Conceptos generales de aprendizaje supervisado

Curso de aprendizaje automático para el INE

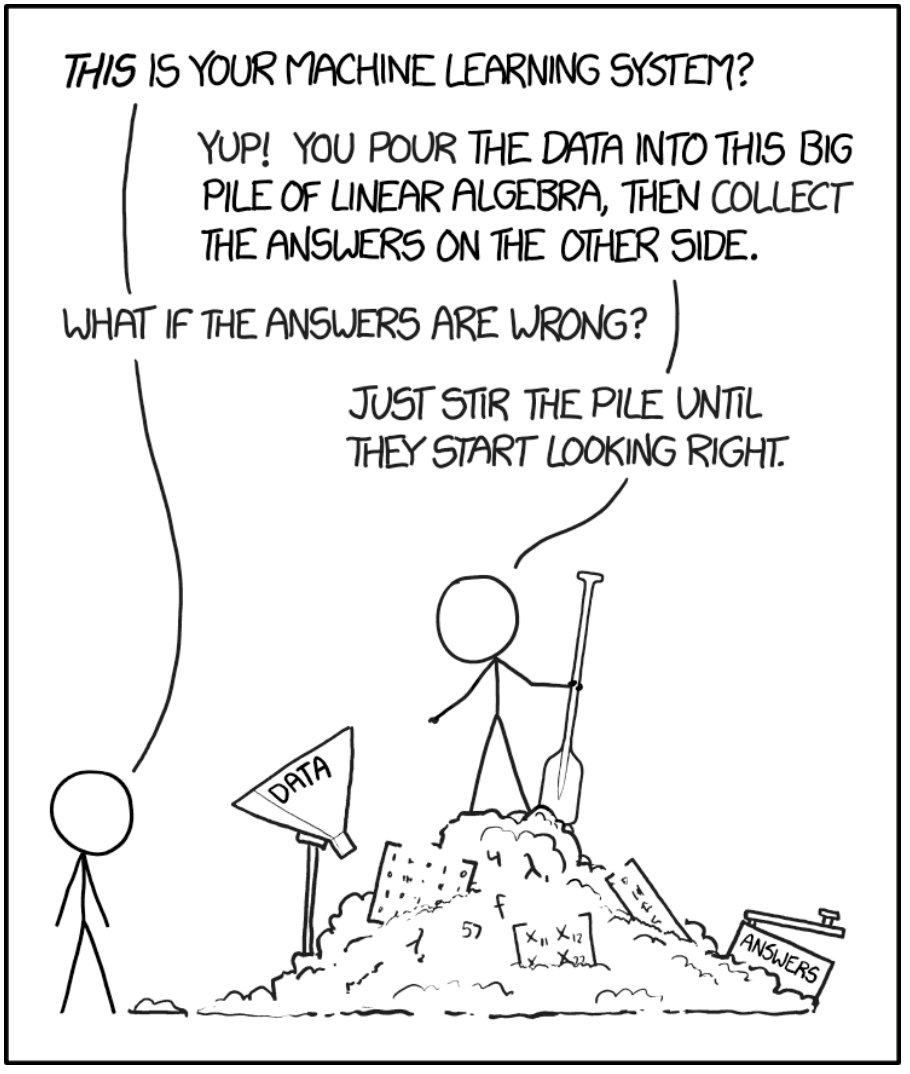
Alberto Torres Barrán

2019-03-21

¿Qué es el Aprendizaje Automático?

De la Wikipedia:

*Machine learning is a subfield of **computer science** that evolved from the study of **pattern recognition** and computational learning theory in artificial intelligence. In 1959, Arthur Samuel defined machinelearning as a "Field of study that gives computers the ability to learn without being **explicitly programmed**". Machine learning explores the study and construction of algorithms that can learn from and make predictions on **data**.*

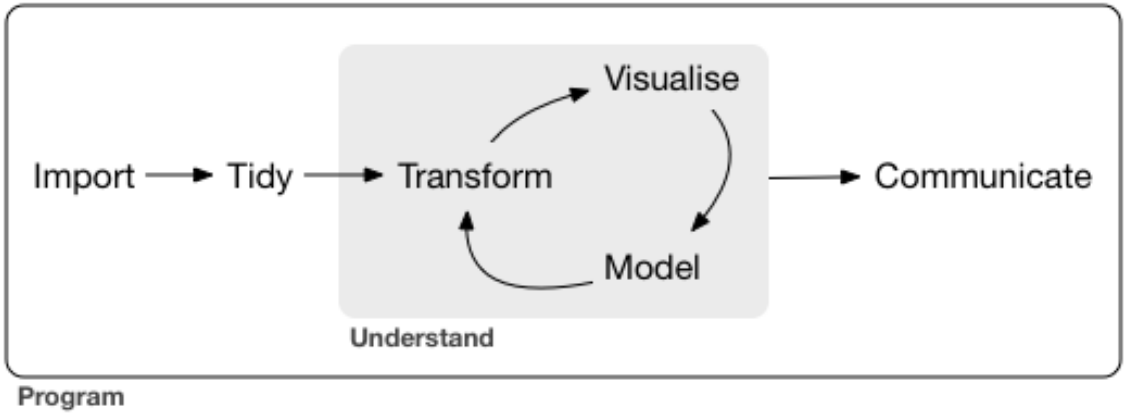


Fuente: xkcd #1838

???


