

# Aprendizaje Automático, Regresores y Clasificadores

Inteligencia Artificial e Ingeniería del Conocimiento

---

Constantino Antonio García Martínez

Universidad San Pablo Ceu

## Bibliografía

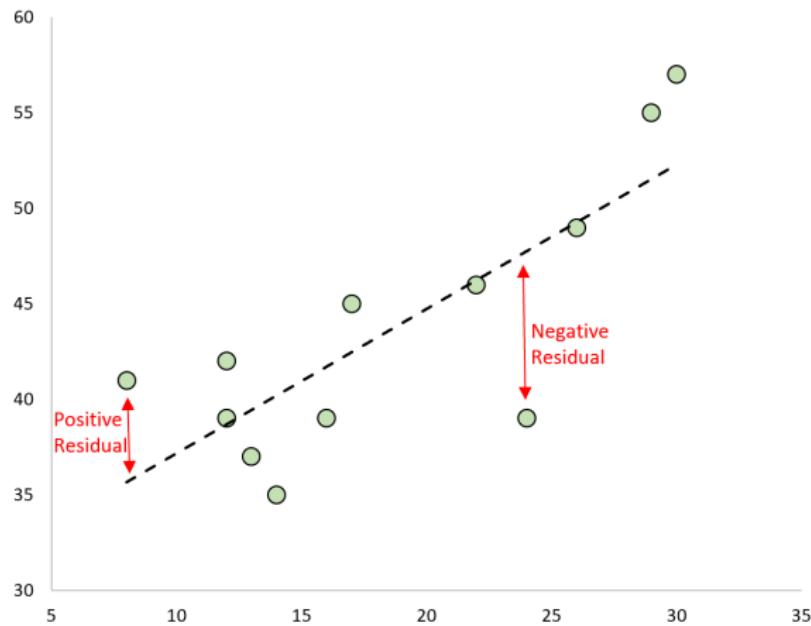
- PMLR Christopher Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
- AIMA Russell, Stuart J., and Peter Norvig. Artificial intelligence: a modern approach. Pearson, 2016.

## **Regresión Lineal, Regresión Logística y Descenso de Gradiente**

---

# Linear Regression

- El modelo es una combinación lineal de características:  
 $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n = \boldsymbol{\beta}^T \mathbf{x}$ .
- Objetivo: Minimizar el Error Cuadrático Medio (MSE). MSE es un ejemplo de **función de pérdida**.
- $\mathcal{L} = MSE = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$

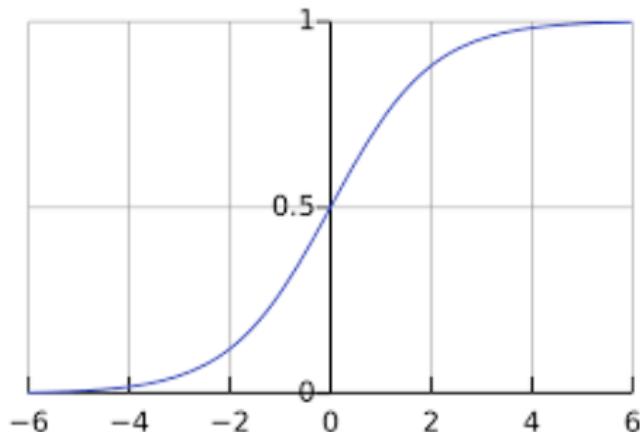


## Logistic Regression

- ¡Recuerda que los Regresores Logísticos son clasificadores!
- Idea básica: Usar regresión para ajustar los identificadores de clase (0 y 1, en un problema binario).
- Problema: Salidas no interpretables. ¿Qué pasa si el modelo produce  $\hat{y} = 1.9$  o  $\hat{y} = -5$ ?
- La solución: permitir que el regresor produzca cualquier número en  $(-\infty, \infty)$  y luego comprimirlo al rango  $(0, 1)$  para interpretarlo como la probabilidad de pertenecer a la clase 1.

