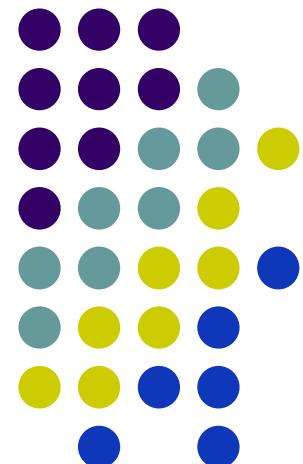


# Presentación

---

**Inteligencia Artificial  
Ingeniería Informática  
en Sistemas de Información**





# Profesores

---

- Alicia Troncoso Lora ([atrolor@upo.es](mailto:atrolor@upo.es))
  - EB
  - Tutorías: despacho 6, ed. 11, 1<sup>a</sup> planta
- Roberto Ruiz Sánchez ([rruisan1@upo.es](mailto:rruisan1@upo.es))
  - EPD12
  - Tutorías: despacho 14, ed. 11, 1<sup>a</sup> planta
- Laura Melgar García ([lmelgar@upo.es](mailto:lmelgar@upo.es))
  - EB y EPD11
  - Tutorías: despacho 11.02.SJ2, ed. 11, 2<sup>º</sup> planta,  
sala juntas 2



# Docencia

---

- Teoría
  - Horario: lunes de 17:30 a 19:00
  - Lugar: edificio 24, aula 1.04
- Prácticas
  - Comienzan semana del 26 de septiembre
  - Horario: martes de 16:00-18:00 (EPD11), martes de 18:00-20:00 (EPD12)
  - Lugar: edificio 24, laboratorio INFB03 (EPD11) y INFB05 (EPD12)



# Asignatura

- **Objetivos:** Resolver aplicaciones reales con técnicas de Inteligencia Artificial
  - Seleccionar las técnicas adecuadas
  - Usar/modificar/implementar prototipos
- **Prerrequisitos:** Se recomienda haber cursado las asignaturas relacionadas con programación y algorítmica
- **Planificación de la asignatura**
  - Guía docente general → Web de EPS
  - Guía docente específica → Aula Virtual



# Distribución del temario

---

- T1: Introducción a la Inteligencia Artificial
- T2: Introducción al Machine Learning
- T3: Regresión Lineal
- T4: Regresión Logística
- T5: Redes Neuronales
- T6: Clustering
- T7: Sistemas de Recomendación
- T8: Búsquedas



# Distribución del temario

SEMANA	<b>EB</b> Lunes 17:30-19:00 (Ed. 24 aula 1.04)	<b>EPD</b> Martes 16:00-18:00 EPD11 (Ed.24 INFB03) Martes 18:00-20:00 EPD12 (Ed.24 INFB05)
19-sep	Presentación + T1: Introducción IA T2: Introducción ML (martes 16:00-18:00)	
26-sep	T3: Regresión	EPD1: Herramientas para Machine Learning
3-oct	T4: Regresión Logística	EPD2: Regresión univariable
10-oct	T5: Redes Neuronales: Representación	EPD2: Regresión multivariable
17-oct	T5: Redes Neuronales: Aprendizaje	EPD3: Regresión Logística
24-oct	T3: Ejercicios	EPD4: Redes Neuronales
31-oct	T4: Ejercicios	
7-nov	T5: Ejercicios	EPD4: Redes Neuronales
14-nov	T6: Clustering	EPD5: Clustering
21-nov	<b>Examen EB Midterm-(T1-T5)</b>	<b>Examen EPD Midterm-(EPD1-EPD4)</b>
28-nov	T7: Sistemas de recomendación	EPD6: Sistemas de recomendación
5-dic	T7: Ejercicios	
12-dic	T8: Búsquedas	EPD7: Herramientas para Búsquedas
19-dic	T8: Ejercicios	EPD8: Búsquedas
VACACIONES NAVIDAD		
23-ene	<b>Examen</b> (jueves 26 enero 2023)	
19-jun	<b>Examen</b> (martes 20 junio 2023)	



# Evaluación

---

EB

25%- Examen Midterm – (T1-T5)

25%- Examen final - Clustering, Sistemas de recomendación y búsquedas

EPD

25%- Examen Midterm – (T1-T5)

25%- Examen final - Clustering, Sistemas de recomendación y búsquedas

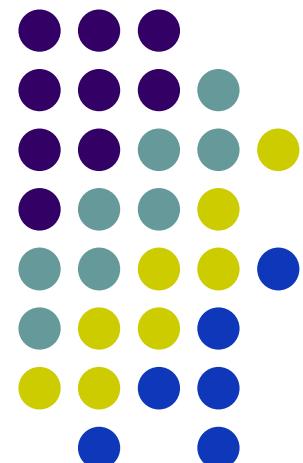


# Bibliografía

- Artificial Intelligence: A modern aproach. Stuart Russel and Peter Norvig. Pearson,2010 (3 Ed.)
- Inteligencia Artificial: Técnicas, métodos y aplicaciones, José T. Palma Méndez, Roque Marín Morales, Mac Graw Hill, 2011.
- Inteligencia Artificial e Ingeniería del Conocimiento, G. Pajares Martin Sanz, M. Santos Peñas, Ra-Ma, 2005.
- Machine Learning. Tom Mitchell. MacGraw-Hill, 1997
- Machine Learning: a Probabilistic Perspective. Kevin Patrick Murphy. Mit Press, 2012
- Data Mining: Practical Machine Learning Tools and Techniques. Ian Witten, Eibe Frank, Mark Hall. Morgan Kaufmann, 2011
- Andrew Ng, University of Stanford – Machine Learning

# Introducción y Aplicaciones de la Inteligencia Artificial

Inteligencia Artificial  
Ingeniería Informática  
en Sistemas de Información





# Contenido

---

- ¿Qué es la Inteligencia Artificial?
- Historia de la Inteligencia Artificial
- Aplicaciones
- Próximos retos

# ¿Qué es la Inteligencia Artificial?

## Definición:



Se trata de dotar de **inteligencia** a las **máquinas**

¿Qué es ser inteligente?

Artificial

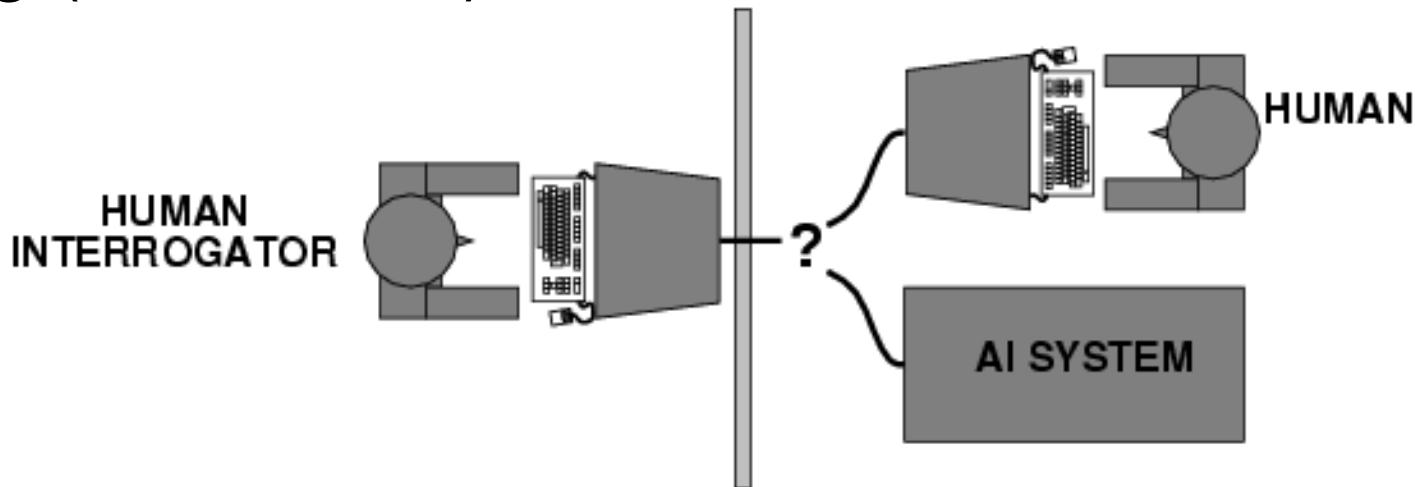
Sistemas que piensan como humanos	Sistemas que piensan racionalmente
Habría que conocer el funcionamiento de la mente humana	Saber hacer inferencias correctas LÓGICA
Sistemas que actúan como humanos	Sistemas que actúan racionalmente
Test de Turing 1950	Saber actuar alcanzando el mejor resultado AGENTES

# ¿Qué es la Inteligencia Artificial?



## Test de Turing:

Alang Turing (1912-1954)



- Matemático inglés, padre de la ciencia de la computación y precursor de la informática moderna.
- Inventó la máquina de Turing que es un modelo matemático por el cual cualquier problema se puede implementar a través de un algoritmo.
- Trabajó en la Segunda Guerra Mundial descifrando códigos nazis y diseñó uno de los primeros computadores electrónicos programables digitales.



# ¿Qué es la Inteligencia Artificial?

## Test de Turing:

- Objetivo: Incapacidad de diferenciar entre respuestas del ordenador y respuestas humanas.
- Para superar el test un computador necesita las capacidades:
  - Procesamiento de lenguaje natural
  - Representación del conocimiento
  - Razonamiento
  - Aprendizaje

+

  - Visión
  - Robótica

} Permiten la interacción física entre persona y ordenador

# ¿Qué es la Inteligencia Artificial?

## Agentes:



- Un agente recibe información del entorno y actúa para conseguir unos resultados según una medida de rendimiento. Es decir, se basan en dos conceptos: percepción y actuación.



Agente = arquitectura + programa

Arquitectura → PC, máquina con ruedas (smart cleaner), smartphone, ...

# ¿Qué es la Inteligencia Artificial?

## ¿Cuál es la metodología de la IA?:



- Aspectos teóricos:
  - Formalizaciones matemáticas, propiedades y algoritmos.
- Aspectos de ingeniería:
  - Construir máquinas útiles.
- Ciencia empírica:
  - Experimentos.



# **Historia de la IA**

## **Campos relacionados:**

- Filosofía:
  - Teorías de razonamiento, fundamentos del aprendizaje, etc.
  - Ejemplos: Aristóteles, Descartes, Leibniz, Russell, ....
- Matemáticas:
  - Teoría de probabilidad, estadística, ....
- Ingeniería:
  - Algoritmos, lógica e inferencia.
  - Lenguajes de programación, sistemas informáticos (hardware y software).
- Otras: psicología, neurociencia, lingüística, ....



# Historia de la IA

---

- George Boole, Gottlob Frege, and Alfred Tarski
  - Formalizaron las leyes del pensamiento humano
- Alan Turing, John von Neumann, and Claude Shannon
  - Empezaron a pensar en computadores
- Fundadores directos de la IA: John McCarthy, Marvin Minsky, Herbert Simon, and Allen Newell
  - El inicio de la IA (1956)



# Historia de la IA

## Hitos (I):

- 1943: McCulloch y Pitts. Primeros modelos de neuronas artificiales.
  - 1951: Minsky. Red de 40 neuronas.
- 1950: Turing. Test de Turing: “Computing Machinery and Intelligence”.
- 1950-60s: Primeros programas de IA, incluyendo:
  - Juego de damas de Samuel.
  - Programa de razonamiento “Teórico Lógico” / “Logic Theorist” de Newell y Simon.
- 1956: Reunión en Dartmouth: término “Inteligencia Artificial”
  - Workshop de 2 meses con 10 asistentes entre ellos Newell y Simon.



# Historia de la IA

## Hitos (II):

- 1966: ELIZA de Weizenbaum: procesamiento lenguaje natural (chatbot “psicoterapeuta”).
  - Reconocía palabras claves.
  - Tenía almacenadas frases hechas.
  - Repetía la frase del usuario transformándola en pregunta.
  - No memorizaba y no aprendía de las conversaciones previas.



# Historia de la IA

## Hitos (II):

- 1966: ELIZA de Weizenbaum: procesamiento lenguaje natural (chatbot “psicoterapeuta”).

```
Eliza

ELIZA: Please tell me what's been bothering you.
YOU: Everything
ELIZA: Tell me more about that.
YOU: Everything bothers me lately
ELIZA: Does talking about this bother you ?
YOU: No, oddly enough
ELIZA: Why 'no' ?
YOU: It just doesn't
ELIZA: I'm not sure I understand you fully.
YOU: It doesn't bother me to talk about it
ELIZA: Do you feel strongly about discussing such things ?
YOU: I don't know, should I?
ELIZA: Don't you really know ?
YOU: Really I don't
ELIZA: Why don't you ?
YOU: Are you here to help me or not?
ELIZA: Why are you interested in whether I am here to help you
or not or not ?
```



# Historia de la IA

## Hitos (III):

- 1969-73: La IA descubre la complejidad computacional.
  - La investigación en redes neuronales casi desaparece.
- 1969-79: Inicio del desarrollo de los sistemas basados en conocimiento.
  - Métodos de búsqueda de propósito general.
  - Sistemas expertos: Simula el razonamiento humano como lo haría un experto en un área de especialización.
    - MYCIN: “Razonaba” el proceso para llegar a diagnóstico de enfermedades infecciosas sanguíneas y recetaba medicamentos personalizados para el paciente.
    - R1: Elaboración de pedidos de sistemas informáticos.





# Historia de la IA

## Hitos (IV):

- 1980: IA se convierte en industria.
- 1986: Regreso de las redes neuronales.
- 1987: IA se convierte en una ciencia.
- 1995- : Sistemas inteligentes.
  - Deep Blue: éxito de gran impacto.



Deep Blue had Kasparov in deep thought  
(CNN)

VS.





# Historia de la IA

## Hitos (V):

**“I could feel human-level intelligence across the room”**

*Gary Kasparov, Campeón del mundo de ajedrez*



**Game 1: 5/3/97:  
Kasparov gana**

**Game 2: 5/4/97:  
Deep Blue gana**

**Game 3: 5/6/97:  
Empate**

**Game 4: 5/7/97:  
Empate**

**Game 5: 5/10/97:  
Empate**

**Game 6: 5/11/97:  
Deep Blue gana**

**El valor del stock  
de IBM se  
incrementó  
en \$18 billones!**

Deep blue: Uno de los computadores modernos más famosos que derrotó a Gary Kasparov al ajedrez



# Aplicaciones:

- **Medicina:**

- Medicina robótica → Da Vinci.
- Análisis médicos y de imágenes.

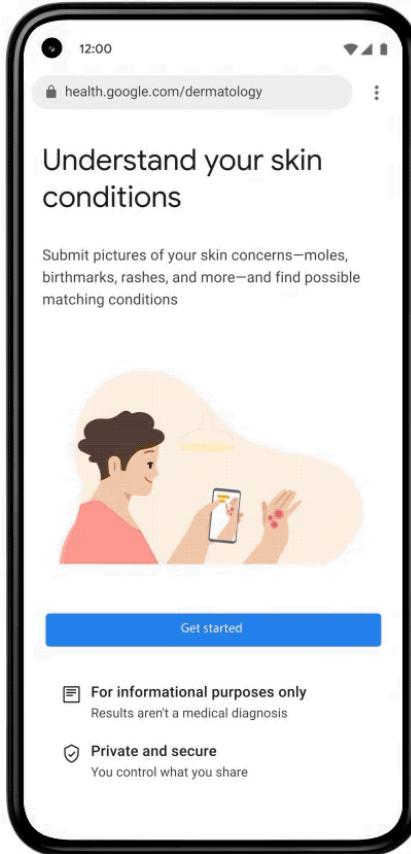




# Aplicaciones:

- **Medicina:**

- 2021: Google da respuesta a problemas de la piel comunes  
(<https://blog.google/technology/health/ai-dermatology-preview-io-2021/>)





# Aplicaciones:

- **Redes sociales:**
  - Chatbots para Whatsapp Business.
  - Instagram contra el acoso, mentiras sobre la edad.
  - Twitter predice los usuarios que difundirán “fake news” antes de que lo hagan: correlación entre el tipo de contenido, el tono usado y (obviamente) la tendencia a publicar “fake news”.
  - TextStyleBrush: Facebook imita la escritura de un usuario solo a partir de una palabra.



# Aplicaciones:

- **Otras:**

- Sistemas de recomendación: información recopilada de millones de usuarios hace que el sistema aprenda y cada vez sea más inteligente. Se identifican los usuarios con gustos y preferencias similares a ti. Te recomiendan los productos que han satisfecho a esos usuarios.
  - Redes sociales: Personas a las que quizás conozcas, personas a las que seguir, etc.
  - Netflix: ¿Qué ver después?
- Reconocimiento de imágenes o texto: Lectores de matrículas, códigos QR o códigos de barra, etc.
- Control por voz



# Aplicaciones:

- **Otras:**

- PageRank de Google: Porcentaje de veces que el usuario ve el resultado de la búsqueda presentado en una posición del ranking determinada.
  - Depende del número de páginas que llevan a esa página  $C(i)$  y del valor de PageRank de dichas páginas  $PR(i)$ . El parámetro  $d$  es un factor de amortiguación que se puede definir entre 0 y 1.

$$PR(A) = (1 - d) + d \sum_{i=1}^n \frac{PR(i)}{C(i)}$$

- Fuente: <http://infolab.stanford.edu/~backrub/google.html>

# En la actualidad...



- **INTERNET DE LAS COSAS**

- ¿Se imagina un frigorífico que le avise de la fecha de caducidad de los alimentos que contiene?
- ¿O que las zapatillas que usa para hacer deporte registren "en la nube" las estadísticas de cuánto corre cada semana y a qué velocidad?
- ¿Y que los inodoros analicen su orina y le recomiendan la dieta alimentaria que más le conviene seguir?
- ¿Qué pasaría si el cepillo de dientes le alertara de cualquier pequeña caries y pidiera por usted cita en el dentista?

# En la actualidad...



- **SMART CITIES**

- Movilidad y transporte
- Seguridad física y lógica
- Gestión de recursos naturales
- Eficiencia energética en la gestión de infraestructuras y edificios
- eGobierno y participación ciudadana

# En la actualidad...



- **BIG DATA**

- Volumen muy elevado
- Datos no estructurados
- Velocidad en la captura y tratamiento



# En la actualidad...

- **CIBERSEGURIDAD**

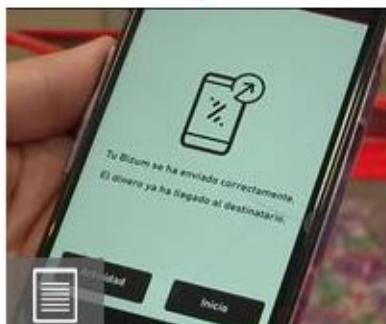
18/09/2020 | 11:30



## **Alertan de una campaña fraudulenta que suplanta a Kutxabank**

La entidad bancaria recuerda que solo pide los datos y claves de acceso cuando se está operando en sus canales oficiales, y recomienda que no se abran los correos de remitentes desconocidos.

16/09/2020 | 08:36



## **Detectan una estafa a través de Bizum, haciéndose pasar por la Seguridad Social**

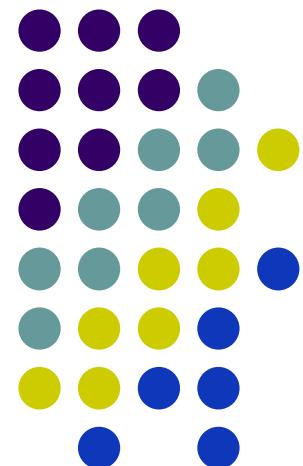
Los ciberdelicuentes engañan a las víctimas haciéndoles creer que tienen pendiente una devolución de tasas por escolarizar a sus hijos.

Técnicas de IA para la defensa del ciberespacio y detección de delitos informáticos

# Introducción a *Machine Learning*

---

**Inteligencia Artificial  
Ingeniería Informática  
en Sistemas de Información**



[Source: Andrew Ng, University of Stanford  
Antonio Bahamonde, Universidad de Oviedo]



# Contenido

---

- ¿Qué es Machine Learning?
- Aprendizaje supervisado
  - Regresión
  - Clasificación
- Aprendizaje no supervisado
  - Clustering
- Conceptos básicos
- Diseño de sistemas de aprendizaje
- Evaluación en Machine Learning
  - Métricas
  - Validación del modelo



# Contenido

---

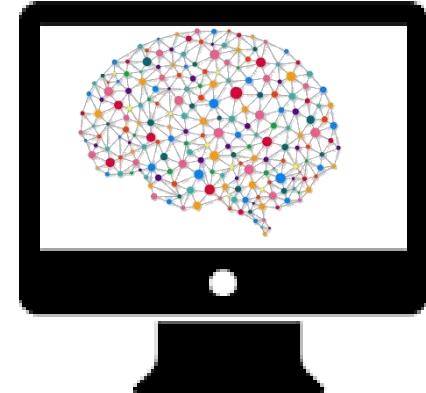
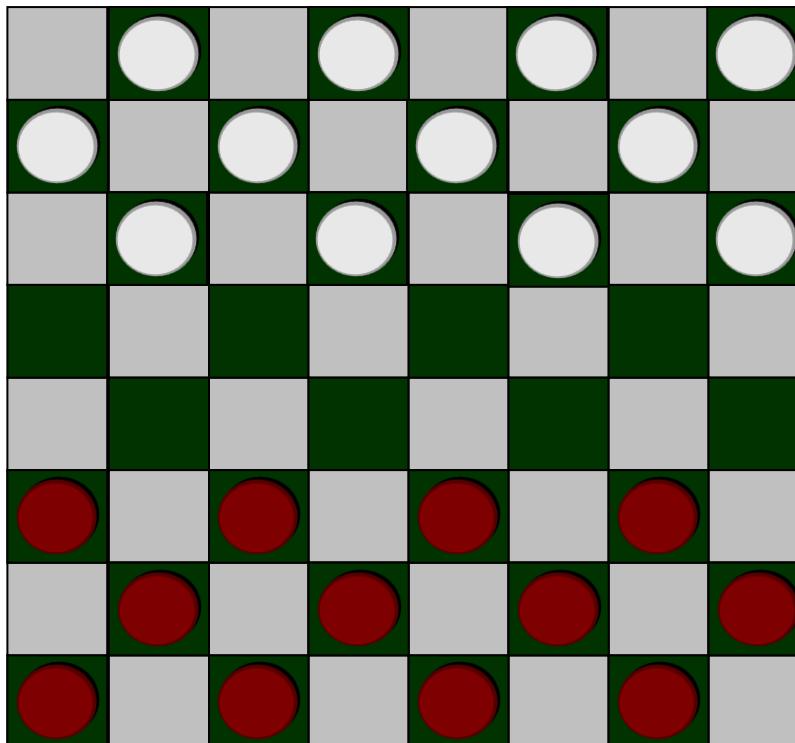
- **¿Qué es Machine Learning?**
- Aprendizaje supervisado
  - Regresión
  - Clasificación
- Aprendizaje no supervisado
  - Clustering
- Conceptos básicos
- Diseño de sistemas de aprendizaje
- Evaluación en Machine Learning
  - Métricas
  - Validación del modelo

# ¿Qué es Machine Learning?



**Arthur Samuel (1959).**

“Field of study that gives computers the ability to learn without being explicitly programmed.”



# ¿Qué es Machine Learning?



**Tom Mitchell (1998).**

“Well-posed Learning Problems”:

Un programa de ordenador APRENDE  
a partir de una experiencia  $E$   
a realizar una tarea  $T$   
(de acuerdo con una medida de rendimiento  $P$ ),

SI

su rendimiento al realizar  $T$ ,  
medido en  $P$ ,  
mejora gracias a la experiencia  $E$ .

# ¿Qué es Machine Learning?



**Tom Mitchell (1998).**

“Well-posed Learning Problems”

- Ejemplo 1) Juego “checkers”:
  - Tarea  $T$ : Jugar a “checkers”.
  - Experiencia  $E$ : datos generados de jugar contra uno mismo muchas veces.
  - Medida de rendimiento  $P$ : Porcentaje de juegos ganados contra nuevos oponentes.

# ¿Qué es Machine Learning?



**Tom Mitchell (1998).**

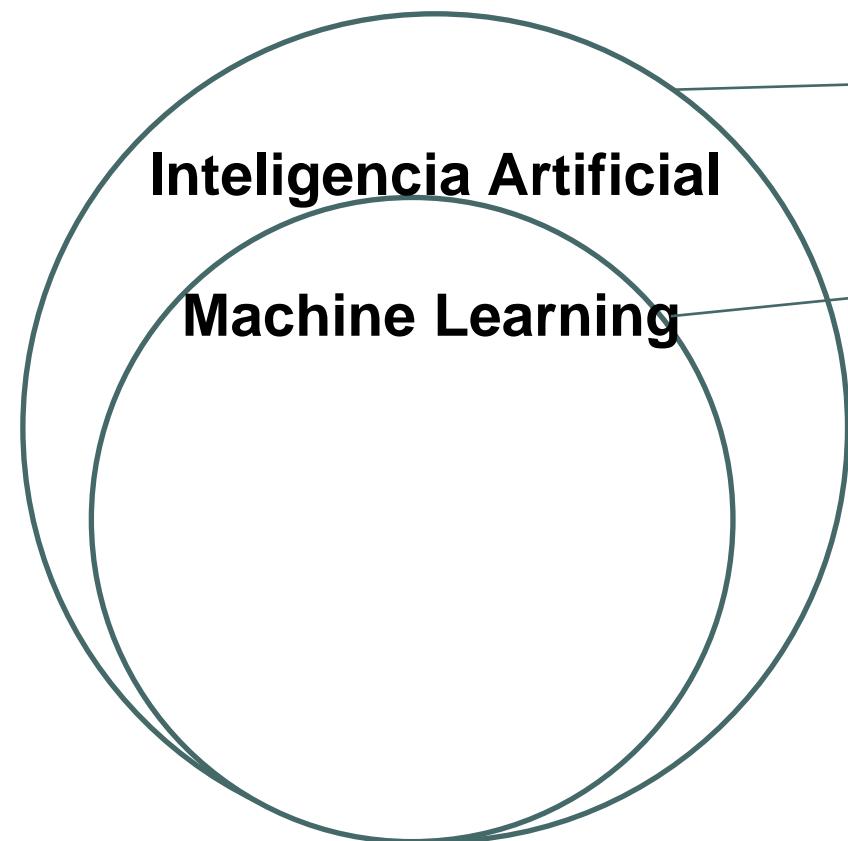
“Well-posed Learning Problems”

- Ejemplo 2) Filtro de emails como spam o no:
  - Tarea  $T$ : Clasificación de emails como spam o no.
  - Experiencia  $E$ : base de datos de emails clasificados/observar las etiquetas de los emails.
  - Medida de rendimiento  $P$ : Porcentaje de emails clasificados correctamente como spam/no spam.

# ¿Qué es Machine Learning?



## Conceptos clave de la ciencia de datos.



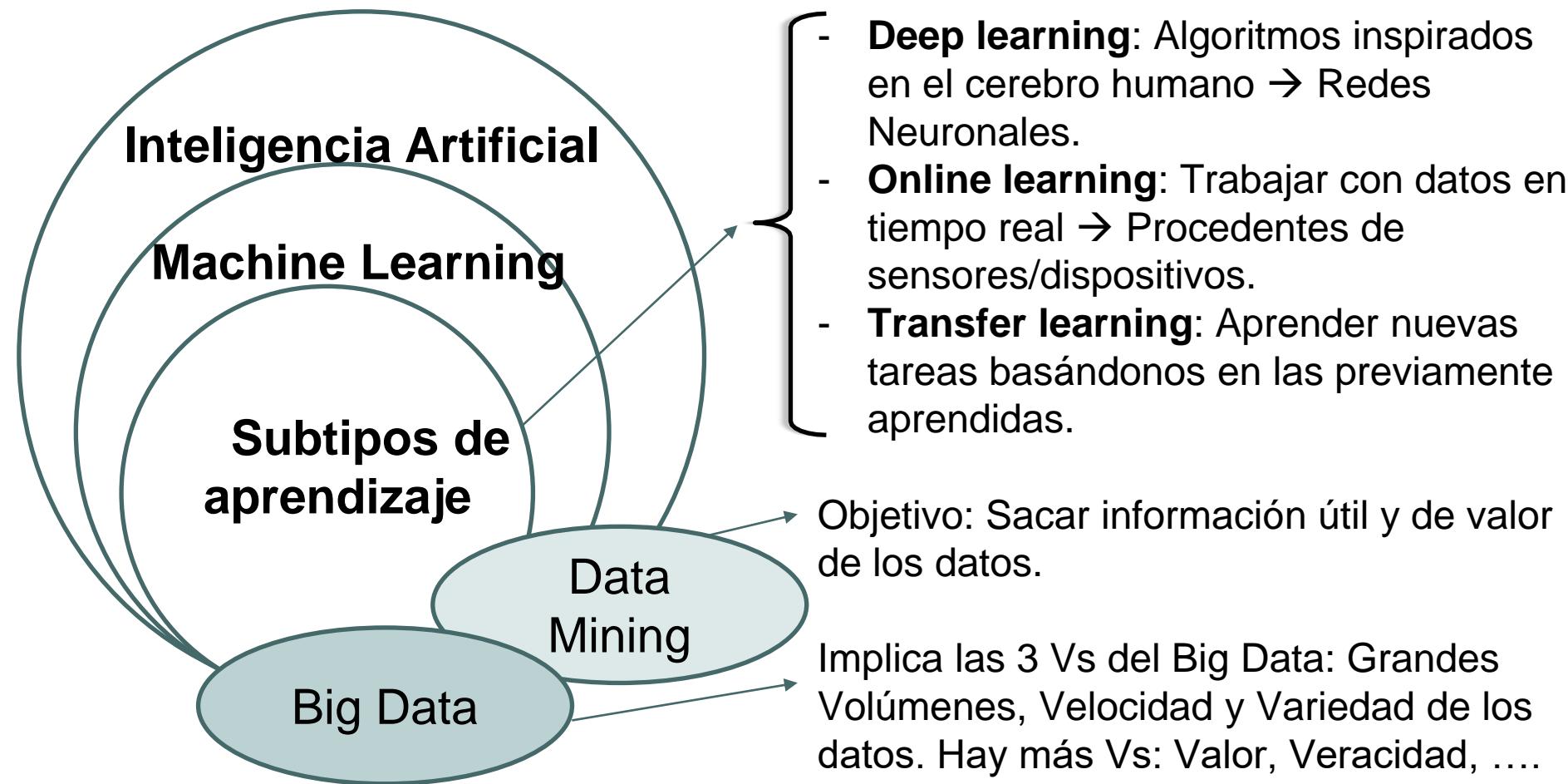
Objetivo: Crear máquinas inteligentes (robótica, medicina, domótica, video juegos...).

Objetivo: Máquinas que puedan aprender a partir de una experiencia.

# ¿Qué es Machine Learning?



## Conceptos clave de la ciencia de datos.



# ¿Qué es Machine Learning?

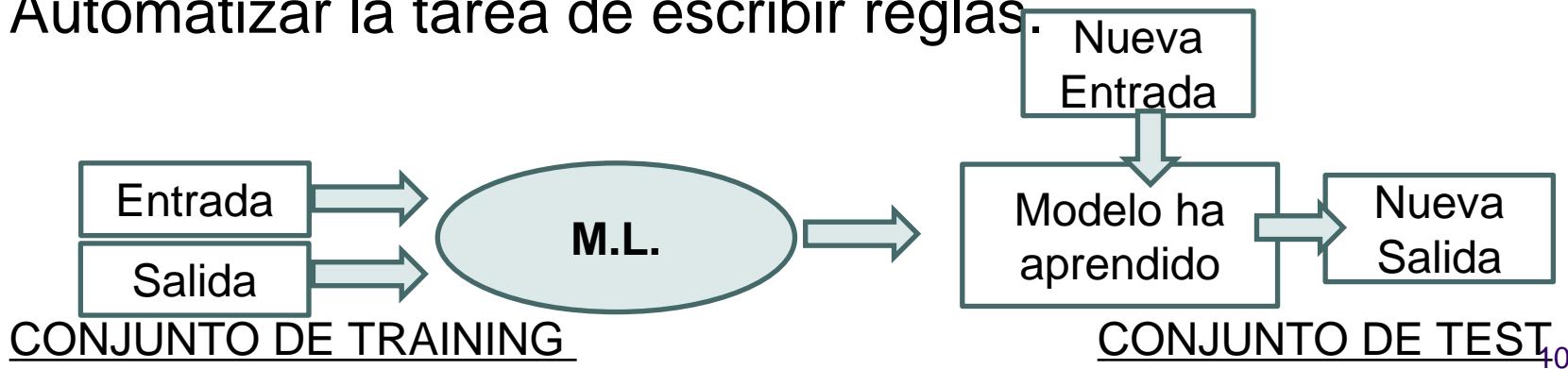


## Idea general de programación:

- Programación tradicional:
  - Automatizar una tarea escribiendo reglas.



- Machine Learning (Idea aprendizaje supervisado):
  - Automatizar la tarea de escribir reglas.



# ¿Qué es Machine Learning?



Algoritmos de Machine Learning:

- Aprendizaje supervisado (T3, T4, T5)
- Aprendizaje no supervisado (T6)
- Otros: sistemas de recomendación (T7).

También dentro de Inteligencia Artificial:  
búsquedas (T8).

# ¿Qué es Machine Learning?



## Tipos de aprendizaje de Machine Learning:

Se diferencian según el objetivo que persiguen:

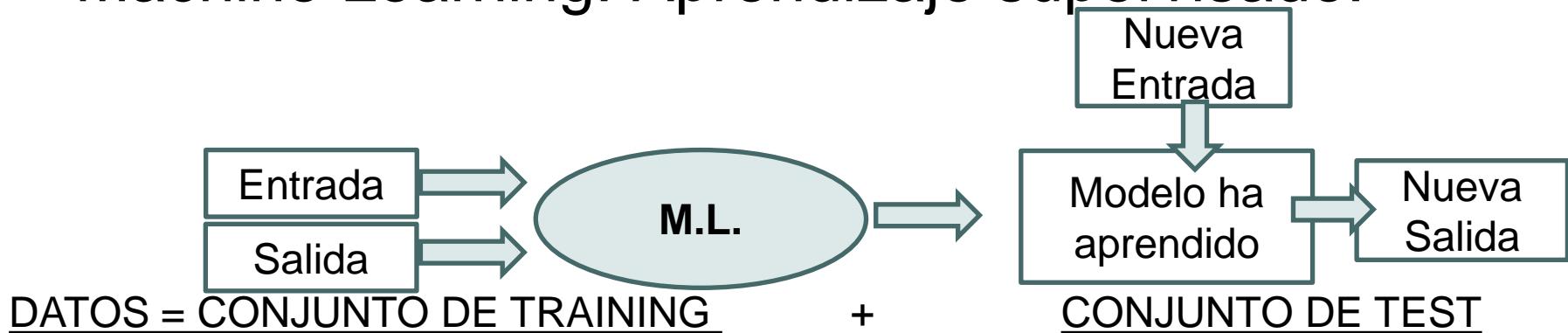
- Aprendizaje **supervisado**.
  - Permite realizar predicciones relacionando todos los atributos con un atributo especial, llamado atributo objetivo o clase.
- Aprendizaje **no supervisado**.
  - Sirve para explorar las propiedades de los datos examinados, no para predecir nuevos datos. Es decir, se usa para entender y resumir los datos.

# ¿Qué es Machine Learning?

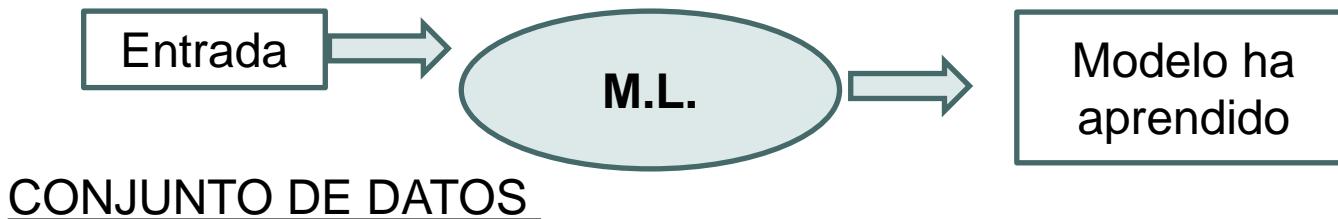


## Tipos de aprendizaje – Idea programación:

- Machine Learning: Aprendizaje supervisado:



- Machine Learning: Aprendizaje no supervisado.





# Contenido

---

- ¿Qué es Machine Learning?
- **Aprendizaje supervisado**
  - Regresión
  - Clasificación
- Aprendizaje no supervisado
  - Clustering
- Conceptos básicos
- Diseño de sistemas de aprendizaje
- Evaluación en Machine Learning
  - Métricas
  - Validación del modelo

# Aprendizaje supervisado

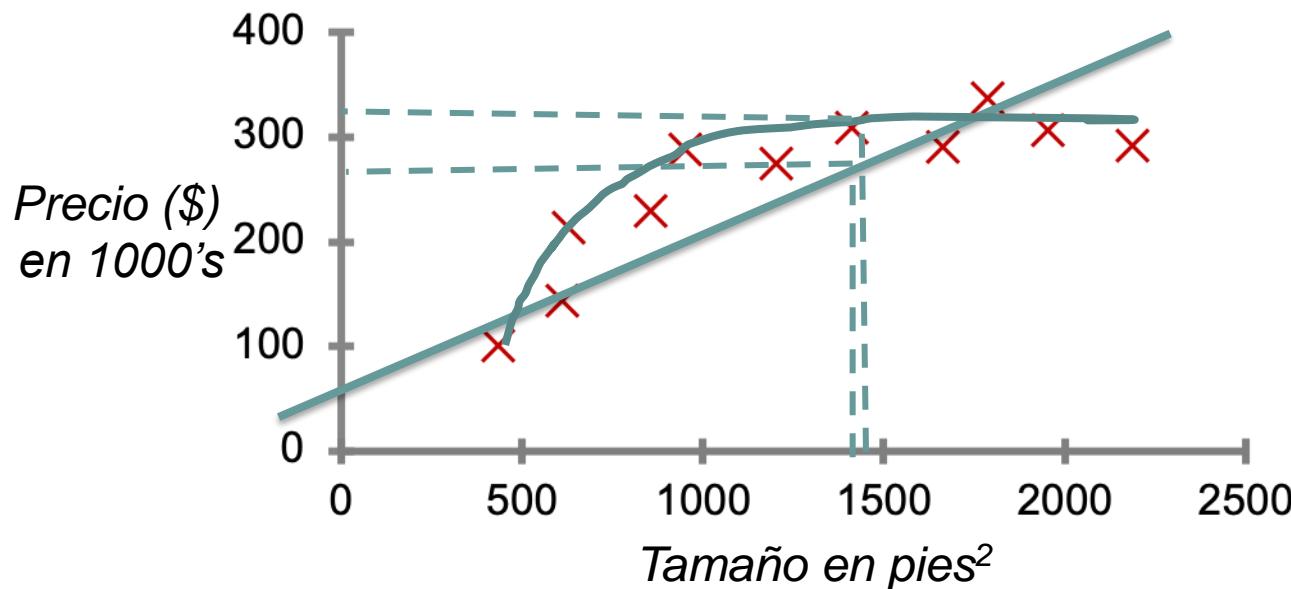


- Regresión:
  - Predice un valor numérico continuo (datos cuantitativos).
- Clasificación:
  - Predice una etiqueta o clase discreta (datos cualitativos).

# Aprendizaje supervisado



**Ejemplo regresión:** Predicción del precio de una casa



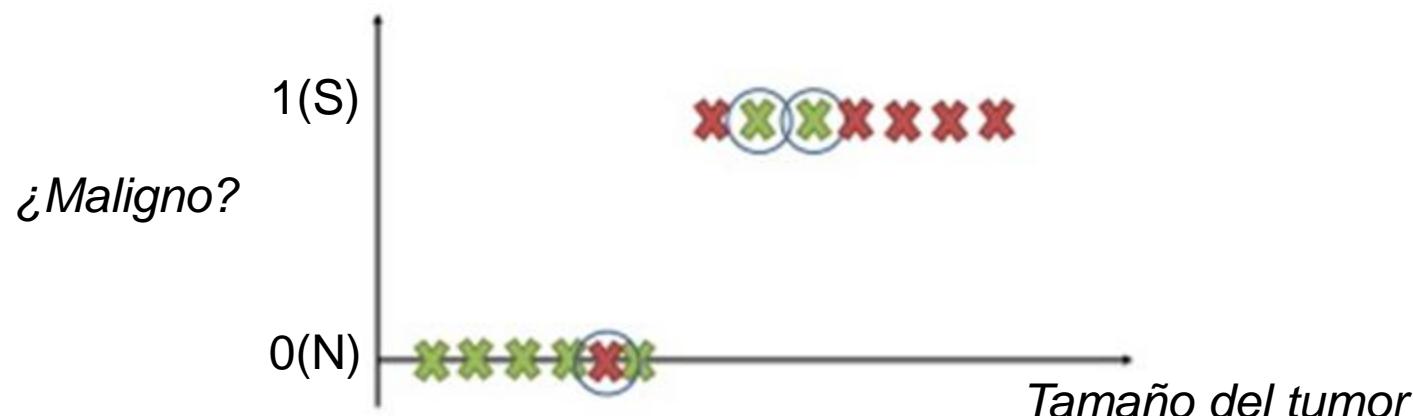
Aprendizaje supervisado  
“Respuestas correctas”  
proporcionadas.

Regresión: Predecir como salida  
un valor continuo (precio de la  
casa)

# Aprendizaje supervisado



**Ejemplo clasificación:** Detectar/predicir un tipo de tumor → {maligno; benigno}



Aprendizaje supervisado  
“Respuestas correctas”  
proporcionadas.

Clasificación: Predecir como salida un valor discreto (tipo de tumor → {maligno, benigno})

# Aprendizaje supervisado



¿Cómo deberías tratar los siguientes problemas?

- Problema 1: Tienes un gran inventario de artículos idénticos. Quieres predecir cuántos de esos artículos se venderán en los próximos 3 meses.
- Problema 2: Quieres examinar las cuentas individuales de los clientes y para cada una decidir si ha sido hackeada o no.

