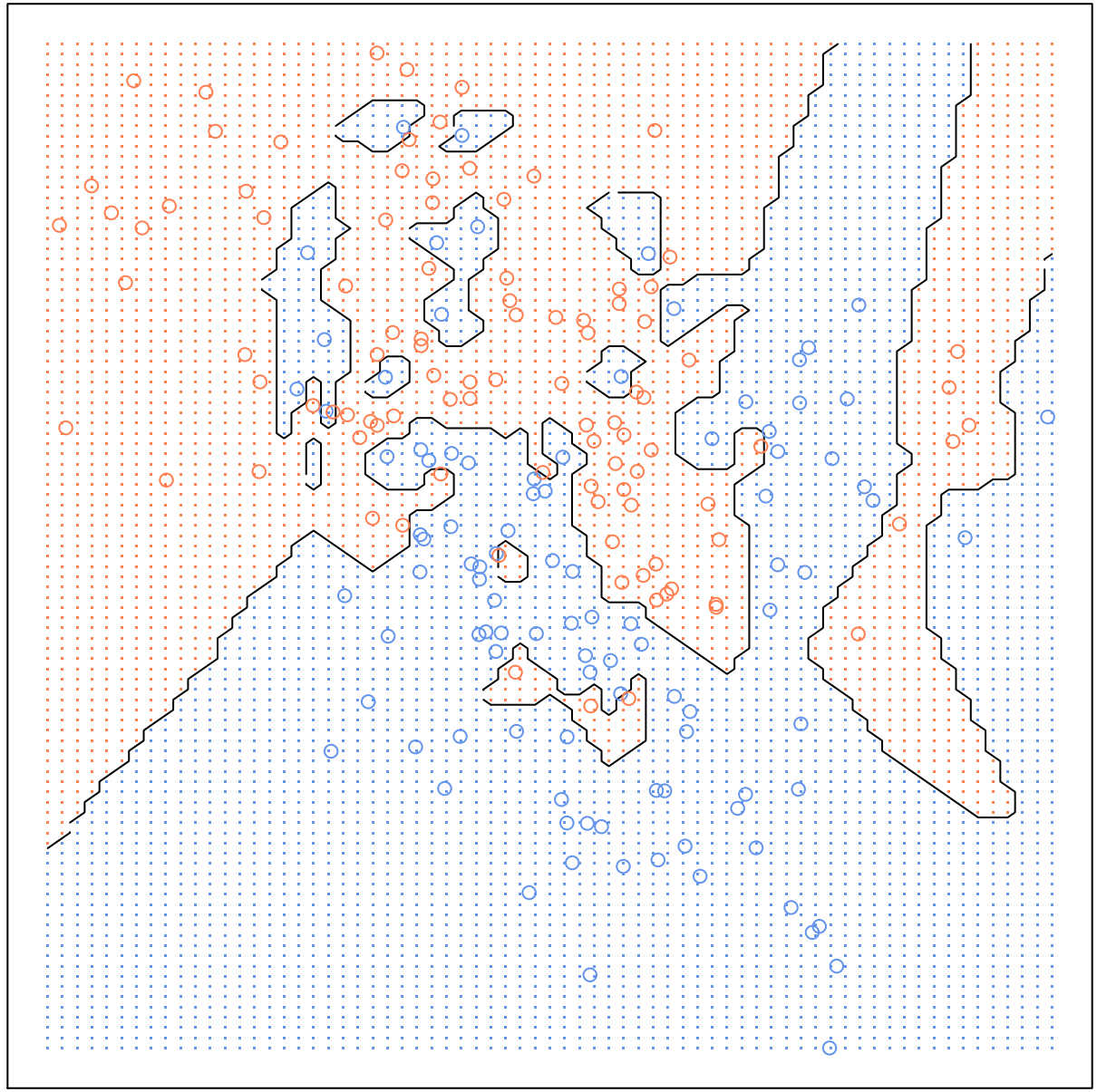
- Cuando queremos predecir nuevos ejemplos, se hace todo el trabajo:
  1. Se calculan las distancias de las nuevas observaciones a predecir con el conjunto de entrenamiento.
  2. Se ordenan y escogen las k observaciones más cercanas.
  3. Se predice la nueva observación como la clase mayoritaria en dichas observaciones.
- Hay que elegir el valor de k y una distancia.