## Logistic Regression

- Si la probabilidad predicha de pertenecer a la clase 1 es  $\hat{p}$ , la salida sin comprimir puede interpretarse como un logaritmo de ratio de probabilidades  $\log \frac{\hat{p}}{1-\hat{p}}$ .
- La operación de compresión se llama sigmoidal:  $\sigma(z) = \frac{1}{1+e^{-z}}$



## Logistic Regression

- Si la probabilidad predicha de pertenecer a la clase 1 es  $\hat{p}$ , la salida sin comprimir puede interpretarse como un logaritmo de ratio de probabilidades  $\log \frac{\hat{p}}{1-\hat{p}}$ .
- La operación de compresión se llama sigmoide:  $\sigma(z) = \frac{1}{1+e^{-z}}$
- El modelo completo es:

$$z = \log \frac{\hat{p}}{1-\hat{p}} = \boldsymbol{\beta}^T \mathbf{x}$$
$$\hat{y} = \hat{p} = \sigma(z)$$

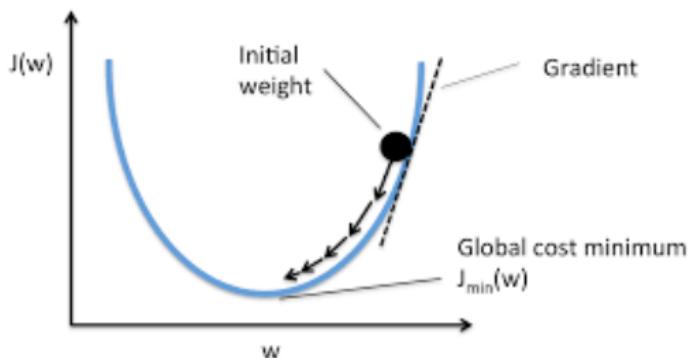
- Objetivo: **entropía cruzada binaria (binary cross-entropy)**.

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)]$$

- Intuición de la entropía cruzada binaria: Mide la disimilitud entre la distribución verdadera (etiquetas reales) y la distribución predicha (salidas del modelo).

## Descenso de Gradiente (Gradient Descent, GD)

- Problema: Dado un conjunto de datos y un modelo de regresión lineal/logística, ¿cómo aprendemos los mejores parámetros/pesos  $\beta$ ?
- Solución: ¡Simplemente minimizar  $\mathcal{L}(\beta)$ !



- ¿Cómo? Hagámoslo iterativamente. Empezamos en un  $\beta$  inicial. Encontramos la dirección que minimiza  $\mathcal{L}(\beta)$  y nos movemos un pequeño paso en esa dirección. Es decir:

$$\beta = \beta - \alpha \nabla_{\theta} \mathcal{L}(\beta)$$

- $\alpha$  es la llamada tasa de aprendizaje.

Code Exercise: Regression from scratch

## **Variantes de Regresión Lineal: Lasso, ElasticNet, y Ridge**

---

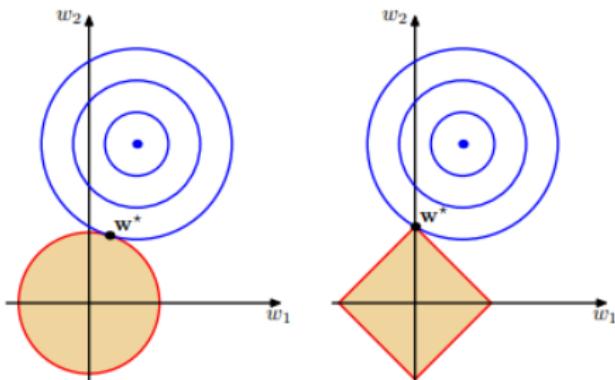
Existen muchas variantes de modelos lineales (tanto para clasificación como regresión) que utilizan **regularización**:

- Ridge Regression (L2):  $\mathcal{L}' = \mathcal{L} + \lambda \sum_{j=1}^n \beta_j^2$
- Lasso (L1):  $\mathcal{L}' = \mathcal{L} + \lambda \sum_{j=1}^n |\beta_j|$
- ElasticNet: Combinación de L1 y L2:  $\mathcal{L}' = \mathcal{L} + \lambda_1 \sum_{j=1}^n |\beta_j| + \lambda_2 \sum_{j=1}^n \beta_j^2$
- Objetivo: Prevenir el sobreajuste, manejar la multicolinealidad

## Exercise: Sklearn

Localiza en Sklearn el nombre de los regresores y clasificadores Lasso, Ridge y ElasticNet. ¿Cómo se llaman  $\lambda$ ,  $\lambda_1$  y  $\lambda_2$  en el código?