Posibilidades:

Ambos como problemas de clasificación.

Ambos como problemas de regresión.

Tratar el problema 1 como clasificación y el 2 como regresión.

Tratar el problema 1 como regresión y el 2 como clasificación.

# Aprendizaje supervisado



¿Cómo deberías tratar los siguientes problemas?

- Problema 1: Tienes un gran inventario de artículos idénticos. Quieres predecir cuántos de esos artículos se venderán en los próximos 3 meses.
- Problema 2: Quieres examinar las cuentas individuales de los clientes y para cada una decidir si ha sido hackeada o no.

Posibilidades:

Ambos como problemas de clasificación.

Ambos como problemas de regresión.

Tratar el problema 1 como clasificación y el 2 como regresión.

Tratar el problema 1 como regresión y el 2 como clasificación.



# Contenido

---

- ¿Qué es Machine Learning?
- Aprendizaje supervisado
  - Regresión
  - Clasificación
- **Aprendizaje no supervisado**
  - Clustering
- Conceptos básicos
- Diseño de sistemas de aprendizaje
- Evaluación en Machine Learning
  - Métricas
  - Validación del modelo

# Aprendizaje no supervisado

---



- Objetivo: explorar las propiedades de los datos examinados, no para predecir nuevos datos. Entender y resumir los datos.

# Aprendizaje no supervisado

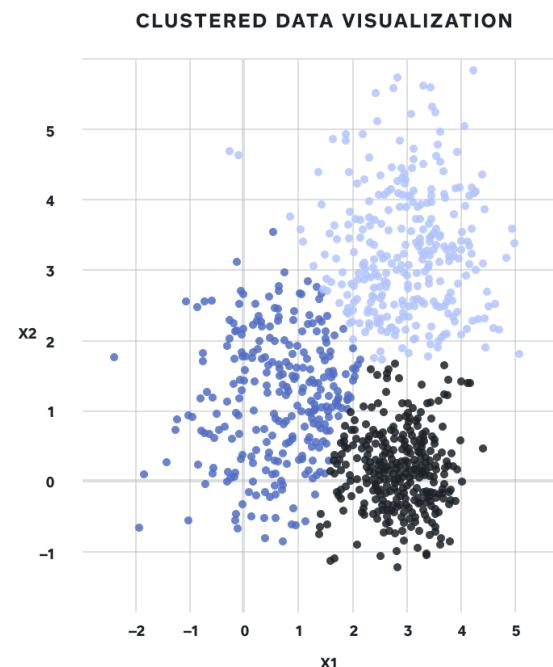
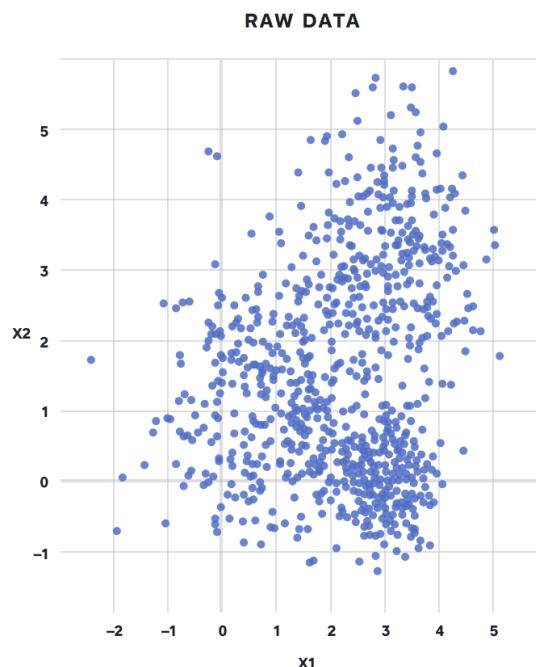


- Clustering:
  - Agrupación de un conjunto de datos (no etiquetados) en grupos de objetos llamados cluster.
    - Cada cluster está formado por una colección de objetos que son similares entre sí, pero que son distintos respecto a los objetos de otros clusters.
- Reglas de asociación:
  - Búsqueda de relaciones dentro del conjunto de datos. Establecer las posibles relaciones entre distintas acciones o sucesos aparentemente independientes.
    - Reconocer cómo la ocurrencia de un suceso o acción puede inducir la aparición de otros.

# Aprendizaje no supervisado



- Ejemplo clustering:



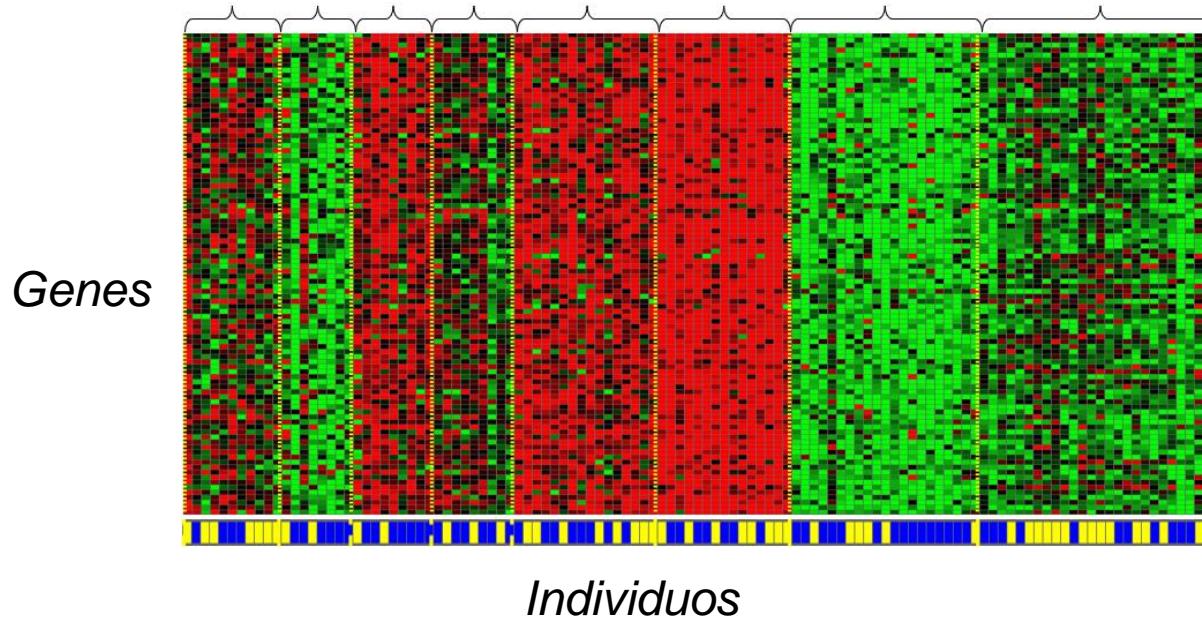
Aprendizaje no supervisado  
Explorar los datos: entender  
y resumir.

Clustering: Agrupaciones de  
datos no etiquetados.

# Aprendizaje no supervisado



- Ejemplo clustering:



Aprendizaje no supervisado  
Explorar los datos: entender  
y resumir.

[Source: Daphne Koller]

Clustering: Agrupaciones de  
datos no etiquetados.

# Aprendizaje no supervisado



¿Cómo deberías tratar los siguientes problemas?

- Problema 1: Un supermercado quiere saber cuáles son los productos que compran los clientes frecuentemente.
- Problema 2: Un servicio de telefonía quiere conocer los grupos de clientes según sus perfiles de consumo.

Posibilidades:

Ambos como problemas de clustering.

Ambos como problemas de reglas de asociación.

Tratar el problema 1 como clustering y el 2 como regla de asociación.

Tratar el problema 1 como regla de asociación y el 2 como clustering.

# Aprendizaje no supervisado



¿Cómo deberías tratar los siguientes problemas?

- Problema 1: Un supermercado quiere saber cuáles son los productos que compran los clientes frecuentemente.
- Problema 2: Un servicio de telefonía quiere conocer los grupos de clientes según sus perfiles de consumo.

Posibilidades:

Ambos como problemas de clustering.

Ambos como problemas de reglas de asociación.

Tratar el problema 1 como clustering y el 2 como regla de asociación.

Tratar el problema 1 como regla de asociación y el 2 como clustering.



# Contenido

---

- ¿Qué es Machine Learning?
- Aprendizaje supervisado
  - Regresión
  - Clasificación
- Aprendizaje no supervisado
  - Clustering
- **Conceptos básicos**
- Diseño de sistemas de aprendizaje
- Evaluación en Machine Learning
  - Métricas
  - Validación del modelo

# Conceptos básicos



- **Dataset o conjunto de datos:**
  - Filas: ejemplos, instancias, puntos u observaciones (*instances*)
  - Columnas: atributos o características (*features*)

- Valor numérico: valor continuo (cuantitativo)
- Valor texto o etiquetas: valores discretos (cualitativos)
- Si es aprendizaje supervisado: necesitamos una columna con el atributo de decisión o clase.

# Conceptos básicos



- Dataset o conjunto de datos:

Atributos o características					Clase
Ejemplo o Instancia	Longitud sépalo (cm)	Anchura sépalo (cm)	Longitud pétalo (cm)	Anchura pétalo (cm)	Clase (Tipo de Flor)
	5.1	3.5	1.4	0.2	Iris-setosa
	4.4	3.2	1.3	0.2	Iris-setosa
	7.0	3.2	4.7	1.4	Iris-versicolor
Valor numérico o continuo	6.4	3.2	4.5	1.5	Iris-versicolor
	6.3	3.3	6.0	2.5	Iris-virginica
	5.8	2.7	5.1	1.9	Iris-virginica

Etiqueta o valor discreto



# Contenido

---

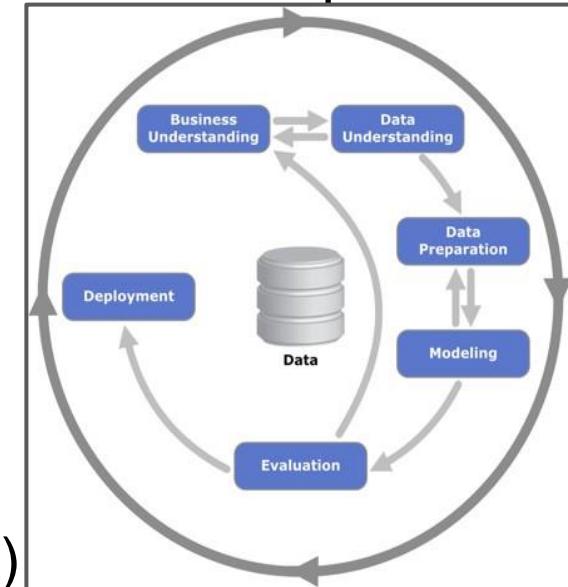
- ¿Qué es Machine Learning?
- Aprendizaje supervisado
  - Regresión
  - Clasificación
- Aprendizaje no supervisado
  - Clustering
- Conceptos básicos
- **Diseño de sistemas de aprendizaje**
- Evaluación en Machine Learning
  - Métricas
  - Validación del modelo

# Diseño de sistemas de aprendizaje



- **Metodología CRISP-DM:**

1. Comprensión del **negocio**: Identificar el problema y plantear objetivos
2. Comprensión de los **datos**
  - Análisis exploratorio de datos
3. **Preparación** de los datos
  - Extracción, transformación y carga (ETL)
  - Preprocesado de datos (limpieza, selección, ...)
4. **Modelado**: elegir, configurar y aplicar
  - Machine learning
5. **Evaluación** de negocio: análisis e interpretación
6. **Despliegue** y difusión



# Diseño de sistemas de aprendizaje



## 2. Comprensión de los datos:

- En el proceso de aprendizaje la experiencia *E* es crucial.
- Esa experiencia serán los datos que usaremos para “aprender”.
- El proceso de aprendizaje se llama entrenamiento o ***training***.
- Los datos que se usan durante el proceso de aprendizaje son los datos de entrenamiento o ***training set***. Deben ser de alta calidad y completos.
- ¿Los datos del problema son representativos y sirven para modelar correctamente el problema?

# Diseño de sistemas de aprendizaje



## **2. Comprensión de los datos:**

- Posibles problemas del conjunto de datos de entrenamiento:
  - Ruido
  - Valores perdidos/ausentes/missings
  - Proporción entre atributos e instancias
  - Problemas de dimensión y mala selección de atributos

# Diseño de sistemas de aprendizaje



## 3. Preparación de los datos:

- ETL:
  - Extracción de datos: repositorios personales, repositorios open data, bases de datos, APIs, Web scrapping, etc.
  - Transformación de datos: cambiar formato/tipo de datos, filtros, reestructuración, creación, limpieza, etc.
  - Carga de datos (*Load*): Guardar después de las transformaciones.
- Preprocesado de datos:
  - Objetivo: Conseguir un conjunto de datos de entrenamiento completo y de buena calidad.
    - Tratamiento de valores ausentes, detección de outliers, normalización de datos, discretización, selección de atributos, etc.

# Diseño de sistemas de aprendizaje



## 4. Modelado:

- En todos los problemas definiremos una función  $f$  desconocida. **Ejemplo:** ¿Qué queremos aprender?
  - Precios casa                     $f: \text{Casa} \rightarrow \mathbb{R}$
  - Filtro del spam                 $f: \text{Email} \rightarrow \{\text{Spam}, \text{No Spam}\}$
  - Segmentación del mercado     $f: \text{Cliente} \rightarrow \{\text{A}, \text{B}, \text{C}, \dots\}$
- Se aprenderá una función  $h$  (hipótesis) y se aproximará a  $f$ . Podrá ser una función lineal/no lineal/....
- Tipos de modelos: “*black box*” / “*symbolic modeling*”.

# Diseño de sistemas de aprendizaje



## 4. Modelado:

- Elegir el modelo de machine learning:
  - Hay diferentes algoritmos para cada tipo de problema (problema de regresión, de clasificación, de clustering, etc.)
  - Pueden aplicarse varios modelos y comparar sus comportamientos y errores (normalmente mediante tests estadísticos).
  - Normalmente los algoritmos tienen parámetros. Se buscan los óptimos.
  - **EVALUACIÓN Y MÉTRICAS DEL MODELADO**



# Contenido

---

- ¿Qué es Machine Learning?
- Aprendizaje supervisado
  - Regresión
  - Clasificación
- Aprendizaje no supervisado
  - Clustering
- Conceptos básicos
- Diseño de sistemas de aprendizaje
- **Evaluación en Machine Learning**
  - Métricas
  - Validación del modelo

# Evaluación en Machine Learning



- Para saber cómo de bueno o malo es el modelo aprendido.
- **Métricas:**
  - **Clustering:** Calidad difícil de determinar.
    - En algunos casos:
      - Comparar resultados con información/documentación externa (personas expertas, otros estudios, ...).
      - Observar la tendencia de los clusters.
      - Índices que calculan una relación entre las distancias dentro de cada cluster (cohesión) y las distancias entre clusters (separación).
      - ...

# Evaluación en Machine Learning



- **Métricas:**

- **Regresión:** Diferencia entre valor real ( $y_i$ ) y valor predicho ( $\hat{y}_i$ ) siendo  $m$  el número de instancias. Las más comunes:

- Error absoluto medio (MAE):

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

- Error cuadrático medio (MSE):

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

- Raíz del error cuadrático medio (RMSE):

$$RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2}$$

# Evaluación en Machine Learning



- **Métricas:**

- **Clasificación:** Se suele usar la matriz de confusión:
  - Tasa de error: Proporción de errores cometidos sobre el conjunto entero de instancias del entrenamiento.
  - Éxito: La clase de la instancia es predicha correctamente.
  - Error: La clase de la instancia es predicha incorrectamente.

		Clase real	
		Positive	Negative
Predicción	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Accuracy	$\frac{TP + TN}{TP + FN + TN + FP}$
Precisión	$\frac{TP}{TP + FP}$
Sensibilidad (recall)	$\frac{TP}{TP + FN}$
F1 Score	$\frac{2 \times precision \times recall}{precision + recall}$

# Evaluación en Machine Learning



- **Conjunto de datos para la evaluación:**
  - Debe ser distinto al conjunto de entrenamiento para evitar conclusiones erróneas.
  - Conjuntos / Sets:
    - Training: Entrenamiento de los modelos.
    - Validation: Ajustar los hiperparámetros con el objetivo de seleccionar el mejor modelo.
    - Test: Evaluación del modelo con registros con los que no se ha entrenado ni ajustado el modelo.



# Evaluación en Machine Learning

---



- **Validación del modelo:**
  - Hold-out.
  - Validación cruzada o *k*-fold cross validation.
  - Validación cruzada con repetición.
  - Validación cruzada “Leave One Out”.

# Evaluación en Machine Learning

---



- **Validación del modelo:**

- Hold-out: Dividir los datos en training y test.
  - Solución sencilla si tenemos muchos datos etiquetados.
  - Normalmente aproximadamente: 70% para training y 30% para test (instancias del test no incluidas en el training ¡IMPORTANTE!).
  - Normalmente:
    - Más grande el conjunto de training → mejor es el modelo
    - Más grande el conjunto de test → más aproximada la estimación del error.

# Evaluación en Machine Learning



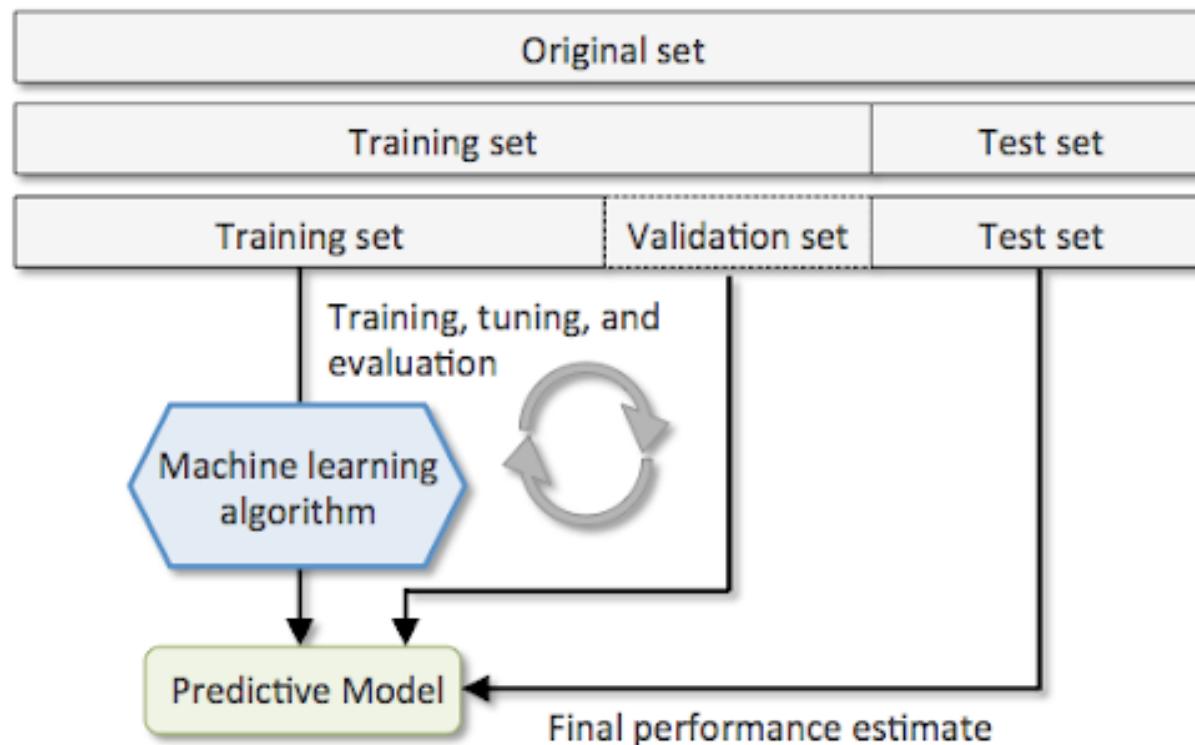
- **Validación del modelo:**

- Hold-out: Dividir los datos en training y test.
  - Problemas:
    - Las muestras podrían no ser representativas (otros métodos de validación más avanzados usan la estratificación para resolver este problema).
    - Número de instancias bajo, y por tanto, el conjunto de test será muy pequeño.
  - Si hay que seleccionar los parámetros óptimos (parameter tuning process):
    - No usar para ese proceso los datos de test.
    - Se usarán los datos del conjunto de validación para optimizar los parámetros.
    - Normalmente: 70% training y validation (70% y 30% respectivamente) + 30% test.

# Evaluación en Machine Learning



- Validación del modelo:
  - Hold-out: Dividir los datos en training y test.



# Evaluación en Machine Learning



- **Validación del modelo:**

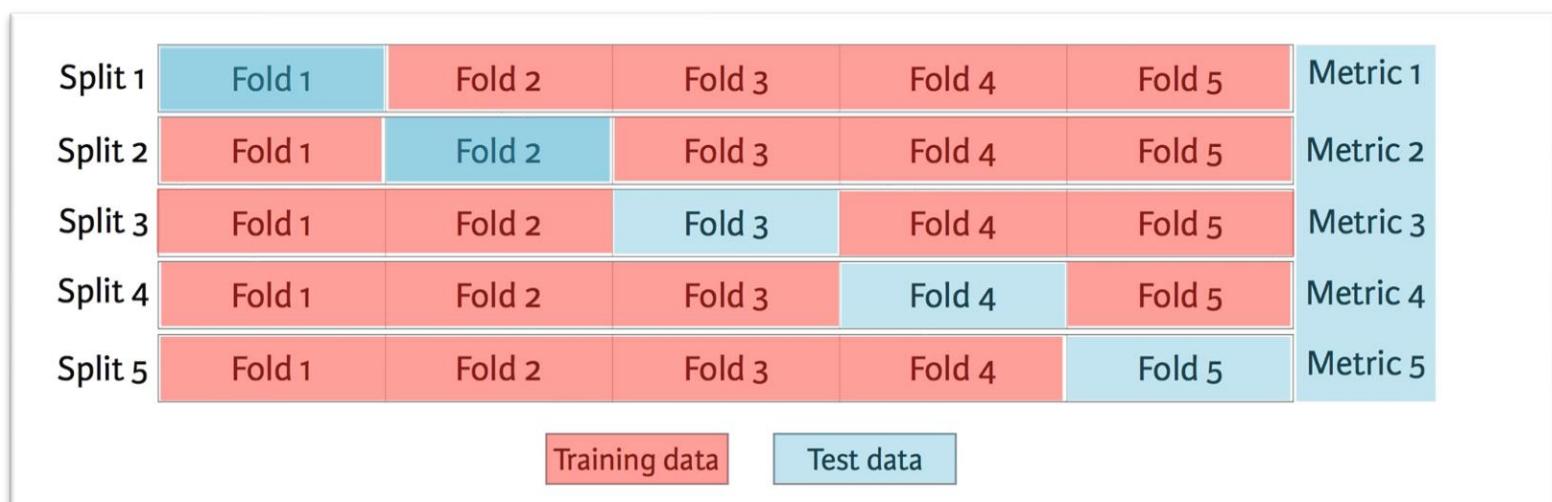
- Validación cruzada o *k*-fold cross validation: Se generan diferentes conjuntos de training y test.
  - Normalmente se estratifican los datos antes: asegura que el training es representativo.
  - El dataset se partitiona en *k* subconjuntos/carpetas/folds.
  - Se entrena el modelo y se testeá *k* veces (cada vez con un conjunto de training y test distinto).
  - Las estimaciones de error se promedian para obtener una estimación de error global.

# Evaluación en Machine Learning



- **Validación del modelo:**

- Validación cruzada o *k*-fold cross validation: Se generan diferentes conjuntos de training y test.

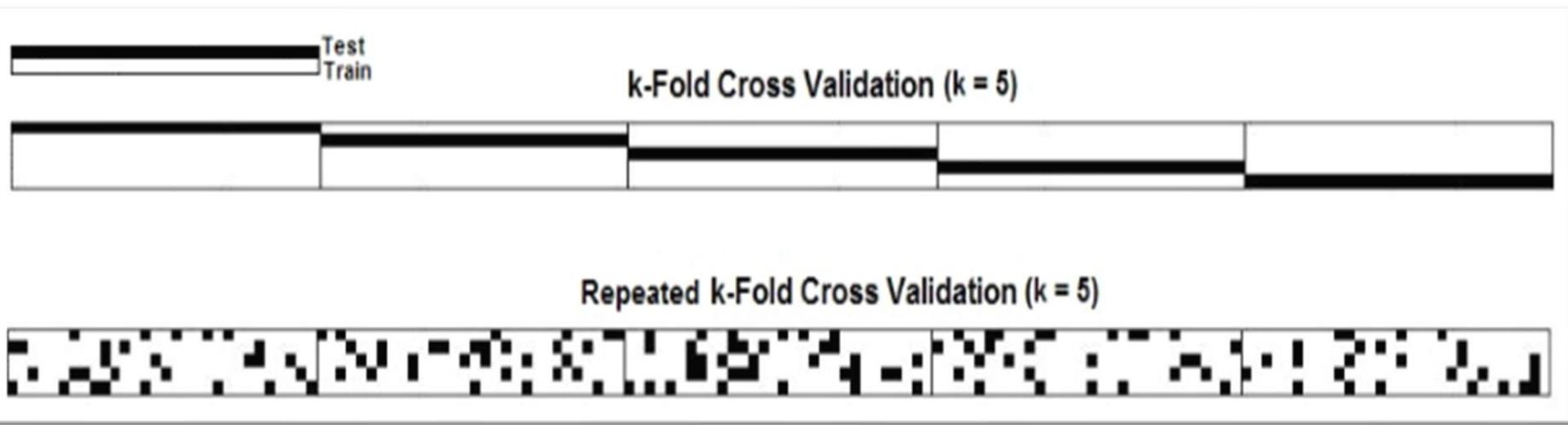


# Evaluación en Machine Learning



- **Validación del modelo:**

- Validación cruzada con repetición:
  - Se hacen  $k$  validaciones cruzadas y en cada una de ellas se mezclan y seleccionan aleatoriamente los datos  $R$  veces.
  - Se realizarán  $k \times R$  ejecuciones.



# Evaluación en Machine Learning



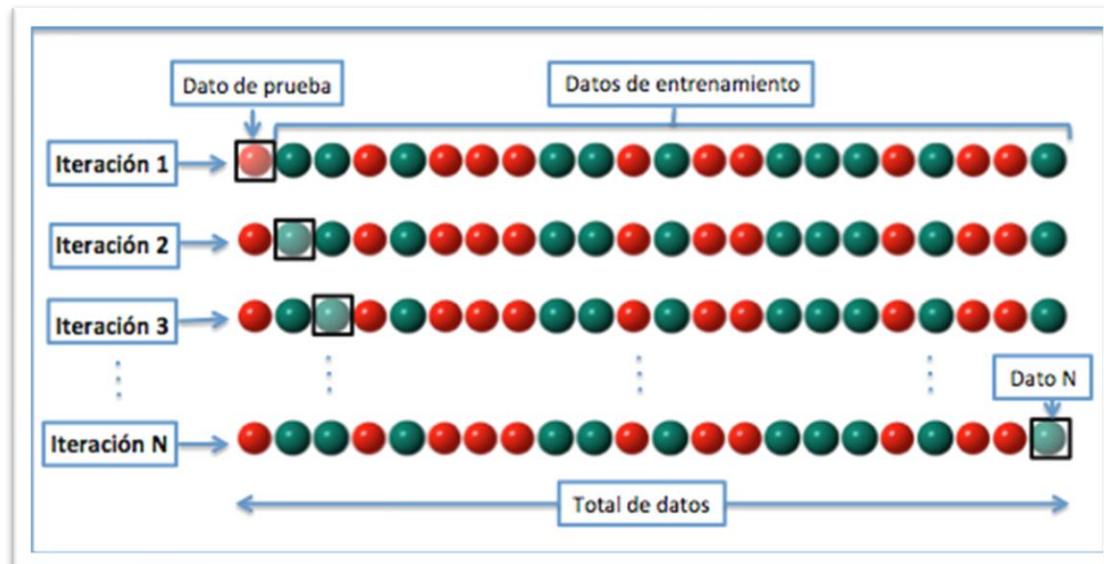
- **Validación del modelo:**

- Validación cruzada “Leave One Out” (LOOCV):
  - Siendo  $n$  el número de instancias de datos:
    - Se realizan  $n$  iteraciones del modelo y en cada una de ellas el conjunto de test es una sola instancia. El conjunto de training está formado por  $n-1$  instancias.
  - Es muy costoso computacionalmente.
  - No implica ningún submuestreo aleatorio.
  - No se puede hacer ninguna versión estratificada.

# Evaluación en Machine Learning

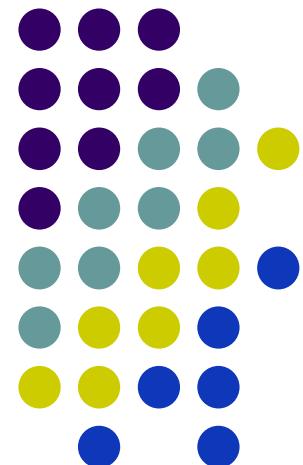


- **Validación del modelo:**
  - Validación cruzada “Leave One Out” (LOOCV):



# Regresión lineal

Inteligencia Artificial  
Ingeniería Informática  
en Sistemas de Información





# Contenido

---

- ❑ Regresión lineal univariable
  - ❑ Representación del modelo
  - ❑ Función coste
  - ❑ Descenso del gradiente
  
- ❑ Regresión lineal multivariable
  - ❑ Representación del modelo
  - ❑ Función coste y descenso del gradiente
  - ❑ Feature Scaling / Escalado de variables
  - ❑ Ecuación normal

# Regresión lineal univariable:



- Aprendizaje supervisado:
  - Dada la “respuesta correcta” para cada ejemplo del entrenamiento.
  - Tipo: Problema de regresión: Predice la salida que es un valor real.
  - El número de características/variables en el conjunto de entrenamiento es 1.

# Regresión lineal univariable:

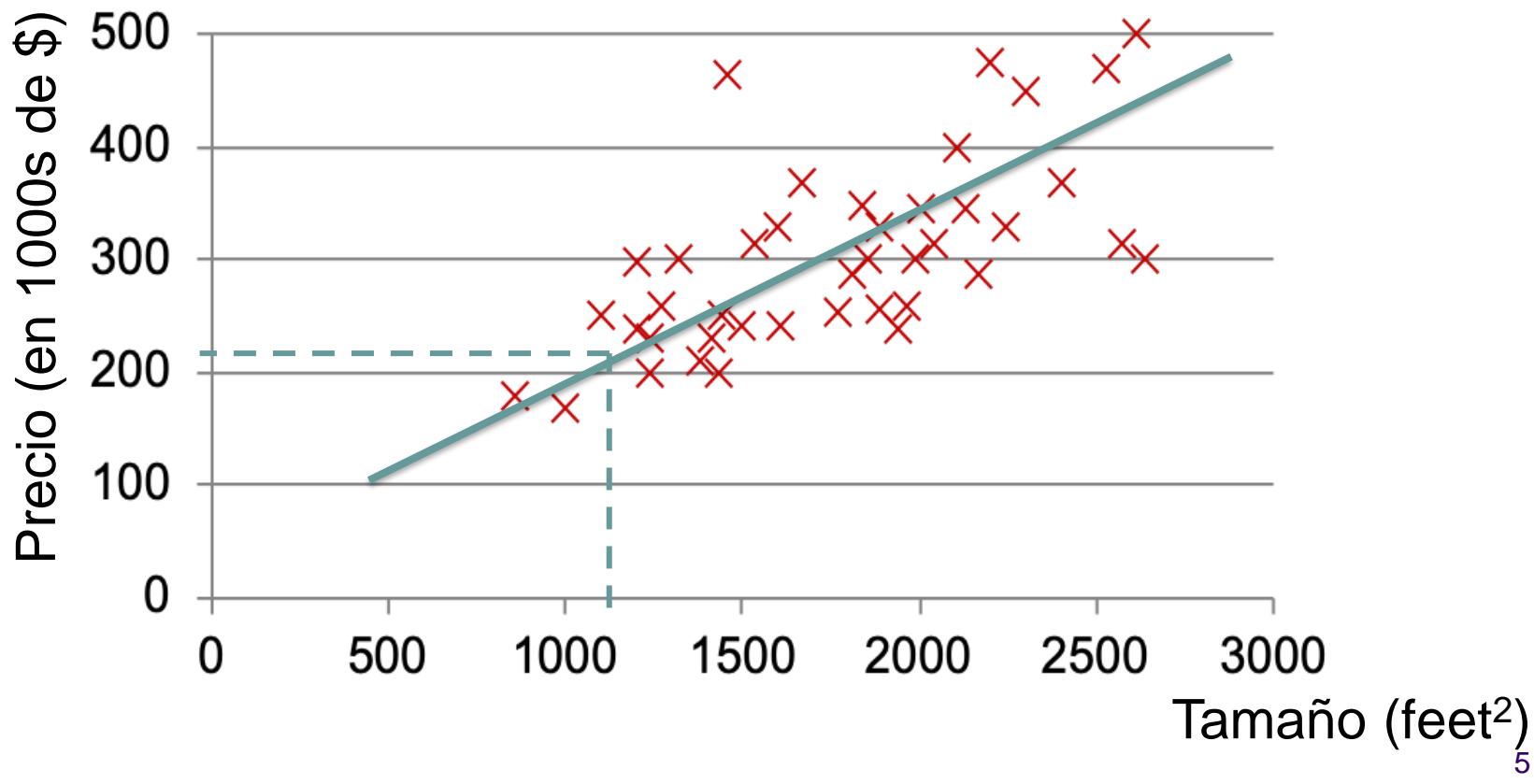


- La realidad se modela con base a la recta que mejor ajusta el conjunto de entrenamiento.
- Se realizan múltiples iteraciones para ir ajustando, en cada una de ellas, los coeficientes de la recta.
- El número de iteraciones viene determinado por una tolerancia de error o explícitamente por el usuario.

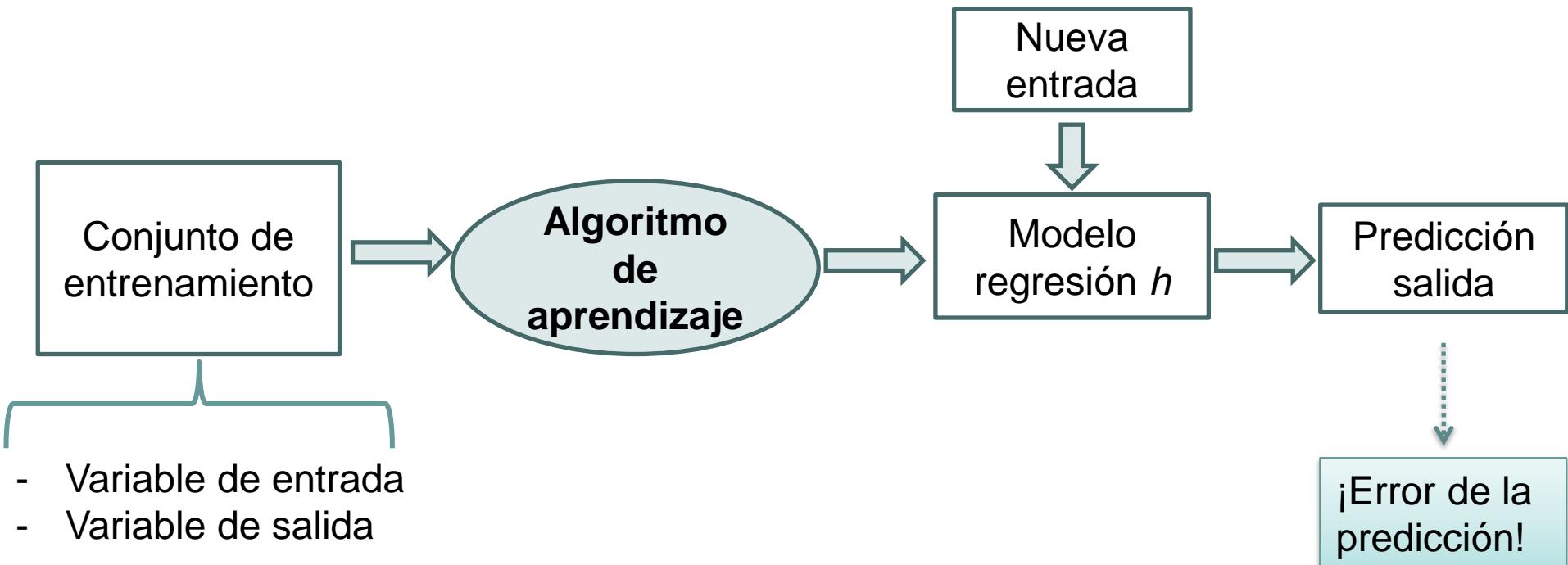
# Regresión lineal univariable:



Problema: Predecir los precios de las casas en Portland (OR) considerando sus tamaños.



# Regresión lineal univariable: Representación del modelo



# Regresión lineal univariable: Representación del modelo



- Notación

$m$  = número de ejemplos / instancias del conjunto de entrenamiento

$x$ 's = variable de entrada / atributo

$y$ 's = variable de salida /variable “target”

$(x^i, y^i)$ : Instancia “ $i$ ” del conjunto de entrenamiento.

# Regresión lineal univariable: Representación del modelo



Problema: Predecir los precios de las casas en Portland (OR) considerando sus tamaños. El conjunto de entrenamiento contiene 200 instancias.

Tamaño ( $x$ )	Precio ( $y$ )
2104	460
1416	232
1534	315
852	178
...	...

$$m = 200$$

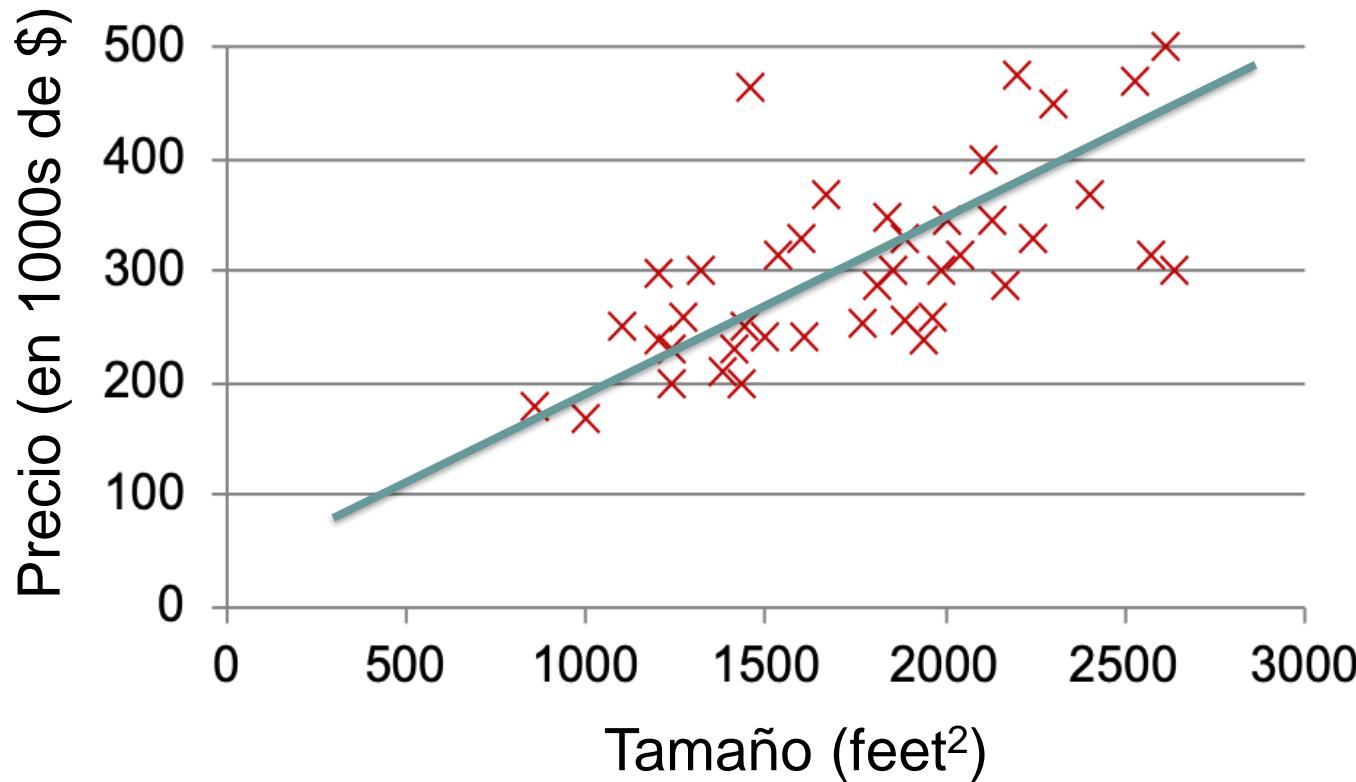
$$(x^2, y^2) = (1416, 232)$$

# Regresión lineal univariable: Representación del modelo



Hipótesis del modelo de regresión lineal univariable:

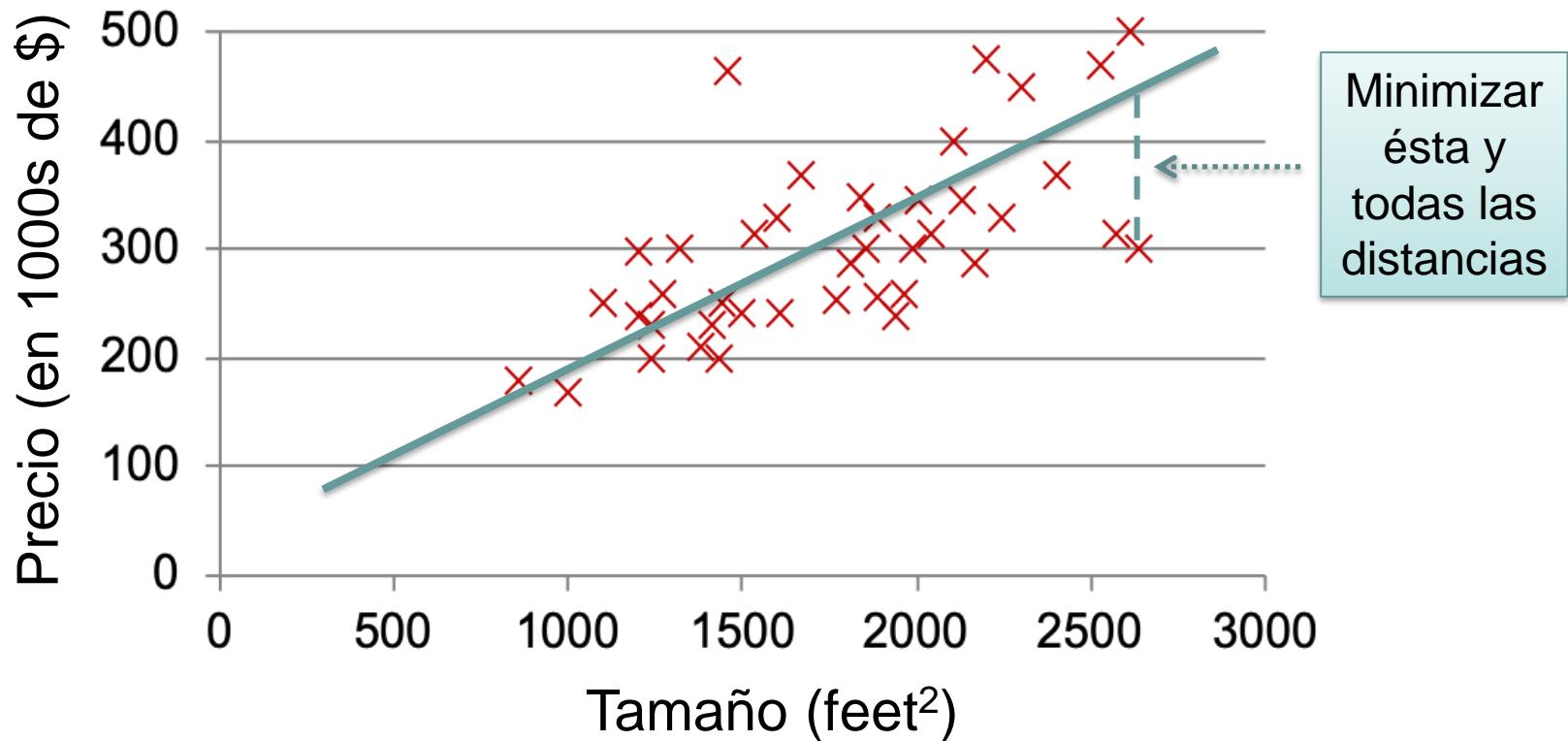
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



# Regresión lineal univariable: Representación del modelo



Idea: Elegir los parámetros  $\theta_0$  y  $\theta_1$  tal que  $h_{\theta}(x)$  se asemeje a  $y$  para el conjunto de datos de entrenamiento  $(x, y)$ .