Predicción y efecto de k

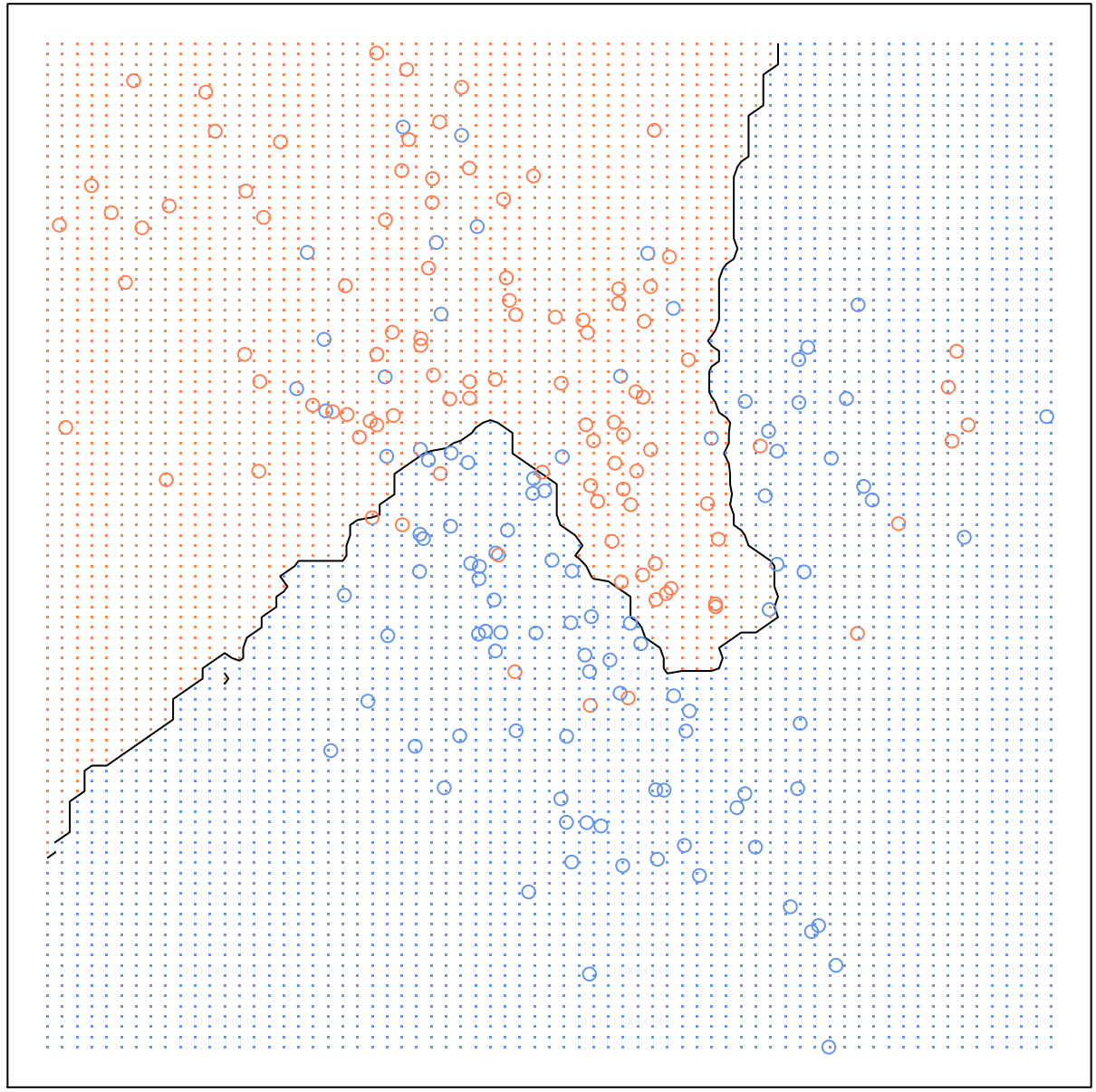


Línea continua $k = 3$, línea discontinua $k = 5$. Fuente

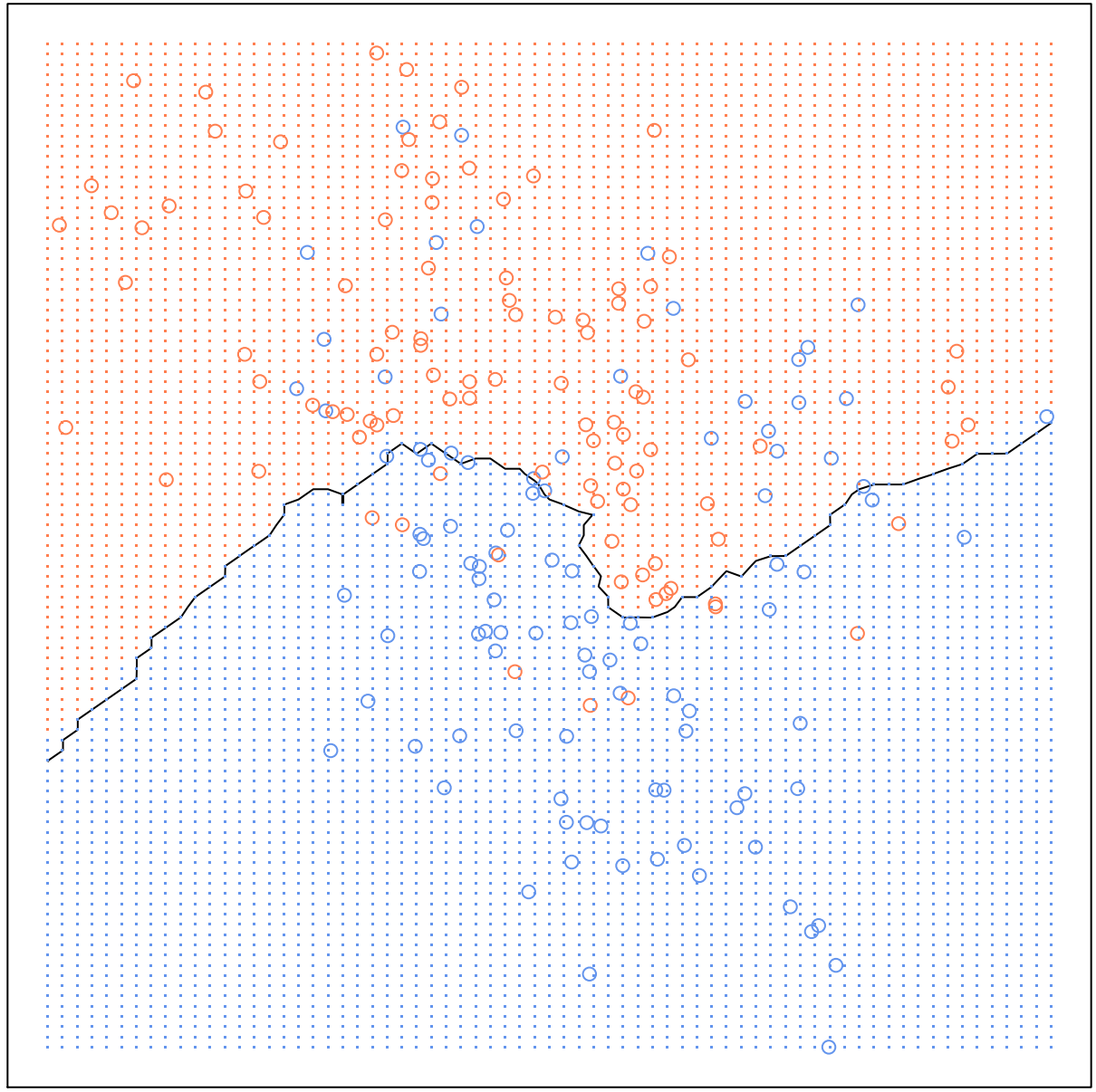
Vecinos próximos, k=1



Vecinos próximos, k=15



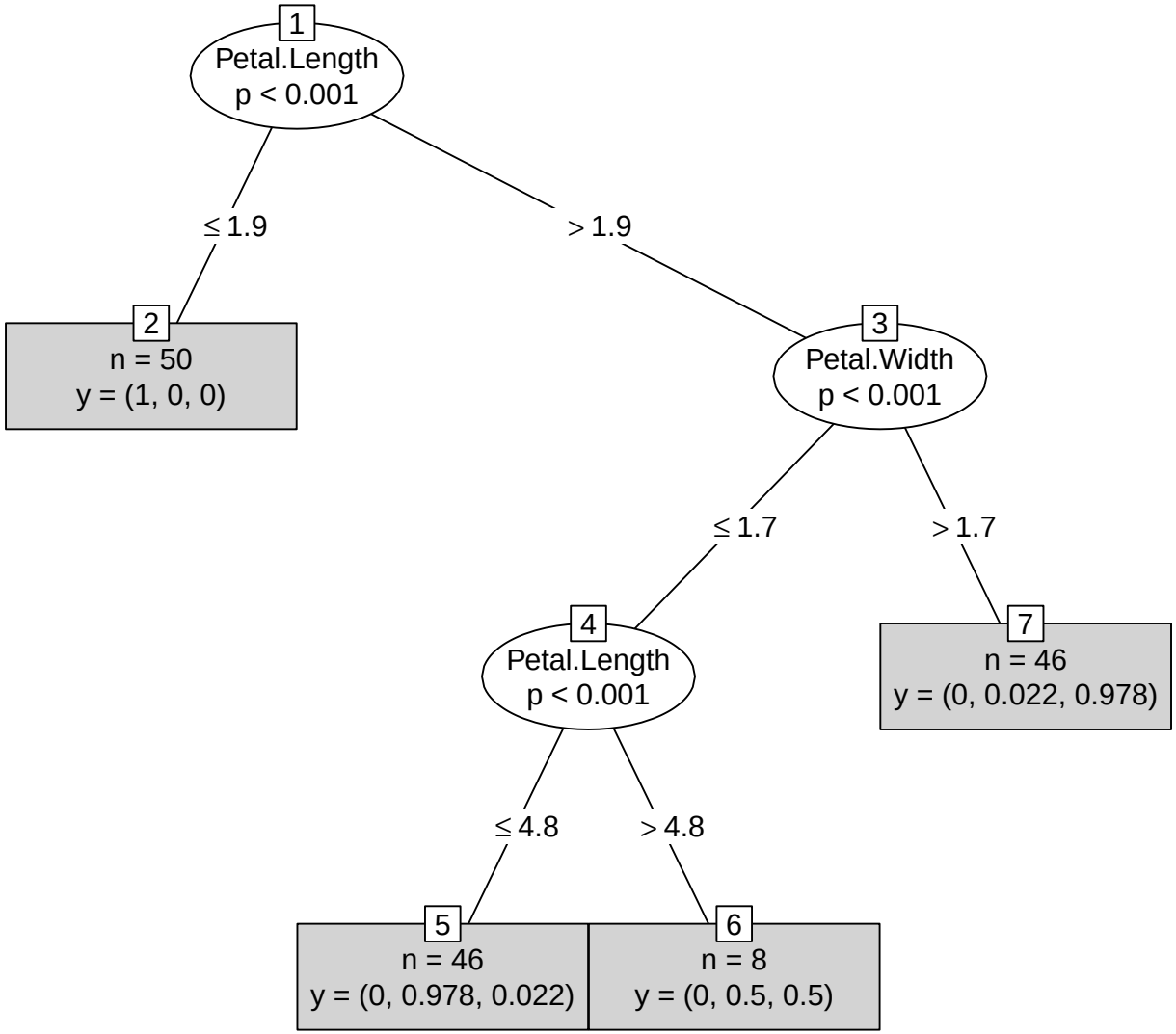
Vecinos próximos, k=50



Árboles de decisión

- Crea modelos de regresión o clasificación en forma de árbol.
- Para construir un árbol, se utiliza el siguiente algoritmo:
  1. Elegir la variable que más información proporciona sobre la respuesta.
  2. Particionar el conjunto de datos, de acuerdo a un valor de la variable anterior.
  3. Para cada uno de los dos subconjuntos, repetir el procedimiento.
  4. Paramos cuando ya nos queda una única observación o según un criterio de parada.