# Comparación de Técnicas de Regularización



- Ridge (L2):
  - Mejor manejo de la multicolinealidad
  - Reduce los coeficientes hacia cero (pero nunca exactamente a cero)
- Lasso (L1):
  - Mejor manejo de la multicolinealidad
  - Puede llevar coeficientes exactamente a cero (selección de características)
- ElasticNet (L1 + L2): Combina los beneficios de Ridge y Lasso
  - Maneja mejor los siguientes escenarios:
    - Cuando  $n < p$  (más características que muestras)
    - Cuando las características están correlacionadas en grupos

---

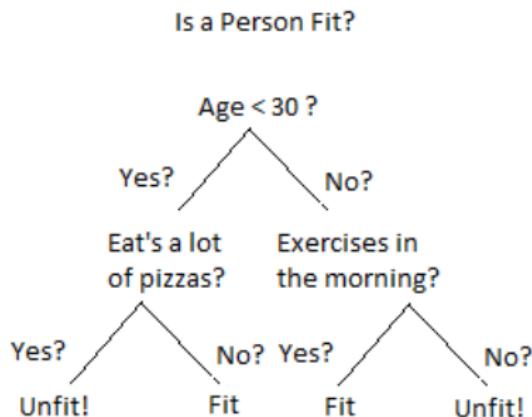
Imagen de PRML

## Decision Trees

---

## Decision Trees: Visión General

- Modelo en forma de árbol de decisiones y sus consecuencias
- Usado tanto para clasificación como regresión (`DecisionTreeClassifier` y `DecisionTreeRegressor`)
- Ventajas: Interpretabilidad, maneja relaciones no lineales
- Desafíos: Sobreajuste, inestabilidad



# Algoritmo de Decision Trees

---

## Algorithm 1 Algoritmo de Decision Trees

---

- 1: Comenzar con todo el conjunto de datos
  - 2: Para cada característica, encontrar la división que minimiza la función de costo
  - 3: Elegir la mejor división entre todas las características
  - 4: Dividir el nodo y crear nodos hijos
  - 5: Repetir pasos 2-4 para cada nodo hijo hasta cumplir los criterios de parada
- 

- Algoritmos históricos:
  - ID3 (Iterative Dichotomiser 3) por Ross Quinlan (1986)
  - C4.5: Sucesor de ID3, maneja atributos continuos
  - C5.0: Sucesor comercial de C4.5, más rápido y eficiente en memoria
  - CART: Soporta tanto clasificación como regresión, usa divisiones binarias

CART usa **impureza de Gini** para clasificación como función de costo, mientras que los otros algoritmos usan **ganancia de información**.

# Selección de la Mejor División

## 1) Seleccionar la división para una característica dada:

- Para cada posible umbral  $t$  en la característica  $f$ , dividir los datos en dos grupos:

$$S_{left} = \{x_i | x_i[f] \leq t\}, \quad S_{right} = \{x_i | x_i[f] > t\}$$

- Calcular la función de costo (ej., impureza Gini/Ganancia de Información, MSE) para la división.
- Elegir el umbral  $t^*$  que minimiza el costo:

$$t^* = \arg \min_t \left( \frac{|S_{left}|}{|S|} \cdot \text{Costo}(S_{left}) + \frac{|S_{right}|}{|S|} \cdot \text{Costo}(S_{right}) \right)$$

## 2) Elegir la mejor división entre todas las características:

- Repetir el proceso anterior para cada característica.
- Elegir la característica y el umbral correspondiente que resulten en el menor costo general.