# Regresión lineal univariable: Función coste



Hay diferentes formas válidas para calcular la diferencia entre los datos reales y los predichos.

- Nosotros usaremos una de las más comunes: Error cuadrático.
- Esa será nuestra función coste y dependerá de  $\theta_0$  y  $\theta_1$ :

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^i) - y^i)^2$$

- El objetivo es encontrar  $\theta_0$  y  $\theta_1$  que minimizan la función coste:  
 $minimizar_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

# Regresión lineal univariable: Función coste



- Hipótesis es una función de  $x$  para cualquier valor fijado de  $\theta_1$  distinto de 0.
- Hipótesis es una recta.
- Cada par de parámetros  $\theta_0$  y  $\theta_1$  definen un modelo  $h$ .
- Función coste es una función de los parámetros  $\theta_0$  y  $\theta_1$ .
- Si la función coste es el error cuadrático, tiene naturaleza convexa y por tanto tiene un solo mínimo (global).
- Encontrar el modelo  $h$  que alcanza el mínimo global.

# Regresión lineal univariable: Función coste



- **Ejemplo 1:** Fijar  $\theta_0$  a 0. 3 instancias para el entrenamiento: (1,1), (2,2), (3,3).

$$h_{\theta}(x) = \theta_1 x \quad J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (\theta_1 x^i - y^i)^2$$

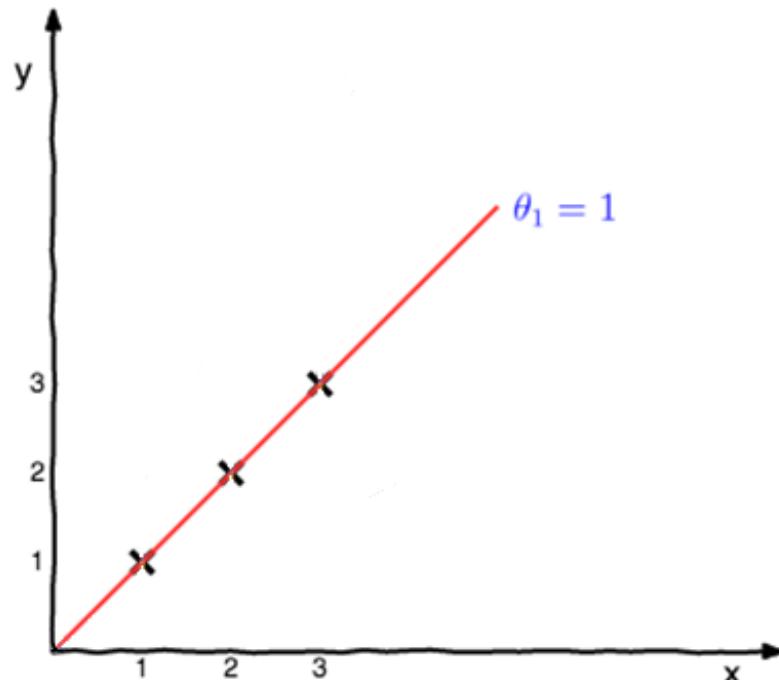
$\theta_1$	x	y	$h_{\theta}(x) = \theta_1 x$	$h_{\theta}(x) - y$	$J(\theta_1)$
1	1	1	1	0	0,000
	2	2	2	0	
	3	3	3	0	
2	1	1	2	1	2,333
	2	2	4	2	
	3	3	6	3	
0,5	1	1	0,5	-0,5	0,583
	2	2	1	-1	
	3	3	1,5	-1,5	

# Regresión lineal univariable: Función coste

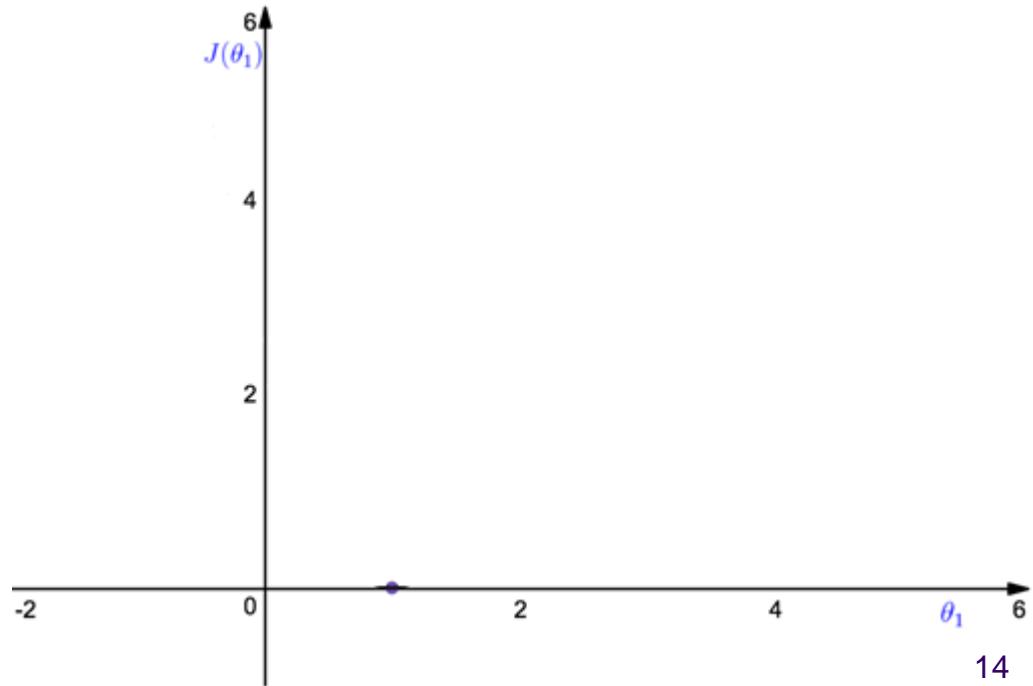


- **Ejemplo 1:** Fijar  $\theta_0$  a 0. 3 instancias para el entrenamiento: (1,1), (2,2), (3,3).

Representación hipótesis



Representación  $\theta_1$  y función coste

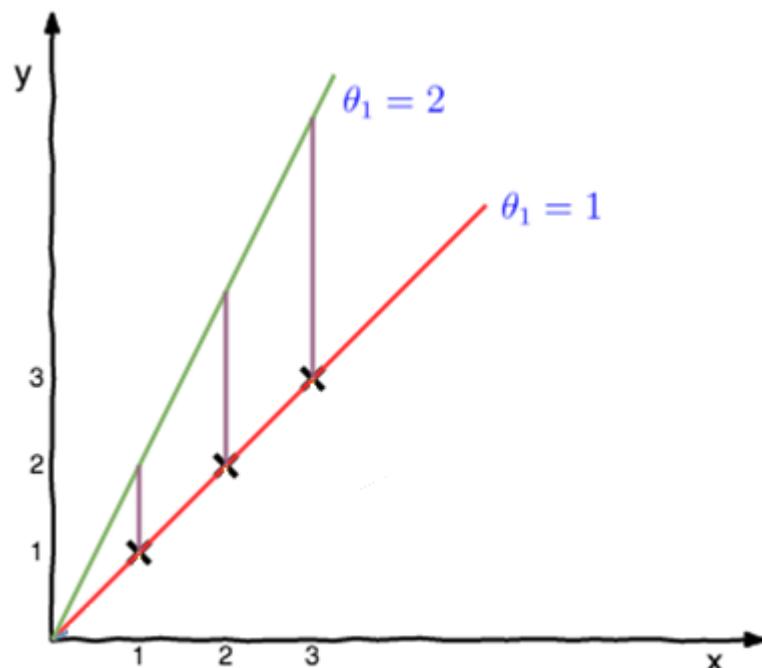


# Regresión lineal univariable: Función coste

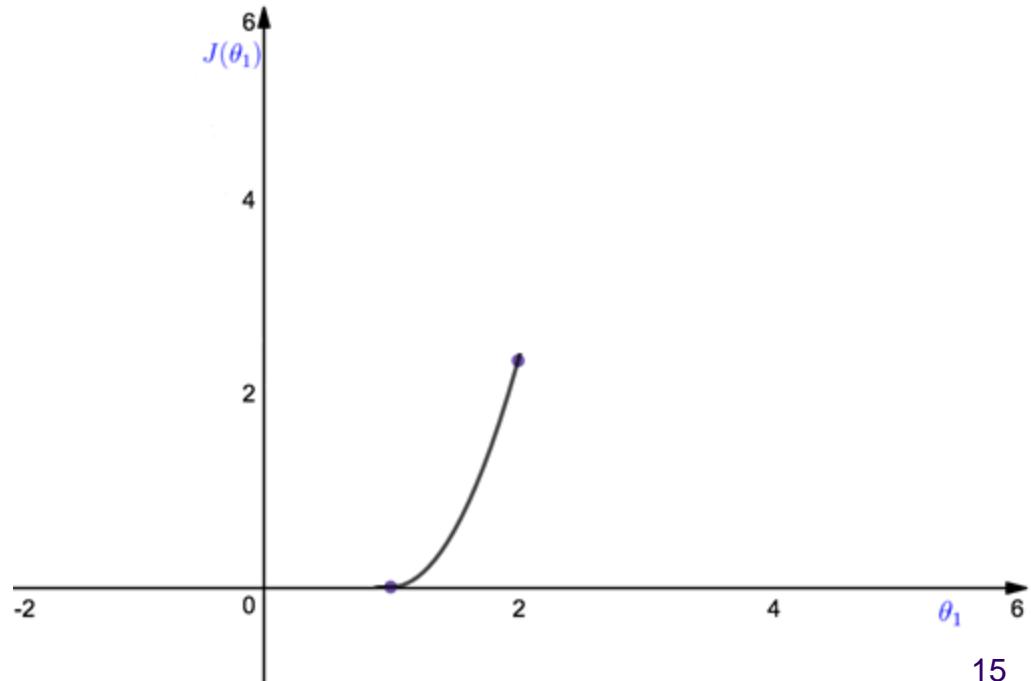


- Ejemplo 1:** Fijar  $\theta_0$  a 0. 3 instancias para el entrenamiento: (1,1), (2,2), (3,3).

Representación hipótesis



Representación  $\theta_1$  y función coste

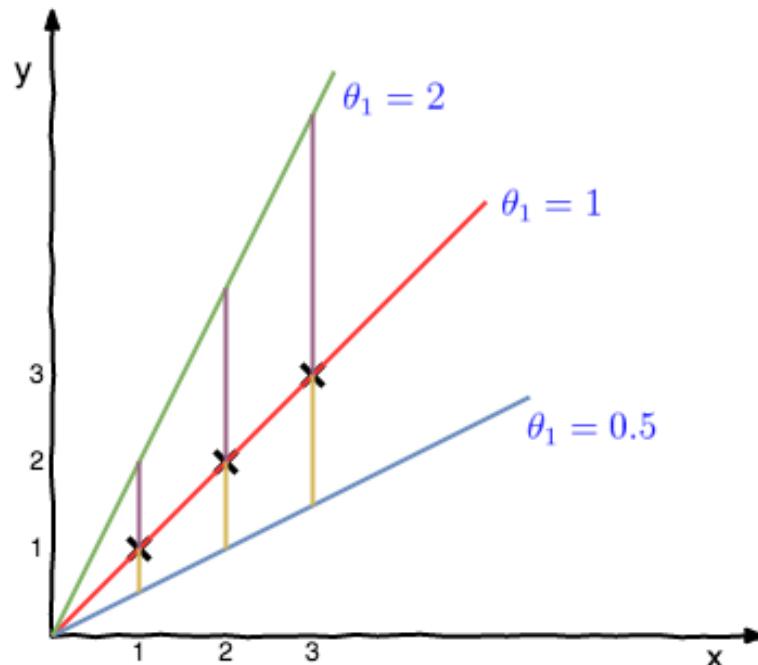


# Regresión lineal univariable: Función coste

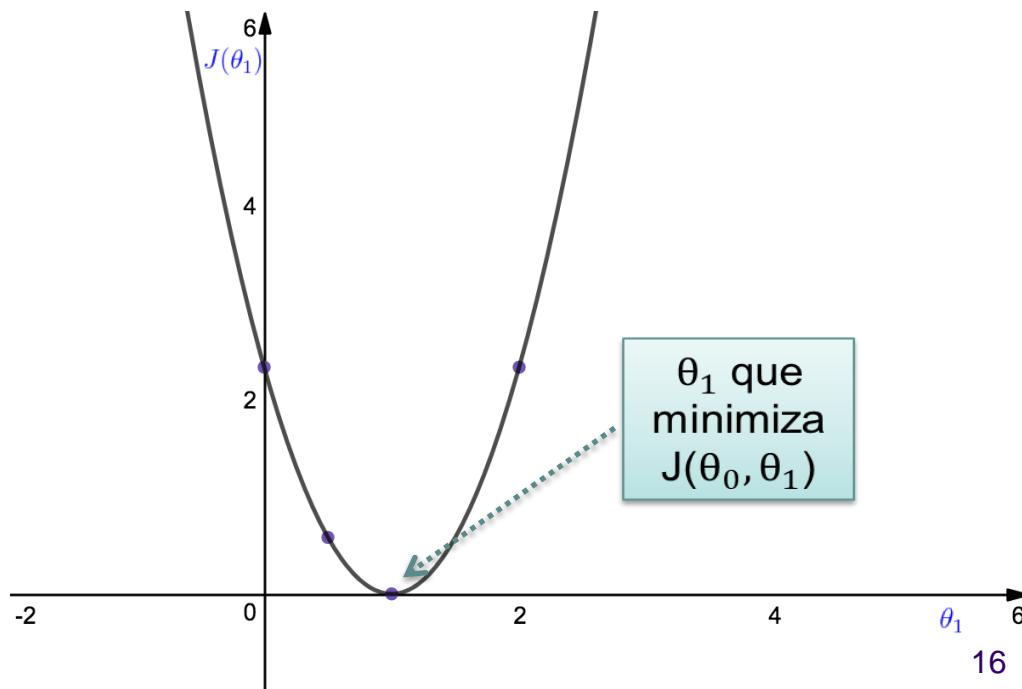


- Ejemplo 1:** Fijar  $\theta_0$  a 0. 3 instancias para el entrenamiento: (1,1), (2,2), (3,3).

Representación hipótesis



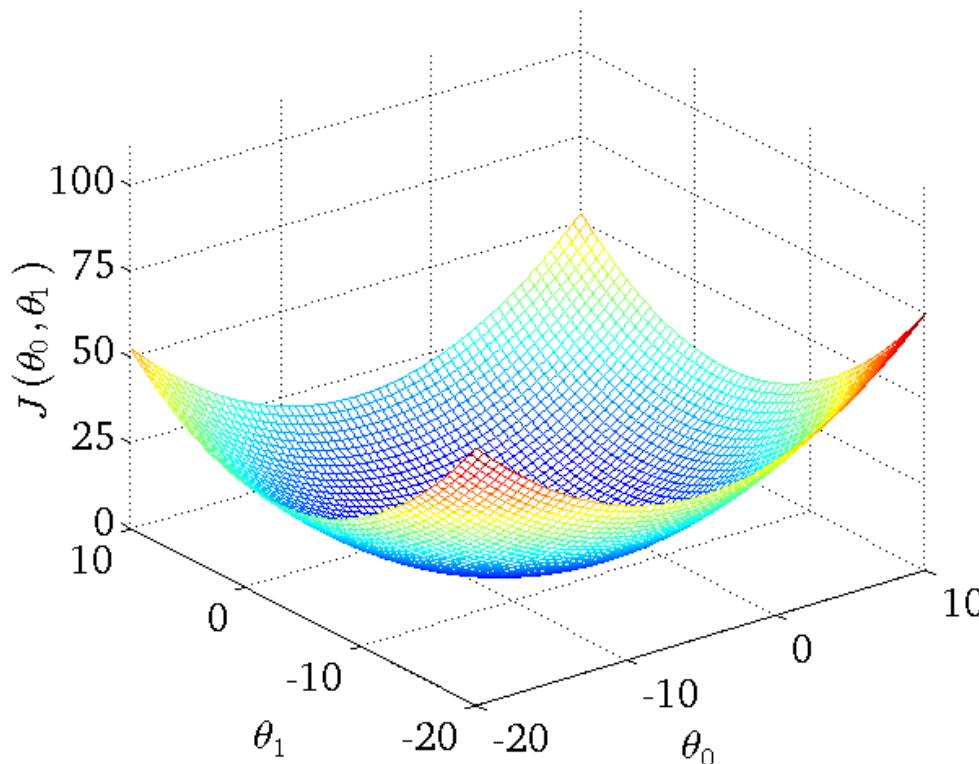
Representación  $\theta_1$  y función coste



# Regresión lineal univariable: Función coste



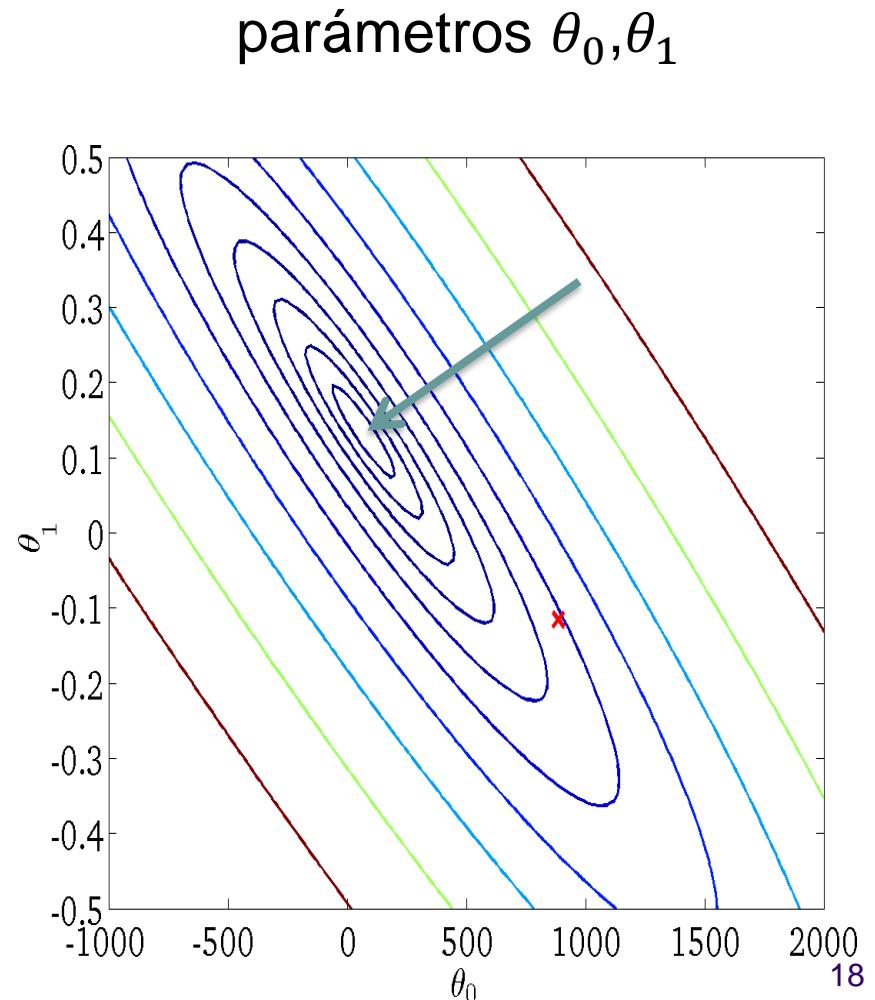
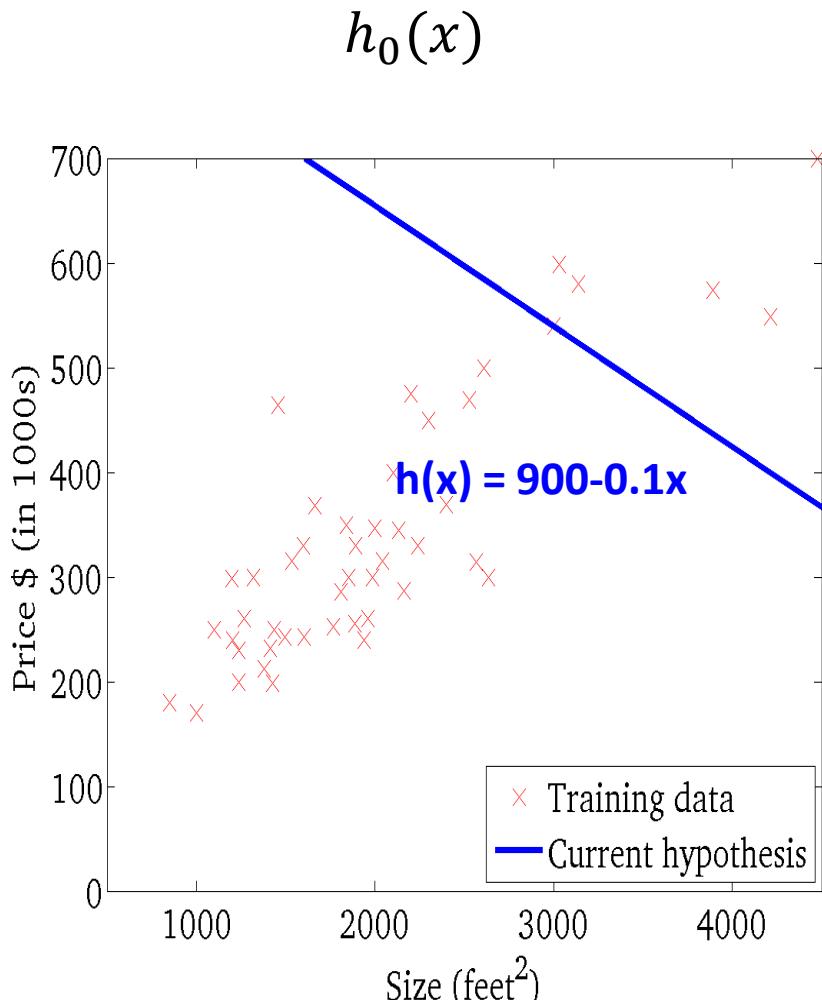
- **Ejemplo 2:** Sin fijar  $\theta_0$  a 0.
  - Se suele recurrir a gráficas 3d “*contour plots*”.



# Regresión lineal univariable: Función coste



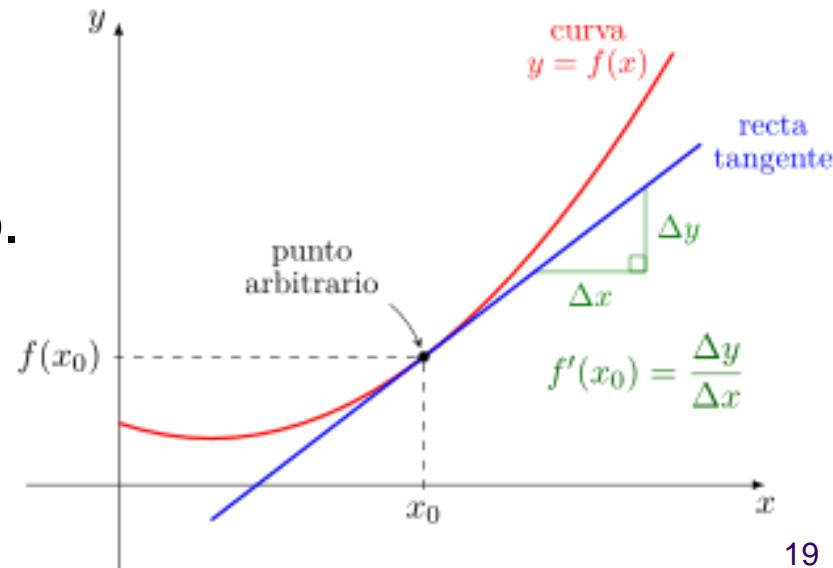
- Ejemplo 2: Sin fijar  $\theta_0$  a 0.



# Regresión lineal univariable: Descenso del gradiente



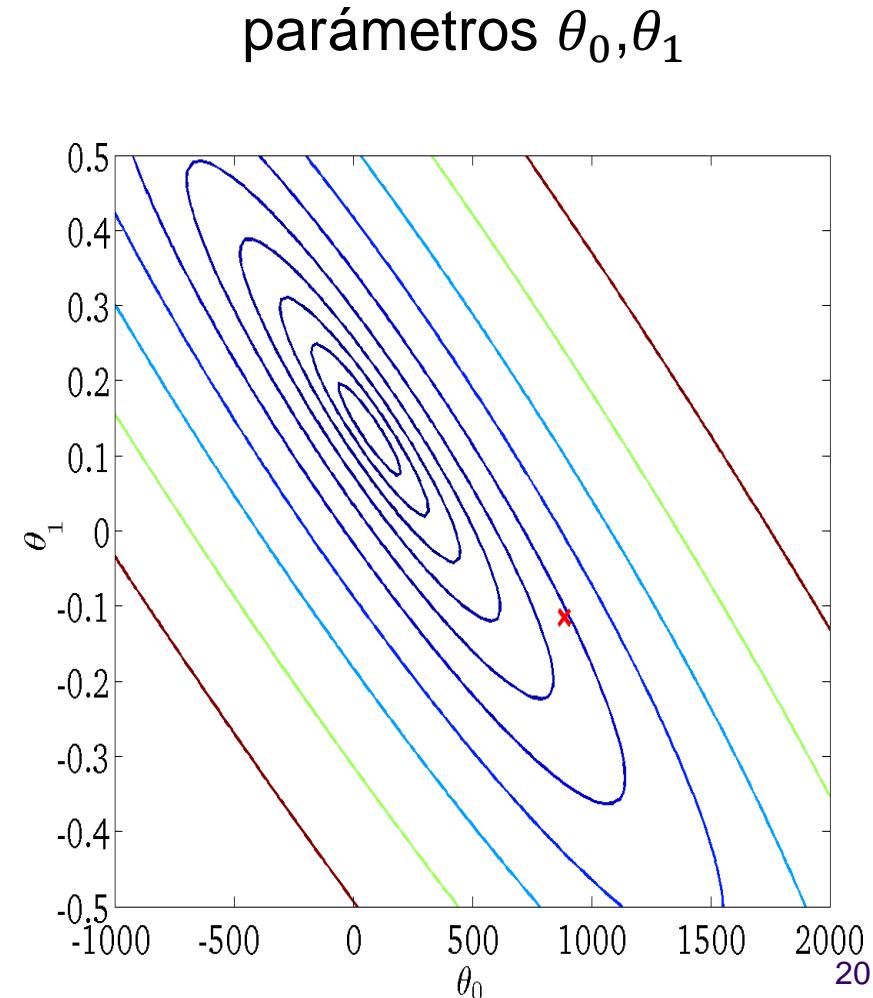
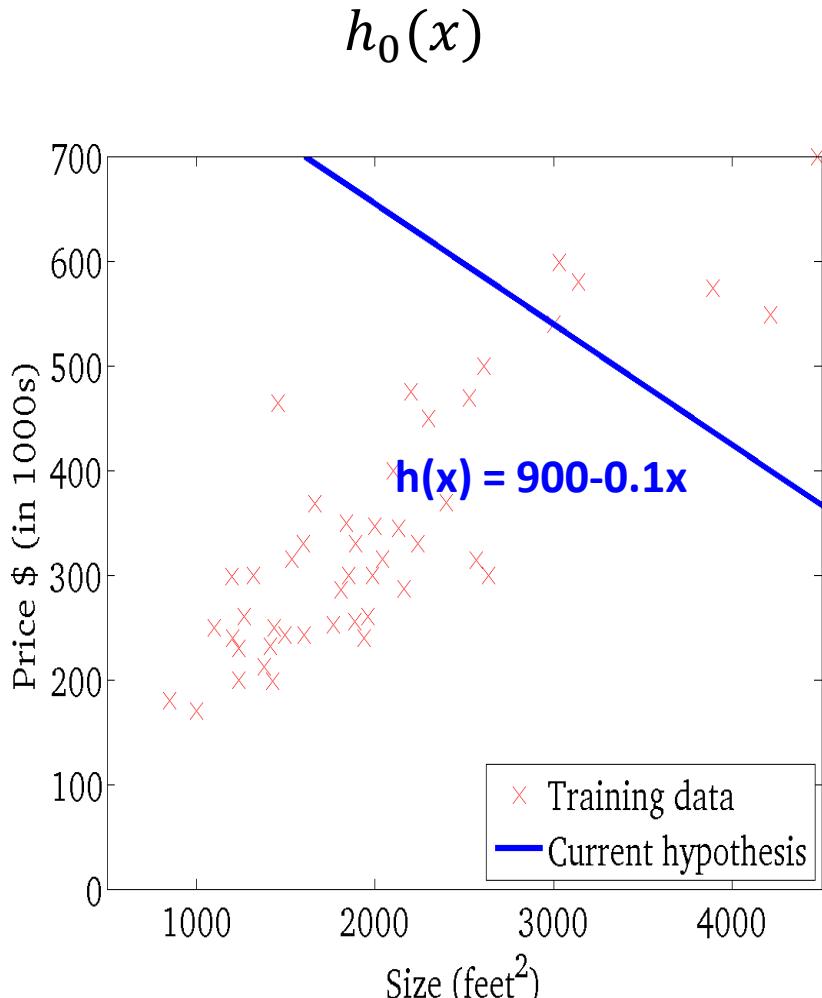
- Algoritmo de optimización iterativo.
- Objetivo: encontrar los mejores parámetros que minimizan la función coste.
- Usa el concepto de la derivada:  
La derivada de una curva en un punto es la pendiente de la recta tangente a esa curva en ese punto.



# Regresión lineal univariable: Descenso del gradiente



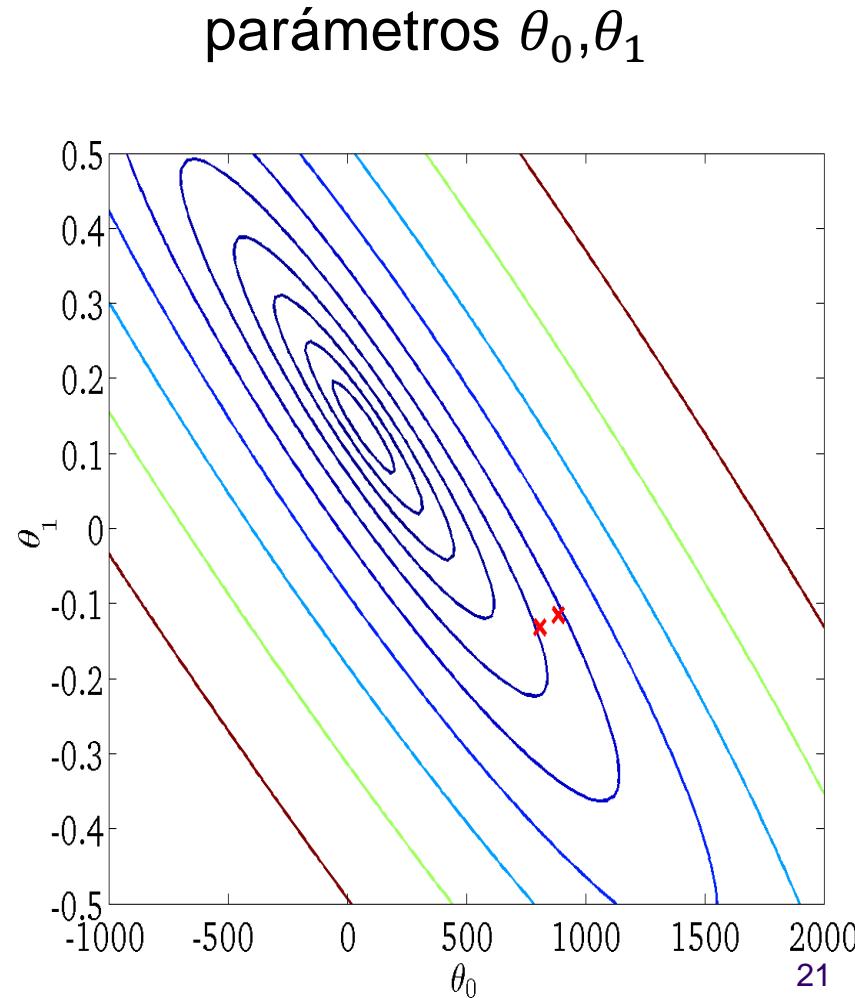
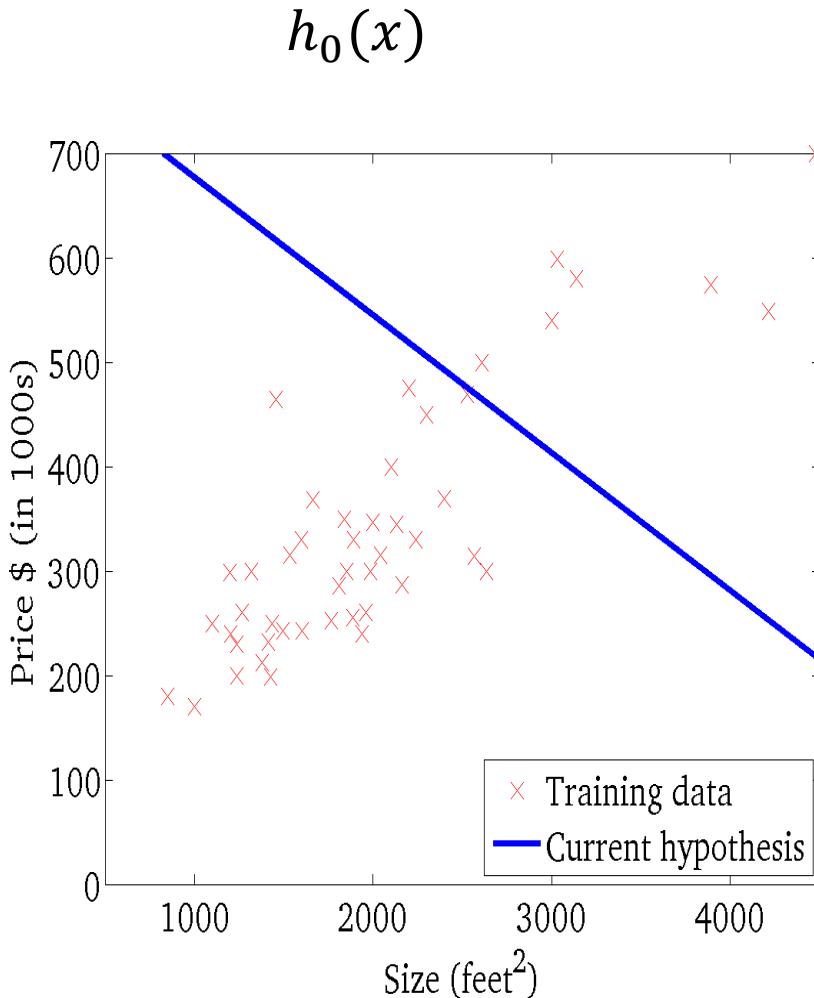
## Método iterativo



# Regresión lineal univariable: Descenso del gradiente



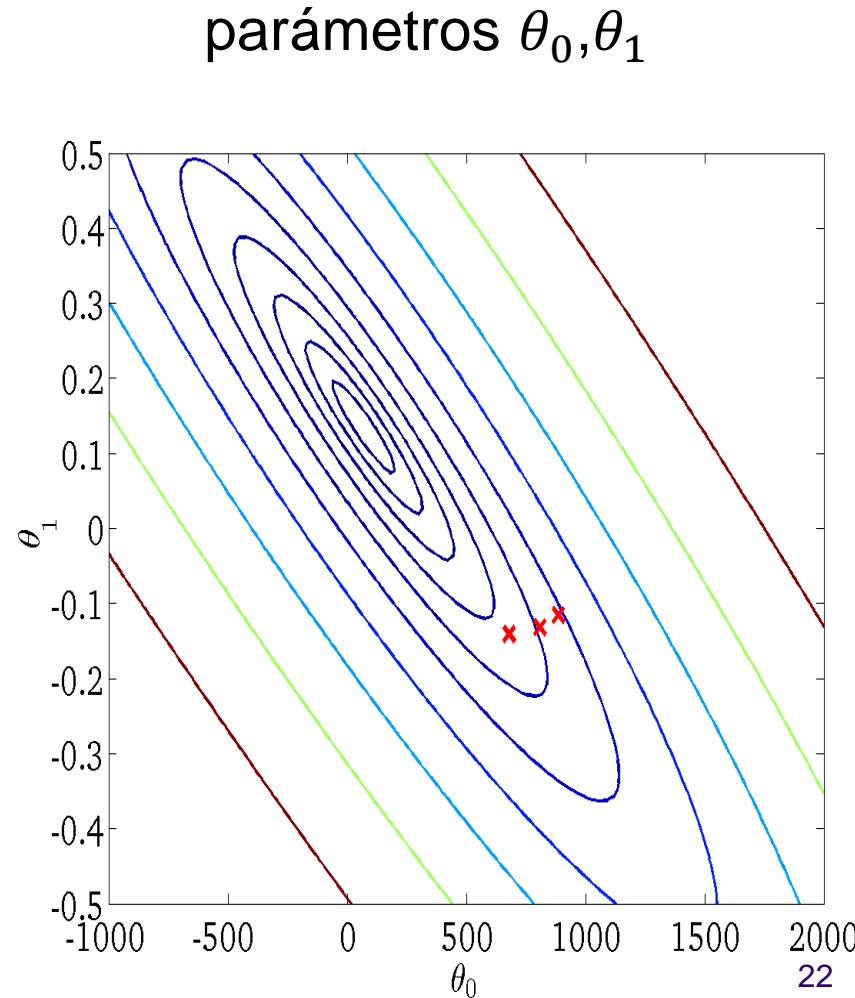
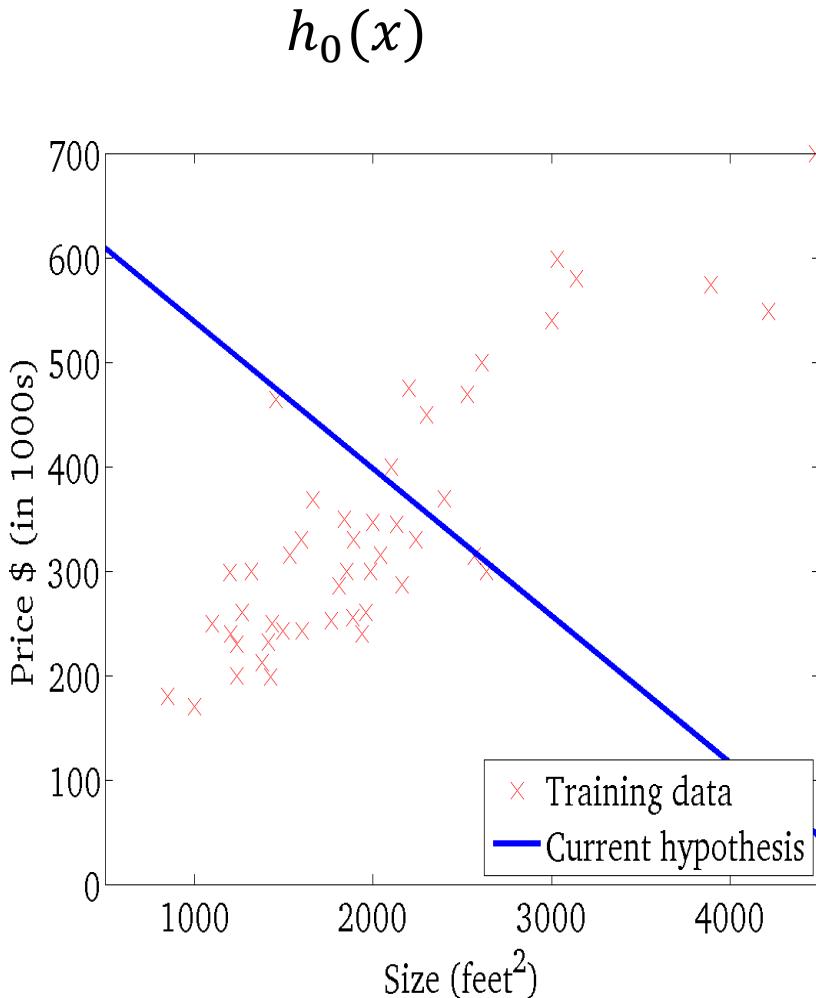
## Método iterativo