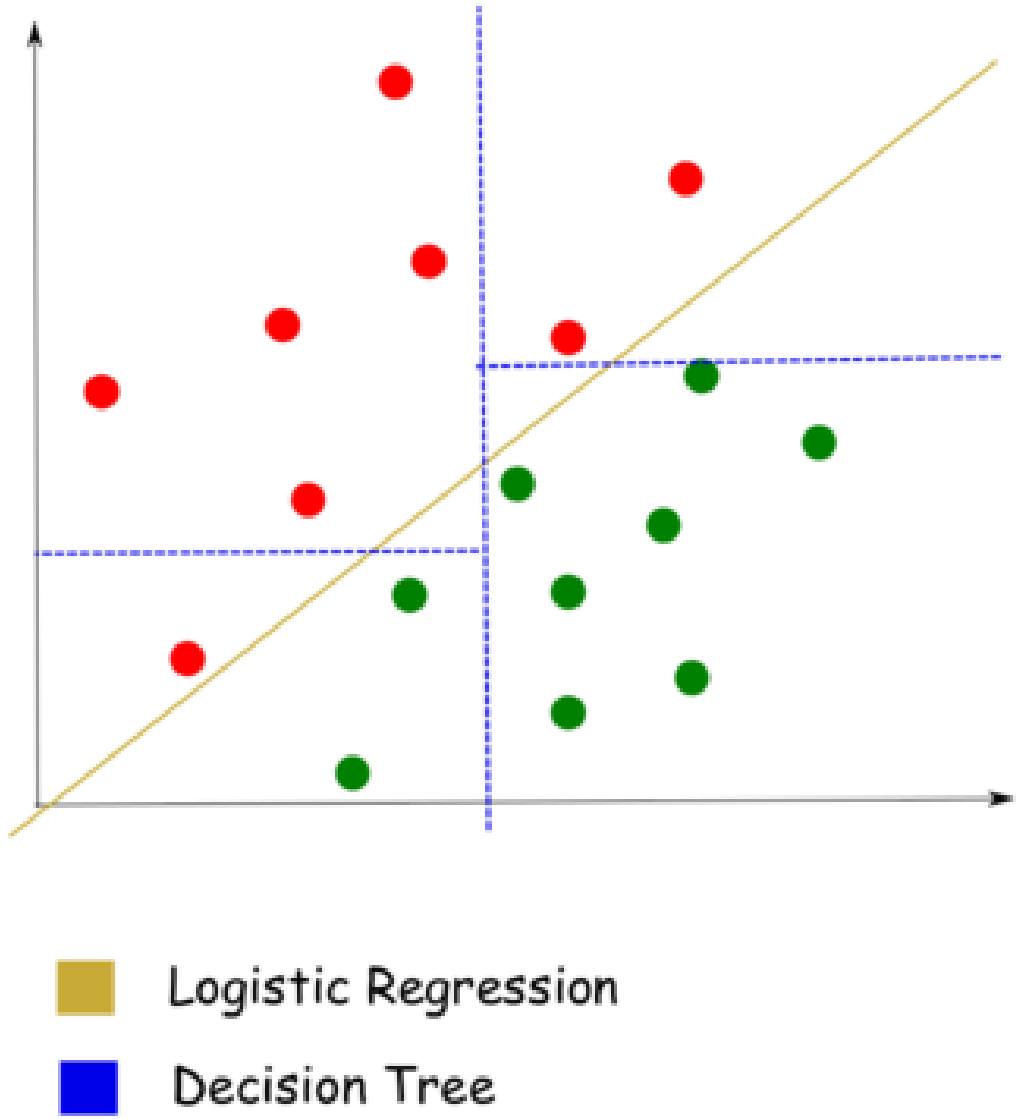
Árboles de decisión: ejemplo



Arboles de decisión: interpretabilidad

- Hasta ahora solo hablamos del error de como medida de calidad de un modelo.
- Sin embargo, a veces es también muy importante que el modelo sea fácil de entender e interpretar.
- Esta es una de las principales ventajas de los árboles de decisión sobre el resto de modelos.
- Otras ventajas son:
  - Es capaz de trabajar con variables categóricas sin necesidad de codificación.
  - Rápido de crear, incluso con muchas observaciones y variables.
- La principal desventaja es que es necesario controlar el crecimiento del árbol, ya que son muy propensos al sobreajuste.