Data Science
Data Mining
Statistics
Artificial Intelligence
Machine Learning

Data science

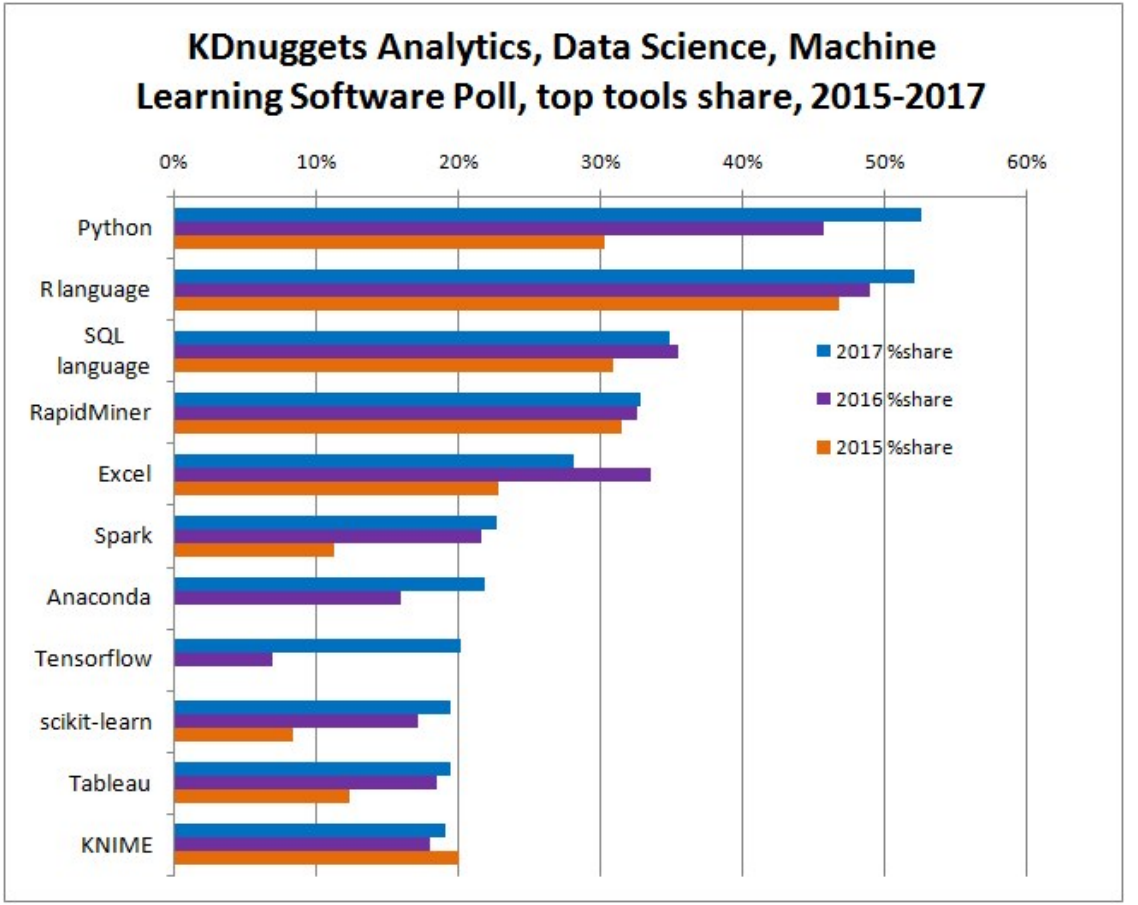


Fuente: R for Data Science

Casos de éxito

- Coches autónomos
- Análisis de imágenes médicas
- Procesamiento de lenguaje natural
- AlphaGo y juegos Atari
- Generación de imágenes
- Sistemas de recomendación

Herramientas



Tipos de aprendizaje

Existen diversos tipos de tareas, dependiendo de la información disponible:

- **supervisado**: tenemos acceso a pares de ejemplos entrada-salida
- **no supervisado**: no tenemos acceso a las salidas
- otros (limitando de alguna forma el acceso a las salidas):
  - *activo*: el algoritmo puede acceder a la salida para nuevos datos de entrada
  - *semi-supervisado*: solo se tienen salidas para algunos datos
  - *refuerzo*: no se tiene el valor de la salida, pero si una indicación de lo lejos o cerca que se encuentra

Referencias

1. Jerome H. Friedman. Data Mining and Statistics: What's the Connection? (1998)
2. Leo Breiman. Statistical Modeling: The Two Cultures (2001)
3. Cross Validated. What is the difference between data mining, statistics, machine learning and AI (2010).
4. Sakthi Dasan Sekar. What is the difference between Artificial Intelligence, Machine Learning, Statistics, and Data Mining (2014)
5. Cross Validated. What exactly is Big Data? (2015)
6. David Donoho. 50 years of Data Science (2015)

Aprendizaje supervisado

- Tenemos disponibles datos con múltiples observaciones:
  - ejemplos (*examples*)
  - muestras (*samples*)
  - ...
- Varias variables por observación:
  - predictores
  - atributos (*atributes*)
  - características (*features*)
  - covariables (*covariates*)
  - variables independientes
  - variables explicativas
  - ...
- Una de ellas es de especial interés:
  - variable respuesta
  - variable dependiente
  - objetivo (*target*)
  - salida (*output*)
  - etiqueta (*label*)
  - ...