# Regresión lineal univariable: Descenso del gradiente



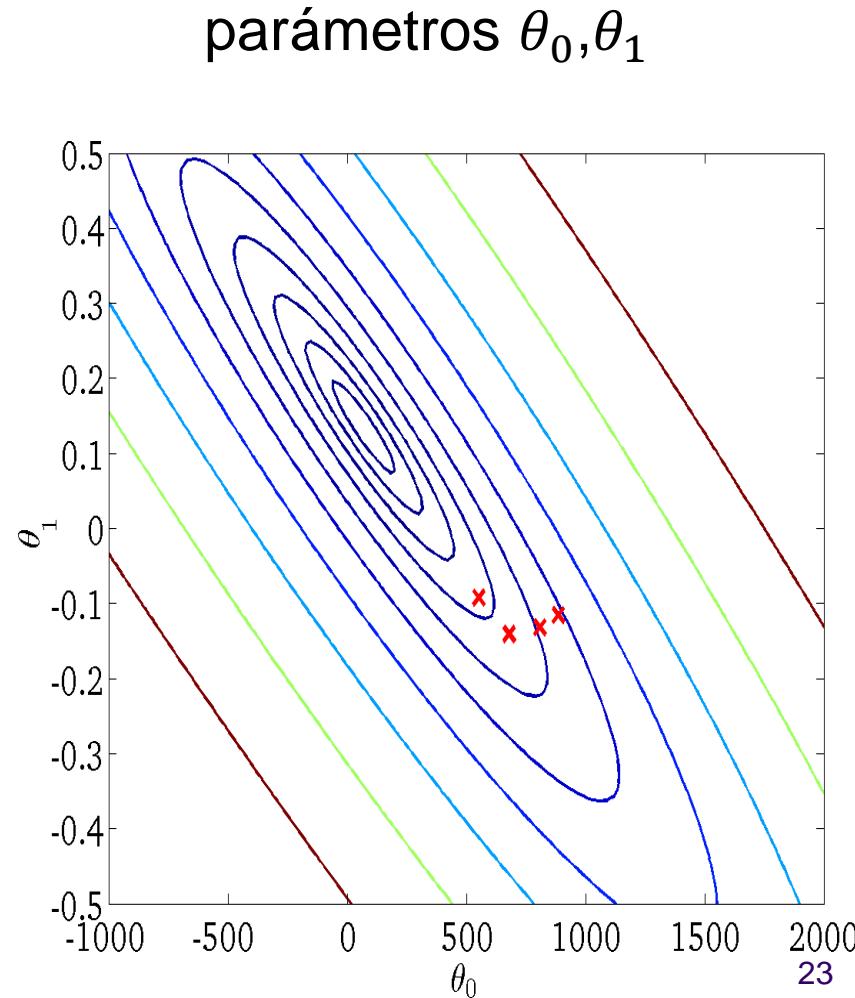
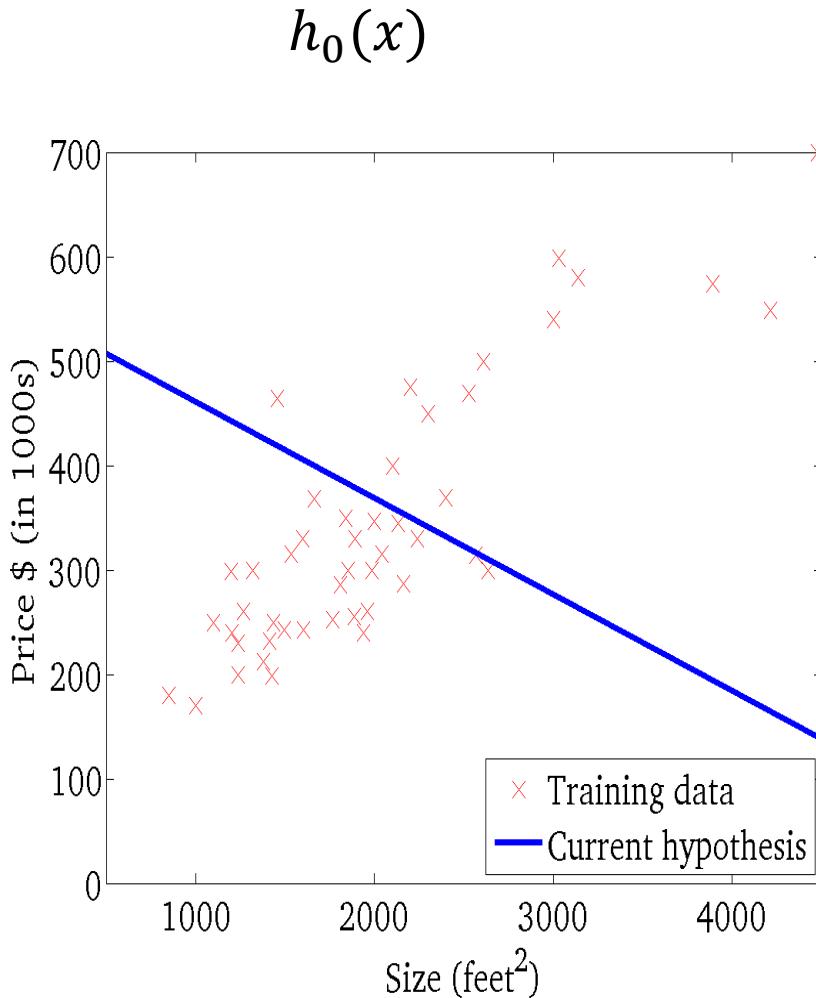
## Método iterativo



# Regresión lineal univariable: Descenso del gradiente



## Método iterativo

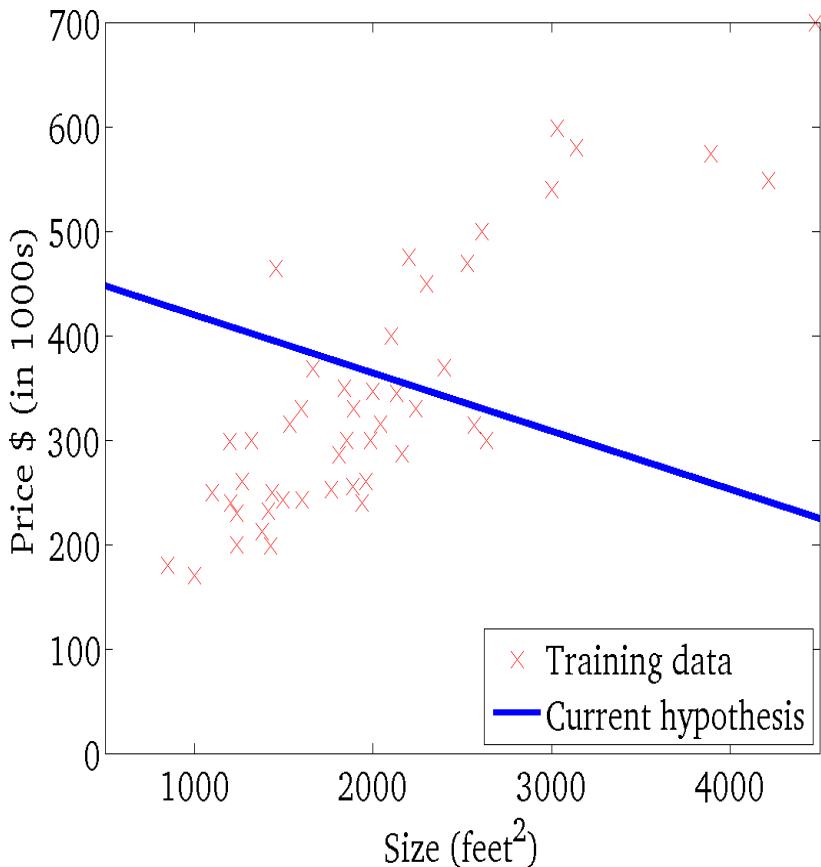


# Regresión lineal univariable: Descenso del gradiente

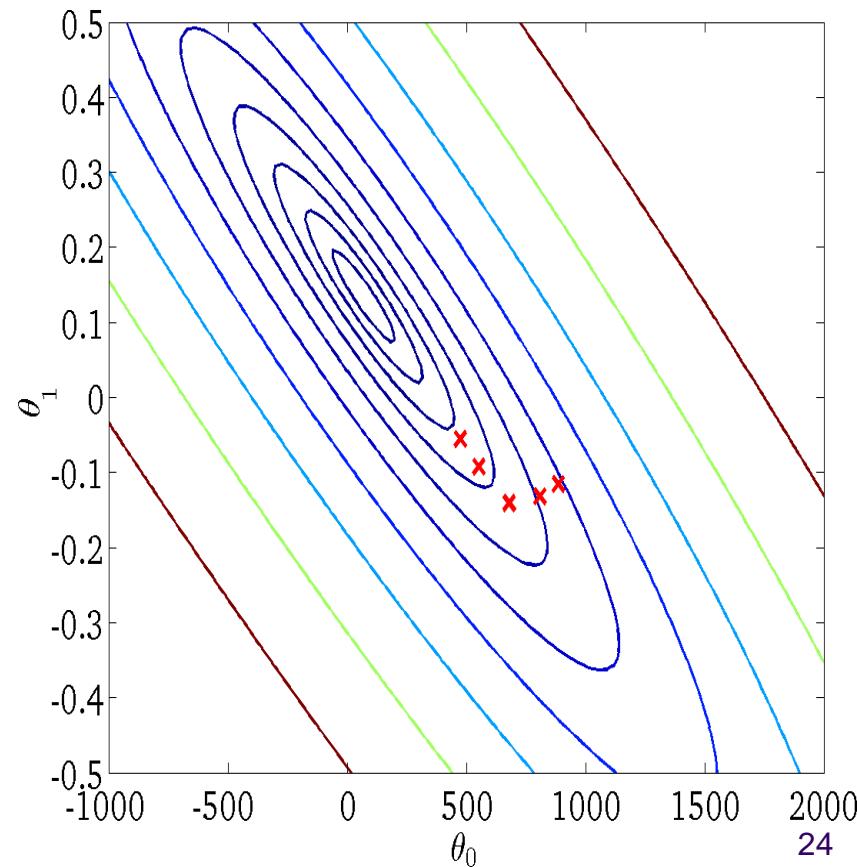


## Método iterativo

$h_0(x)$



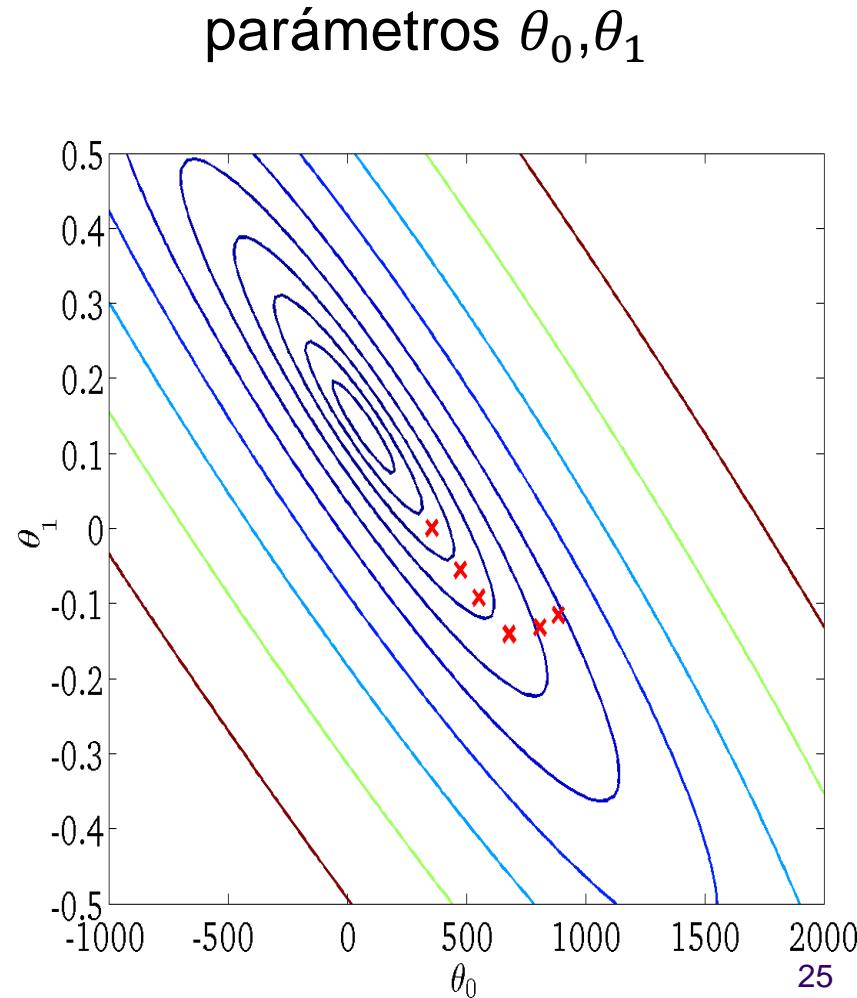
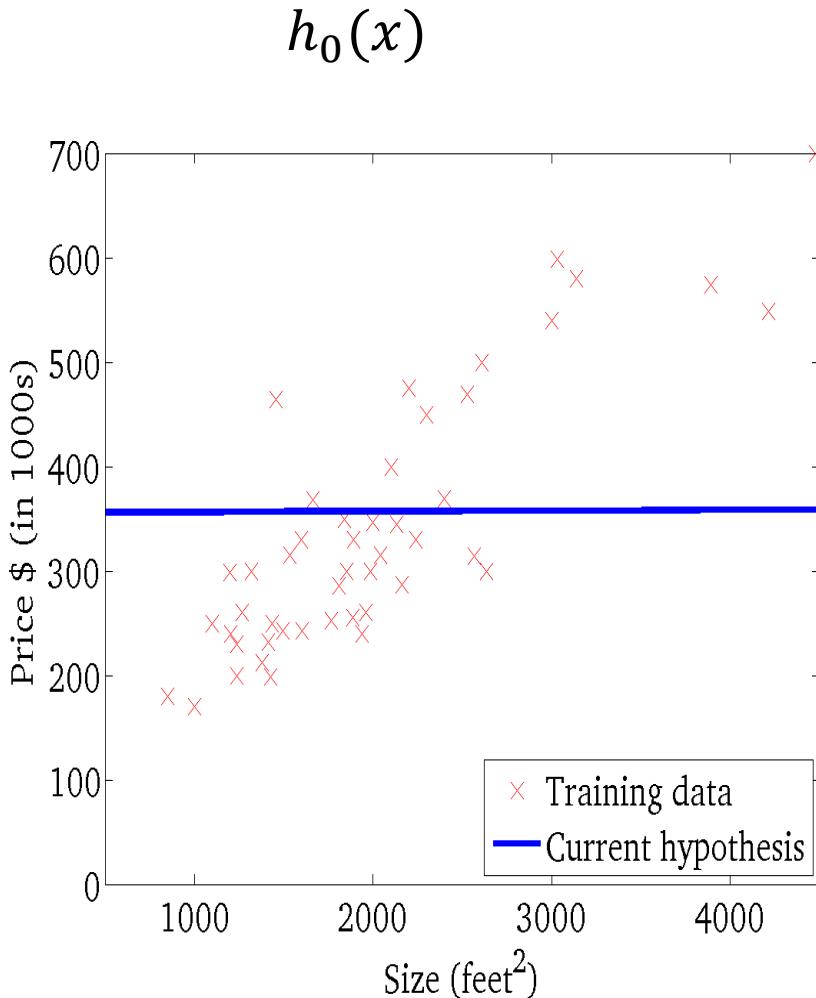
parámetros  $\theta_0, \theta_1$



# Regresión lineal univariable: Descenso del gradiente



## Método iterativo

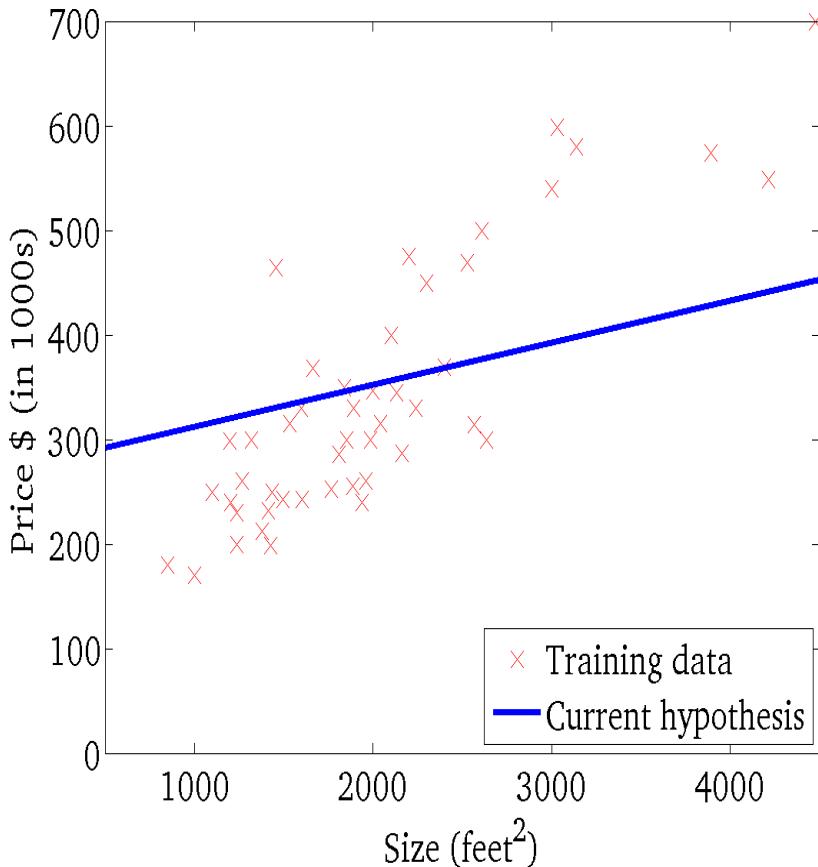


# Regresión lineal univariable: Descenso del gradiente

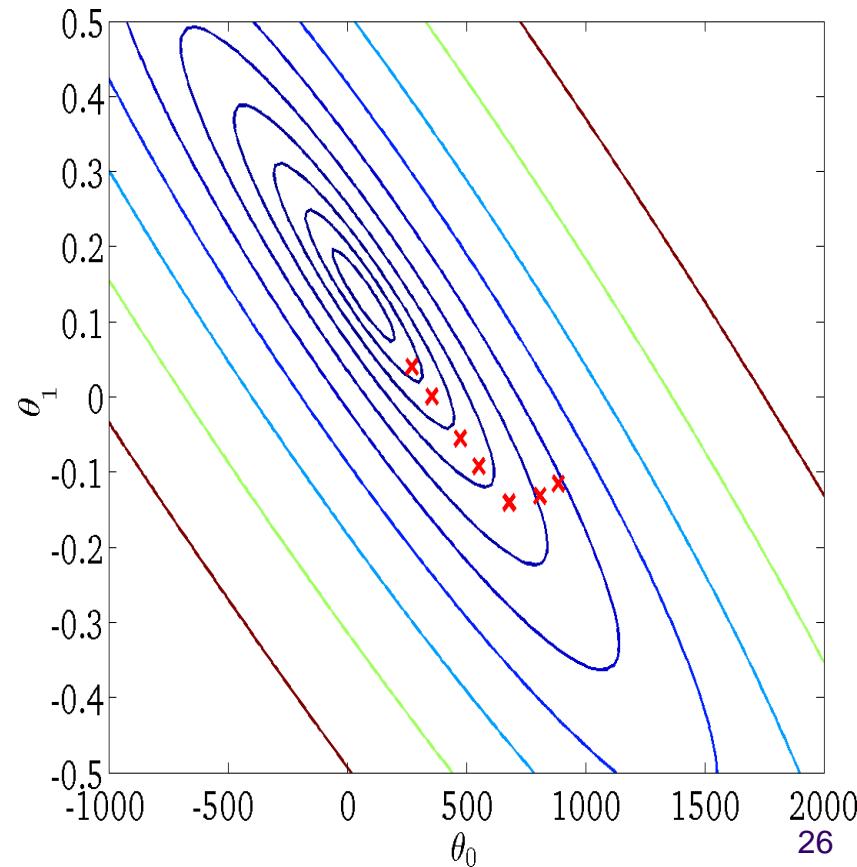


## Método iterativo

$h_0(x)$



parámetros  $\theta_0, \theta_1$

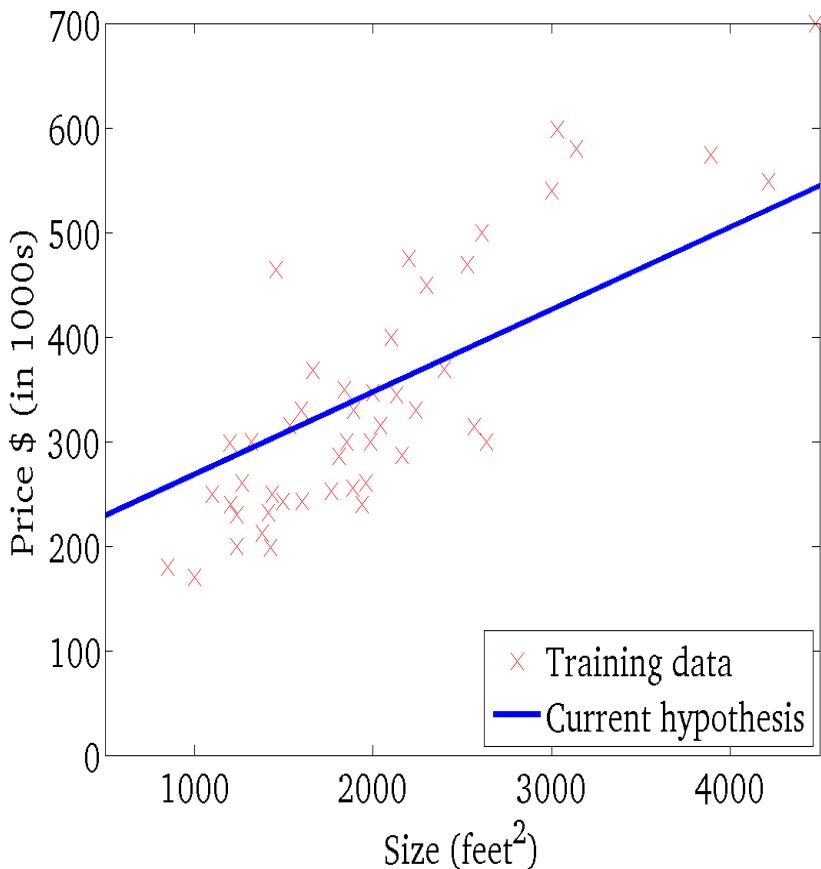


# Regresión lineal univariable: Descenso del gradiente

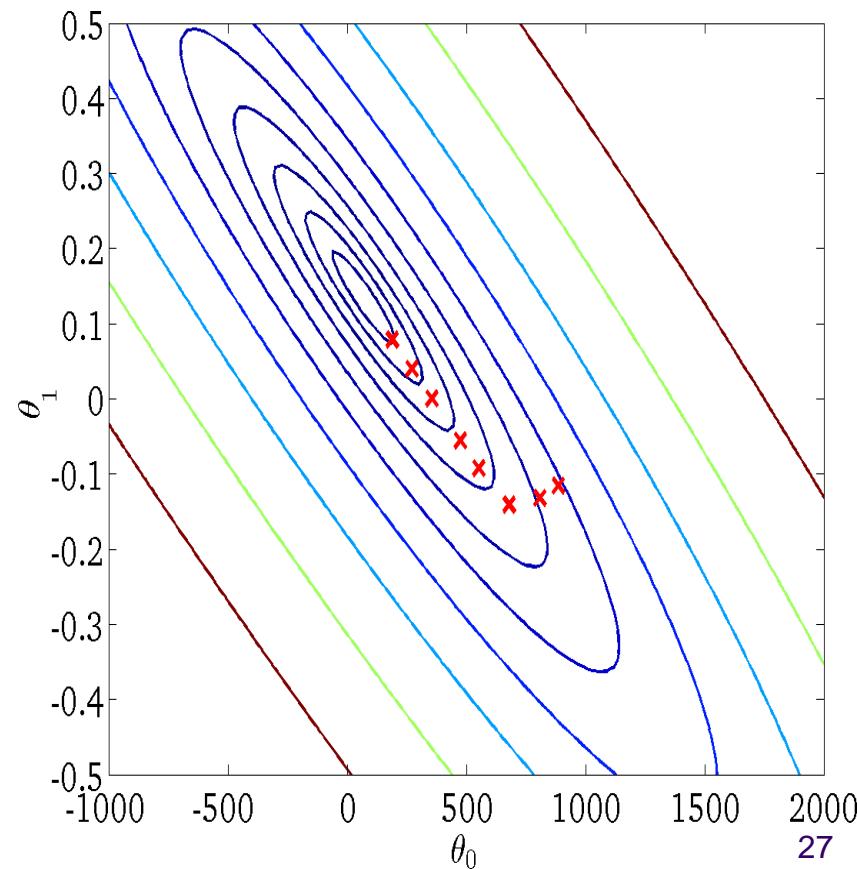


## Método iterativo

$h_0(x)$



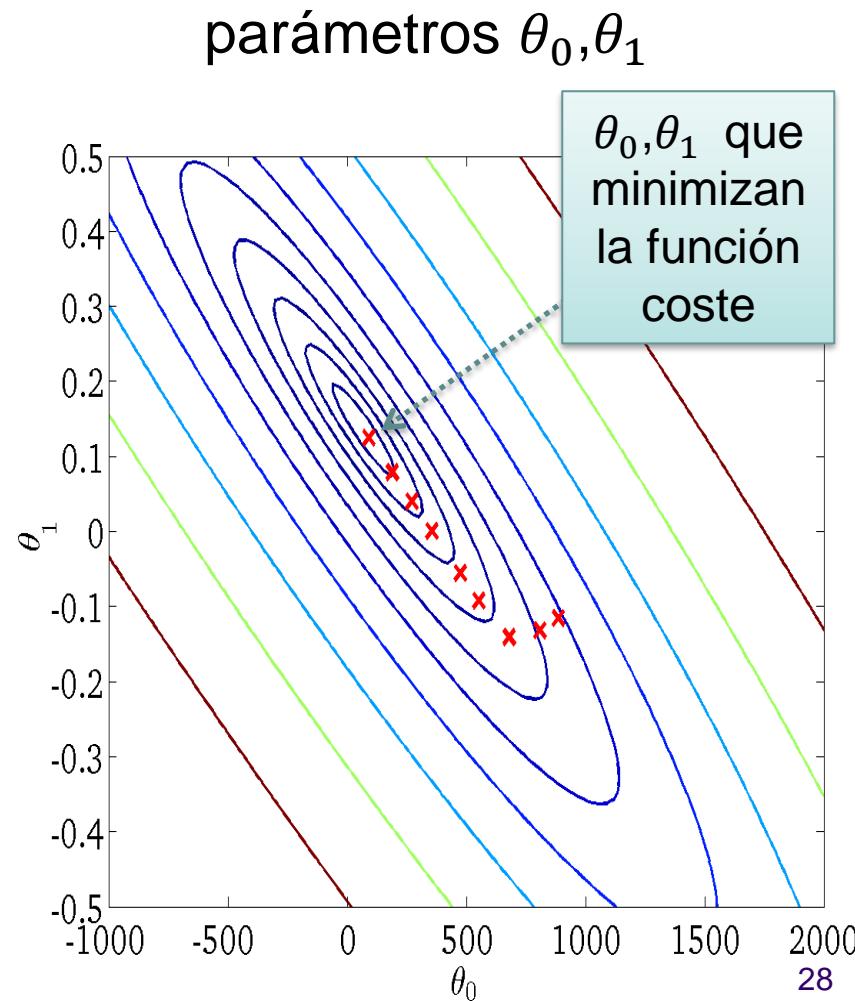
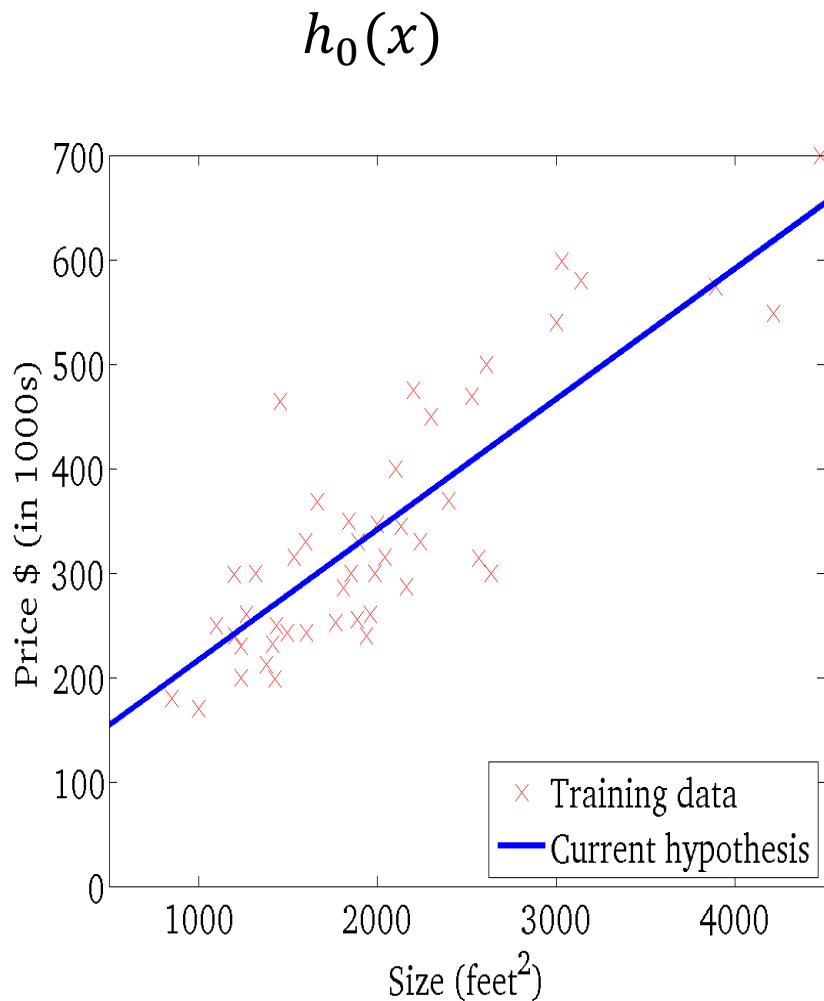
parámetros  $\theta_0, \theta_1$



# Regresión lineal univariable: Descenso del gradiente



## Método iterativo

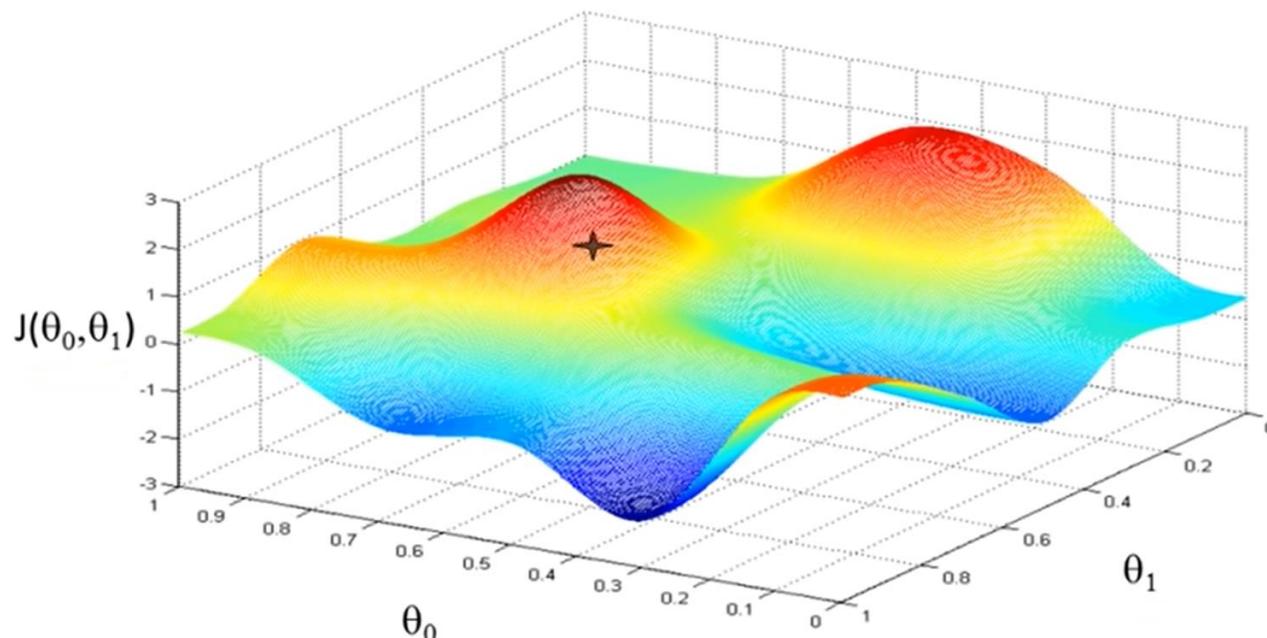


# Regresión lineal univariable: Descenso del gradiente



## Perspectiva geométrica:

- “Contour plot”: Para una función coste  $J(\theta_0, \theta_1)$  distinta de la función coste error cuadrático. Empezamos inicializando  $\theta_0$  y  $\theta_1$  con valores “cualesquiera”.

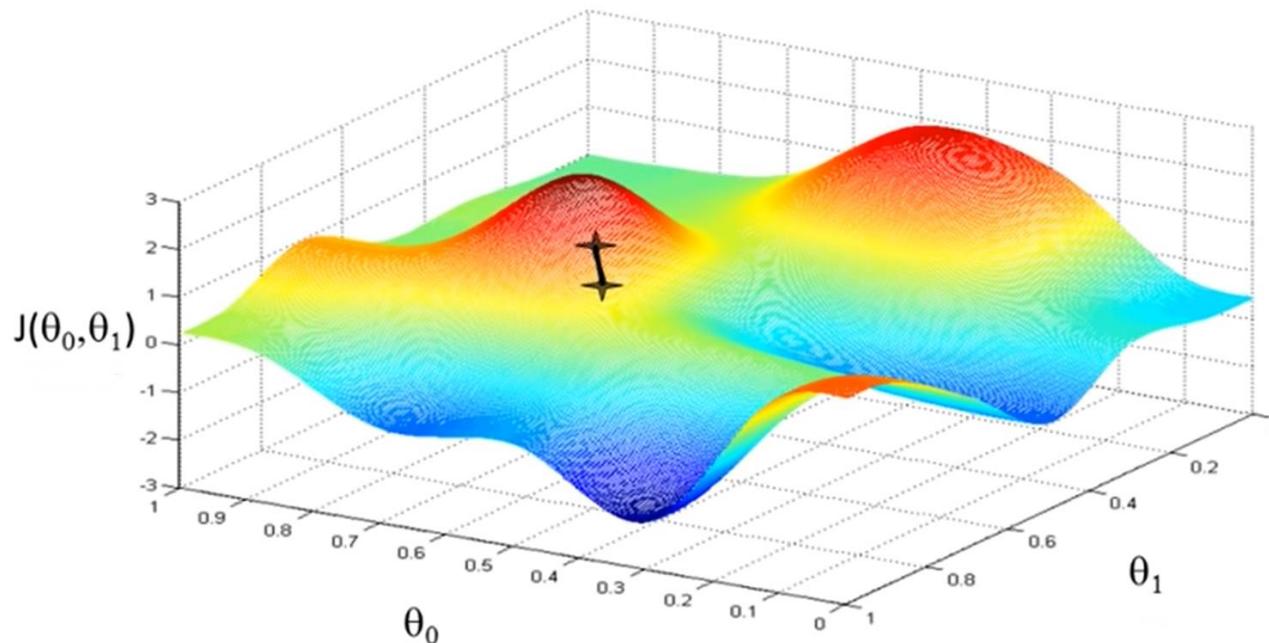


# Regresión lineal univariable: Descenso del gradiente



## Perspectiva geométrica:

- A continuación, se busca una dirección donde poder disminuir el valor de la función coste lo máximo posible modificando lo mínimo posible el valor de los parámetros  $\theta_0$  y  $\theta_1$ .

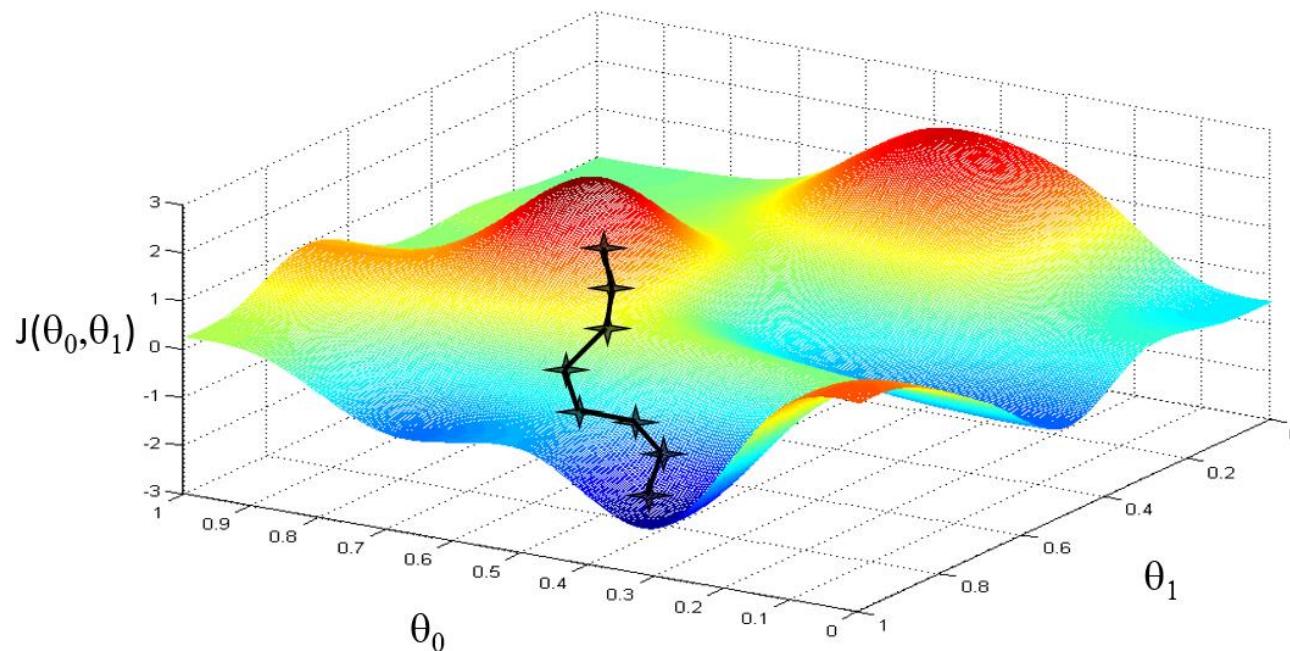


# Regresión lineal univariable: Descenso del gradiente



## Perspectiva geométrica:

- Repetimos hasta encontrar un mínimo óptimo local.

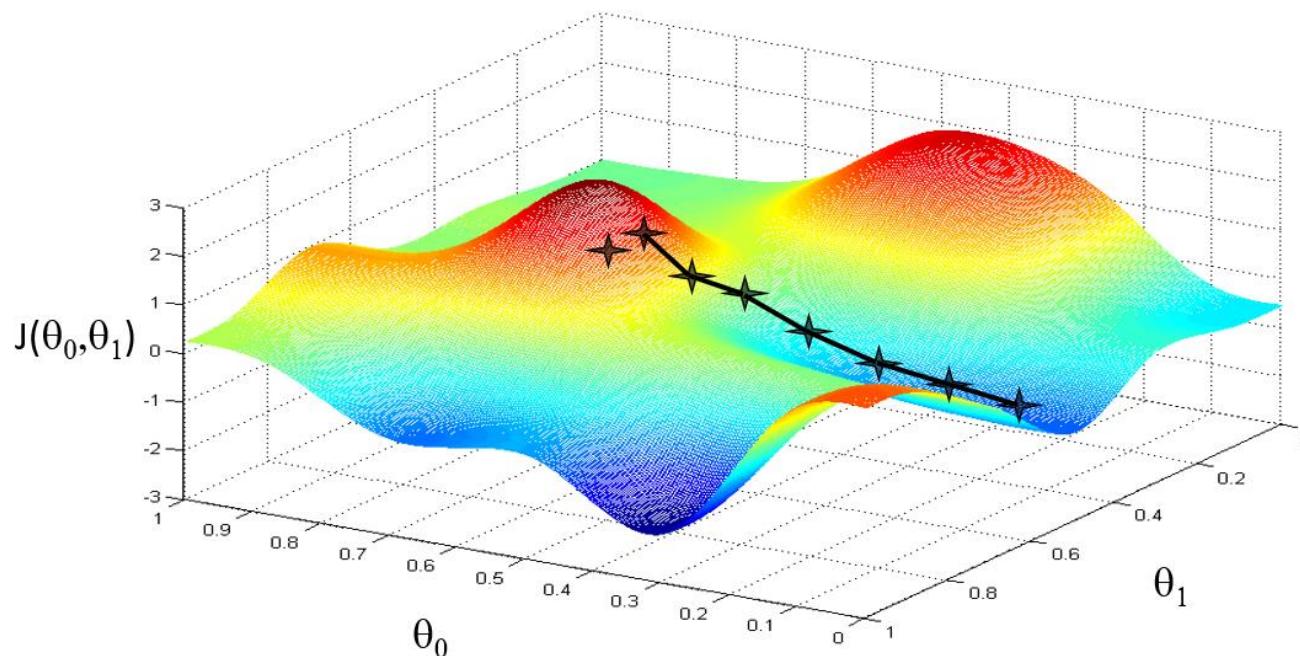


# Regresión lineal univariable: Descenso del gradiente



## Perspectiva geométrica:

- Si se inicializan  $\theta_0$  y  $\theta_1$  con otros valores (cercaos a los anteriores), se puede llegar a otro mínimo óptimo local.