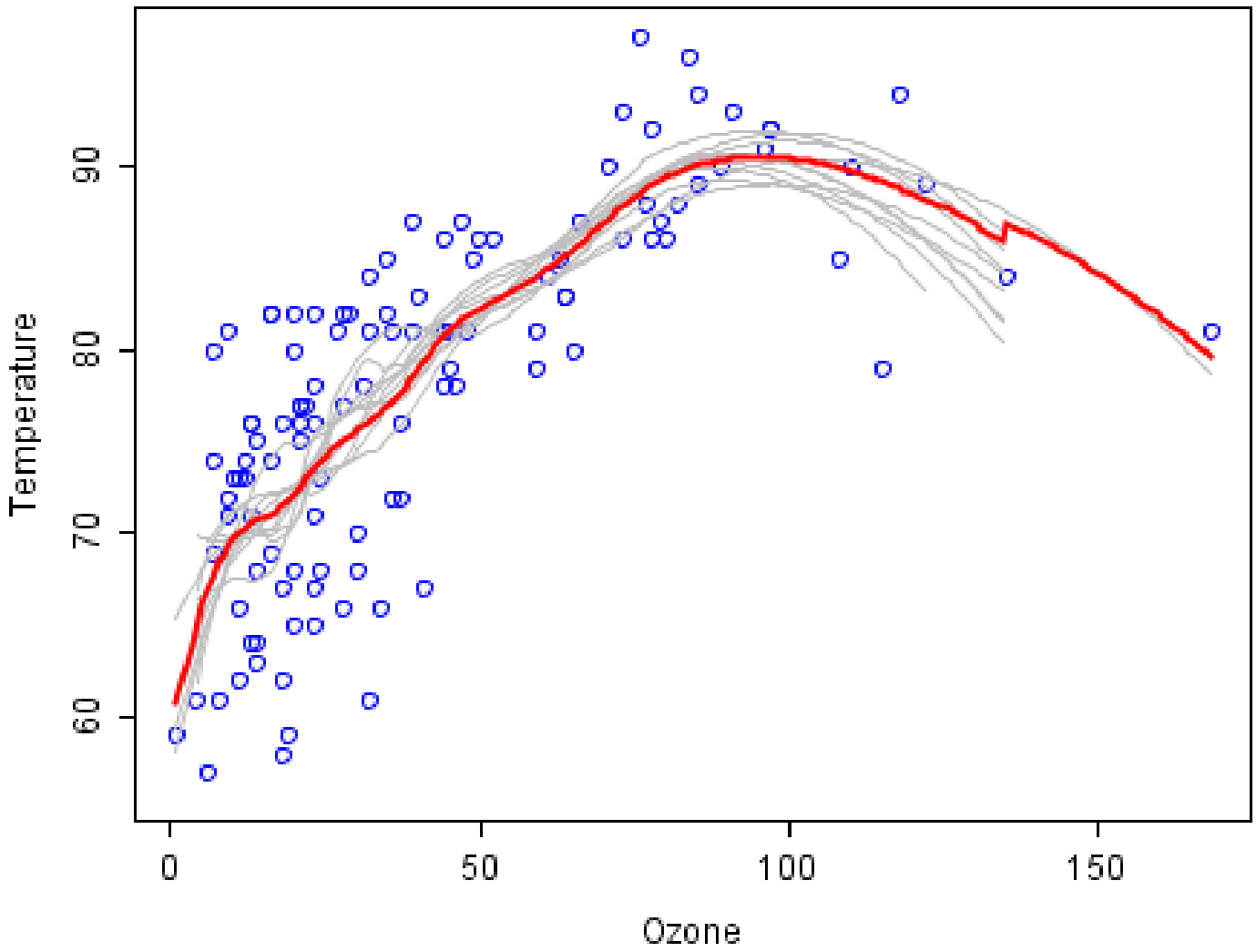
Árboles de decisión vs regresión logística



Métodos de conjunto: *bagging*

- La idea de combinar varios modelos es muy natural.
- Queremos combinar clasificadores simples y además que tengan diversidad, es decir, que no aprendan todos lo mismo.
- La diversidad la podemos introducir entrenando cada clasificador con una pequeña variación del conjunto de entrenamiento.
- Dado un conjunto de entrenamiento, *bagging* crea m nuevos del mismo tamaño mediante muestreo uniforme con reemplazamiento.
- Se puede ver que cada uno de estos m conjuntos tiene un 63% de muestras no repetidas.