# Detour: Entropía a través de Longitudes de Código

## Ejemplo: Frecuencias de Palabras

Palabra	Frecuencia
"cat"	$50\% = \frac{1}{2}$
"dog"	$25\% = \frac{1}{4}$
"stats"	$12.5\% = \frac{1}{8}$
"AI"	$12.5\% = \frac{1}{8}$

## Codificación Fija vs. Óptima

### Longitud Fija (2 bits):

Palabra	Código
cat	00
dog	01
stats	10
AI	11

### Codificación Óptima:

Palabra	Código	Longitud
cat	0	1 bit
dog	10	2 bits
stats	110	3 bits
AI	111	3 bits

## Longitud Media del Código

- Fija: 2 bits siempre
- Óptima:  $0.5 \cdot 1 + 0.25 \cdot 2 + 0.125 \cdot 3 + 0.125 \cdot 3 = 1.75$  bits

## Detour: Entropía a través de Longitudes de Código

### Idea Clave: Bits Necesarios vs. Probabilidad

- Si  $p = \frac{1}{2}$       Necesita 1 bit       $-\log_2(\frac{1}{2}) = 1$
- Si  $p = \frac{1}{4}$       Necesita 2 bits       $-\log_2(\frac{1}{4}) = 2$
- Si  $p = \frac{1}{8}$       Necesita 3 bits       $-\log_2(\frac{1}{8}) = 3$

### Patrón General

Cuando la probabilidad es  $p = (\frac{1}{2})^n$ , necesitamos  $n$  bits

Resolviendo para  $n$ :  $n = -\log_2(p)$

### Entropía

Entropía = Bits promedio necesarios =  $\sum p(x) \cdot (-\log_2(p(x)))$

## Diferentes Interpretaciones de la Entropía

- **Teoría de Codificación:** Número mínimo promedio de bits necesarios para codificar eventos en una distribución
- **Teoría de la Información:** Contenido de información esperado o sorpresa de los eventos (eventos raros son más sorprendentes)
- **Decision Trees:** Medida de impureza del conjunto de datos (cero para conjuntos puros, máximo cuando las clases son igualmente probables)

## Formulación Matemática

- Entropía:  $H(S) = -\sum_{i=1}^c p_i \log_2(p_i)$
- Ganancia de Información:  $IG(S, A) = H(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} H(S_v)$

## Entendiendo la Ganancia de Información

- Representa la reducción en entropía después de dividir por el atributo A
- $H(S)$ : Entropía (incertidumbre) original del conjunto de datos
- $\sum \frac{|S_v|}{|S|} H(S_v)$ : Promedio ponderado de entropía después de la división
- Mayor ganancia = Mayor reducción en incertidumbre = Mejor división

# Decision Trees: Criterios de Parada y Selección de Profundidad

## ¿Cuándo Dejar de Hacer Crecer el Árbol?

- Se alcanza la profundidad máxima
- El nodo se vuelve puro (todas las muestras de la misma clase)
- Muy pocas muestras en el nodo

## El Compromiso de la Profundidad

- **Muy Superficial** → Subajuste (demasiado simple)
- **Muy Profundo** → Sobreajuste (demasiado complejo)

## Code Exercise: Visualización de Decision Trees

## Code Exercise: Decision Trees y Sklearn

Lee los parámetros de DecisionTree y relacionalos con todos los conceptos explicados.

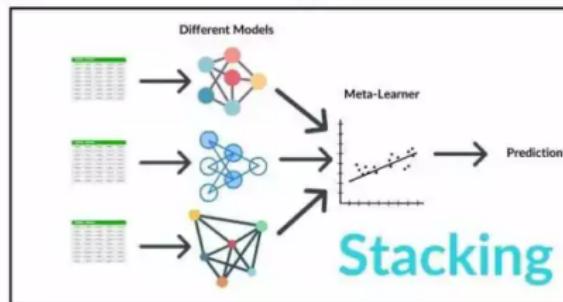
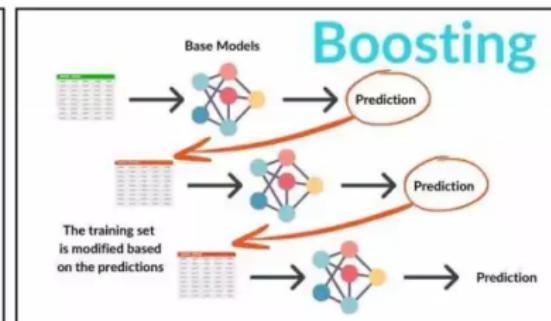
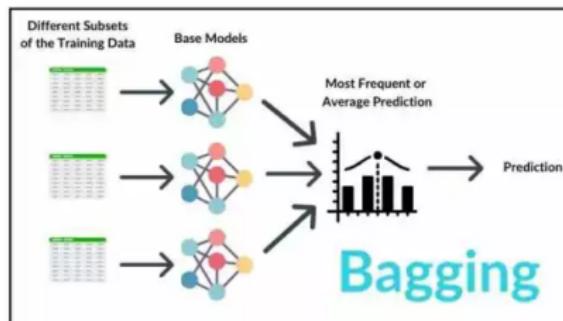
## Research Project: Decision Trees from Scratch