# Regresión lineal univariable: Descenso del gradiente



## Perspectiva matemática:

- Repetir hasta convergencia, para cada uno de los parámetros de la función coste:  $\theta_0$  y  $\theta_1$  .
- Actualización de los parámetros de forma simultánea.
- Learning rate o tasa de aprendizaje ( $\alpha$ ): Controla cómo de grande será el cambio de los valores de los parámetros (“el salto”).

# Regresión lineal univariable: Descenso del gradiente



## Perspectiva matemática:

Repetir hasta convergencia (para  $j=0$  y  $j=1$ ) haciendo la asignación simultáneamente {

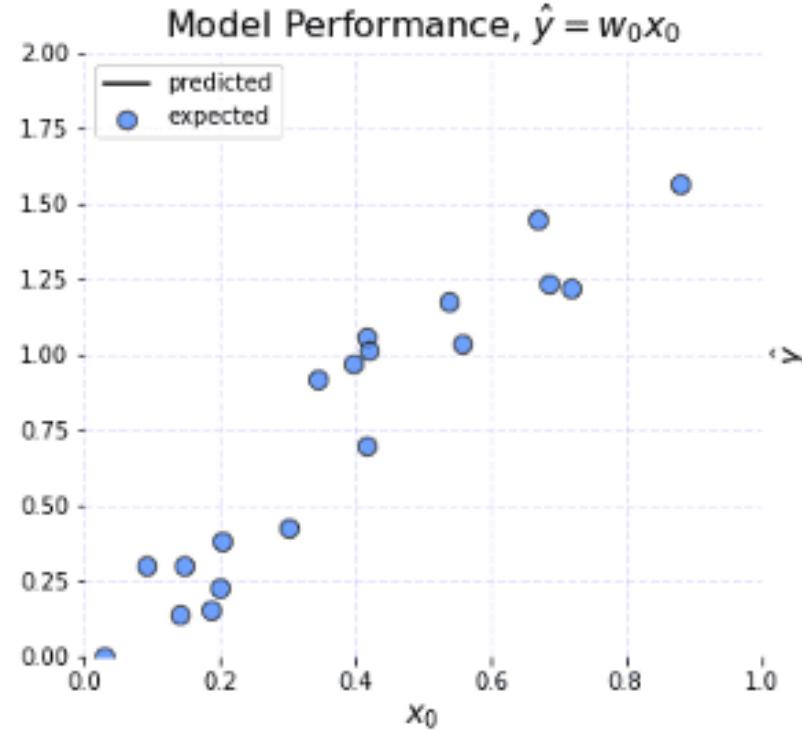
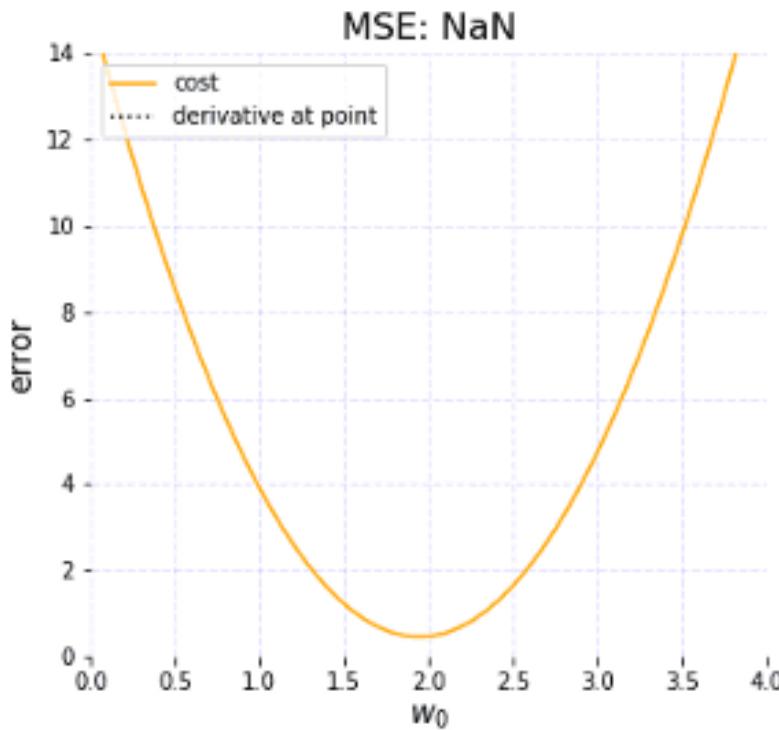
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

}

# Regresión lineal univariable: Descenso del gradiente



- Ejemplo: Fijar  $\theta_0$  a 0.

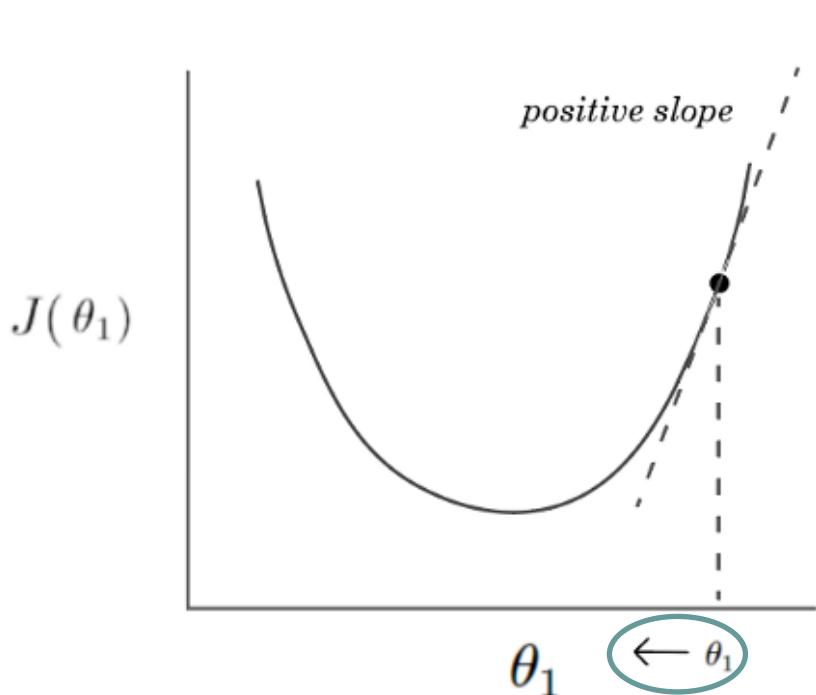


Fuente: <https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-gradient-descent-fcd5e0fc077d>

# Regresión lineal univariable: Descenso del gradiente



- Ejemplo: Fijar  $\theta_0$  a 0.
  - Pendiente positiva:



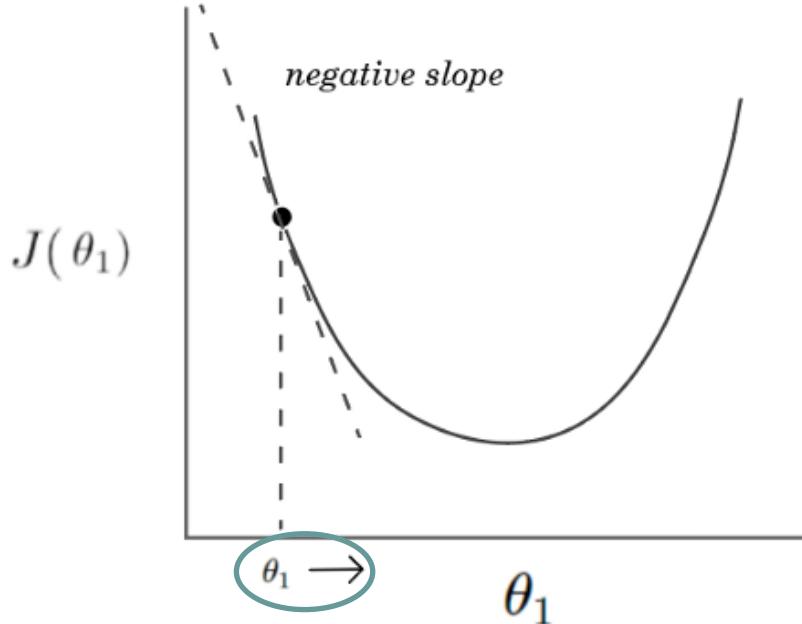
$$\theta_1 := \theta_1 - \alpha \underbrace{\frac{\partial}{\partial \theta_1} J(\theta_1)}_{\text{número positivo}}$$

El valor de  $\theta_1$  decrece cuando la derivada es positiva, es decir, se acerca al mínimo objetivo.

# Regresión lineal univariable: Descenso del gradiente



- Ejemplo: Fijar  $\theta_0$  a 0.
  - Pendiente negativa:



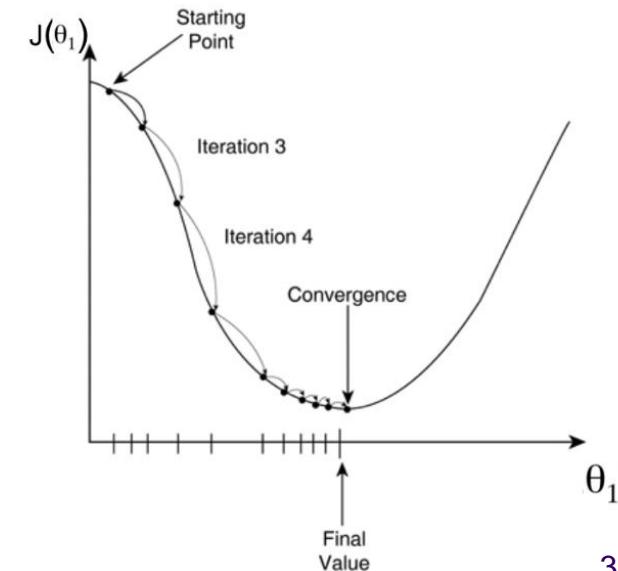
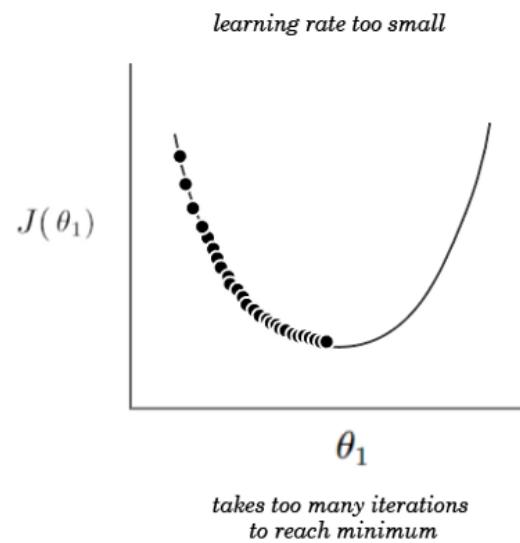
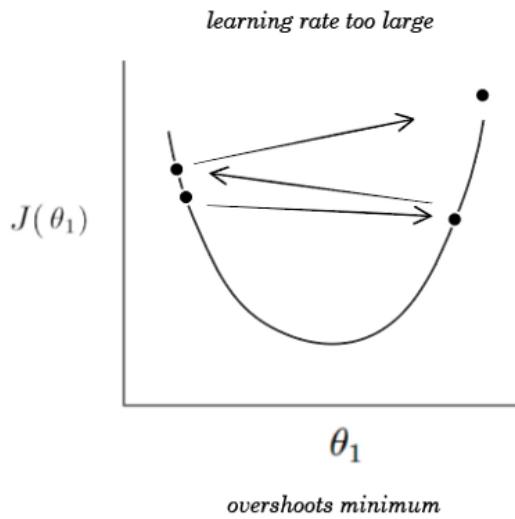
$$\theta_1 := \theta_1 - \alpha \underbrace{\frac{\partial}{\partial \theta_1} J(\theta_1)}_{\text{número negativo}}$$

El valor de  $\theta_1$  crece cuando la derivada es negativa, es decir, se acerca al mínimo objetivo.

# Regresión lineal univariable: Descenso del gradiente



- En cuanto al learning rate:
  - Un valor demasiado pequeño: puede provocar que el algoritmo del descenso de gradiente sea muy lento.
  - Un valor demasiado grande: puede provocar que el algoritmo del descenso de gradiente consiga converger o no.
  - Learning rate es un valor fijo y se calcula de forma experimental.

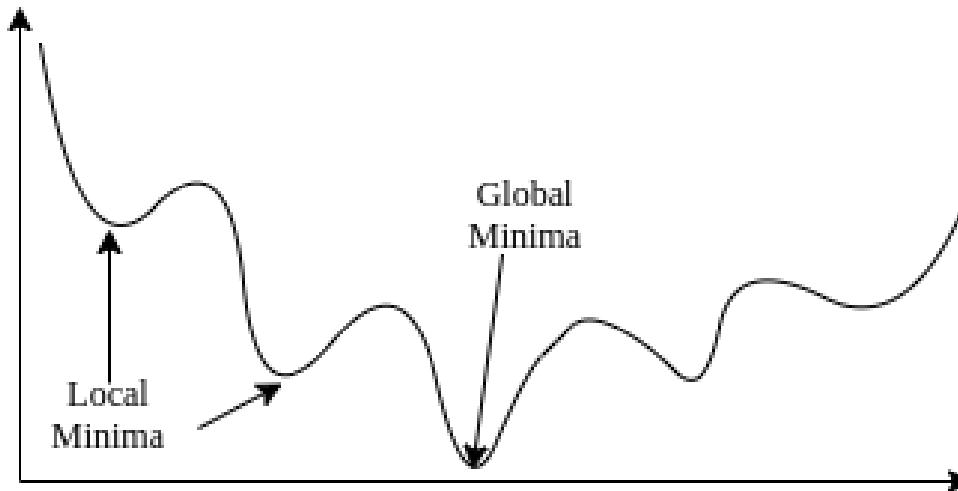


# Regresión lineal univariable: Descenso del gradiente



- Detección de un mínimo local:
  - Pendiente es 0 → Derivada será 0 → los parámetros no se actualizan.
  - ¡Puede que no sea el mínimo global!.

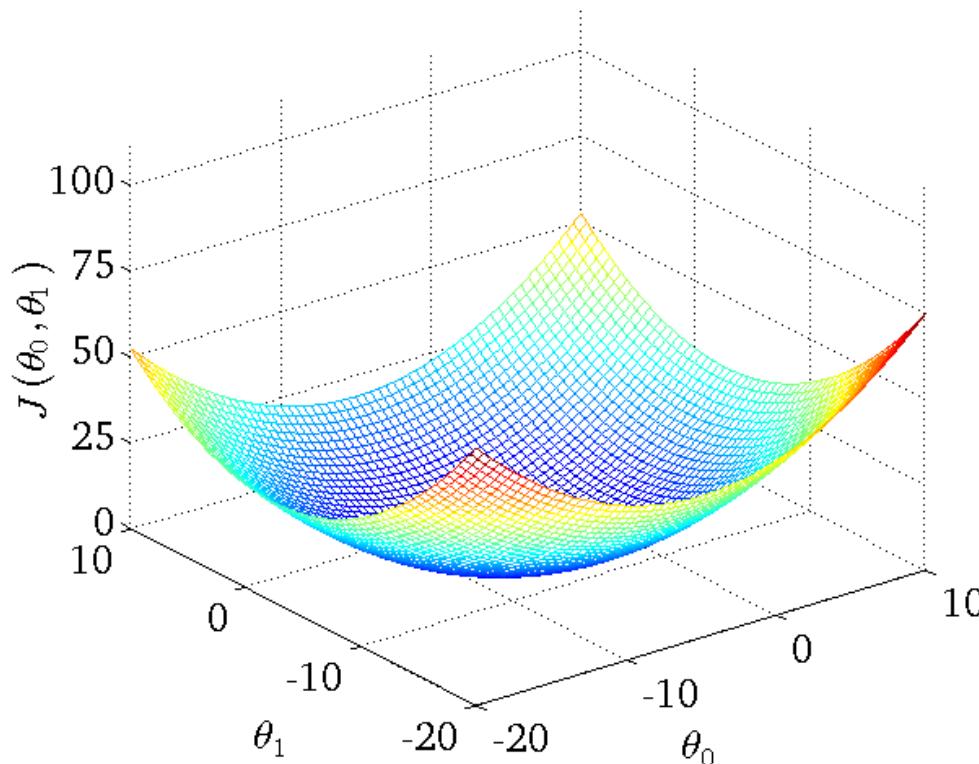
## Función coste no convexa



# Regresión lineal univariable: Descenso del gradiente



- Si la función coste es el error cuadrático:
  - No existe el problema del mínimo local/global ya que es siempre una función **convexa**.



# Regresión lineal univariable: Descenso del gradiente



- Si la función coste es el error cuadrático:

Repetir hasta convergencia con actualización simultánea {

$$\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} \left( \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 x^i - y^i)^2 \right)$$

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} \left( \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 x^i - y^i)^2 \right)$$

}

# Regresión lineal univariable: Descenso del gradiente



- Si la función coste es el error cuadrático:

Repetir hasta convergencia con actualización simultánea {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (\theta_0 + \theta_1 x^i - y^i)$$

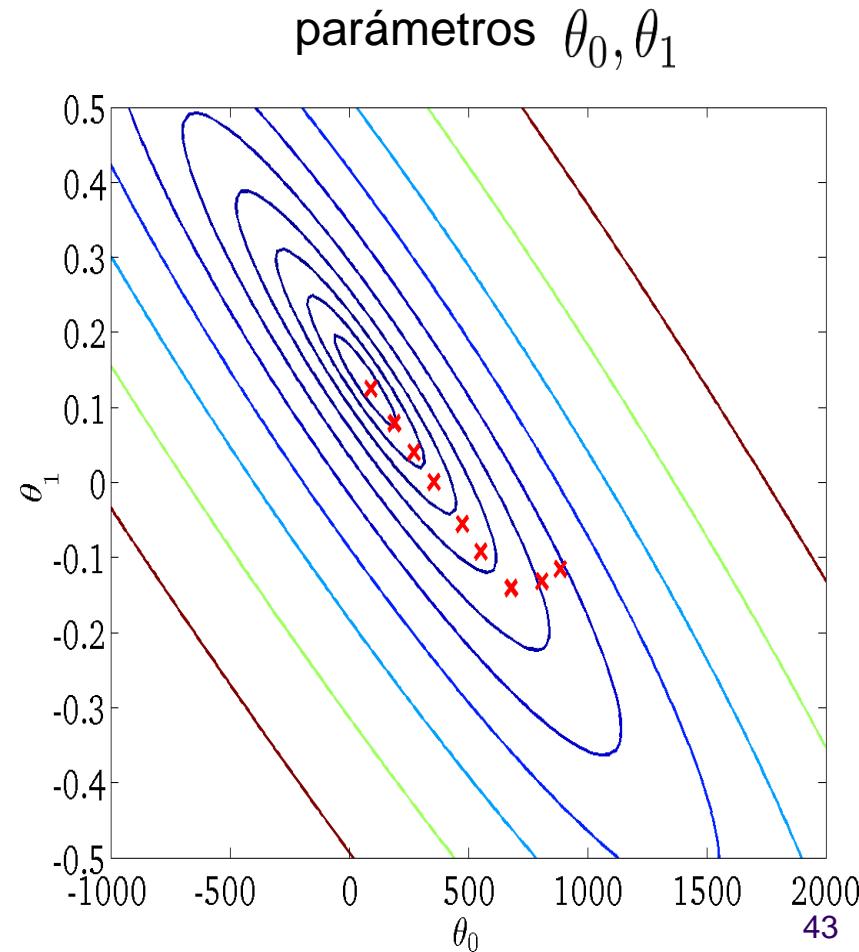
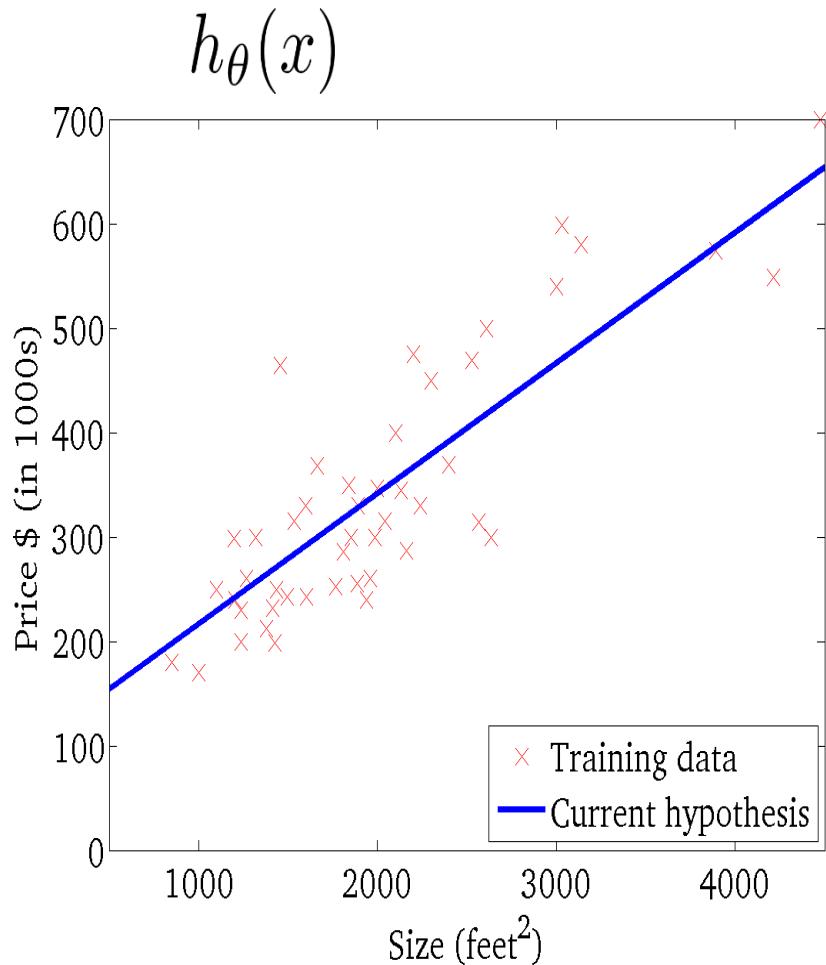
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (\theta_0 + \theta_1 x^i - y^i) x^i$$

}

# Regresión lineal univariable: Descenso del gradiente



- Si la función coste es el error cuadrático:





# Contenido

---

- Regresión lineal univariable
  - Representación del modelo
  - Función coste
  - Descenso del gradiente
  
- Regresión lineal multivariable
  - Representación del modelo
  - Función coste y descenso del gradiente
  - Feature Scaling / Escalado de variables
  - Ecuación normal

# Regresión lineal multivariable:



- Aprendizaje supervisado:
  - Dada la “respuesta correcta” para cada ejemplo del entrenamiento.
  - Tipo: Problema de regresión: Predice la salida que es un valor real.
  - El número de características/variables en el conjunto de entrenamiento es mayor que 1.
  - Los conceptos generales son los mismos que los de la regresión lineal univariable.

# Regresión lineal multivariable:

## Representación del modelo



- Notación

$m$  = número de ejemplos / instancias del conjunto de entrenamiento

$n$  = número de atributos / variables / características

$x^{(i)}$  = instancia “ $i$ ” del conjunto de entrenamiento ( $n$  valores)

$x_j^{(i)}$  = valor de la variable “ $j$ ” de la instancia “ $i$ ” del conjunto de entrenamiento (1 valor)

# Regresión lineal multivariable:

## Representación del modelo



- Notación

Problema: Predecir los precios de las considerando sus tamaños, el número de habitaciones, el número de plantas y los años de antigüedad de la casa. El conjunto de entrenamiento contiene 200 instancias.

Tamaño ( $x_1$ )	Número de habitaciones ( $x_2$ )	Número de plantas ( $x_3$ )	Años de antigüedad ( $x_4$ )	Precio ( $y$ )
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

# Regresión lineal multivariable:

## Representación del modelo



- Notación

Problema:

Tamaño ( $x_1$ )	Número de habitaciones ( $x_2$ )	Número de plantas ( $x_3$ )	Años de antigüedad ( $x_4$ )	Precio ( $y$ )
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$$m=200,$$

$$n=4, \quad x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}, \quad x_3^2 = 2$$

# Regresión lineal multivariable: Representación del modelo



Hipótesis del modelo de regresión lineal univariable:

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

Hipótesis del modelo de regresión lineal multivariable,  
siendo  $n$  el número de variables/atributos del dataset:

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

**Ejemplo:** Si  $n = 4$

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

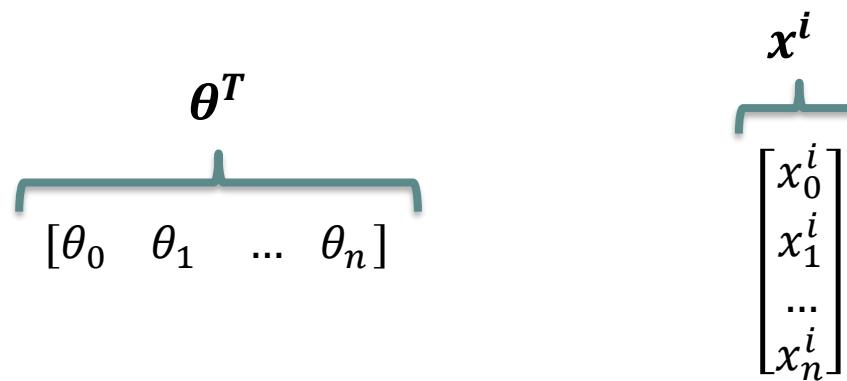
*Normalmente, por conveniencia:  $x_0 = 1$ !*

# Regresión lineal multivariable: Representación del modelo



Otra manera de expresar la hipótesis del modelo de regresión lineal multivariable, siendo  $n$  el número de variables/atributos del dataset:

$$h_{\theta}(x^i) = \theta_0 x_0^i + \theta_1 x_1^i + \theta_2 x_2^i + \cdots + \theta_n x_n^i = \theta^T x^i$$



Considerando  $\theta$  un vector columna de  $n+1$  elementos.

# Regresión lineal multivariable: Representación del modelo



Otra manera de expresar la hipótesis del modelo de regresión lineal multivariable, siendo  $n$  el número de variables/atributos del dataset:

$$h_{\theta}(x^i) = \theta_0 x_0^i + \theta_1 x_1^i + \theta_2 x_2^i + \cdots + \theta_n x_n^i = \theta^T x^i$$

## Vectorización del modelo:

$$h_{\theta}(x) = X\theta = \begin{bmatrix} x_0^1 & x_1^1 & \dots & x_n^1 \\ x_0^2 & x_1^2 & \dots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_0^m & x_1^m & \dots & x_n^m \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

# Regresión lineal multivariable: Función coste



Si la función coste es el error cuadrático:

- Hipótesis:

$$h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

- Parámetros:

$$\theta = \theta_0, \theta_1, \dots, \theta_n$$

- Función coste:

$$J(\theta) = J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

# Regresión lineal multivariable: Función coste



Si la función coste es el error cuadrático:

- Función coste:

$$J(\theta) = J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{2m} \begin{bmatrix} \theta^T x^{(1)} - y^{(1)} & \dots & \theta^T x^{(m)} - y^{(m)} \end{bmatrix} \cdot \begin{bmatrix} \theta^T x^{(1)} - y^{(1)} \\ \vdots \\ \theta^T x^{(m)} - y^{(m)} \end{bmatrix}$$

$$= \frac{1}{2m} (X \cdot \theta - y)^T (X \cdot \theta - y)$$

# Regresión lineal multivariable: Descenso del gradiente



Si la función coste es el error cuadrático:

- Descenso del gradiente:

Repetir con actualización simultánea para cada  $j=0, \dots, n$  {

$$\begin{aligned}\theta_j &:= \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n) \\ &= \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}\end{aligned}$$

}

$$\theta = \theta - \alpha \frac{1}{m} (X^T \cdot (X \cdot \theta - y))$$

# Regresión lineal multivariable: Descenso del gradiente



Opciones para comprobar si está funcionando correctamente:

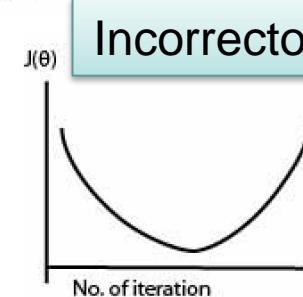
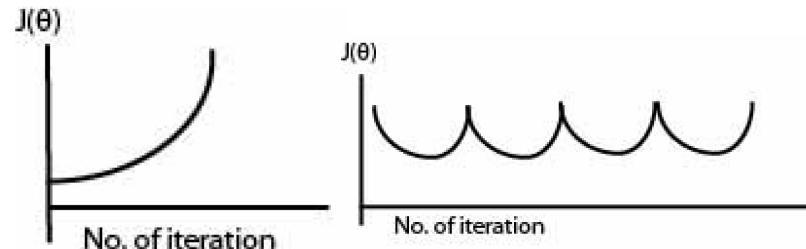
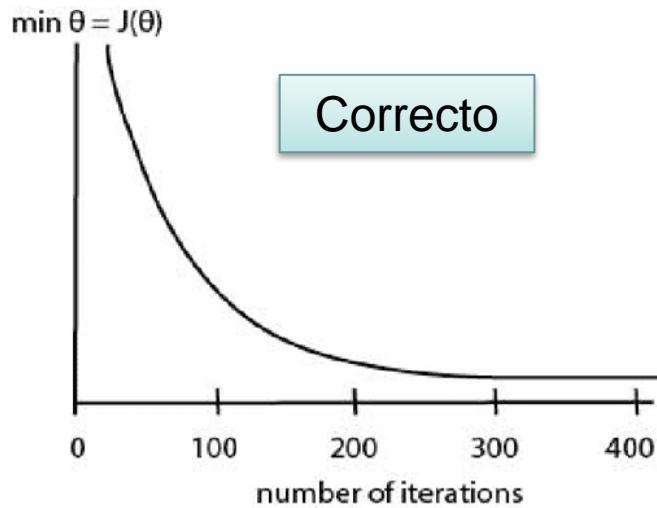
- Test de convergencia automático:
  - Declarar que converge si  $J(\theta)$  decrece por menos un valor pequeño épsilon  $\varepsilon$  en una iteración.
  - Es difícil definir cuál es ese épsilon  $\varepsilon$  ( $10^{-3}, 10^{-4}, \dots$ ).
- Gráfica número de iteraciones vs  $J(\theta)$ 
  - Muy útil visualmente (Siguiente diapositiva).
  - $J(\theta)$  debe descender y estabilizarse a lo largo de las iteraciones.

# Regresión lineal multivariable: Descenso del gradiente



Opciones para comprobar si está funcionando correctamente:

- Gráfica número de iteraciones vs  $J(\theta)$

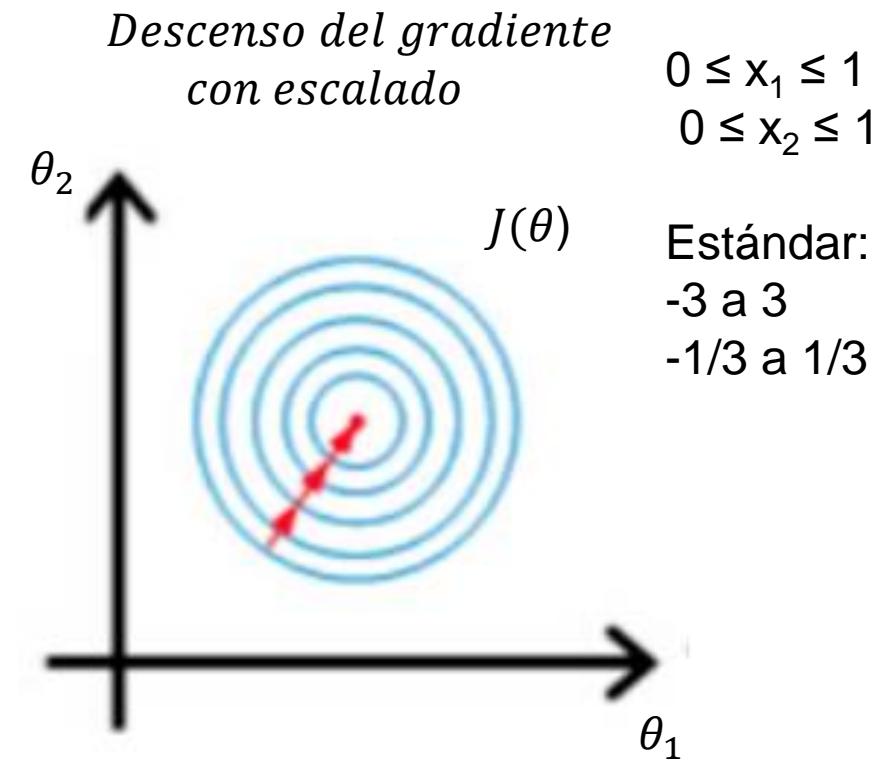
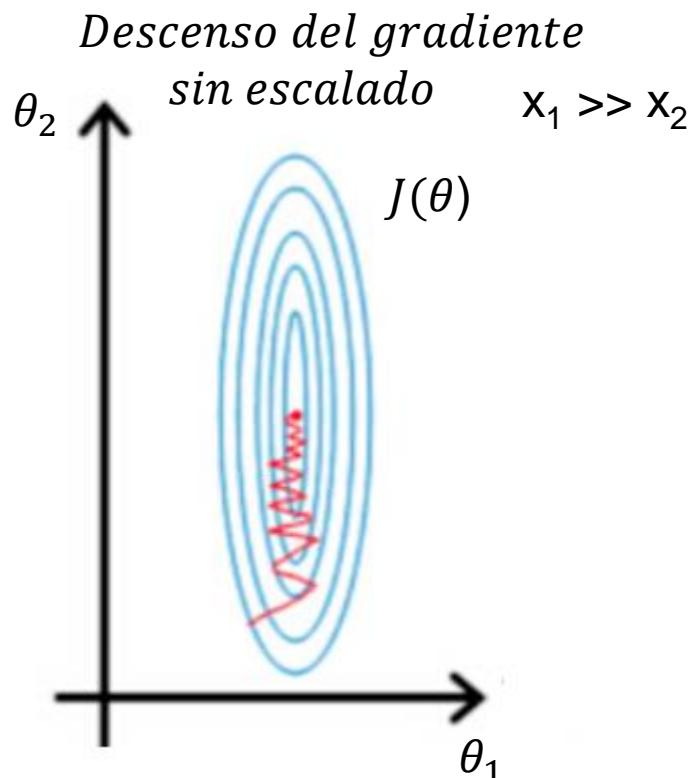


Si no converge podemos intentar comprobar si es debido a que el learning rate no es muy acertado. Probar por ejemplo:  $\alpha = \dots 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1 \dots (x3, x10)$ .

# Regresión lineal multivariable: Feature Scaling / Escalado de variables



Cuando trabajamos con más de una variable es importante que estén en la misma escala o rango para que el descenso del gradiente converja rápidamente.



# Regresión lineal multivariable: Feature Scaling / Escalado de variables



Opciones:

- Normalización y estandarización:

siendo  $x_{normalizada}^i_j$  el valor normalizado de la variable  $j$  de la instancia  $i$ , siendo  $x_{estandarizada}^i_j$  el valor estandarizado de la variable  $j$  de la instancia  $i$ ,  $x_j^i$  el valor sin normalizar de la misma,  $\mu_j$  la media de la variable  $j$  de todas las instancias,  $max_j$  y  $min_j$  el máximo y el mínimo de la misma y  $\sigma_j$  la desviación estándar de la variable  $j$ .

$$x_{normalizada}^i_j = \frac{x_j^i - \mu_j}{max_j - min_j}$$

$$x_{estandarizada}^i_j = \frac{x_j^i - \mu_j}{\sigma_j}$$

# Regresión lineal multivariable:

## Ecuación normal



- Método para resolver  $\theta$  **analíticamente**.
- El feature scaling no es necesario.
- Intuición: para minimizar una función, hacemos la derivada y la igualamos a 0.

$$J(\theta) = J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \quad \theta \in \mathbb{R}^{n+1}$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \dots = 0$$

$$\theta = (X^T X)^{-1} X^T y$$

¡Cuidado! No todas las matrices pueden ser invertidas.

# Regresión lineal multivariable: Ecuación normal



- Notación

**m** instancias:  $(x^1, y^1), \dots, (x^m, y^m)$ ; **n** variables

$x^i$  identifica la instancia “ $i$ ”  
del entrenamiento:

$$x^i = \begin{bmatrix} x_0^i \\ x_1^i \\ \vdots \\ x_n^i \end{bmatrix} \in \mathbb{R}^{n+1}$$

**X** es la “design matrix”:

$$X = \begin{bmatrix} \cdots & (x^1)^T & \cdots \\ \cdots & (x^2)^T & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & (x^m)^T & \cdots \end{bmatrix}$$

**y** es el vector con los  
datos de salida reales

$$y = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix}$$

# Regresión lineal multivariable: Ecuación normal



- Ejemplo 1:  $m = 5$

$x_0$	Size (feet <sup>2</sup> ) $x_1$	Number of bedrooms $x_2$	Number of floors $x_3$	Age of home (years) $x_4$	Price (\$1000) $y$
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
1	3000	4	1	38	540

$$\boldsymbol{\theta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \\ 1 & 3000 & 4 & 1 & 38 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \\ 540 \end{bmatrix}$$

# Regresión lineal multivariable:

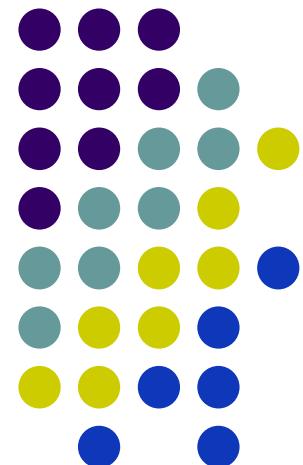
## Ecuación normal



- **Descenso del gradiente:**
  - Hay que elegir  $\alpha$
  - Necesita muchas iteraciones
  - Trabaja bien incluso si hay muchas variables ( $n > 10000$ )
- **Ecuación normal:**
  - No hay que elegir  $\alpha$
  - No hay que iterar
  - Hay que computar  $\theta = (X^T X)^{-1} X^T y$ 
    - Si  $n$  es muy grande, puede ser muy lento  
( $n=100, 1000, 10000$ )

# Regresión logística

Inteligencia Artificial  
Ingeniería Informática  
en Sistemas de Información



# Contenido



- Regresión logística binaria
  - Hipótesis
  - Función coste
  - Descenso del gradiente
  - Optimización avanzada
- Clasificación multiclasse
  - OneVsAll
- Sobreajuste
  - Regularización: regresión lineal y regresión logística

# Regresión logística binaria:



- Aprendizaje supervisado:
  - Dada la “respuesta correcta” para cada ejemplo del entrenamiento.
  - Tipo: Problema de clasificación: Predice la salida que toma siempre valores discretos.
  - ¡Cuidado!: Aunque se llame Regresión logística, no es un problema de regresión.