Objetivos

1. Predecir el valor de la variable respuesta para nuevas observaciones
2. Obtener información sobre la relación entre las variables independientes y la salida

Tipos de problemas

1. Regresión, si la variable respuesta es continua
2. Clasificación, si la variable respuesta es discreta
3. Otros: por ejemplo,
    - salida continua pero valores enteros
    - salida discreta pero los valores tienen un orden

Aprendizaje estadístico

Dados:

- Espacio de las muestras de entrada: $\mathcal{X}$
- Conjunto de posibles salidas: $\mathcal{Y}$
- Conjunto de **entrenamiento**: $S = \{x_i,\, y_i\}_{i=1}^{n}$, contenido en el espacio $\mathcal{X} \times \mathcal{Y}$

Objetivo:

- Aprender una regla de predicción (hipótesis), $h : \mathcal{X} \to \mathcal{Y}$

Asumimos:

- Los ejemplos se han generado por una distribución de probabilidad desconocida $\mathcal{P}$
- Existe una función de pérdida $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ que mide como de lejos se encuentra $h(x)$ de y
- El conjunto posible de hipótesis $(\mathcal{F})$ es finito

Minimización del riesgo empírico

Elegir h tal que minimice el riesgo esperado

$$R(h) = \int_{\mathcal{X} \times \mathcal{Y}} L(h(x), y)\, dP(x, y)$$

Problema: cómo podemos calcular R si P es desconocida?

Podemos evaluar la función de pérdida en el conjunto S (riesgo empírico):

$$\hat{R}(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(x_i), y_i)$$

Si n suficientemente grande, esperamos que $\hat{R}(x) \sim R(x) \rightarrow$ minimizar el riesgo empírico es una buena aproximación de minimizar el riesgo esperado

Descomposición del error

Minimizador del riesgo:

$$h^* = \arg\min_{h \in \mathcal{F}} \; R(h)$$

Minimizador del riesgo empírico:

$$\hat{h}^* = \arg\min_{h \in \mathcal{F}} \; \hat{R}(h)$$

Riesgo de Bayes o error de Bayes:

$$R^* = \inf_{h} \; R(h)$$

Nota: sobre todas las funciones $h : \mathcal{X} \to \mathcal{Y}$, no solo las contenidas en $\mathcal{F}$!!

La differencia entre el riesgo y el error de Bayes es:

$$R(h) - R^* = \underbrace{\left(R(h) - R(\hat{h}^*)\right)}_{\text{error optimización}} + \underbrace{\left(R(\hat{h}^*) - R(h^*)\right)}_{\text{error estimación}} + \underbrace{\left(R(h^*) - R^*\right)}_{\text{error aproximación}}$$

Error optimización: como de buena es la optimización que llevó a la hipótesis h, relativa a al óptimo del riesgo empírico

- Disminuye al mejorar el algoritmo de optimización

Error de estimación: surge por aproximar el riesgo esperado con el riesgo empírico

- Disminuye si aumentamos el conjunto de datos de entrenamiento n

Error de aproximación: surge por aproximar la mejor función posible por la mejor función dentro de $\mathcal{F}$

- Disminuye si reemplazamos $\mathcal{F}$ por otra clase más flexible

Ejemplo

- Elegimos $\mathcal{F}$ como las clase de funciones del tipo $f(x) = w_0 + x^T w$
- Función de pérdida: $L(y, f(x)) = (y - f(x))^2$
- Riesgo empírico:

$$R(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w_0 + x_i^T w)^2$$

Regresión lineal

Dado el conjunto de entrenamiento $S = \{y_i, x_i\}_{i=1}^n$

- Agrupamos todos los ejemplos de entrada x_i en una matrix $\mathbf{X}$ de tamaño $n \times d$
- Agrupamos todas las salidas en un vector columna y de tamaño $n \times 1$

Expresamos el riesgo empírico en notación matricial:

$$R(w) = (y - \mathbf{X}w)^T(y - \mathbf{X}w)$$

Gradiente:

$$\nabla_w R(w) = \mathbf{X}^T(y - \mathbf{X}w) = \mathbf{X}^T y - \mathbf{X}^T\mathbf{X}w$$

Minimizamos el riesgo empírico:

$$\nabla_w R(w) = 0 \quad \Rightarrow \quad w^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$$

Recuperamos mínimos cuadrados ordinarios!

Posibles problemas del modelo:

- Teóricos:
    1. asumimos que y depende linealmente de x
    2. asumimos que el modelo está especificado correctamente (no faltan variables)
- Numéricos:
    1. hay menos variables que observaciones
    2. no hay dos variables con correlación perfecta

Selección de modelos

- Para medir la calidad del modelo, podemos calcular el riesgo o error empírico en el conjunto de entrenamiento
- Este error se puede disminuir de forma casi arbitraria aumentando la complejidad de la clase de funciones
- **Ejemplo**: en el caso de la regresión lineal, podemos añadir nuevas variables que sean expansiones polinómicas de las ya existentes
- Interesa el **error de generalización**, es decir, el error en nuevas observaciones no usadas para entrenar el modelo
- Partir los datos iniciales en dos conjuntos:
    1. conjunto de entrenamiento
    2. conjunto de test