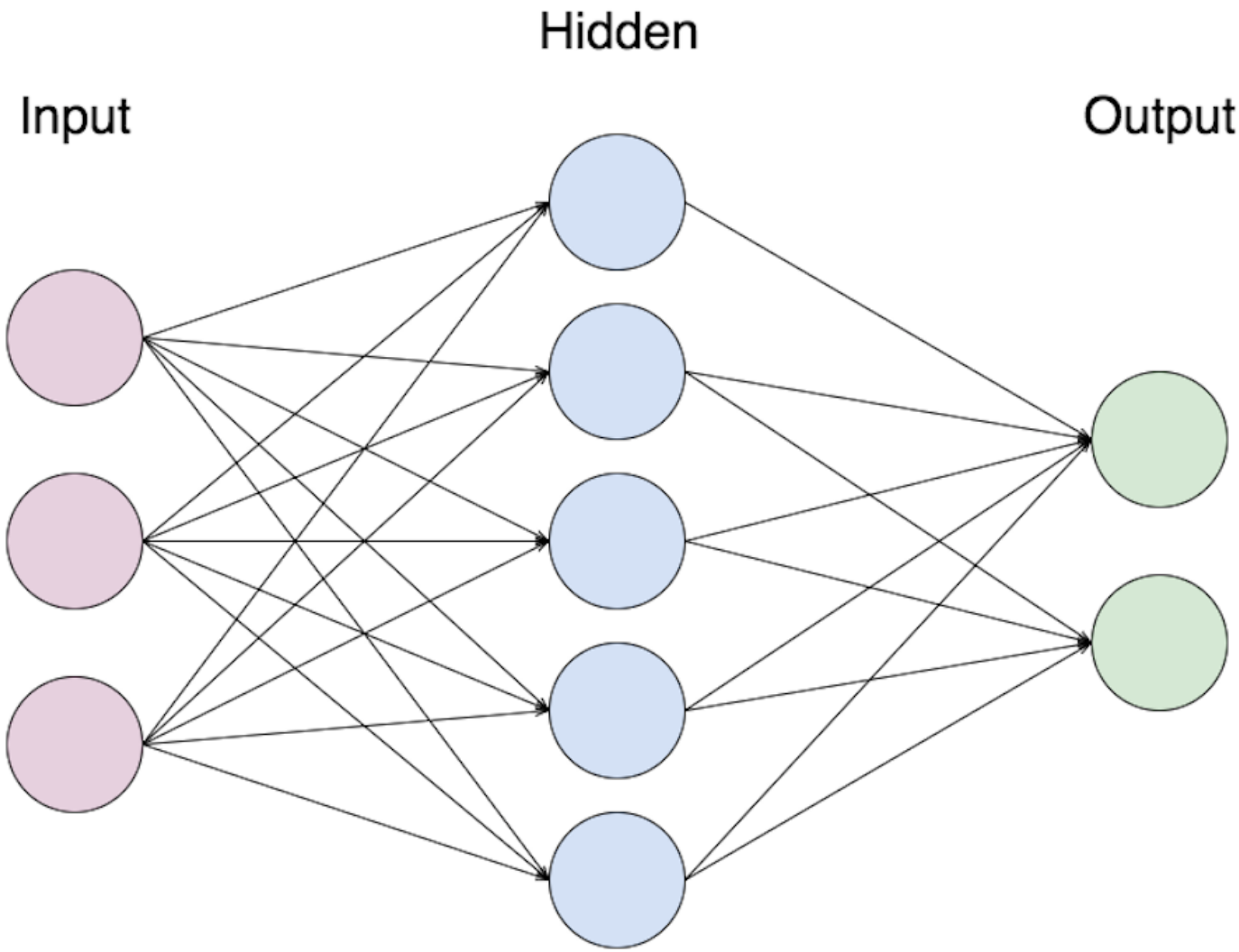
Métodos de conjunto: ejemplo



Random Forest

- Random Forest combina los árboles de decisión con la técnica de `bagging` para generar un conjunto de modelos.
- Para cada uno de las particiones de *bagging*, se entrena un árbol de decisión.
- Finalmente se combinan las predicciones de los árboles, por ejemplo con el voto mayoritario.
- Con respecto a los árboles de decisión, estos conjuntos:
  - Pierden la capacidad interpretativa, aunque son capaces de dar una medida de lo importantes que son las variables.
  - Mejoran notablemente el error de predicción.
  - Solucionan el problema de sobreajuste.
  - Escogen un subconjunto aleatorio de variables en cada partición, para incrementar todavía más la diversidad.

Redes neuronales



Redes neuronales (cont.)

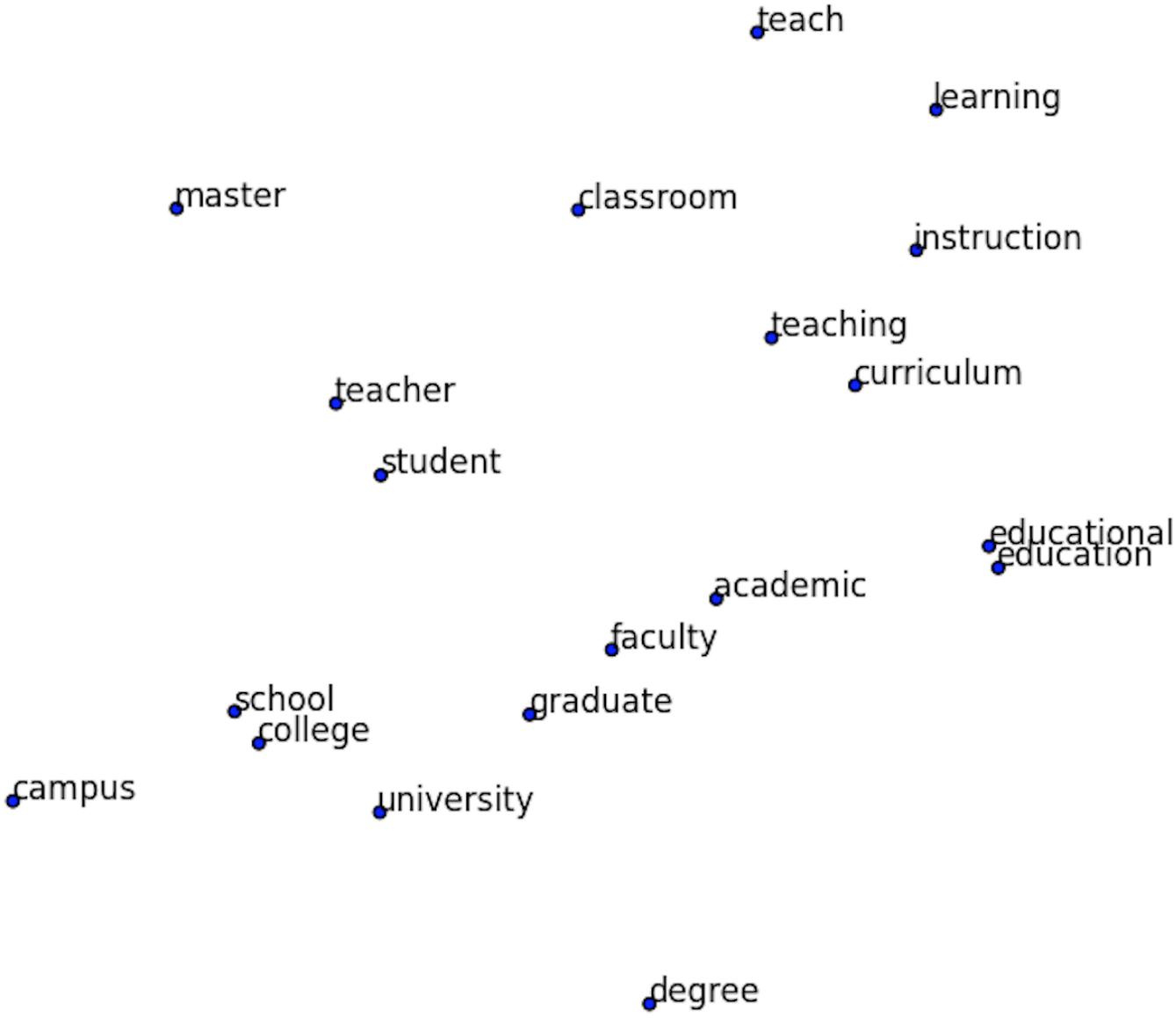
- Inspiradas en el funcionamiento de los sistemas nerviosos biológicos.
- Están compuestas de distintas capas, que a su vez están compuestas de neuronas.
- Las neuronas toman unos datos de entrada, los agregan y les aplican una función no lineal, devolviendo una única salida.
- Se pueden usar para problemas de regresión y de clasificación modificando ligeramente la capa de salida.
- Muy flexibles, se pueden añadir capas si la relación a aproximar es muy compleja.
- Cuantas mas capas son necesarios más datos para ajustarlas correctamente.