# Regresión logística binaria:



- Ejemplos:

- Clasificación binaria:

- Predecir si lloverá o no en una ciudad.
    - Predecir si un tumor será benigno o maligno.

$$y \in \{0, 1\}$$

$0$ : Clase negativa;     $1$ : Clase positiva

- Clasificación multi-clase:

- Predecir el tipo de flor según sus características: problema común “iris dataset”.

$$y \in \{0, 1, 2\}$$

$0$ : *iris – setosa*;     $1$ : *iris – versicolor*;     $2$ : *iris – virginica*

# Regresión logística binaria:



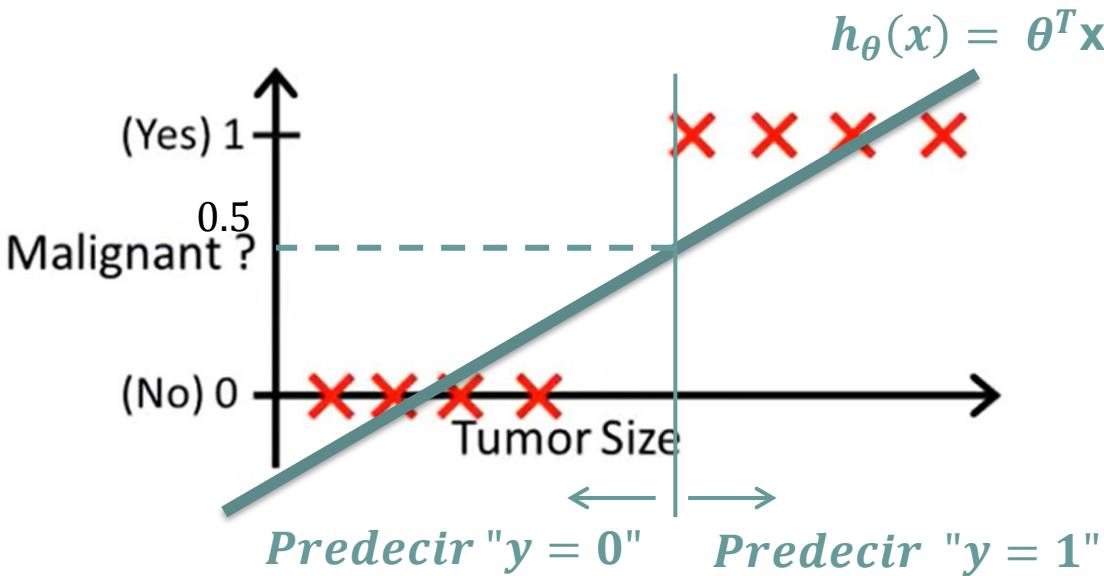
- No debemos usar la regresión lineal ya que está dirigida a problemas de regresión:
  - Hipótesis regresión logística binaria:  $0 \leq h_{\theta}(x) \leq 1$
  - Hipótesis regresión lineal:  $h_{\theta}(x)$  *puede ser*  $> 1$  o  $< 0$

# Regresión logística binaria:



- No debemos usar la regresión lineal ya que está dirigida a problemas de regresión:
  - Ejemplo usando regresión lineal  $h_\theta(x)$  con umbral en 0.5 para predecir si un tumor es maligno o benigno:

*si  $h_\theta(x) \geq 0.5$ , entonces:  $y = 1$*   
*si  $h_\theta(x) < 0.5$ , entonces:  $y = 0$*

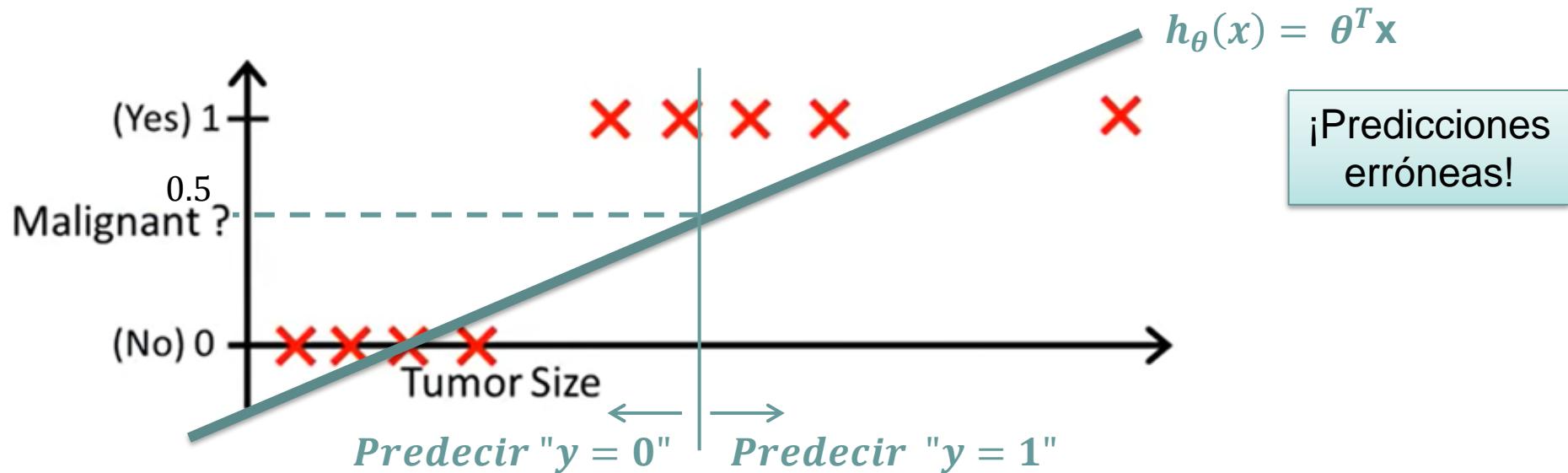


# Regresión logística binaria:



- No debemos usar la regresión lineal ya que está dirigida a problemas de regresión:
  - Ejemplo usando regresión lineal  $h_\theta(x)$  con umbral en 0.5 para predecir si un tumor es maligno o benigno:

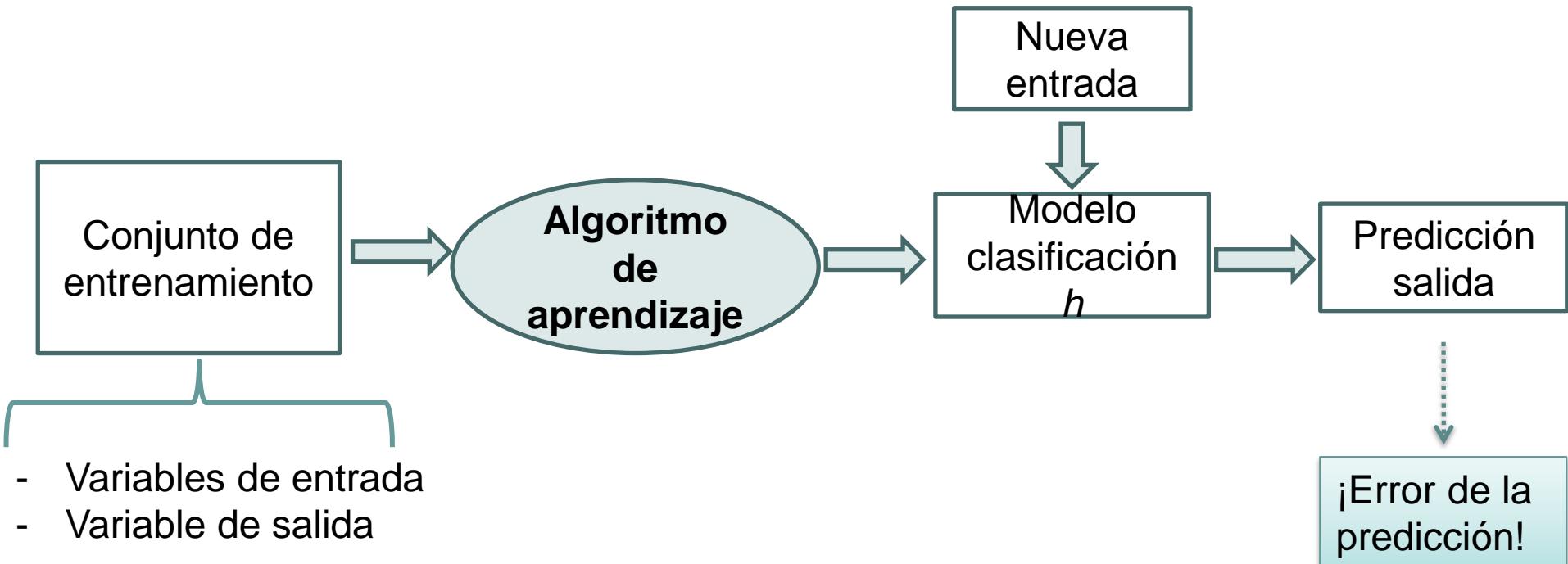
*si  $h_\theta(x) \geq 0.5$ , entonces:  $y = 1$*   
*si  $h_\theta(x) < 0.5$ , entonces:  $y = 0$*



# Regresión logística binaria:



## Representación del modelo:



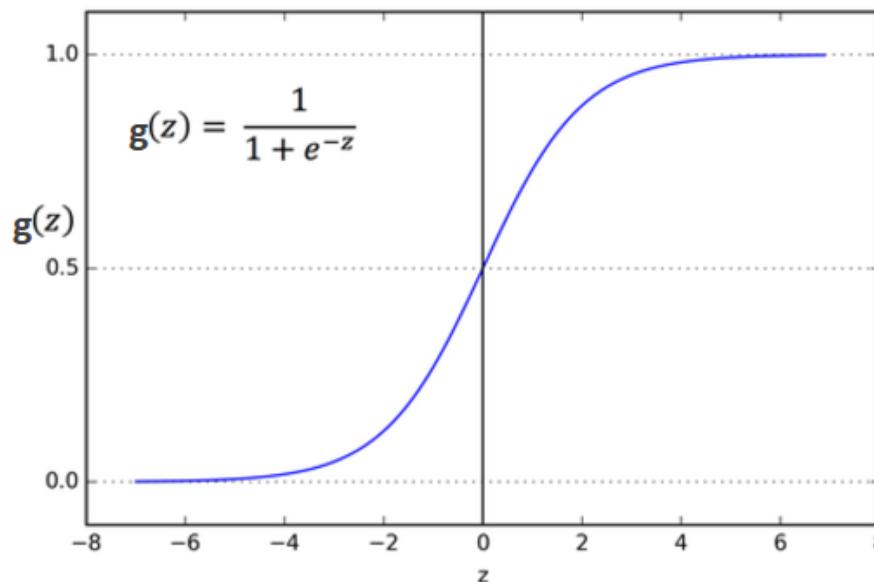
# Regresión logística binaria: Hipótesis



Hipótesis del modelo de regresión logística binaria:

- Usa la función sigmoide o logística:

$$g(z) = \frac{1}{1 + e^{-z}}$$



# Regresión logística binaria: Hipótesis



Hipótesis del modelo de regresión logística binaria:

- Usa la función sigmoide o logística:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Hipótesis regresión lineal:  $h_{\theta}(x) = \theta^T x$

$h_{\theta}(x)$  puede ser  $> 1$  o  $< 0$

Hipótesis regresión logística:  $h_{\theta}(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$

$$0 \leq h_{\theta}(x) \leq 1$$

# Regresión logística binaria: Hipótesis

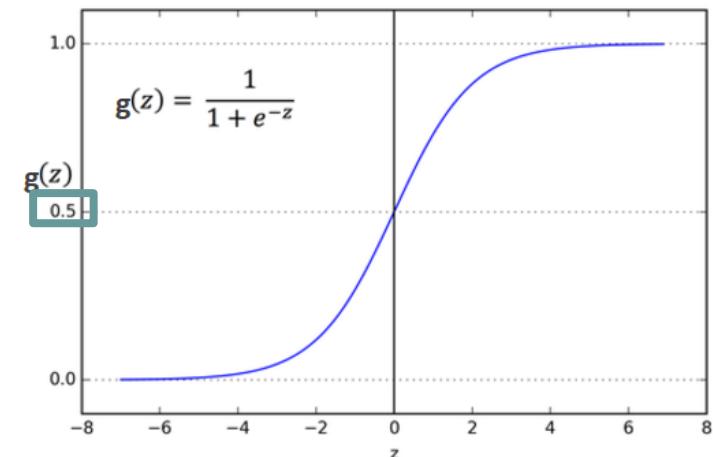


Hipótesis del modelo de regresión logística binaria:

- Suponemos que:

$$y = 1 \text{ si } h_{\theta}(x) \geq 0.5$$

$$y = 0 \text{ si } h_{\theta}(x) < 0.5$$



- $y = 1 \text{ si } h_{\theta}(x) \geq 0.5$

$g(z) \geq 0.5$  cuando  $z \geq 0$ ;

entonces:  $h_{\theta}(x) = g(\theta^T x) = g(z) \geq 0.5$  cuando  $\theta^T x = z \geq 0$

- $y = 0 \text{ si } h_{\theta}(x) < 0.5$

$g(z) < 0.5$  cuando  $z < 0$ ;

entonces:  $h_{\theta}(x) = g(\theta^T x) = g(z) < 0.5$  cuando  $\theta^T x = z < 0$

# Regresión logística binaria: Hipótesis



Hipótesis del modelo de regresión logística binaria:

- Se considera que  $h_{\theta}(x)$  es la probabilidad estimada de que  $y=1$  para una nueva entrada:

$$h_{\theta}(x) = g(\theta^T x) = P(y = 1 | x; \theta)$$

- Ejemplo: Si  $h_{\theta}(x) = 0.7$  , entonces la probabilidad de que  $y=1$  para esa nueva entrada será del 70%.
- Si es clasificación binaria: la probabilidad de que  $y=1$  + la probabilidad de que  $y=0$  es 1.

$$P(y = 0 | x; \theta) + P(y = 1 | x; \theta) = 1.$$

# Regresión logística binaria: Hipótesis



Hipótesis del modelo de regresión logística binaria:

- **Ejemplo 1: Frontera de decisión lineal.** Suponemos:

$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) = g(-3 + x_1 + x_2)$$

- Entonces:  $\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$
- Considerando la diapositiva anterior:
$$y = 1 \quad si \quad \theta^T x = z \geq 0 ; \quad y = 0 \quad si \quad \theta^T x = z < 0$$
- Por tanto:
  - $y = 1 \quad si \quad -3 + x_1 + x_2 \geq 0$
  - $y = 0 \quad si \quad -3 + x_1 + x_2 < 0$

# Regresión logística binaria: Hipótesis



Hipótesis del modelo de regresión logística binaria:

- Ejemplo 1: Frontera de decisión lineal. (Sigue):

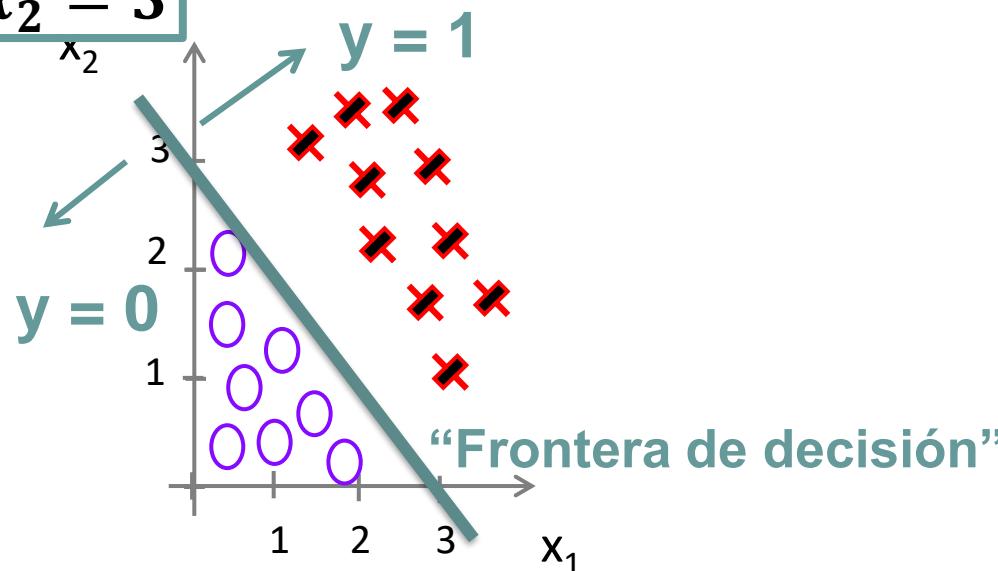
- Por tanto:

- $y = 1 \text{ si } -3 + x_1 + x_2 \geq 0 ; \quad x_1 + x_2 \geq 3$

- $y = 0 \text{ si } -3 + x_1 + x_2 < 0 ; \quad x_1 + x_2 < 3$

- La “Decision Boundary” o frontera de decisión será:

$$x_1 + x_2 = 3$$



# Regresión logística binaria: Hipótesis



Hipótesis del modelo de regresión logística binaria:

- **Ejemplo 2: Frontera de decisión no lineal.** Suponemos:

$$\begin{aligned} h_{\theta}(x) &= g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2) \\ &= g(-1 + x_1^2 + x_2^2) \end{aligned}$$

- Entonces:  $\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$
- Considerando la diapositiva anterior:

$$y = 1 \quad si \quad \theta^T x = z \geq 0 ; \quad y = 0 \quad si \quad \theta^T x = z < 0$$

- Por tanto:
  - $y = 1 \quad si \quad -1 + x_1^2 + x_2^2 \geq 0$
  - $y = 0 \quad si \quad -1 + x_1^2 + x_2^2 < 0$

# Regresión logística binaria: Hipótesis



Hipótesis del modelo de regresión logística binaria:

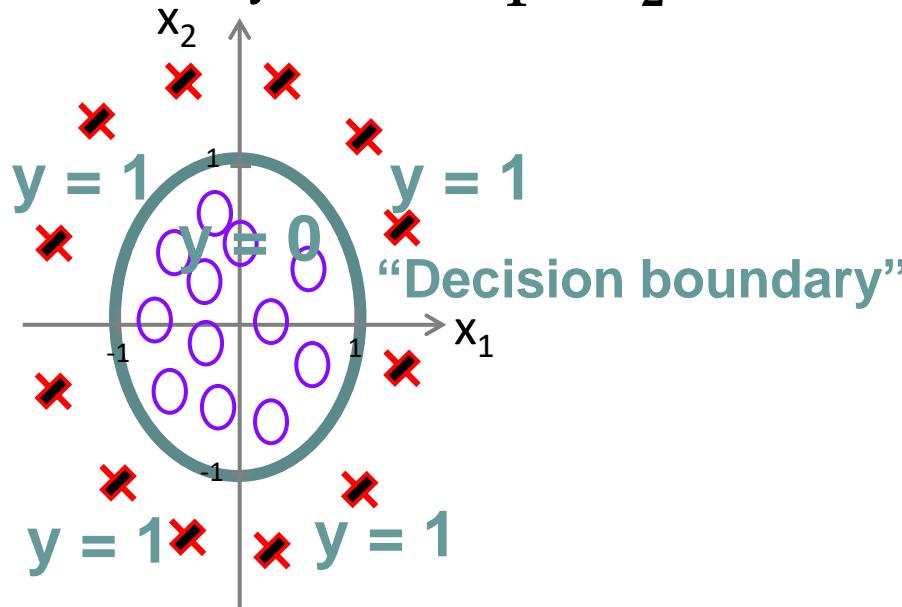
- Ejemplo 2: Frontera de decisión no lineal. (Sigue):

- Por tanto:

- $y = 1 \text{ si } -1 + x_1^2 + x_2^2 \geq 0 ; \quad x_1^2 + x_2^2 \geq 1$

- $y = 0 \text{ si } -1 + x_1^2 + x_2^2 < 0; \quad x_1^2 + x_2^2 < 1$

- La “Decision Boundary” será:  $x_1^2 + x_2^2 = 1$



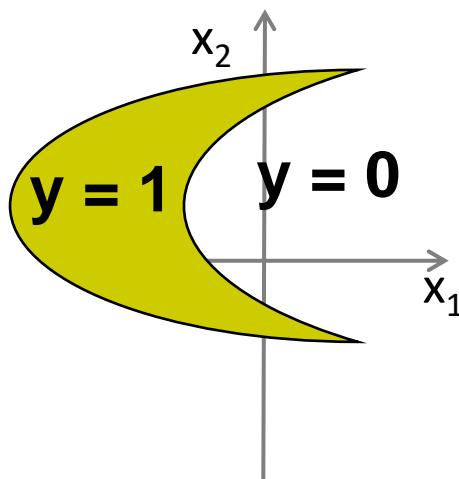
# Regresión logística binaria: Hipótesis



Hipótesis del modelo de regresión logística binaria:

- **Ejemplo 3: Frontera de decisión no lineal.** Suponemos:

$$h_{\theta}(x) = g \left( \begin{array}{l} \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 \\ + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots \end{array} \right)$$



# Regresión logística binaria:

## Función coste



- Notación clasificación binaria:

$m$  = número de ejemplos / instancias del conjunto de entrenamiento

$n$  = número de atributos / variables / características

**Training** :  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad \mathbb{R}^{n+1}, x_0 = 1 \quad y \in \{0,1\}$$

- Hipótesis:  $h_{\theta}(x) = \frac{1}{1+e^{-\theta^T x}}$  tenemos que elegir los  $\theta$ .

# Regresión logística binaria:

## Función coste



- Partimos de la función coste que habíamos seleccionado para la regresión lineal:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^i) - y^i)^2$$

$$= \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h_\theta(x^i) - y^i)^2 = \frac{1}{m} \sum_{i=1}^m Cost(h_\theta(x^i), y^i)$$

- Debido a que para la regresión logística binaria:  
$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$
 es una función sigmoidal no lineal, al sustituir en la fórmula de  $J(\theta)$  tendremos una función no convexa (no solo un mínimo global).
- Objetivo: trabajar con una función coste que sea convexa.

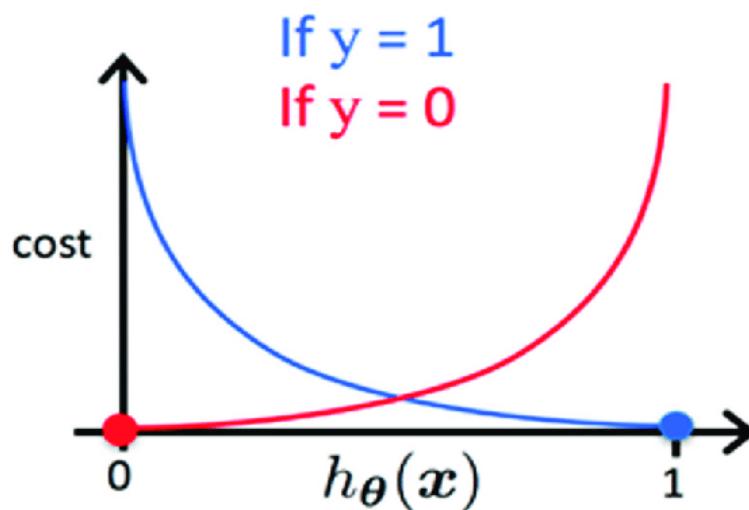
# Regresión logística binaria: Función coste



- Función coste elegida para la regresión logística binaria:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m Cost(h_\theta(x^i), y^i)$$

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{si } y = 1 \\ -\log(1 - h_\theta(x)) & \text{si } y = 0 \end{cases}$$



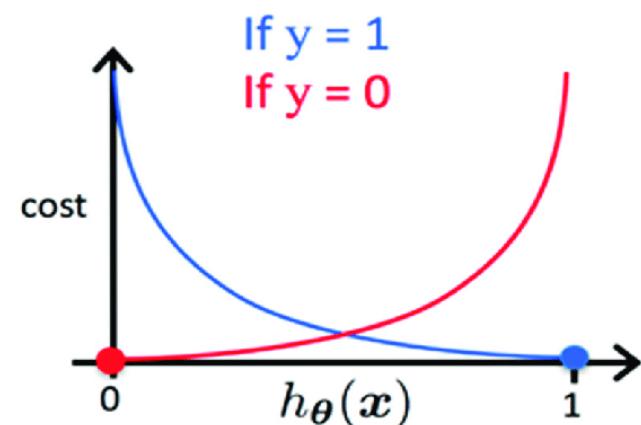
# Regresión logística binaria: Función coste



- Función coste elegida para la regresión logística binaria:

$$Cost(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{si } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{si } y = 0 \end{cases}$$

- Si  $y=1$ :
  - Y si  $h_{\theta}(x) = 1$ , entonces Coste = 0.
  - Y si  $h_{\theta}(x) \rightarrow 0$ , entonces Coste  $\rightarrow \infty$ .



# Regresión logística binaria:

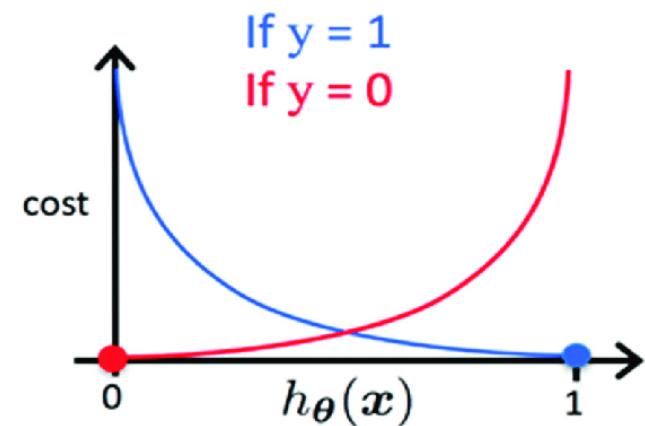
## Función coste



- Función coste elegida para la regresión logística binaria:

$$Cost(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{si } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{si } y = 0 \end{cases}$$

- Si  $y=0$ :
  - Y si  $h_{\theta}(x) = 0$ , entonces Coste = 0.
  - Y si  $h_{\theta}(x) \rightarrow 1$ , entonces Coste  $\rightarrow \infty$ .



# Regresión logística binaria:

## Función coste



- Función coste elegida para la regresión logística binaria:
  - Es convexa.
  - Simplificada:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m Cost(h_\theta(x^i), y^i)$$

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{si } y = 1 \\ -\log(1 - h_\theta(x)) & \text{si } y = 0 \end{cases}$$

$$J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^m y^i \log(h_\theta(x^i)) + (1 - y^i) \log(1 - h_\theta(x^i)) \right]$$

# Regresión logística binaria: Descenso del gradiente



- Objetivo:
  - Encontrar parámetros  $\Theta$  que minimicen la función coste:  
$$\min_{\theta} J(\theta)$$
  - Sustituir esos parámetros  $\Theta$  en la hipótesis

$$h_{\theta}(x) = \frac{1}{1+e^{-\theta^T x}}$$

de la regresión logística binaria y encontrar la predicción correspondiente a  $x$ , interpretando la hipótesis como:

$$P(y = 1|x; \theta)$$



# Regresión logística binaria: Descenso del gradiente

- Considerando la función coste previamente definida:

$$J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^m y^i \log(h_\theta(x^i)) + (1 - y^i) \log(1 - h_\theta(x^i)) \right]$$

Repetir hasta convergencia con actualización simultánea para todos los  $\theta_j$  {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

# Regresión logística binaria: Descenso del gradiente



- Considerando la función coste previamente definida:

$$J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^m y^i \log(h_\theta(x^i)) + (1 - y^i) \log(1 - h_\theta(x^i)) \right]$$

Repetir hasta convergencia con actualización simultánea para todos los  $\theta_j$  {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^i) - y^i) x_j^i$$

}

Igual que en regresión lineal! Tener en cuenta que  $h_\theta(x^i)$  es distinta.

# Regresión logística binaria:

## Descenso del gradiente



- Para implementar el algoritmo del descenso del gradiente podemos usar un bucle o vectorizar.

- Vectorización:**

- Función Sigmoid:

```
def sigmoid(z):  
    return 1 / (1 + np.exp(-z))
```

- Función Coste:

```
def costFunction(theta, X, y):  
    m = len(y)  
    h = sigmoid(np.dot(X, theta))  
    return -(1/m) * np.sum(y * np.log(h) + (1-y)*np.log(1-h))
```

- Función Descenso del Gradiente:

# Regresión logística binaria:

## Descenso del gradiente



- Para implementar el algoritmo del descenso del gradiente podemos usar un bucle o vectorizar.
- Vectorización:

- Función Descenso del Gradiente:

```
def gradientFunction(theta, X, y):  
    m = len(y)  
    h = sigmoid(np.dot(X, theta))  
    return (1/m) * (np.dot(X.T, (h-y)))
```

# Regresión logística binaria: Optimización avanzada



- Hay más algoritmos para optimizar a parte del descenso del gradiente:
  - No necesitan que se seleccione la tasa de aprendizaje o learning rate manualmente.
  - Normalmente convergen antes.
  - Son mucho más complejos tanto de entender como de implementar.
  - Ejemplos: Conjugate gradient (CG), BFGS, L-BFGS.

```
import scipy.optimize as op  
...  
...  
op.fmin_cg(maxiter = n_iter, f = costFunction, x0 = initial_theta.flatten(),  
            fprime = gradientFunction, args = (X, y.to_numpy().flatten()))
```

# Contenido

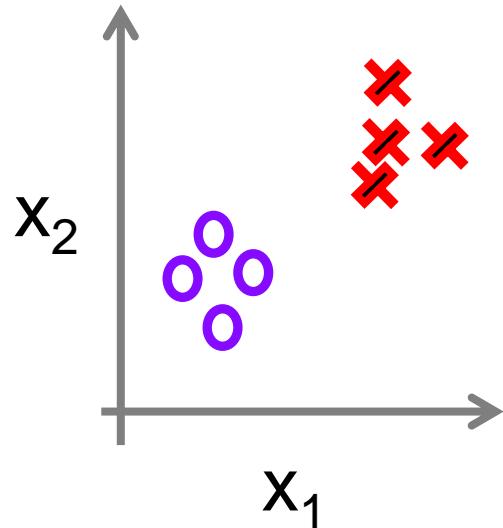


- Regresión logística binaria
  - Hipótesis
  - Función coste
  - Descenso del gradiente
  - Optimización avanzada
- Clasificación multiclasa
  - OneVsAll
- Sobreajuste
  - Regularización: regresión lineal y regresión logística

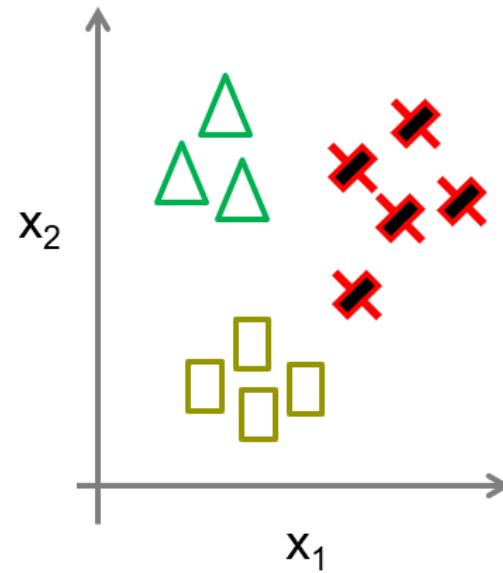
# Clasificación multiclase:



Clasificación binaria:



Clasificación multi-clase:

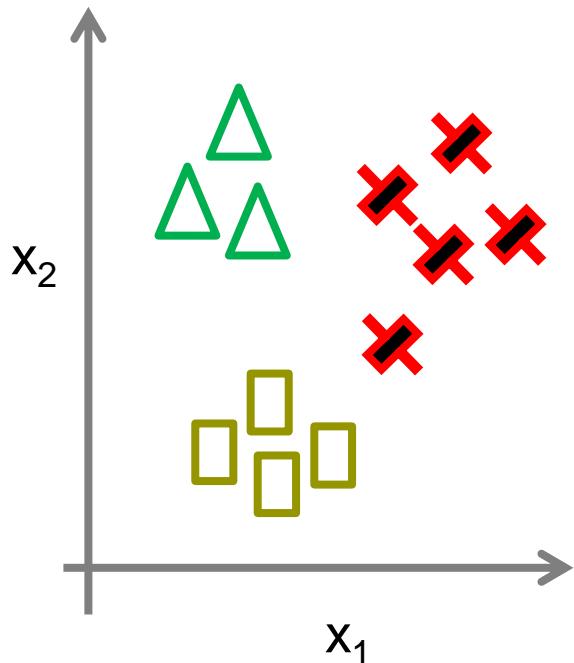


- Ejemplo: Iris dataset:

$y \in \{0, 1, 2\}$  : 0: *i.setosa*; 1: *i.versicolor*; 2: *i.virginica*

# Clasificación multiclas:

## OneVsAll

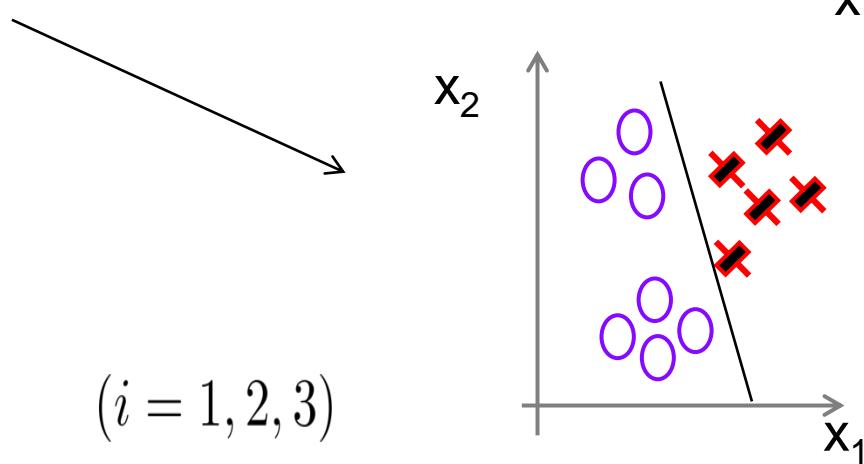
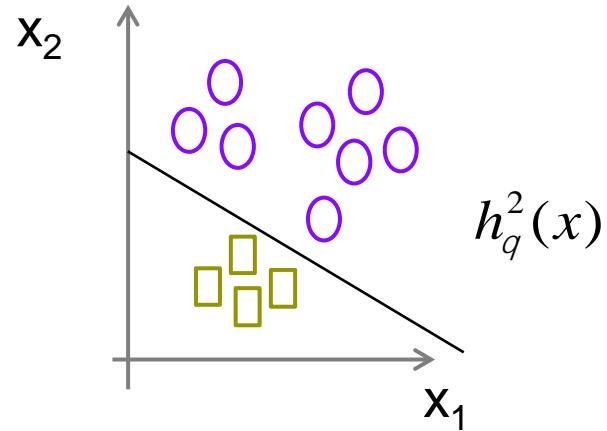
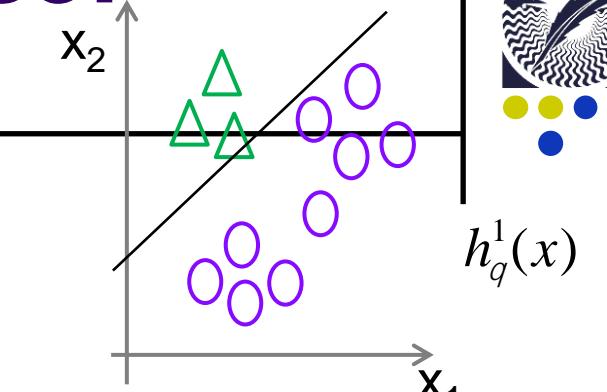
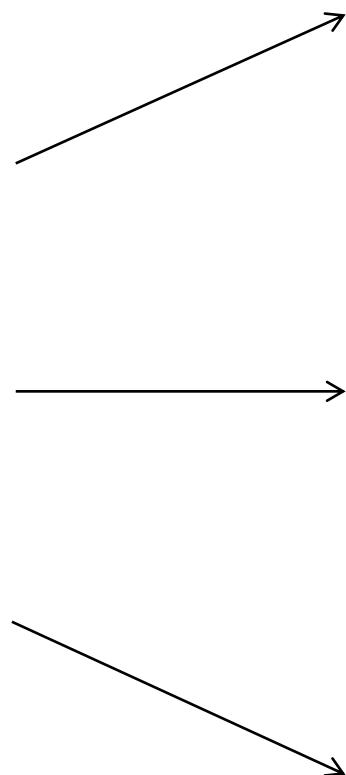


Class 1:

Class 2:

Class 3:

$$h_{\theta}^{(i)}(x) = P(y = i|x; \theta) \quad (i = 1, 2, 3)$$





# Clasificación multiclas:

## OneVsAll

- Entrenar un clasificador de regresión logística binaria  $h_{\theta}^i(x)$  para cada **clase  $i$**  para predecir la **probabilidad de que  $y = i$** .
- Para hacer la predicción de una nueva entrada  $x$  , seleccionar la clase  $i$  que maximiza:

$$\max_i h_{\theta}^i(x)$$

# Contenido



- Regresión logística binaria
  - Hipótesis
  - Función coste
  - Descenso del gradiente
  - Optimización avanzada
- Clasificación multiclasa
  - OneVsAll
- Sobreajuste
  - Regularización: regresión lineal y regresión logística

# Sobreajuste:



- “Overfitting” o sobreajuste:
  - Problema en la generalización
  - Suele ocurrir cuando hay muchos atributos en el dataset.

- La hipótesis se ajusta muy bien al conjunto de training

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^i) - y^i)^2 \approx 0$$

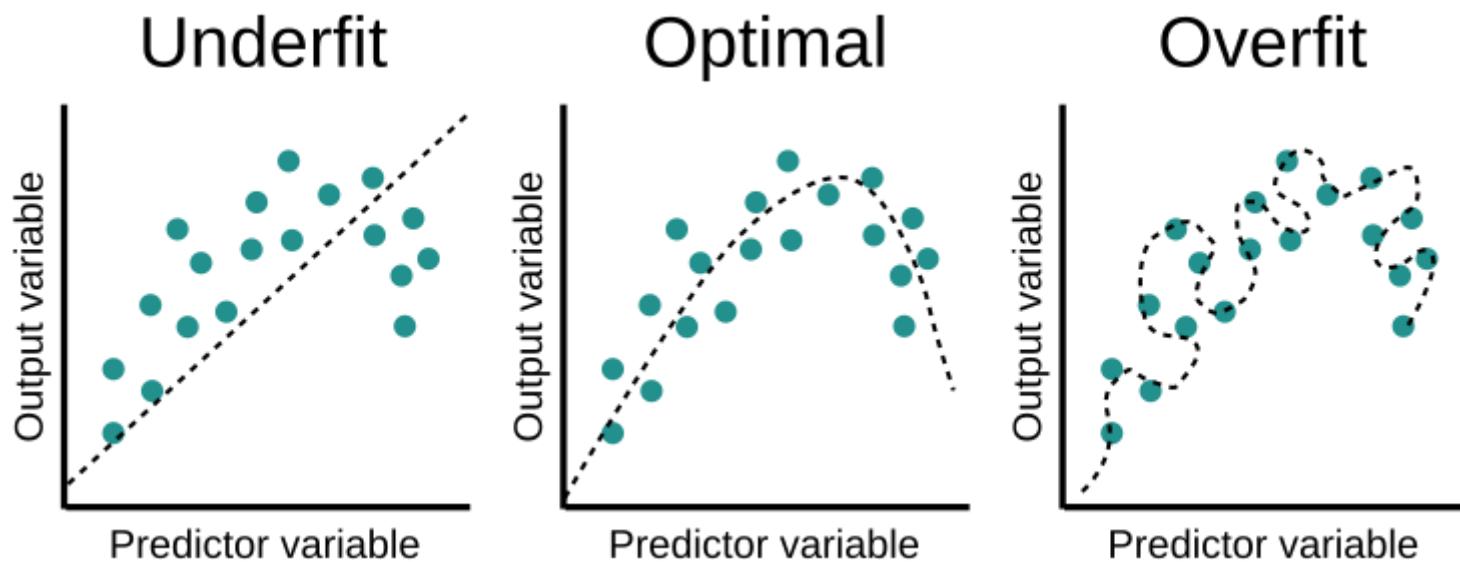
pero no generaliza correctamente para nuevas instancias (no incluidas en el conjunto de training).

- “Underfitting” o subajuste

# Sobreajuste:



- **Ejemplo 1: Regresión lineal**



$$\theta_0 + \theta_1 x$$

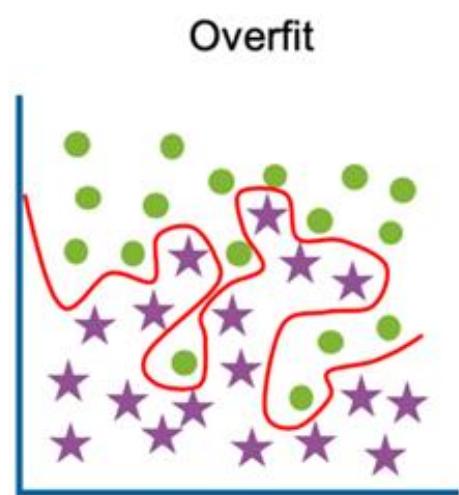
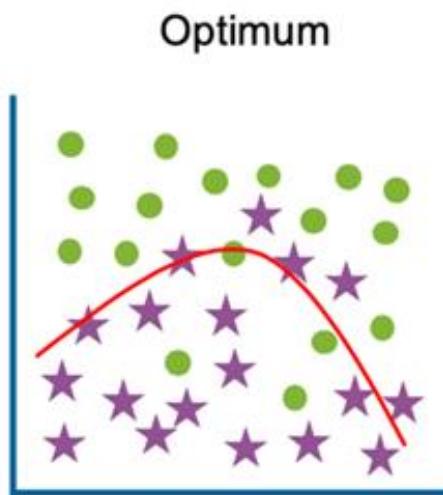
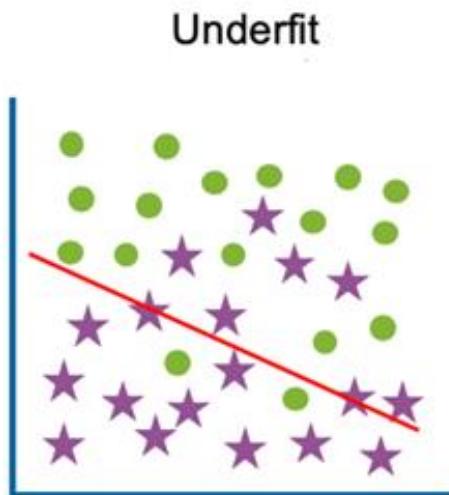
$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\begin{aligned}\theta_0 + \theta_1 x + \theta_2 x^2 + \\ + \theta_3 x^3 + \theta_4 x^4\end{aligned}$$

# Sobreajuste:



- **Ejemplo 2: Regresión logística**



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$$\begin{aligned} & g(\theta_0 + \theta_1 x + \theta_2 x_2 \\ & + \theta_3 x_1^2 + \theta_4 x_2^2 \\ & + \theta_5 x_1 x_2) \end{aligned}$$

$$\begin{aligned} & g(\theta_0 + \theta_1 x + \theta_2 x_1^2 \\ & + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 \\ & + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots) \end{aligned}$$



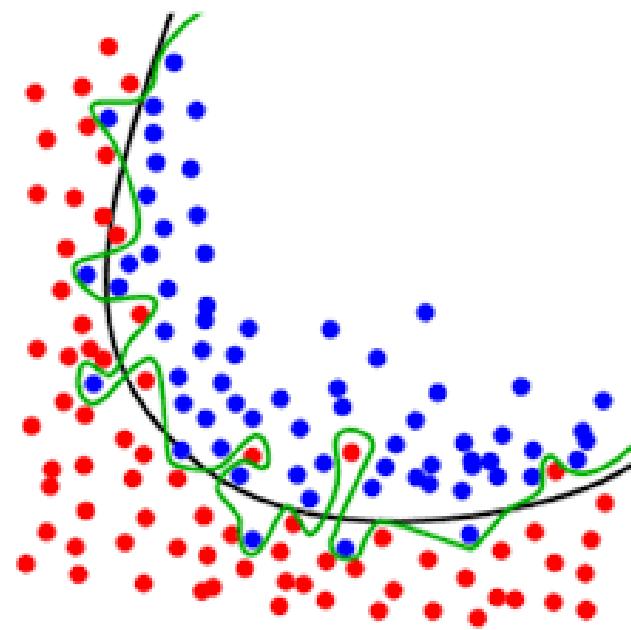
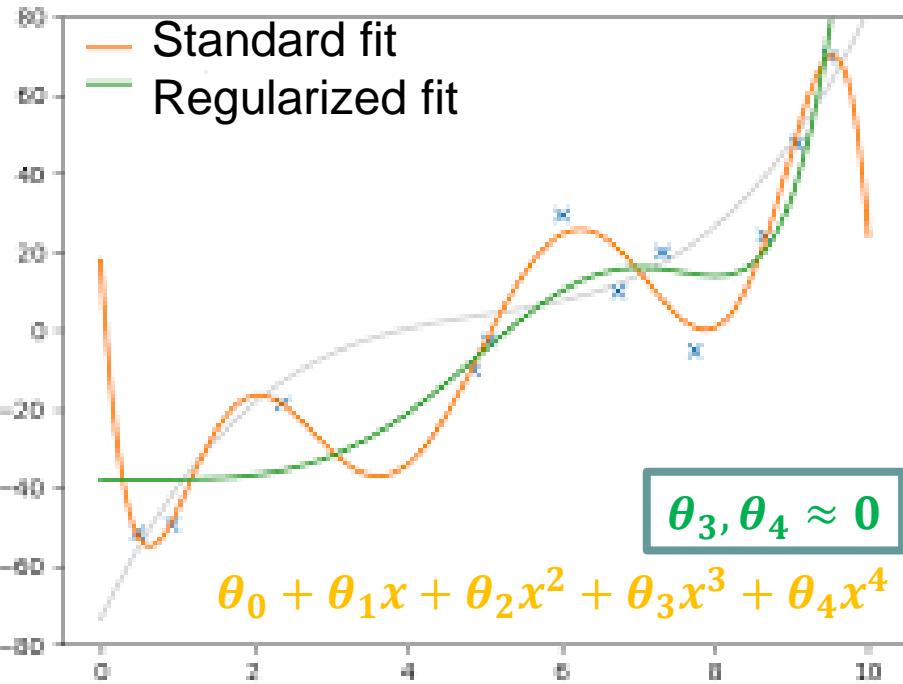
# Sobreajuste:

- Opciones para abordar el problema del “Overfitting”:
  - Reducir el número de variables:
    - Seleccionando manualmente las consideradas más importantes.
    - Usar algoritmo de selección de variables.
  - Regularización:
    - Interesante cuando todas las variables influyen “algo” en la predicción  $y$ .
    - Se mantienen todas las variables, reduciendo las magnitudes/valores de los parámetros  $\theta_j$  (se penalizan esos parámetros en la optimización de la función coste).