Equilibrio sesgo-varianza

- Asumimos que los datos han sido generados por

$$Y = f(X) + \epsilon$$

con $\mathbb{E}[\epsilon] = 0$ y $\text{Var}(\epsilon) = \sigma^2$

- El error esperado de un estimador $\hat{f}\,(X)$ en el punto x (usando pérdida cuadrática) es

$$\text{EPE} = \mathbb{E}[(Y - \hat{f}\,(x))^2]$$

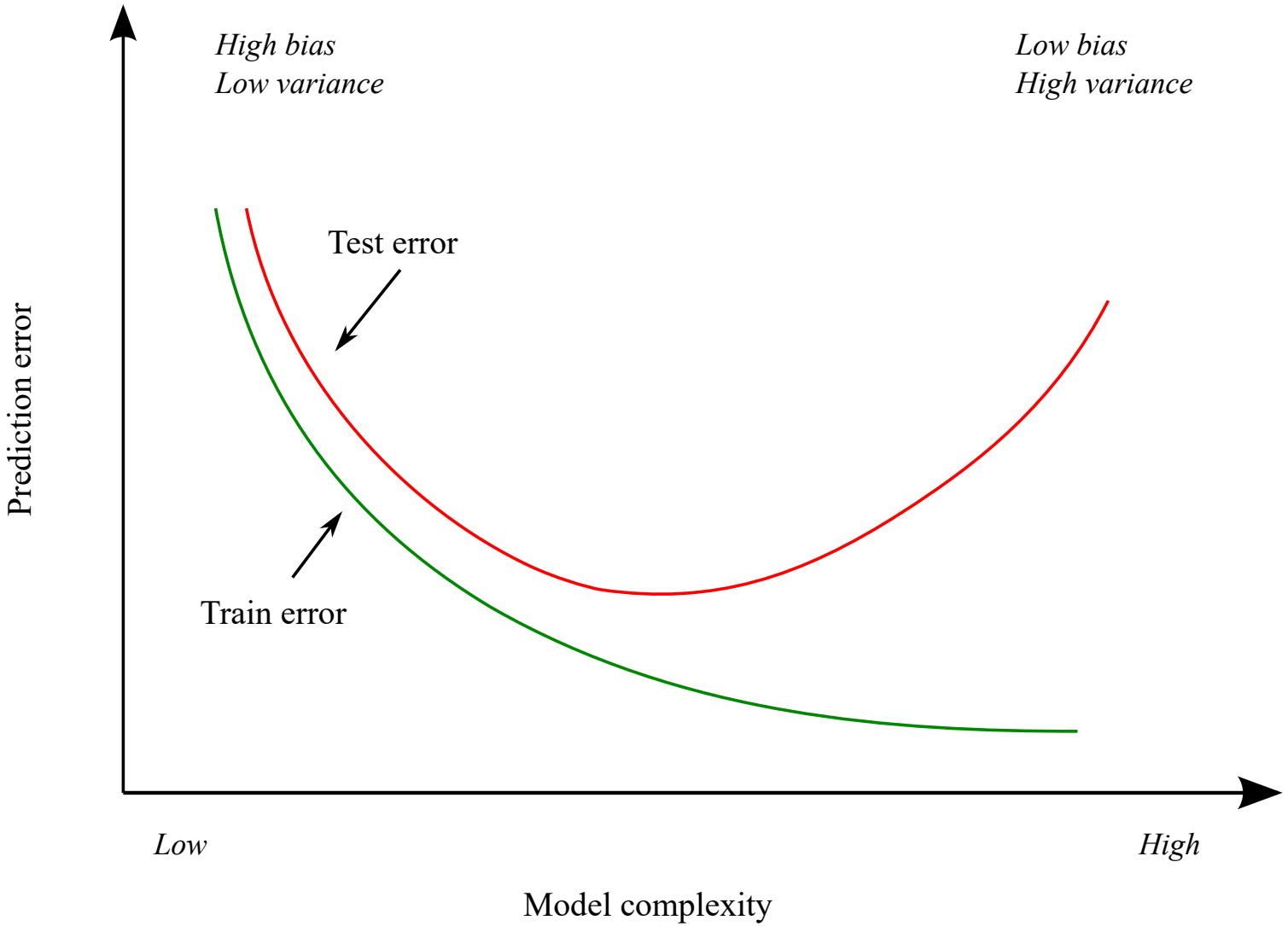
- Podemos descomponerlo en:

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


Low Variance
High Variance
Low Bias
High Bias

Sobreajuste

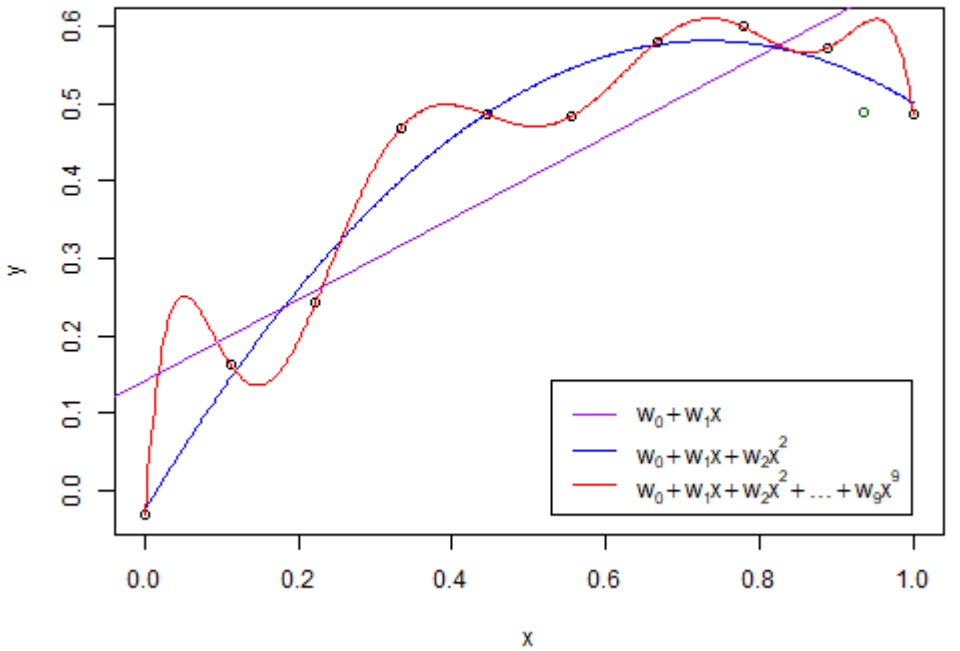
- Los términos de sesgo y varianza son opuestos: si disminuimos uno aumenta el otro y viceversa
- El término de ruido es inherente a los datos
- Si el modelo es muy simple, el estimador está sesgado y no se ajusta bien a los datos (infraajuste)
- Si el modelo es demasiado complejo, es muy sensible a pequeñas variaciones en los datos
- Además, el error de test será mucho más alto que el error de entrenamiento (**sobreajuste**)
- **Solución**: encontrar un equilibrio que minimice el error en el conjunto de test