Representación

- Las redes neuronales se pueden ver como un modelo en dos pasos:
  1. Las capas intermedias crean una nueva "representación" de las variables de entrada.
  2. La capa de salida ajusta un modelo líneal simple (regresión lineal, regresión logística).
- Esta representación interna puede ser útil para entender nuestros datos.

Ejemplo word2vec

- Red neuronal que se entrena con grupos de n palabras con el objetivo de intentar predecir la palabra del medio: *quick brown* **fox** *jumps over*
- Podemos proyectar el resultado de las capas intermedias en 2D.
- Se ve como la distancia entre palabras representa su parecido en cuanto a significado semántico.
- Dicha representación es útil para otro tipo de tareas relacionadas con procesamiento de texto: análisis de sentimiento, categorización, etc.


teach
learning
master
classroom
instruction
teaching
curriculum
teacher
student
educational
education
academic
faculty
school
graduate
college
campus
university
degree

Índice

Diseño iterativo

- Si el resultado del modelo no es tan bueno como nos gustaría no hay que perder la esperanza, ya es muy común que el análisis se realize de forma iterativa.

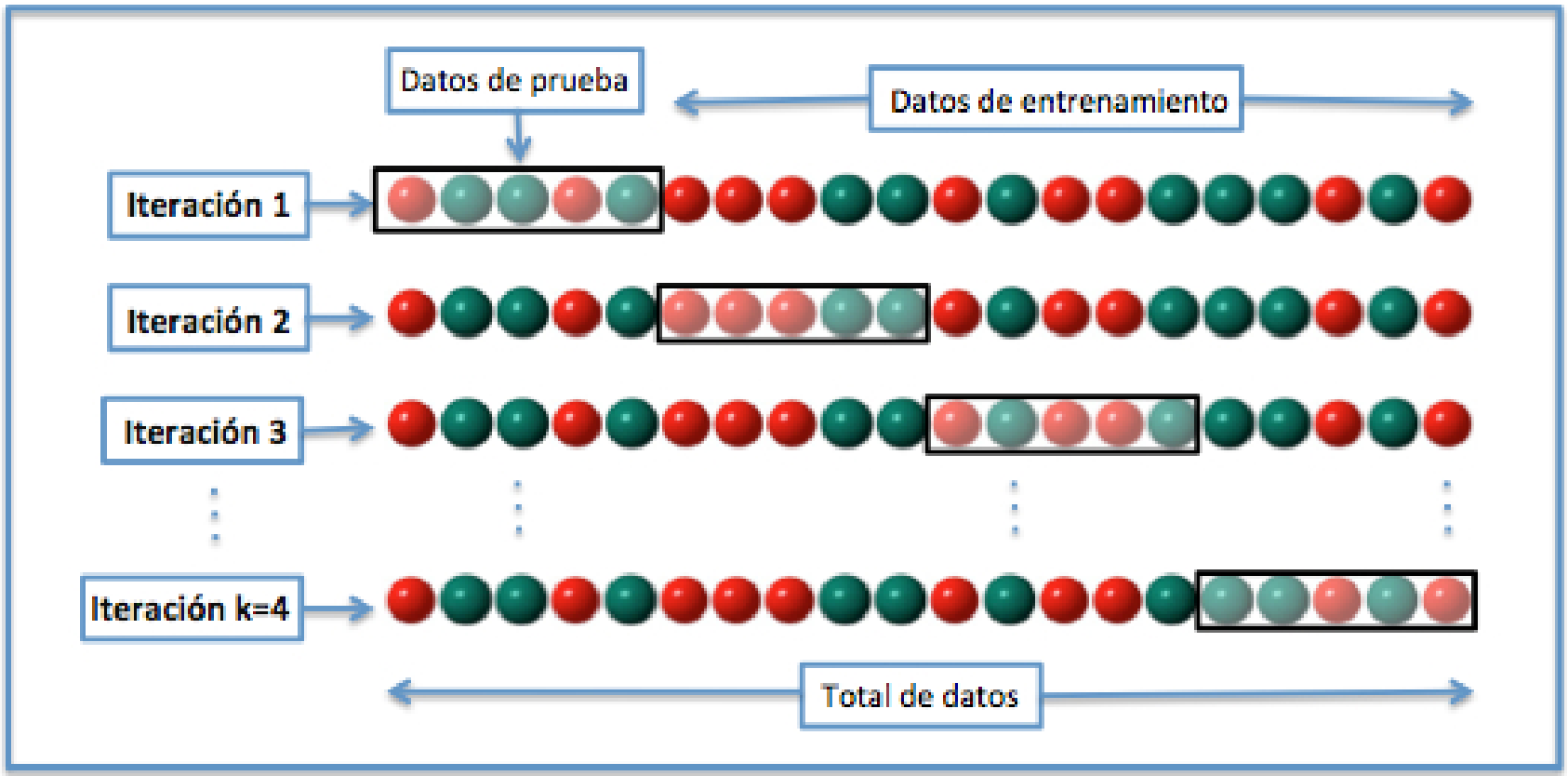
- Podemos probar varias cosas:
  1. Ajustar mejor el modelo: muchos de los modelos que hemos visto tienen parámetros que influyen mucho en el rendimiento (más a continuación).
  2. Probar otros modelos: una opción muy común al hacer el análisis de unos datos es comenzar con modelos más simples e ir moviéndonos hacia modelos más complejos.
  3. Obtener más datos: no siempre es posible, pero en general con más datos se consigue mejor resultado que un algoritmo más inteligente.