# Sobreajuste: Regularización



- Valores pequeños para algunos parámetros  $\theta$  para conseguir:
  - Hipótesis “más sencillas”.
  - Menos predisposto al “overfitting”.



# Sobreajuste: Regularización – regresión lineal



- **Función coste** para regresión lineal con regularización:

$$J(\theta) = \frac{1}{2m} \left[ \sum_{i=1}^m (h_\theta(x^i) - y^i)^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

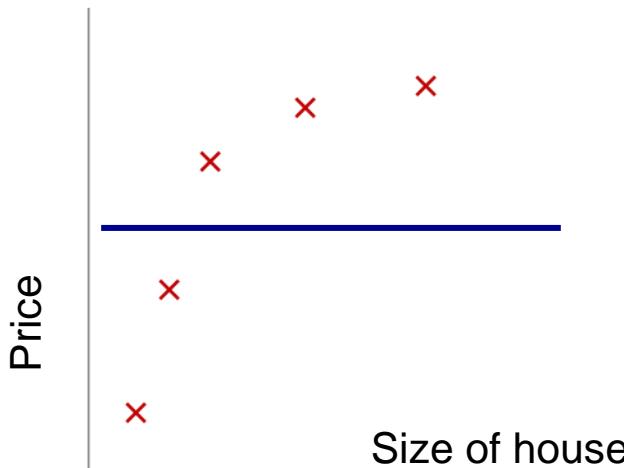
(A)                          (B)

- $\lambda \sum_{j=1}^n \theta_j^2$  : es el término de la regularización.
- $\lambda$  (lambda): es el parámetro de la regularización. Tiene 2 objetivos:
  - Conseguir un buen ajuste a los datos del entrenamiento. **(A)**
  - Conseguir una hipótesis relativamente simple para evitar el “overfitting”. **(B)**

# Sobreajuste: Regularización – regresión lineal



- **Función coste** para regresión lineal con regularización:
  - Hay que seleccionar un  $\lambda$  bueno.
    - Si demasiado grande (sobre  $\lambda = 10^{10}$ ): podemos acabar con todos los  $\theta_1, \theta_2, \dots, \theta_n$  siendo muy cercanos a cero y por lo tanto la hipótesis podría ser:  $h_{\theta}(x) = \theta_0$  que daría lugar a “underfitting”.



- Hay algoritmos que lo seleccionan automáticamente.



# Sobreajuste: Regularización – regresión lineal

- **Descenso del gradiente** para regresión lineal con regularización:

Repetir {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^i) - y^i) x_0^i$$

$\frac{\partial}{\partial \theta_0} J(\theta)$

$$\theta_j := \theta_j - \alpha \left( \frac{1}{m} \sum_{i=1}^m (h_\theta(x^i) - y^i) x_j^i - \frac{\lambda}{m} \theta_j \right)$$
$$= \theta_j \left( 1 - \alpha \frac{\lambda}{m} \right) - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^i) - y^i) x_j^i$$

$\frac{\partial}{\partial \theta_j} J_{regularizada}(\theta)$

$$(j = 1, 2, 3, \dots, n)$$

}



# Sobreajuste: Regularización – regresión lineal

- **Ecuación normal** para regresión lineal con regularización: Si  $\lambda > 0$

$$X = \begin{bmatrix} (x^1)^T \\ \dots \\ (x^m)^T \end{bmatrix} \quad y = \begin{bmatrix} y^1 \\ \dots \\ y^m \end{bmatrix} \quad \min_{\theta} J(\theta)$$

- Sin regularización:  $\theta = (X^T X)^{-1} X^T y$
- Con regularización:  $\theta = (X^T X + \lambda \begin{bmatrix} 0 & & & & 0 \\ & 1 & & & \\ & & 1 & & \\ 0 & & & \ddots & \\ & & & & 1 \end{bmatrix})^{-1} X^T y$ 
  - ¡Va a ser siempre invertible!
  - Ejemplo: si  $n=2$   $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$



# Sobreajuste: Regularización – regresión logística

- **Función coste** para regresión logística binaria con regularización:

$$J(\theta) = - \left[ \frac{1}{m} \sum_{i=1}^m y^i \log(h_\theta(x^i)) + (1 - y^i) \log(1 - h_\theta(x^i)) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

# Sobreajuste: Regularización – regresión logística



- **Descenso del gradiente** para regresión logística binaria con regularización:

Repetir {

$$\boxed{\theta_0} := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^i) - y^i)x_0^i$$

$$\boxed{\theta_j} := \theta_j - \alpha \left( \frac{1}{m} \sum_{i=1}^m (h_\theta(x^i) - y^i)x_j^i - \frac{\lambda}{m} \theta_j \right)$$

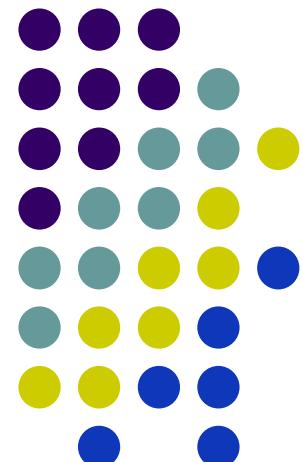
$$(j = 1, 2, 3, \dots, n)$$

}

Vectorización en python.

# Redes Neuronales I: Representación y conocimiento

**Inteligencia Artificial**  
*Ingeniería Informática*  
*en Sistemas de Información*



UNIVERSIDAD  
**PABLO DE OLAVIDE**  
SEVILLA



# Contenido

---

- Motivación
- Neurona artificial
- Red neuronal
  - Funciones de activación
- Conocimiento
  - Forward-Propagation
- Ejemplos

# Motivación: Origen

---



- Modelo computacional que intenta imitar el comportamiento de las neuronas en el cerebro
- Muy populares en los años 80, pero en los 90 decayeron (invierno de la IA)
- Ahora han resurgido gracias a los avances tecnológicos y el deep learning

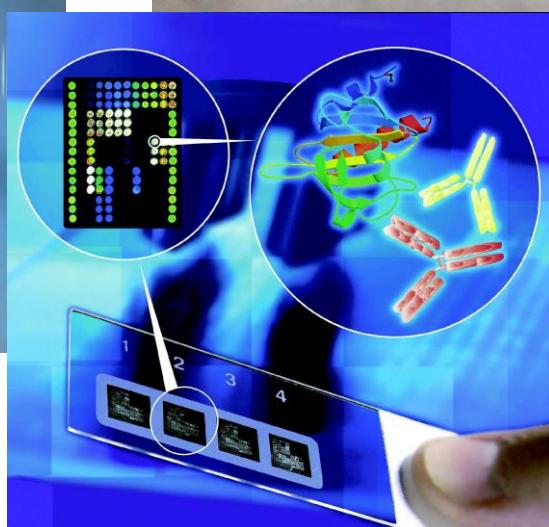
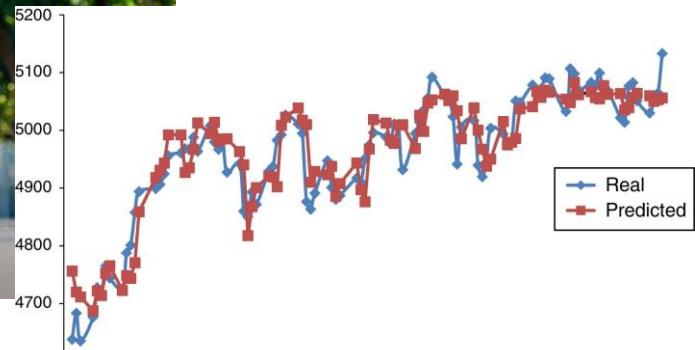
# Motivación: ¿Para qué?



DL9CD D 5036

↓  
OCR

DL9CD5036

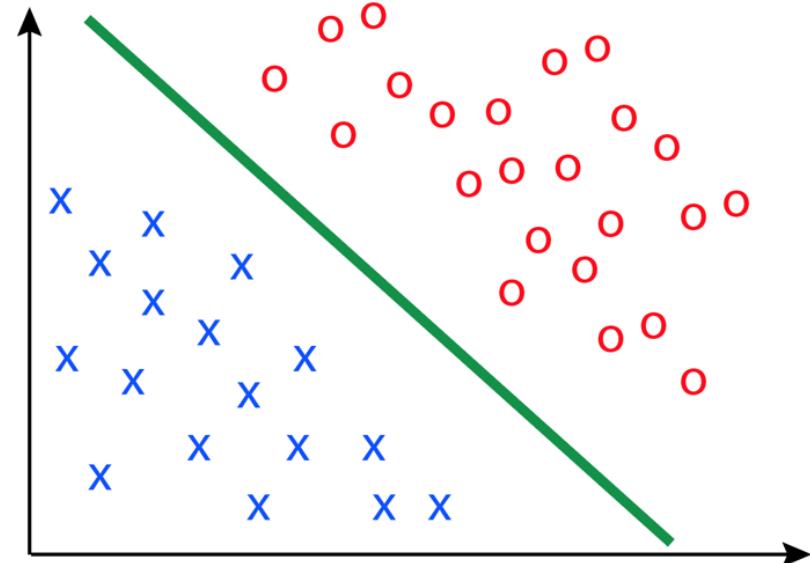


...y mucho más!

# Motivación: ¿Cómo?



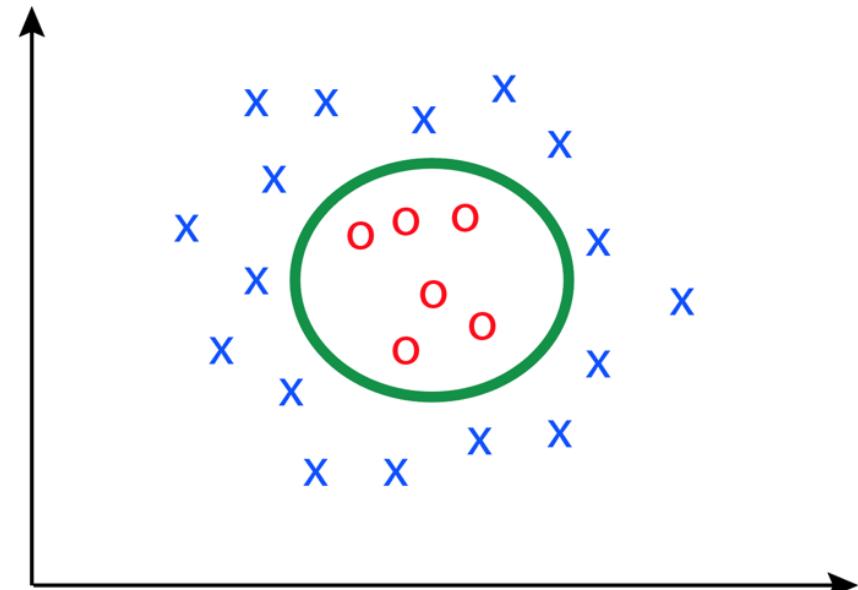
- Buscando fronteras de decisión bien ajustadas
  - Lineales



# Motivación: ¿Cómo?



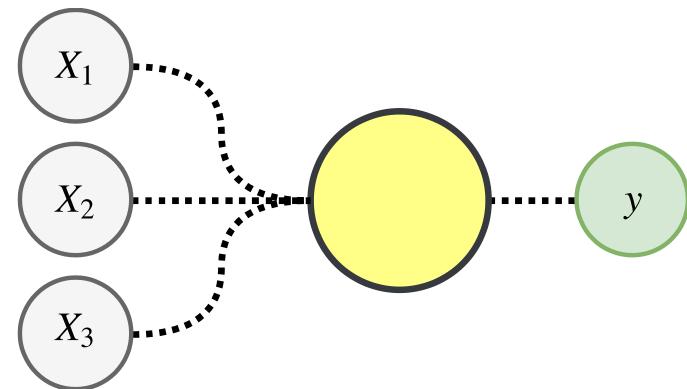
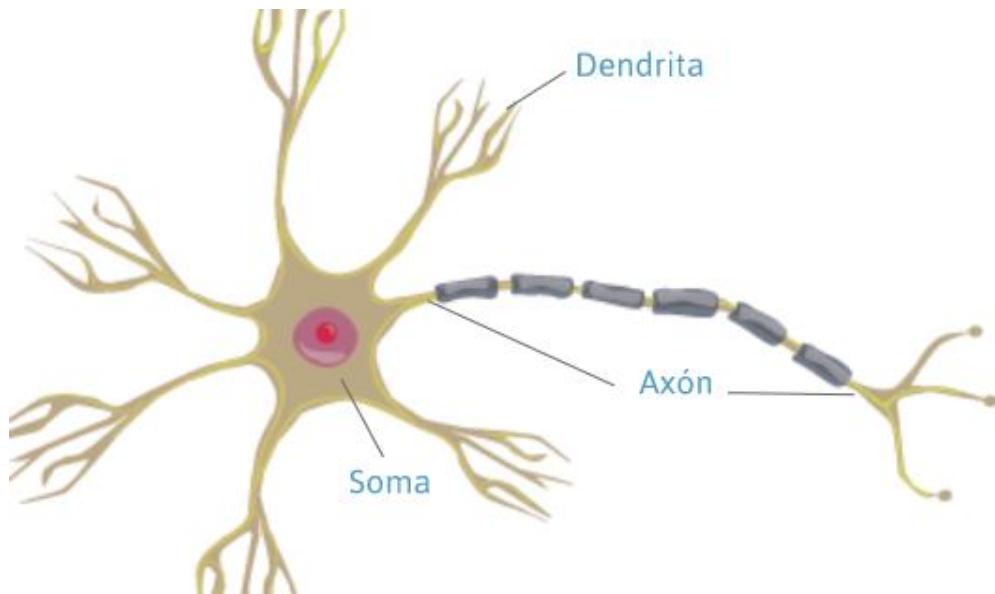
- Buscando fronteras de decisión bien ajustadas
  - No lineales



# Neurona artificial: Neurona biológica vs artificial



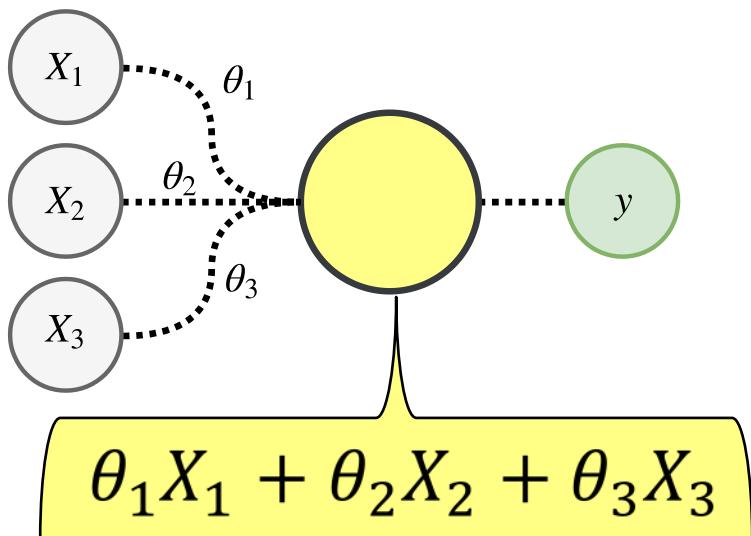
- Es la unidad básica de procesamiento
- Recibe estímulos de entrada
- Realiza un procesamiento
- Ofrece una salida



# Neurona artificial: Procesamiento de una neurona



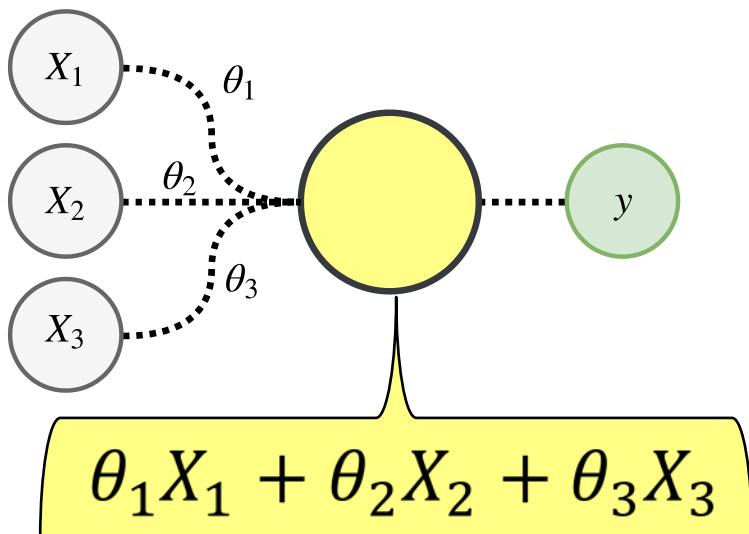
- Usar valores de entrada para hacer una suma ponderada dada por los **pesos ( $\theta$ )** asignados a cada variable de entrada (intensidad con la que afecta)



# Neurona artificial: Procesamiento de una neurona



- Usar valores de entrada para hacer una suma ponderada dada por los **pesos ( $\theta$ )** asignados a cada variable de entrada (intensidad con la que afecta)

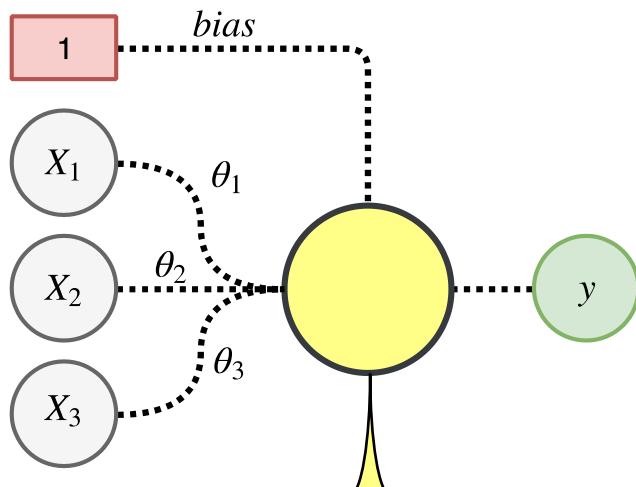


¿Regresión lineal?  
T.3- pag 49

# Neurona artificial: Procesamiento de una neurona



- Sí, se aplica un modelo de regresión lineal
- Los pesos varían la inclinación de la recta
- Término independiente (**bias** o **sesgo**), que **valdrá 1**



$$b + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3$$

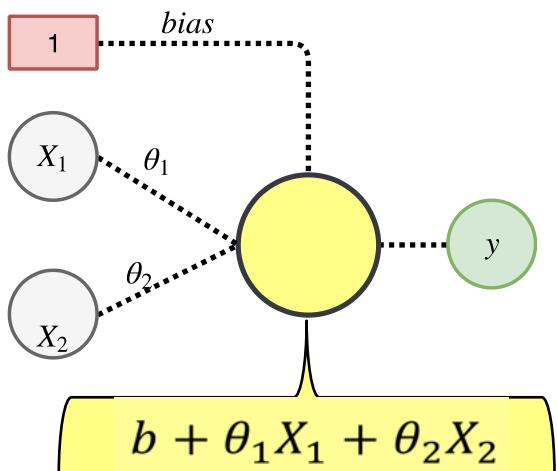


¡Sigue siendo una  
regresión lineal!

# Neurona artificial: Codificación básica



PUERTA AND



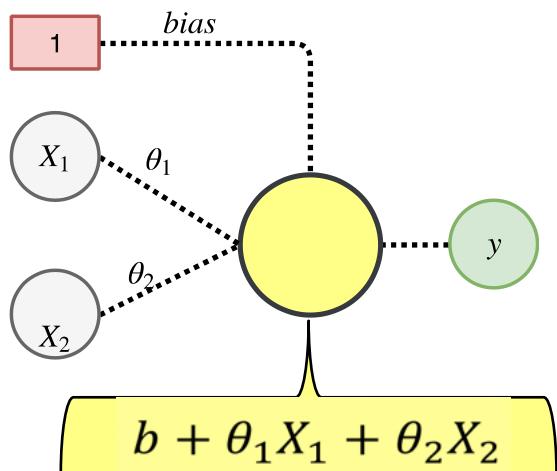
**Objetivo:**  
Variar bias y pesos hasta cumplir las condiciones.

X1	X2	Objetivo	Y

# Neurona artificial: Codificación básica



PUERTA AND



*Bias: -3*

$\theta_1 = 2$

$\theta_2 = 4$

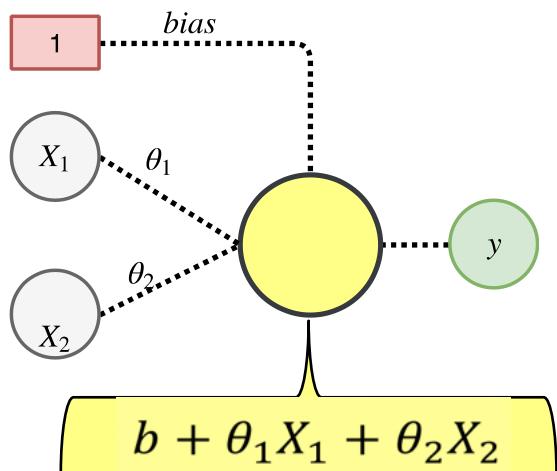
**¡NO CONSEGUIDO!**

X1	X2	Objetivo	Y
			-3
			-1
			1
			3

# Neurona artificial: Codificación básica



PUERTA AND



Bias: 2  
 $\theta_1 = -4$   
 $\theta_2 = 1$

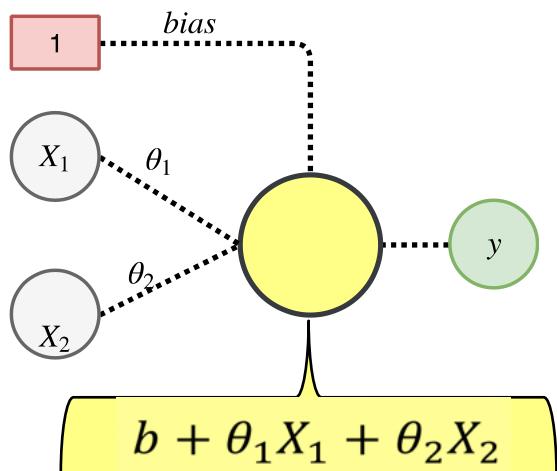
¡NO CONSEGUIDO!

X1	X2	Objetivo	Y
			2
			-2
			3
			-1

# Neurona artificial: Codificación básica



PUERTA AND



Bias: 2

$$\theta_1 = 1$$

$$\theta_2 = 1$$

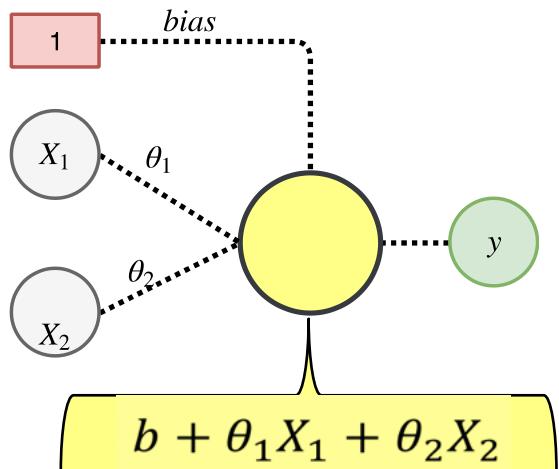
¡NO CONSEGUIDO!

X1	X2	Objetivo	Y
			2
			3
			3
			4

# Neurona artificial: Codificación básica



PUERTA AND



Bias: -6

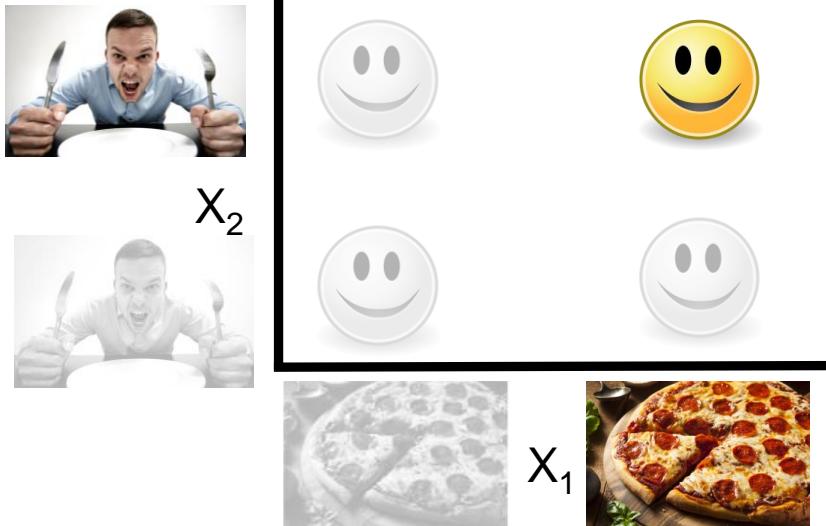
$\theta_1 = 4$

$\theta_2 = 5$

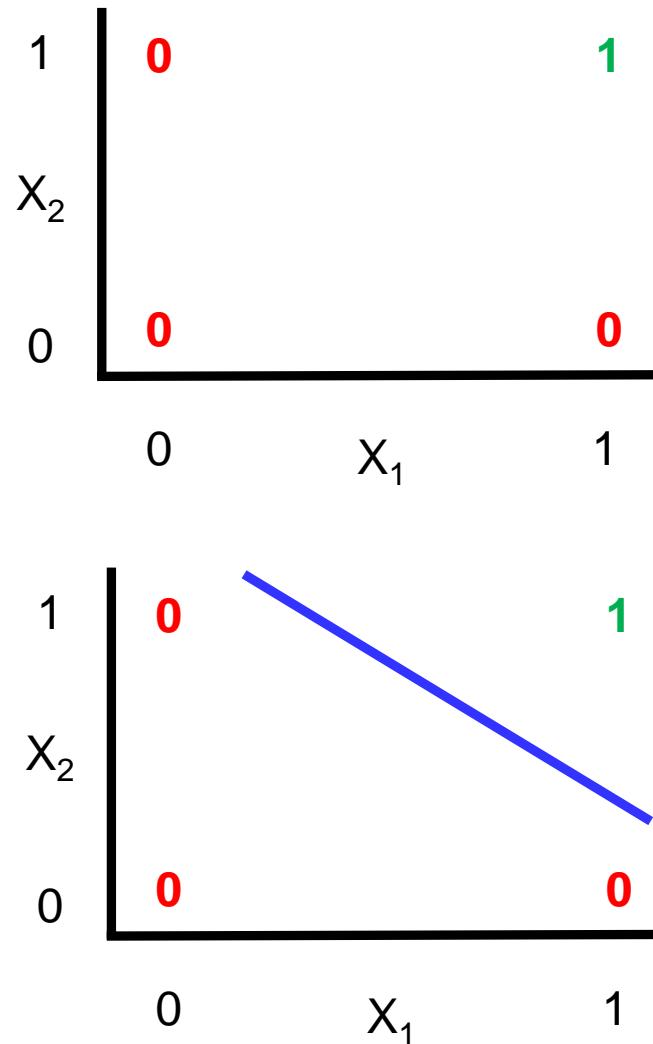
¡CONSEGUIDO!

X1	X2	Objetivo	Y
			-6
			-2
			-1
			3

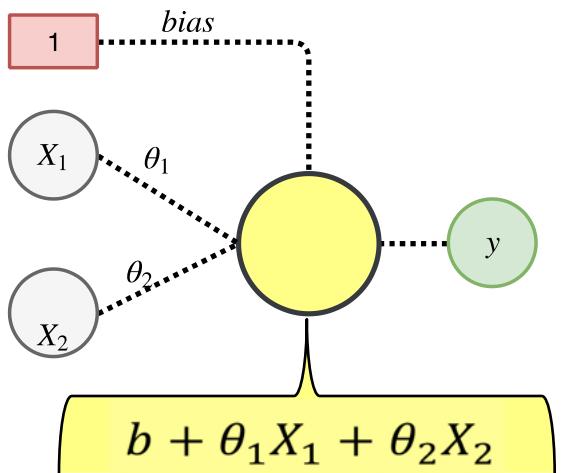
# Neurona artificial: Codificación básica



**PUERTA AND**

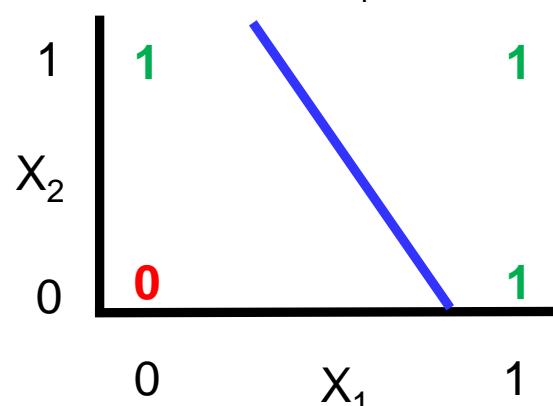
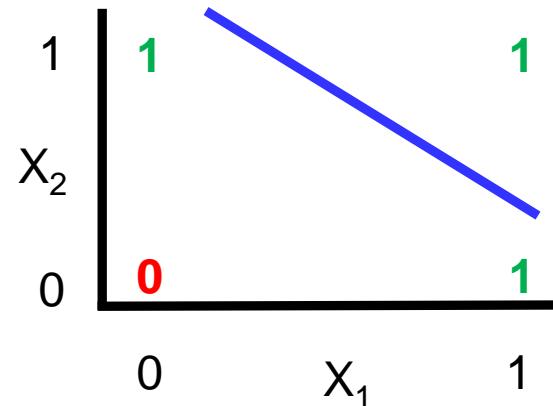
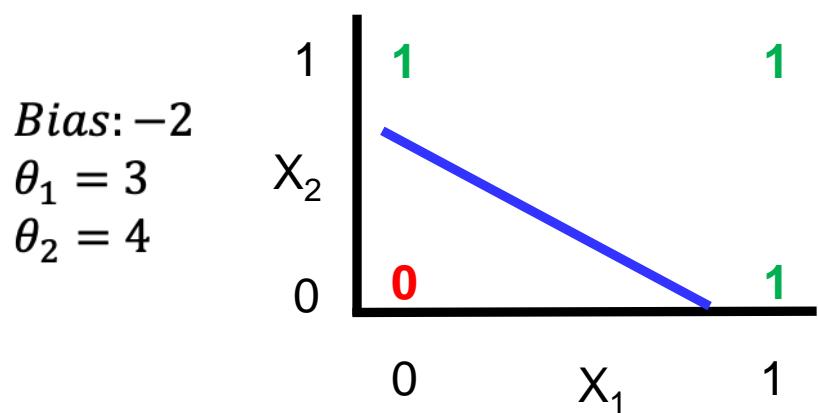


# Neurona artificial: Codificación básica



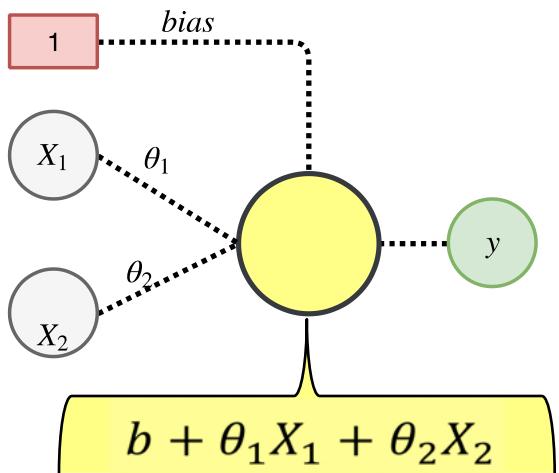
$$\begin{aligned} &\text{Bias: } -6 \\ &\theta_1 = 4 \\ &\theta_2 = 5 \end{aligned}$$

$$\begin{aligned} &\text{Bias: } -4 \\ &\theta_1 = 2 \\ &\theta_2 = 3 \end{aligned}$$



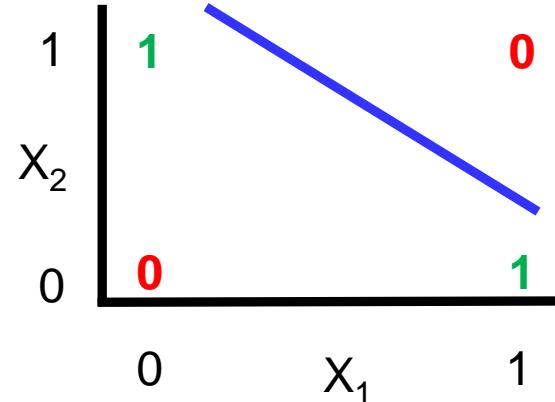
**PUERTA OR**

# Neurona artificial: Codificación básica

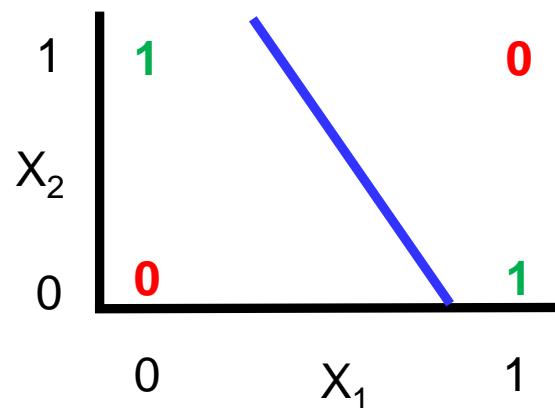


## PUERTA XOR

$$\begin{aligned} \text{Bias: } & -6 \\ \theta_1 = & 4 \\ \theta_2 = & 5 \end{aligned}$$



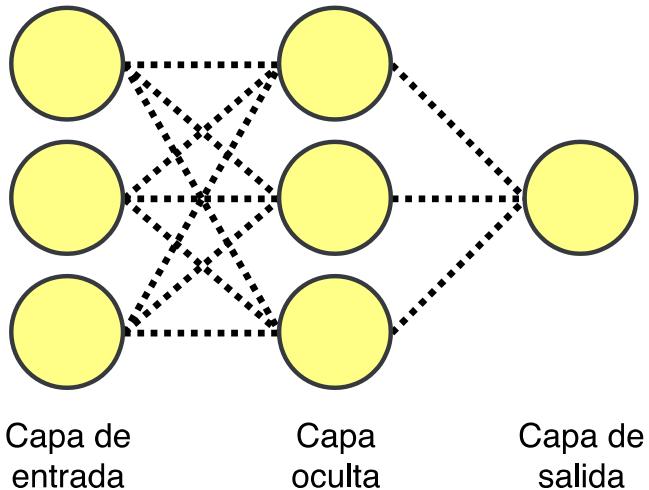
$$\begin{aligned} \text{Bias: } & -4 \\ \theta_1 = & 2 \\ \theta_2 = & 3 \end{aligned}$$



No tiene solución si usamos una sola neurona porque no son separables linealmente.

¿La solución?: ¡Añadir otra neurona!

# Red neuronal: Organización

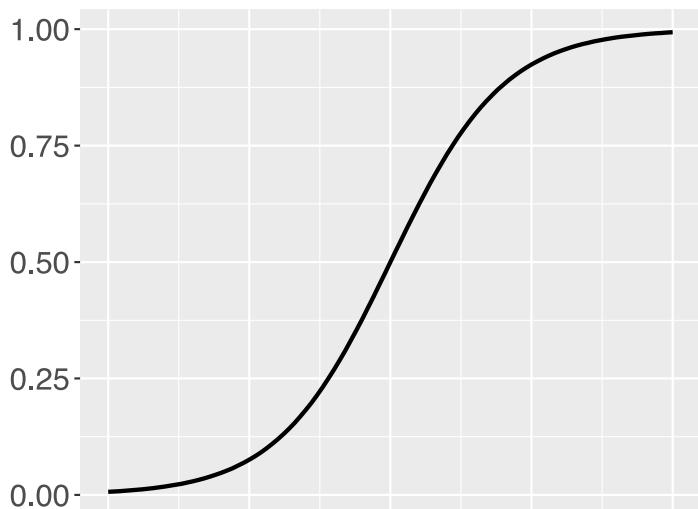


- Organizado en capas y neuronas
- Combinar neuronas para modelos complejos
- Permite aprendizaje jerarquizado
- Pero... ¡concatenar regresiones lineales equivale a una sola recta!