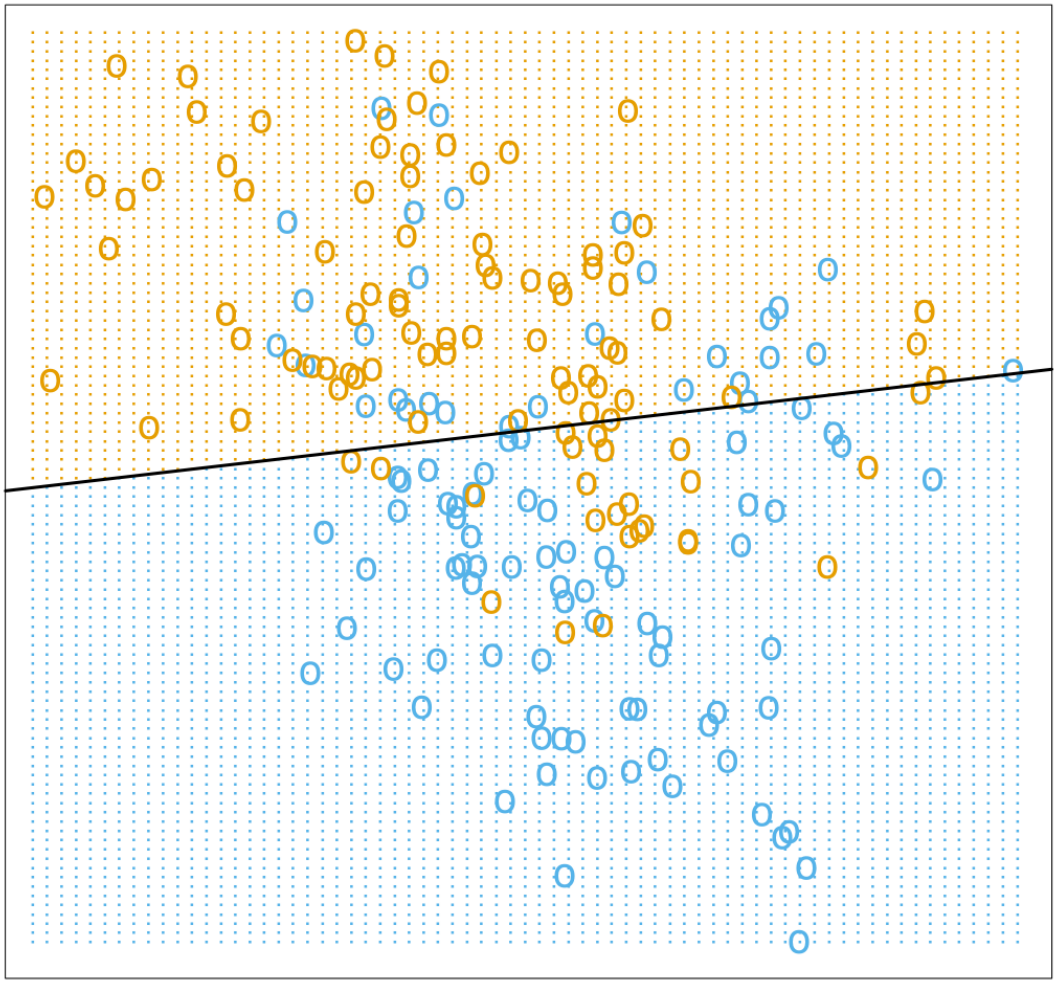

High bias
Low variance
Low bias
High variance
Test error
Train error
Prediction error
Low
High
Model complexity

Simulación

```
set.seed(1)
n <- 10
x <- seq(0, 1, length.out = n)
y <- 1.5*x - x^2 + rnorm(n, 0, 0.05)
data <- data.frame(x=x, y=y)

x_new <- seq(0, 1, length.out=500)
newdata <- data.frame(x=x_new)

fit1 <- lm(y ~ x + I(x^2), data=data)
fit2 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5)
                 + I(x^6) + I(x^7) + I(x^8) + I(x^9),
           data=data)
fit3 <- lm(y ~ x, data=data)

y_pred1 <- predict(fit1, newdata=newdata)
y_pred2 <- predict(fit2, newdata=newdata)

ntest <- 1
xtest <- runif(ntest)
ytest <- 1.5*xtest - xtest^2 + rnorm(ntest, 0, 0.05)
```

```
plot(data)
lines(x_new, y_pred1, col="blue")
lines(x_new, y_pred2, col="red")
abline(fit3, col="purple")
points(xtest, ytest, col="darkgreen")
legend("bottomright",
       c(expression(w[0] + w[1]*x),
         expression(w[0] + w[1]*x + w[2]*x^2),
         expression(w[0] + w[1]*x + w[2]*x^2 + ldots + w[9]*x^9)),
       lty=1, lwd=1.5, col=c("purple", "blue", "red"), inset=0.04)
```