## **Ensembles y Random Forest**

---

# Métodos de Ensemble: Visión General

- Combinan múltiples modelos para mejorar el rendimiento
- Tipos: Bagging, Boosting, Stacking
- Objetivo: Reducir la varianza y el sesgo, mejorar la generalización



- **Bagging (Bootstrap Aggregating)**

- Crear múltiples subconjuntos de datos con reemplazamiento
- Entrenar un modelo en cada subconjunto
- Combinar predicciones (votación para clasificación, promedio para regresión)
- Reduce la varianza, ayuda a prevenir el sobreajuste

- **Boosting**

- Método de ensemble secuencial
- Cada modelo intenta corregir los errores del modelo anterior
- Algoritmos populares: AdaBoost, Gradient Boosting
- Reduce el sesgo y la varianza

## Stacking

- También conocido como *Stacked Generalization*
- Combina predicciones de múltiples modelos usando un *meta-learner*
- Proceso:
  - Entrenar varios *modelos base diversos* en el conjunto de datos original
  - Usar modelos base para hacer predicciones en un conjunto de validación
  - Entrenar un *meta-learner* con las predicciones de los modelos base
- El meta-aprendiz aprende cómo combinar mejor las predicciones de los modelos base
- Puede (debe) combinar diferentes tipos de modelos (ej., decision trees, SVMs, redes neuronales)
- A menudo usado en competiciones de machine learning para lograr alta precisión

### Research Project: Stacking

- **Random Forest**

- Ensemble de decision trees
- Ideas clave:
  - Muestreo Bootstrap (bagging)
  - Selección aleatoria de características en cada división
- Ventajas: Robusto, maneja datos de alta dimensionalidad
- Desventajas: Menos interpretable que un único decision tree

- **XGBoost (Extreme Gradient Boosting)**

- Implementación avanzada de gradient boosting
- Características clave:
  - Regularización para prevenir el sobreajuste
  - Maneja valores faltantes
  - Procesamiento paralelo eficiente
- Alto rendimiento en muchas competiciones de machine learning

**Research Project: XGBoost**

## Interpretabilidad de Modelos

---

# Interpretabilidad de Modelos: De Modelos Simples a Complejos

## Modelos Verdaderamente Interpretables

- Modelos Lineales:
  - Interpretación directa de coeficientes
  - Contribución clara de características
- Árbol de Decisión Individual:
  - Caminos de decisión explícitos
  - Representación visual

## Modelos Complejos (ej., Ensembles)

- Solo ofrecen importancia de características
- Ocultan interacciones complejas
- Sacrifican interpretabilidad por rendimiento

### Punto Clave

Importancia de características  $\neq$  Verdadera interpretabilidad. Mientras que los métodos ensemble pueden decirnos qué características son importantes, no pueden explicar cómo estas características interactúan o por qué se hacen predicciones específicas.

**Para Saber más**

---

## Para Saber Más

1. Leer el trabajo original "A Mathematical Theory of Communication" de Claude Shannon.
2. Implementa Decision Trees desde cero.
3. Leer "XGBoost: A Scalable Tree Boosting System" de Chen y Guestrin y jugar con la librería xgboost de python.
4. Aprender sobre Support Vector Machines (SVM).