Conjunto de validación

- Para escoger los valores de los parámetros de un modelo, podemos mirar el error de entrenamiento para varios valores y escoger el de menor error.
- Al hacer esto, hemos visto que el error de entrenamiento es una medida sesgada y por tanto los valores de los parámetros pueden no funcionar bien en el conjunto de test.
- Por tanto, necesitamos un tercer conjunto llamado conjunto de **validación** donde se va a medir el error para cada valor de los parámetros.
- Finalmente, escogemos el que tenga menor error de validación.
- El conjunto de test sigue sin tocar, y se usa como antes para estimar el error de generalización.

Validación cruzada

- Si no tenemos muchos datos y no queremos hacer un conjunto de validación, podemos usar validación cruzada.



Fuente

Selección de variables y reducción de dimensión

- Una forma de reducir el tiempo de computación de los algoritmos de aprendizaje es reduciendo el número de variables.
- Esto se puede hacer de dos maneras:
  1. Seleccionando únicamente un subconjunto de las variables de acuerdo a algún criterio de relevancia para predecir la respuesta.
  2. Proyectando nuestros datos a un espacio de menor dimensión.
- La primera opción tiene la ventaja de que las variables siguen siendo interpretables, es decir, tienen significado semántico.
- El método más común de proyección es PCA o análisis de componentes principales.

Índice

Datos a gran escala

- Hemos visto que la mayoría de los conjuntos de datos se pueden tratar en un portátil o servidor de gama media–alta.
- Sin embargo en ocasiones tenemos conjuntos de datos que:
  1. directamente no se pueden cargar en memoria
  2. se pueden cargar pero el proceso de aprendizaje es prohibitivamente lento.
- Lo primero es preguntarse si de verdad hacen falta tantos datos, ya que no todos los problemas de aprendizaje son suficientemente complejos para requerir millones de ejemplos.
- Ejemplo: clasificar imágenes de dígitos vs clasificar imágenes de objetos.
- Si lo anterior falla, tenemos que recurrir a técnicas y herramientas de computación de alto rendimiento.

Paralelización

- Se refiere a ejecutar un proceso a la vez en múltiples procesadores.
- No siempre es posible paralelizar un algoritmo, al menos no de forma fácil.
- Existen dos tipos principales que nos interesan:
  - A nivel de operaciones: paraleliza la ejecución de las operaciones algebraicas.
  - A nivel de datos: nuestro problema se puede dividir en tareas independientes, que se pueden ejecutar en varios procesadores a la vez.
- En cualquier caso esto no soluciona el problema de la memoria RAM.

Computación distribuida

- Si tenemos disponible un clúster de servidores, podemos intentar aprovecharlos todos para hacer nuestros cálculos.
- Esto es trivial si, por ejemplo:
  1. Estamos buscando parámetros óptimos.
  2. Estamos haciendo grupos de modelos, como `randomForest`.
  3. Estamos ajustando un modelo para distintas particiones de entrenamiento y test.
- En otro caso, tenemos que usar herramientas que faciliten el trabajo como Hadoop o Spark.
- Hadoop consiste principalmente en un sistema de ficheros distribuido, es decir, los datos se dividen en bloques y se reparten en los nodos, que es donde se realiza el cálculo.