Ejemplo de clasificación en 2 dimensiones [ESL]

Vecinos próximos

- Modelo sencillo que usa las observaciones cercanas a x para realizar la predicción:

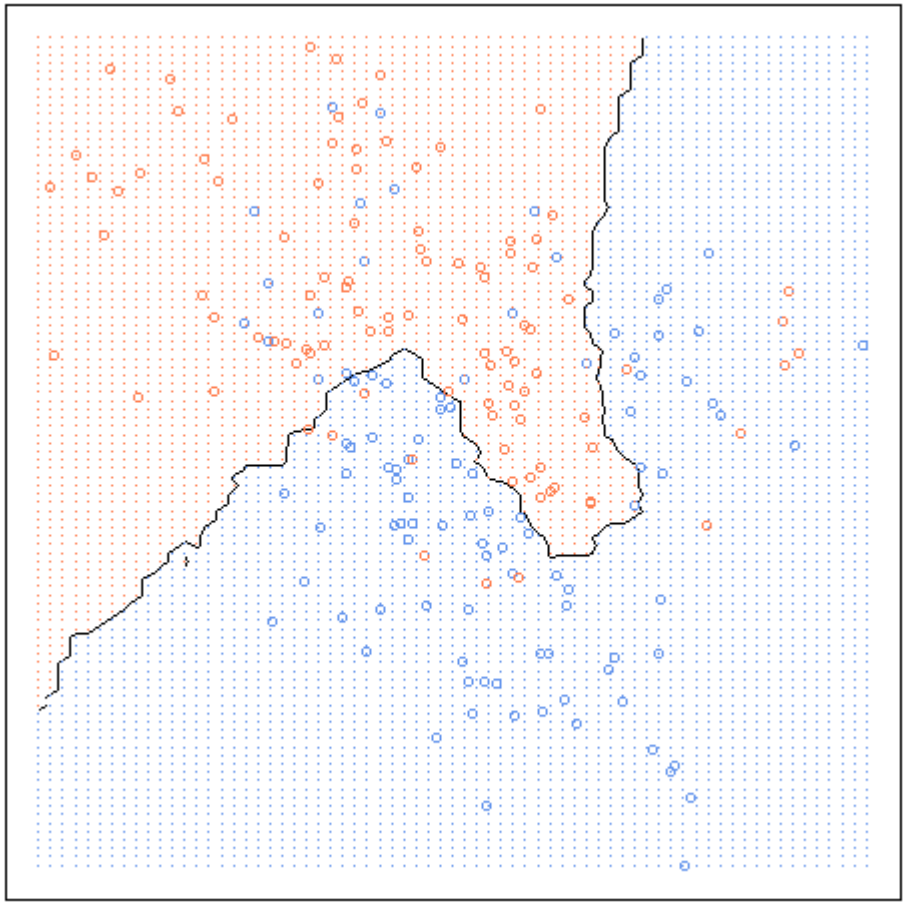
$$f(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

  donde $N_k(x)$ son las k observaciones más cercanas

- Necesaria una métrica (por ej. distancia euclidea)
- Se puede usar tanto para problemas de clasificación como regresión
- Muy sensible al valor de k

```r
library(class)

n <- nrow(iris)

# muestreo aleatorio
idx <- sample(n, n*0.75)

# partir en conjuntos de entrenamiento y test
train <- iris[idx, ]
test <- iris[-idx, ]

# separar variables indenpendientes de la clase (variable respuesta)
# entrenamiento
y_train <- train[,  5]
X_train <- train[, -5]

# test
y_test <- test[,  5]
X_test <- test[, -5]

# modelo knn
y_pred <- knn(X_train, X_test, y_train, k=3)

# tasa acierto
mean(y_test == y_pred)*100
```

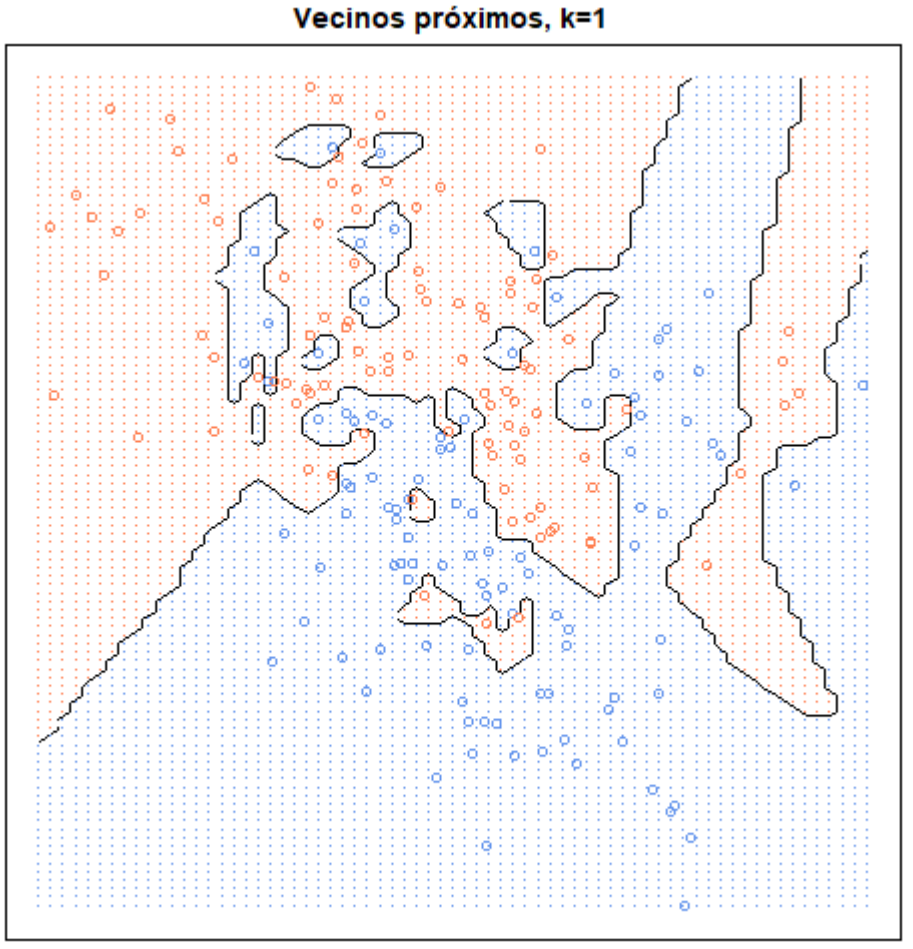
```
## [1] 97.36842
```

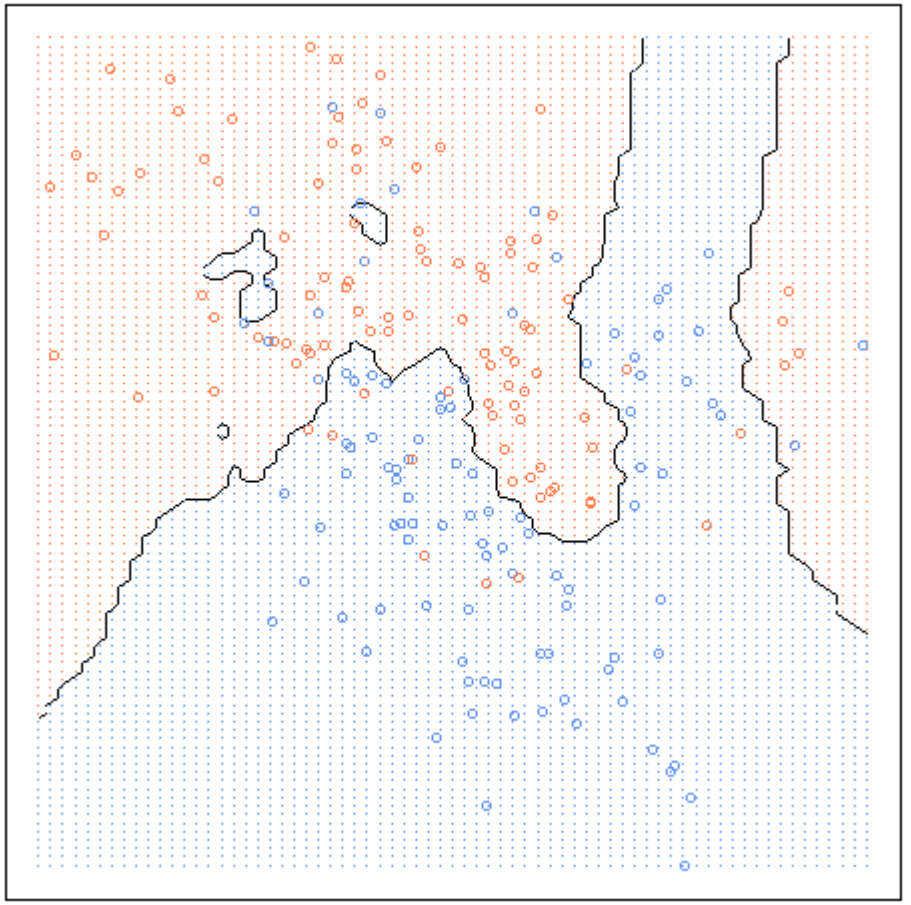

Vecinos próximos, k=15

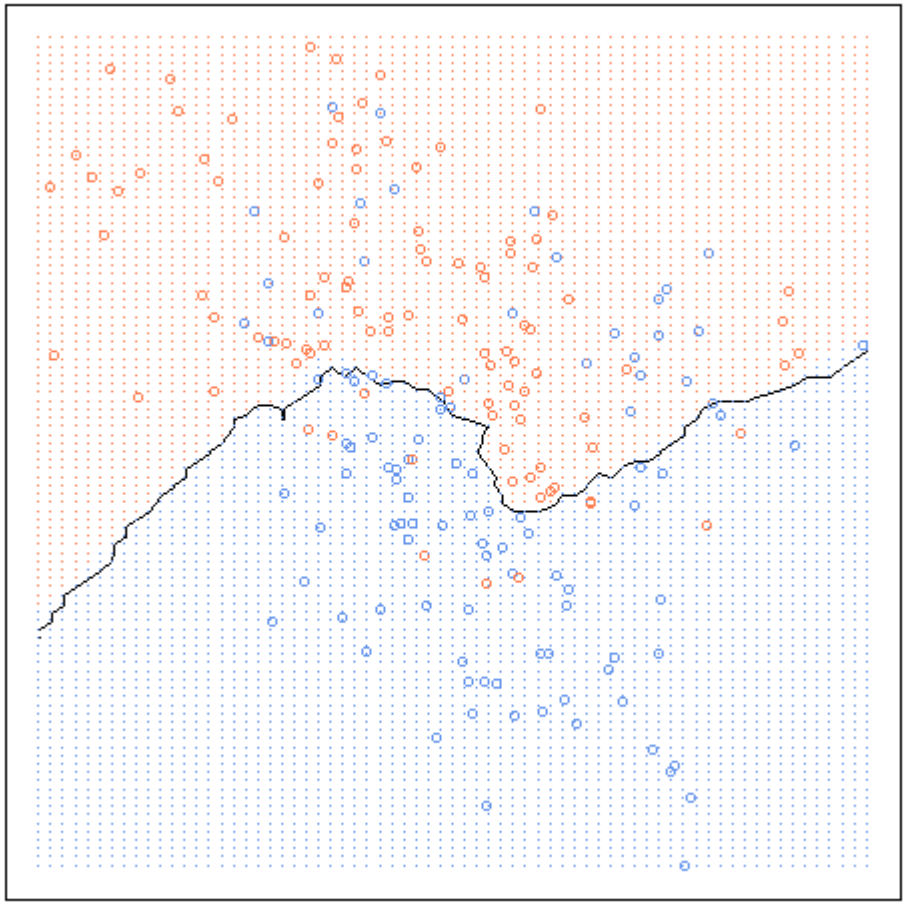
Regresión lineal vs vecinos próximos

- La frontera de decisión de la regresión lineal es suave: tiene poca varianza pero potencialmente mucho sesgo
- k-vecinos próximos no asume ninguna estructura en los datos:
  - la frontera de decisión depende localmente solo de los k puntos más cercanos
  - tiene poco sesgo pero mucha varianza, ya que es muy inestable
- Elegir un modelo u otro depende de los datos del problema

Vecinos próximos: dependencia de k




Vecinos próximos, k=5


Vecinos próximos, k=50

Selección de hiper-parámetros

- k es un **hiper-parámetro** que controla la complejidad del modelo
- Podemos realizar un argumento similar a la comparación con la regresión lineal:
  - Para k grande, la frontera es más suave pero tiene (potencialmente) mayor sesgo
  - Para k pequeño la frontera es muy inestable (mayor varianza), pero menos sesgo
- Nota: usar el error de entrenamiento para elegir el valor de k es mala idea, para $k = 1$ tenemos error 0!!
- Los distintos valores de k se pueden comparar usando el conjunto de test

Conjunto de validación

- Elegir k como el valor que minimiza error de test $\rightarrow$ error de test ya **no** es una buena estimación del rendimiento del modelo en nuevos datos
- Lo mismo ocurre si elegimos la clase de funciones (modelo) usando el error de test
- **Solución**: crear un tercer conjunto, conjunto de validación, para seleccionar hiper-parámetros y comparar modelos
- Finalmente, reportar el error de test como estimación del poder de generalización del modelo
- Existen otras formas que veremos más adelante (por ej. validación cruzada)

Regularización

- A menudo se puede reducir la varianza de un estimador a cambio de introducir un pequeño sesgo
- Este término también puede inducir propiedades en la solución, por ej. *sparsity*
- Para ello limitamos la complejidad del modelo añadiendo a la función de pérdida un término de **regularización**

$$\hat{f} = \arg\min_{f} \{L(y, f(x)) + \lambda J(f)\}$$

- Muchos modelos en aprendizaje automático encajan en este paradigma

Ejemplo

- El estimador de mínimos cuadrados es el mejor estimador no sesgado (mejor = menos varianza)
- Un término de regularización muy habitual es la norma l_2:

$$||w||_2^2 = w^T w$$

- Junto con la función de pérdida de la regresión lineal, el modelose conoce como regresión ridge:

$$w^* = \arg\min_w \{(y - \mathbf{X}w)^T (y - \mathbf{X}w) + \lambda w^T w\}$$

- Tomando derivadas e igualando a 0 la solución es

$$w^* = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} (\mathbf{X}^T y)$$

donde $\mathbf{I}$ es la matriz identidad

Regresión ridge en R

- La función `lm.ridge()` de la librería `MASS` entrena una regresión lineal con regularización *ridge* para varios valores del parámetro λ
- La librería `ridge` implementa la función `linearRidge()` que selecciona automáticamente el valor óptimo del parámetro λ usando el método propuesto en Cule et al (2012)

Aprendizaje supervisado en la práctica

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:

    1. Texto (tweets, páginas web, documentos): word2vec, bag-of-words, modelos n-gram
    2. Imágenes: valores RGB de los píxeles, intensidad de gris
    3. Audio: transformada de Fourier, coeficientes MFCC
    4. Video: secuencia de frames
    5. Series temporales: añadir *lags* como variables

Valores que faltan

- Es importante distinguir cuando una variable tiene valor 0 o 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?

Valores extremos

- Distinguir si un valor es erróneo o válido pero extremo es muy complicado y dependiente del dominio
- Existen diversas reglas para identificarlos
- Pueden perjudicar a ciertos algoritmos de aprendizaje, mientras que otros son robustos frente a este tipo de datos

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 (10,000 – 100,000 EUR) 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 I
  - ...
- En ocasiones normalizar las variables tambié 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 todo tipo de fuentes de datos
- Muy pocos algoritmos de aprendizaje son capaces de tratarlas directamente
- Por tanto, tenemos que 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
- Lo más utilizar una codificación *dummy* o *one-hot encoding*

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 tienen correlación 1
- 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}
- Cuidado con las distancias!
- Ejemplo 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

Variables categóricas en R

- Las variables categóricas en R se codifican con el tipo `factor`
- Muchas funciones que implementan algoritmos de aprendizaje con interfaz para fórmulas aceptan factores directamente y los convierte usando una codificación *dummy*
- Si el algoritmo no tiene interfaz para fórmula y solo acepta variables numéricas, tenemos que convertirlas explicitamente
- Una opción es la función `dummy_cols()` de la librería `fastDummies`