# Red neuronal: Funciones de activación

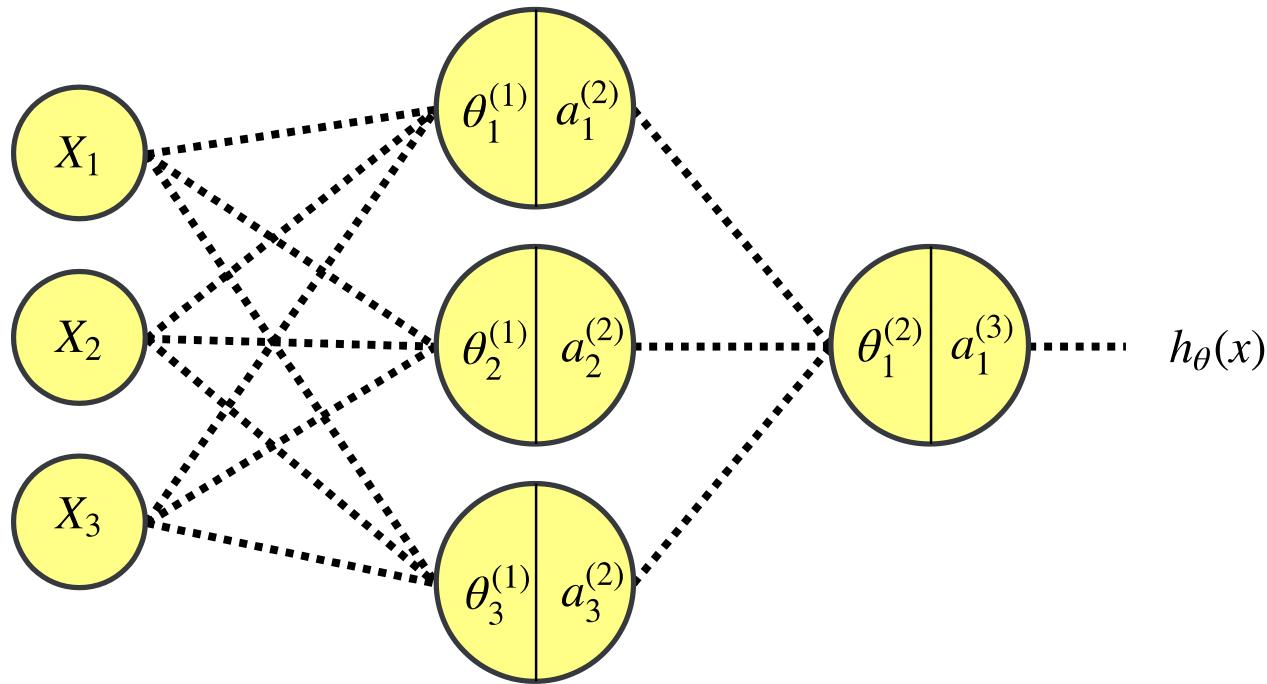


- Solución: **Funciones de activación**
- Aplica transformaciones no lineales, permitiendo encadenar las salidas de cada neurona
- Entre ellas está la **sigmoide**, usada en clasificación por ofrecer una probabilidad [0,1]



$$g(x) = \frac{1}{1 + e^{-x}}$$

# Red neuronal: Arquitectura general



$\theta_i^{(j)}$  = Pesos de la neurona  $i$  en la capa  $j$

$a_i^{(j)}$  = Activación de la neurona  $i$  en la capa  $j$

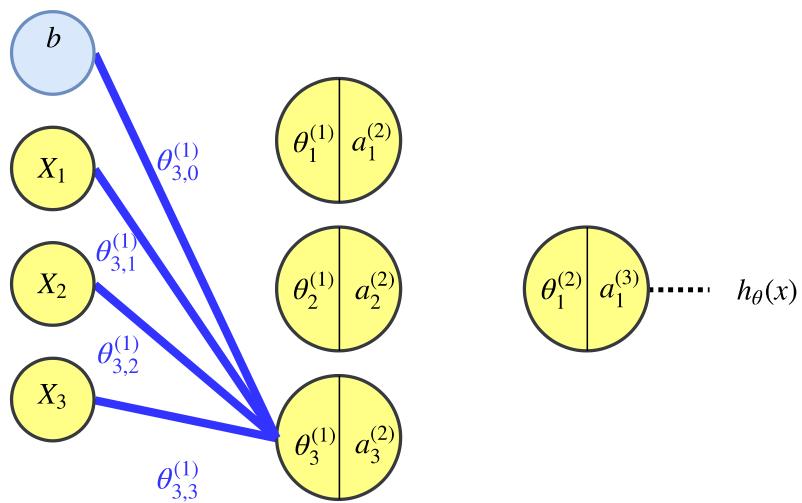
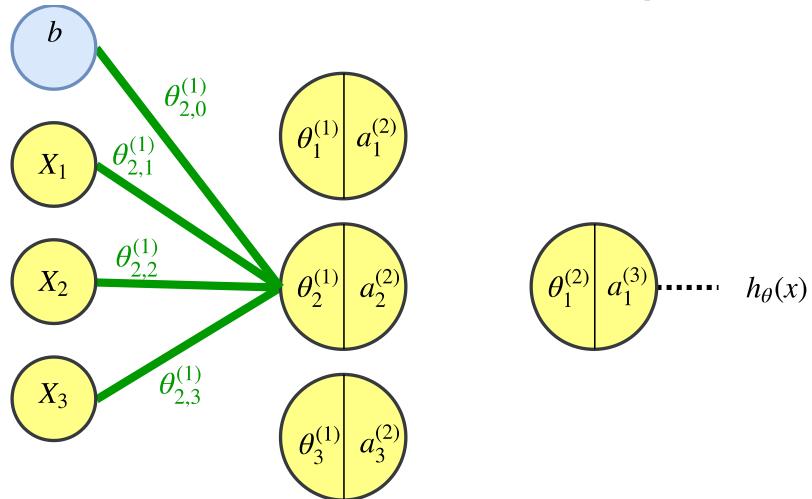
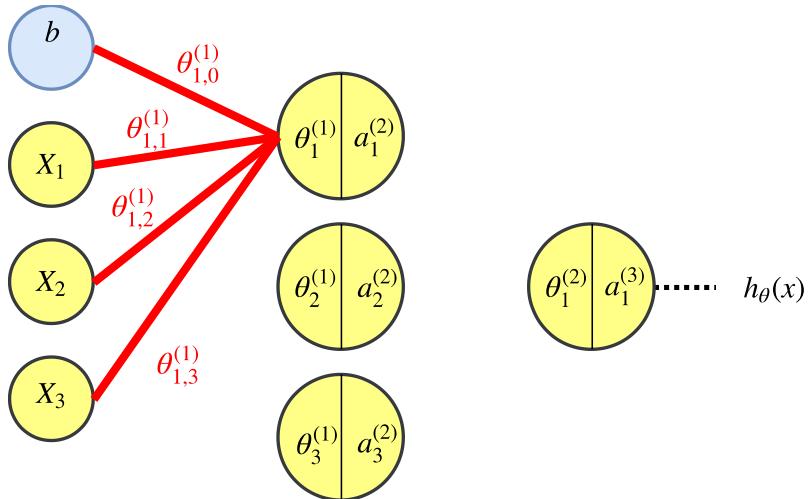
$h_\theta(x)$  = Valor de salida de la red

# Modelado de conocimiento: Conceptos



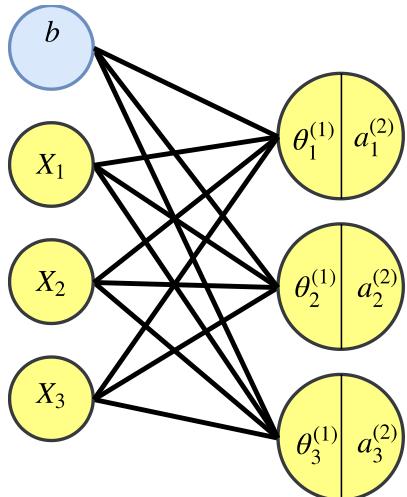
- La extracción de conocimiento se realiza a través de un entrenamiento de la red, que consta de dos fases:
  - **Forward-propagation:** Forma en la que se modela el transpaso de conocimiento a través de las neuronas y capas para obtener la salida final
  - **Back-propagation:** Cálculo de los pesos para obtener el modelo. Se resuelve un problema de optimización

# Conocimiento: Forward-propagation



$$\Theta^{(1)} = \begin{bmatrix} (\theta_1^{(1)})^T \\ (\theta_2^{(1)})^T \\ (\theta_3^{(1)})^T \end{bmatrix} = \begin{bmatrix} \theta_{1,0}^{(1)} & \theta_{1,1}^{(1)} & \theta_{1,2}^{(1)} & \theta_{1,3}^{(1)} \\ \theta_{2,0}^{(1)} & \theta_{2,1}^{(1)} & \theta_{2,2}^{(1)} & \theta_{2,3}^{(1)} \\ \theta_{3,0}^{(1)} & \theta_{3,1}^{(1)} & \theta_{3,2}^{(1)} & \theta_{3,3}^{(1)} \end{bmatrix}$$

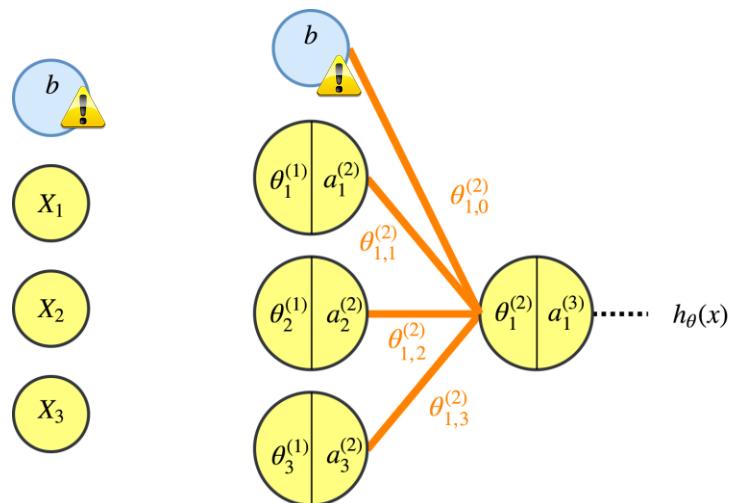
# Conocimiento: Forward-propagation



$$a_1^{(2)} = g(\theta_{1,0}^{(1)}b + \theta_{1,1}^{(1)}X_1 + \theta_{1,2}^{(1)}X_2 + \theta_{1,3}^{(1)}X_3)$$

$$\theta_1^{(2)} \quad a_1^{(3)} \dots h_\theta(x) \quad a_2^{(2)} = g\left(\theta_{2,0}^{(1)}b + \theta_{2,1}^{(1)}X_1 + \theta_{2,2}^{(1)}X_2 + \theta_{2,3}^{(1)}X_3\right)$$

$$a_3^{(2)} = g(\theta_{3,0}^{(1)}b + \theta_{3,1}^{(1)}X_1 + \theta_{3,2}^{(1)}X_2 + \theta_{3,3}^{(1)}X_3)$$



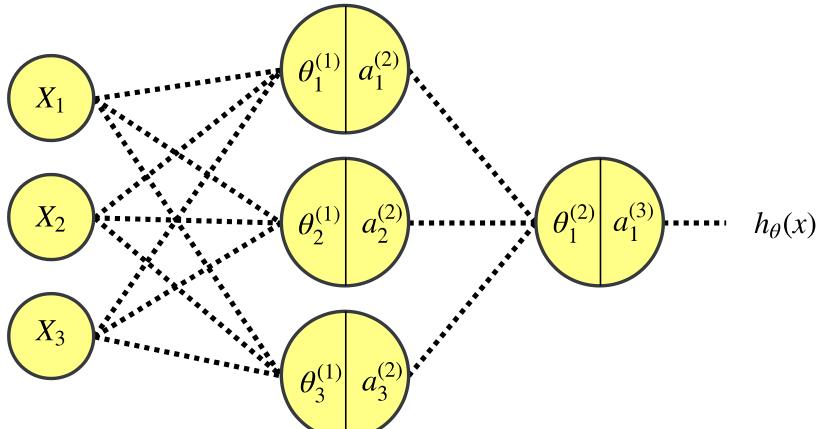
$$\Theta^{(2)} = [(\theta_1^{(2)})^T] = [\theta_{1,0}^{(2)} \quad \theta_{1,1}^{(2)} \quad \theta_{1,2}^{(2)} \quad \theta_{1,3}^{(2)}]$$

$$a_1^{(3)} = g(\theta_{1,0}^{(2)}b + \theta_{1,1}^{(2)}a_1^{(2)} + \theta_{1,2}^{(2)}a_2^{(2)} + \theta_{1,3}^{(2)}a_3^{(2)})$$



¡Añadir +1 correspondiente al sesgo en todas las capas!

# Conocimiento: Todo en 1 - Forward-propagation



$\theta_i^j$  = pesos de la neurona i en la capa j

$a_i^j$  = Activación de la neurona i en la capa j

$$a_1^{(2)} = g(\theta_{1,0}^{(1)} b + \theta_{1,1}^{(1)} X_1 + \theta_{1,2}^{(1)} X_2 + \theta_{1,3}^{(1)} X_3)$$

$$a_2^{(2)} = g(\theta_{2,0}^{(1)} b + \theta_{2,1}^{(1)} X_1 + \theta_{2,2}^{(1)} X_2 + \theta_{2,3}^{(1)} X_3)$$

$$a_3^{(2)} = g(\theta_{3,0}^{(1)} b + \theta_{3,1}^{(1)} X_1 + \theta_{3,2}^{(1)} X_2 + \theta_{3,3}^{(1)} X_3)$$

$$\Theta^{(1)} = \begin{bmatrix} (\theta_1^{(1)})^T \\ (\theta_2^{(1)})^T \\ (\theta_3^{(1)})^T \end{bmatrix} = \begin{bmatrix} \theta_{1,0}^{(1)} & \theta_{1,1}^{(1)} & \theta_{1,2}^{(1)} & \theta_{1,3}^{(1)} \\ \theta_{2,0}^{(1)} & \theta_{2,1}^{(1)} & \theta_{2,2}^{(1)} & \theta_{2,3}^{(1)} \\ \theta_{3,0}^{(1)} & \theta_{3,1}^{(1)} & \theta_{3,2}^{(1)} & \theta_{3,3}^{(1)} \end{bmatrix}$$

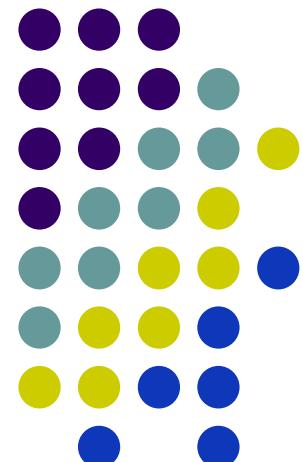
$$\Theta^{(2)} = [(\theta_1^{(2)})^T] = [\theta_{1,0}^{(2)} \quad \theta_{1,1}^{(2)} \quad \theta_{1,2}^{(2)} \quad \theta_{1,3}^{(2)}]$$

$$a_1^{(3)} = g(\theta_{1,0}^{(2)} b + \theta_{1,1}^{(2)} a_1^{(2)} + \theta_{1,2}^{(2)} a_2^{(2)} + \theta_{1,3}^{(2)} a_3^{(2)})$$

**Solución final:**  $h_\theta(x) = a_1^{(3)}$

# Redes Neuronales II: Aprendizaje de la red

**Inteligencia Artificial  
*Ingeniería Informática  
en Sistemas de Información***



UNIVERSIDAD  
**PABLO DE  
OLAVIDE**  
SEVILLA



# Contenido

---

- Breve repaso
- Función de coste en NN
- Descenso del gradiente
- Back-Propagation
- Implementación
  - Unroll
  - Coste y descenso del gradiente
  - Comprobación del gradiente
  - Inicialización

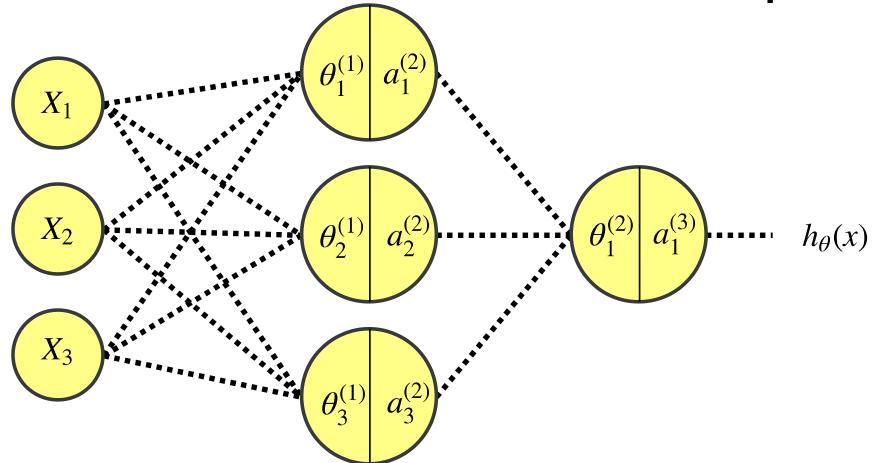
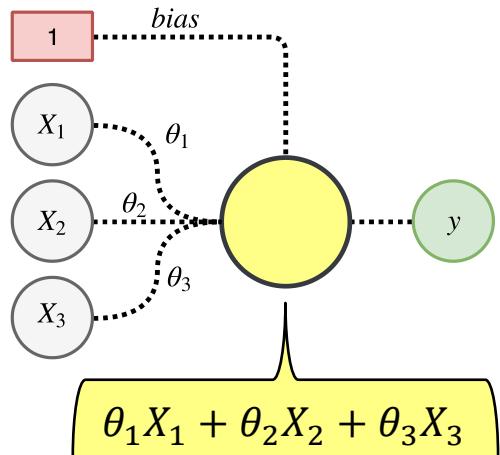
# Breve repaso:

## Conceptos



- Neurona artificial (perceptrón)
- Codificación básica (regresión lineal)
  - Puerta AND
  - Puerta OR
  - Puerta XOR – Imposible con una neurona
- Estructura de una red neuronal
- Función de activación
- Propagación del conocimiento (forward-propagation)

# Breve repaso: Neurona & Forward-Propagation



$$a_1^{(2)} = g(\theta_{1,0}^{(1)} b + \theta_{1,1}^{(1)} X_1 + \theta_{1,2}^{(1)} X_2 + \theta_{1,3}^{(1)} X_3)$$

$$a_2^{(2)} = g(\theta_{2,0}^{(1)} b + \theta_{2,1}^{(1)} X_1 + \theta_{2,2}^{(1)} X_2 + \theta_{2,3}^{(1)} X_3)$$

$$a_3^{(2)} = g(\theta_{3,0}^{(1)} b + \theta_{3,1}^{(1)} X_1 + \theta_{3,2}^{(1)} X_2 + \theta_{3,3}^{(1)} X_3)$$

$$\Theta^{(1)} = \begin{bmatrix} (\theta_1^{(1)})^T \\ (\theta_2^{(1)})^T \\ (\theta_3^{(1)})^T \end{bmatrix} = \begin{bmatrix} \theta_{1,0}^{(1)} & \theta_{1,1}^{(1)} & \theta_{1,2}^{(1)} & \theta_{1,3}^{(1)} \\ \theta_{2,0}^{(1)} & \theta_{2,1}^{(1)} & \theta_{2,2}^{(1)} & \theta_{2,3}^{(1)} \\ \theta_{3,0}^{(1)} & \theta_{3,1}^{(1)} & \theta_{3,2}^{(1)} & \theta_{3,3}^{(1)} \end{bmatrix}$$

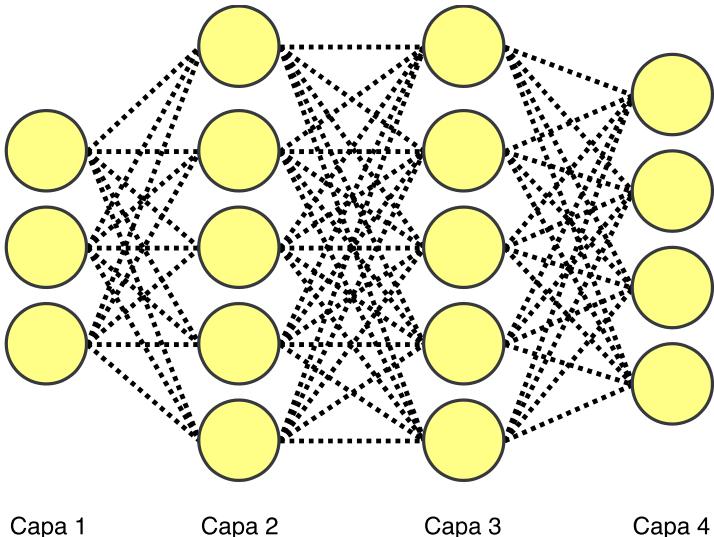
$$\Theta^{(2)} = \begin{bmatrix} (\theta_1^{(2)})^T \end{bmatrix} = \begin{bmatrix} \theta_{1,0}^{(2)} & \theta_{2,1}^{(2)} & \theta_{2,2}^{(2)} & \theta_{2,3}^{(2)} \end{bmatrix}$$

$$a_1^{(3)} = g(\theta_{1,0}^{(2)} b + \theta_{1,1}^{(2)} a_1^{(2)} + \theta_{1,2}^{(2)} a_2^{(2)} + \theta_{1,3}^{(2)} a_3^{(2)})$$

**Solución final:**  $h_\theta(x) = a_1^{(3)}$

# Función de coste en NN

## Problemas de clasificación



La solución se presenta en un vector:

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$$

$m$  = Número ejemplos en el dataset

$L$  = Número total de capas

$s_l$  = Número de neuronas (sin contar el sesgo) en cada capa

### Clasificación binaria

$$y = 0 \text{ or } 1$$

1 unidad de salida

### Clasificación multiclas

$$y \in \mathbb{R}^K \quad \text{E.g. } \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Peatón      coche      moto      camión

$K$  unidades de salida

# Función de coste en NN

## Formulación matemática



$$h_{\Theta}(x) \in \mathbb{R}^K \quad (h_{\Theta}(x))_i = i^{th} \text{ output}$$

$$J(\Theta) = -\frac{1}{m} \left[ \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_{\Theta}(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_{\Theta}(x^{(i)}))_k) \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2$$

Nuestro objetivo

$$\min_{\Theta} J(\Theta)$$

Necesitamos calcular

$$J(\Theta) \quad \frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$$

# Backpropagation

## Intuición



- Objetivo: Conocer la culpa que tiene cada neurona en el error final
- Al llegar a la primera capa, tendremos el error para cada neurona
- Esto nos permite autoajustar los parámetros iniciales, consiguiendo el entrenamiento de la red

# Backpropagation

## Intuición



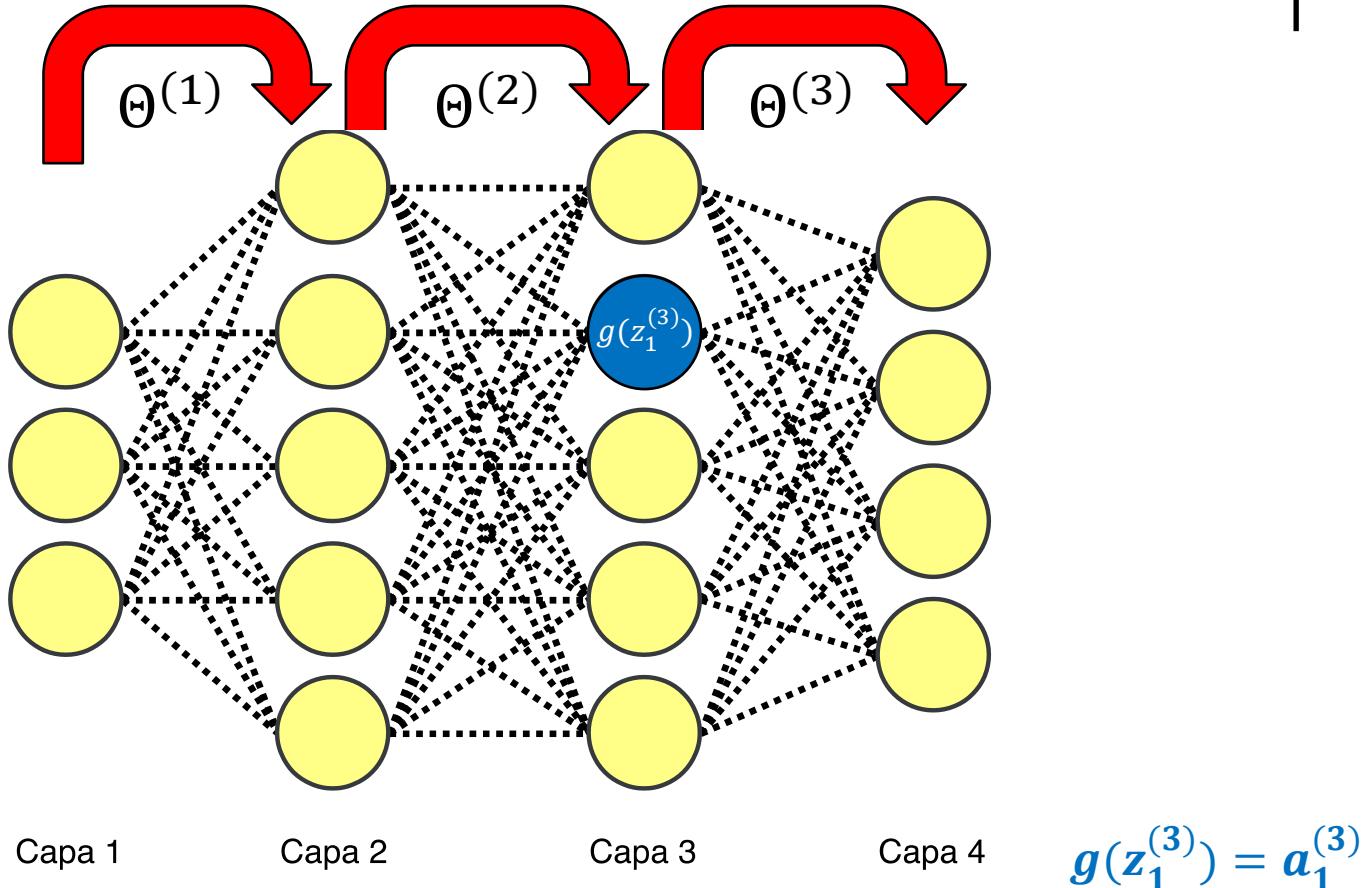
- ¿Cómo se hace? Usando los errores para calcular las derivadas parciales y así formar el vector gradiente

$$\frac{\partial J(\theta)}{\partial \Theta} = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \Theta^{(1)}} \\ \frac{\partial J(\theta)}{\partial \Theta^{(2)}} \\ \vdots \end{bmatrix}$$

- ¿Lo siguiente? Optimizar los parámetros con el algoritmo del descenso del gradiente, minimizando la función de coste

# Backpropagation

## Intuición



$$z_1^{(3)} = \theta_{1,0}^{(2)} * 1 + \theta_{1,1}^{(2)} * a_1^{(2)} + \theta_{1,2}^{(2)} * a_2^{(2)} + \theta_{1,3}^{(2)} * a_3^{(2)} + \theta_{1,4}^{(2)} * a_4^{(2)}$$

# Backpropagation

## Formulación



$$\frac{\partial J(\theta)}{\partial \Theta_{i,j}^l} = \frac{\partial z_i^{l+1}}{\partial \Theta_{i,j}^l} \cdot \frac{\partial J(\theta)}{\partial z_i^{l+1}} = a_j^l \cdot \frac{\partial J(\theta)}{\partial z_i^{l+1}} = a_j^l \cdot \delta_i^l$$

Necesitamos calcular  $\delta^l$   $l = L, L - 1, \dots, 2$

Se necesita calcular la función de coste y el gradiente para llamar a las funciones de optimización avanzada disponibles en Python

# Backpropagation

## Formulación



Para cada unidad de salida (capas  $L = 4$ )

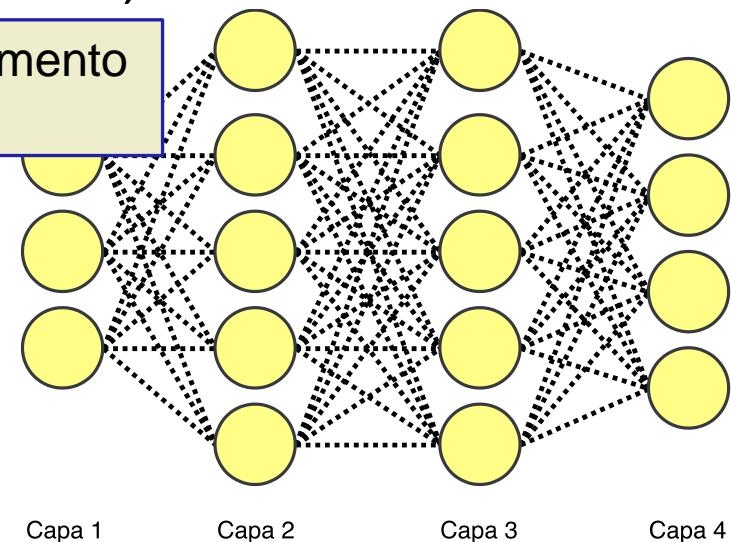
$$\delta_j^{(4)} = a_j^{(4)} - y_j$$

Producto elemento a elemento  
np.multiply(--,--)

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} . * g'(z^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} . * g'(z^{(2)})$$

(No  $\delta^{(1)}$ )



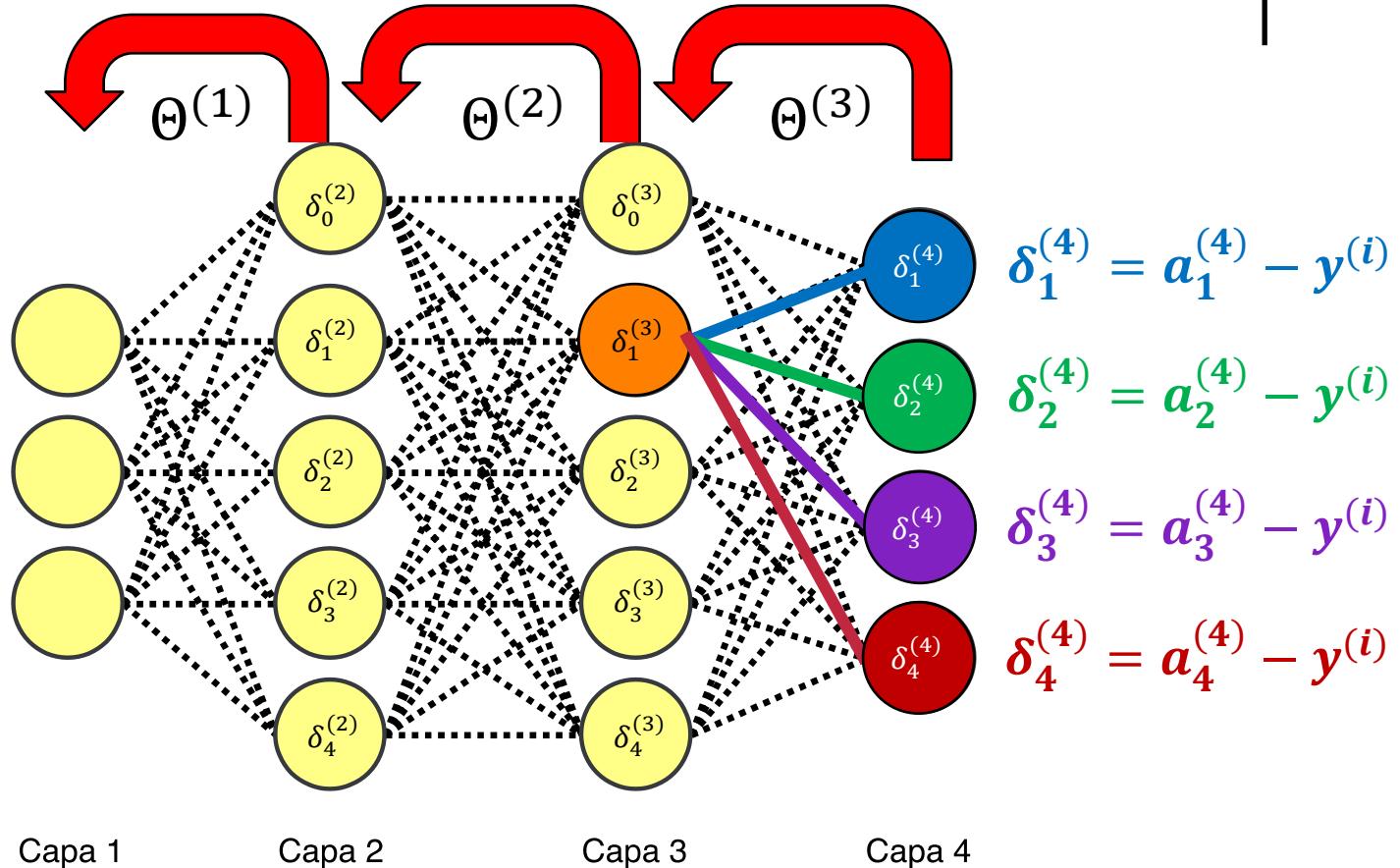
$g'(z^{(3)}) = \text{np.multiply}(a^{(3)}, 1-a^{(3)})$   
 $g'(z^{(2)}) = \text{np.multiply}(a^{(2)}, 1-a^{(2)})$

**Interpretación:**

$\delta_j^{(l)}$  = “error” del nodo  $j$  en la capa  $l$

# Backpropagation

## Intuición



$$\delta_1^{(3)} = (\Theta_{2,1}^{(3)} * \delta_1^{(4)} + \Theta_{2,2}^{(3)} * \delta_2^{(4)} + \Theta_{2,3}^{(3)} * \delta_3^{(4)} + \Theta_{2,4}^{(3)} * \delta_4^{(4)}) * g'(z^{(3)})$$

# Backpropagation

## Pseudocódigo



Conjunto de entrenamiento  $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

Iniciar  $\Delta_{ij}^{(l)} = 0$  para todos  $l, i, j$

Para  $i = 1$  to  $m$

  Iniciarizar  $a^{(1)} = x^{(i)}$

    Forward-propagation para calcular  $a^{(l)}$  para  $l = 2, 3, \dots, L$

    Usando  $y^{(i)}$ , calcular  $\delta^{(L)} = a^{(L)} - y^{(i)}$

    Calcular  $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

fin

$$D_{ij}^{(l)} := \frac{1}{m} \left[ \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} \right] \text{ if } j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \quad \text{if } j = 0$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

# Implementación en Python unroll



- Optimización avanzada

```
def costFunction(theta, ...)    def gradFunction(theta, ...)  
...  
...  
return jVal           vector n+1           return gradient  
  
opTheta=fmin_cg(... ,f=costFunction,fprime=gradFunction,...)
```

Ej. En una red con 4 capas ( $L=4$ ):

$\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$  - matrices (`Theta1, Theta2, Theta3`)  
 $D^{(1)}, D^{(2)}, D^{(3)}$  - matrices (`D1, D2, D3`)

Hay que “Desenrollar” las matrices en vectores

# Implementación en Python unroll

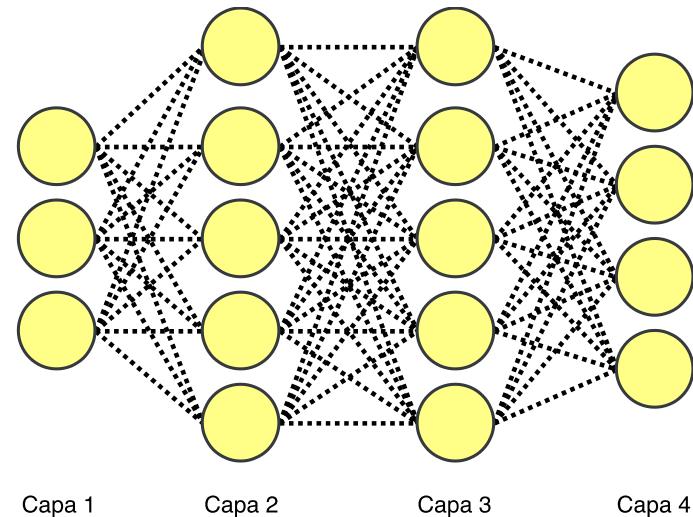


- Ejemplo:

$$l_1 = 3 ; l_2 = 5 ; l_3 = 5 ; l_4 = 4$$

$$\theta^{(1)} \in \mathbb{R}^{5 \times 4} ; \theta^{(2)} \in \mathbb{R}^{5 \times 6} ; \theta^{(3)} \in \mathbb{R}^{4 \times 6}$$

$$D^{(1)} \in \mathbb{R}^{5 \times 4} ; D^{(2)} \in \mathbb{R}^{5 \times 6} ; D^{(3)} \in \mathbb{R}^{4 \times 6}$$



```
thetaVec = np.hstack(Theta1.ravel(order='F') ,  
Theta2.ravel(order='F') , Theta3.ravel(order='F'))  
DVec = np.hstack(---,---,---)
```

```
Theta1 = reshape(thetaVec(1:20),5,4,'F')  
Theta2 = reshape(thetaVec(21:51),5,6,'F')  
Theta3 = reshape(thetaVec(52:76),4,6,'F')
```

# Implementación en Python

## unroll



Tenemos los parámetros iniciales ( $L = 4$ )

```
# Desenrollar para tener vector
initialTheta = np.hstack(θ(1).ravel(order='F'),
θ(2).ravel(order='F'), θ(3).ravel(order='F'))

# Calcular parámetros Theta óptimos
ThetaOptVec = fmin_cg(..., f=costFunction, fprime=gradFunction, ...)

# Enrollar usando reshape
De ThetaOptVec sacar Theta1, Theta2, Theta3
Predecir usando Theta1, Theta2 y Theta3
```

# Implementación en Python unroll



```
def costFunction(thetaVec):
```

De thetaVec obtener  $\Theta^{(1)}$ ,  $\Theta^{(2)}$ ,  $\Theta^{(3)}$  # usar reshape

Usar forward-prop para calcular  $J(\Theta)$

```
return jval
```

```
def gradtFunction(thetaVec):
```

De thetaVec obtener  $\Theta^{(1)}$ ,  $\Theta^{(2)}$ ,  $\Theta^{(3)}$  # usar reshape

Usar back-prop para calcular  $D^{(1)}$ ,  $D^{(2)}$ ,  $D^{(3)}$

# Desenrollar para obtener un vector

```
gradientVec = np.hstack(D(1).ravel(order='F'),  
D(2).ravel(order='F'), D(3).ravel(order='F'))
```

```
return gradientVec
```

# Implementación en Python

## Función de coste y gradiente



Ej: Red con 3 capas sin aplicar regularización

Forward propagation para una instancia del dataset

```
def forward(Theta1,Theta2,X,i):
    # bias + neuronas de la capa 1
    a1 = np.hstack((np.ones(1), X[i]))
    z2 = Theta1 @ a1
    a2 = sigmoid(z2)
    # bias + neuronas de la capa 2
    a2 = np.hstack((np.ones(1), a2))
    z3 = Theta2 @ a2
    a3 = sigmoid(z3) # Salida de la capa 3
    return a1, a2, a3
```

# Implementación en Python

## Comprobación del gradiente



¿Cómo sabemos si el gradiente D1, D2,... está bien calculado?

$$\frac{\nabla J(q)}{\|q\|} @ \frac{J(q + e) - J(q - e)}{2e}$$

Comprobar que `gradApprox ≈ gradientVec`

# Implementación en Python

## Comprobación del gradiente



- Implementar back-prop para calcular gradientVec
- Implementar numerical gradient check para calcular gradApprox
- Asegurarse que son similares
- Desactivar comprobación del gradiente y usar el algoritmo de back-prop para el entrenamiento

Es importante desactivar la comprobación porque el algoritmo se volvería muy lento

# Implementación en Python

## Inicialización



### Valor inicial de $\Theta$

Se necesita un valor inicial de  $\Theta$  para el descenso del gradiente y el método de optimización avanzado

```
ThetaOptVec = fmin_cg(..., x0=initialTheta,...)
```

¿Debemos iniciar Theta a zeros(...)? ¡NO!  
los pesos serían los mismos en todas las neuronas y capas

- Malos resultados de la red neuronal
- Todas las neuronas codifican la misma información
- Representaciones redundantes

# Implementación en Python

## Inicialización



### Inicialización aleatoria: Rompiendo la simetría

Iniciar cada  $\Theta_{ij}^{(l)}$  a un valor aleatorio entre  $[-\epsilon, \epsilon]$

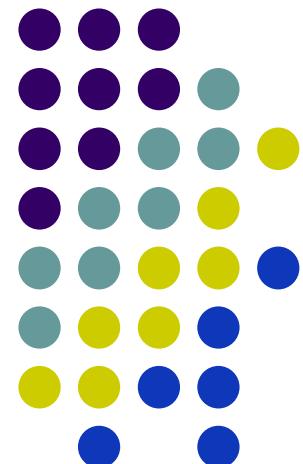
```
Theta1 = np.random.rand(10,11) * (2*INIT_EPS) - INIT_EPS;
```

```
Theta2 = np.random.rand(1,11) * (2*INIT_EPS) - INIT_EPS;
```

# Clustering

---

**Inteligencia Artificial  
Ingeniería Informática  
en Sistemas de Información**





# Contenido

---

- Clustering: Conceptos
- K-Means
  - Representación del modelo
  - Función coste
  - Inicialización de los centroides
  - Elegir K

# Clustering: Conceptos

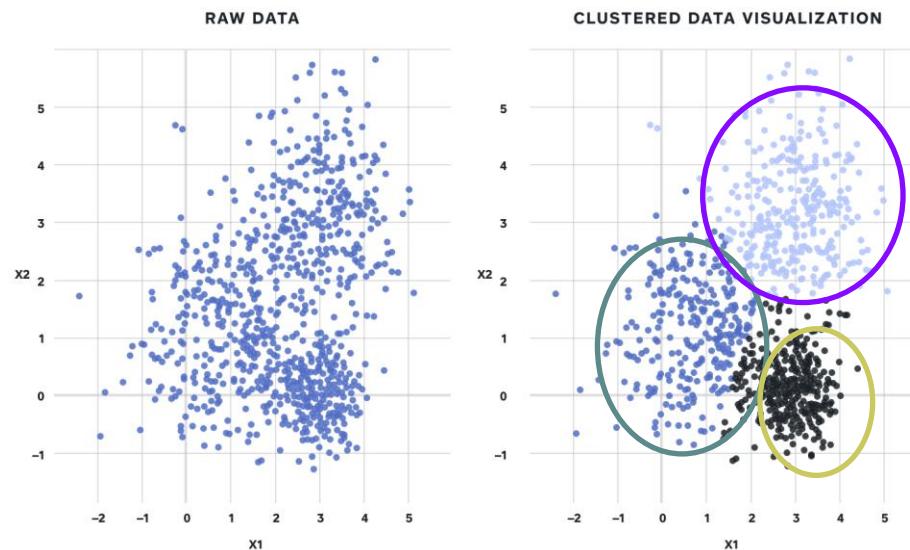


- Aprendizaje no supervisado:
  - Sirve para explorar las propiedades de los datos examinados, no para predecir nuevos datos. Es decir, se usa para entender y resumir los datos.
  - El dataset no tendrá atributo objetivo o clase ( $y$ ) como en los problemas de tipo aprendizaje supervisado.

# Clustering: Conceptos



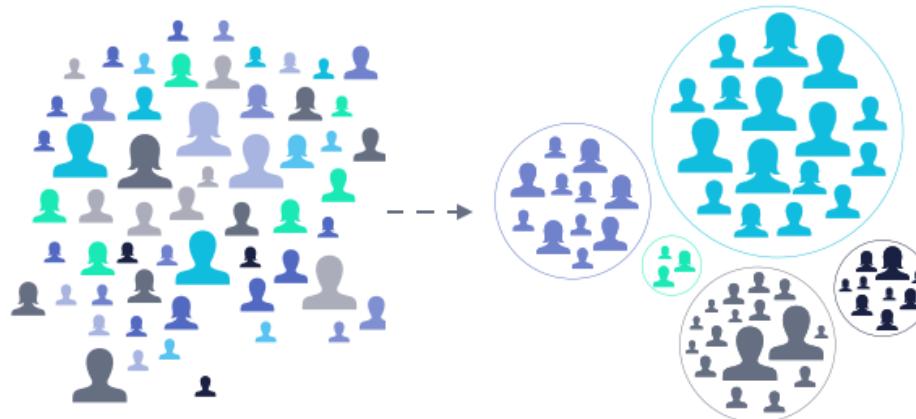
- Clustering:
  - Agrupación de un conjunto de datos (no etiquetados) en grupos de objetos llamados cluster.
  - Cada cluster está formado por una colección de objetos que son similares entre sí, pero que son distintos respecto a los objetos de otros clusters.



# Clustering: Conceptos



- Clustering:
  - Ejemplo: Segmentación del mercado/clientes:
    - Personalizar la publicidad, el contenido de tu página web, los mails: basándote en criterios de comportamiento, geográficos, psicográficos, demográficos, etc. de cada cliente.





# K-Means:

## Representación del modelo

- Es uno de los algoritmos de clustering más famosos y usados.
- Notación:

$m$  = número de ejemplos / instancias del conjunto de entrenamiento (training)

$n$  = número de atributos / variables / características

**Training** :  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

$x \in \mathbb{R}^n \rightarrow$  no vamos a añadir la columna  $x_0$

$K$  = número de clusters

**Centroides** :  $\mu_k$  Representan el centro del cluster

$c^{(i)}$ : Índice (de 1 a K) del centroide más cercano a  $x^{(i)}$

$\mu_{c^{(i)}}$ : Centroide del cluster al que se ha asignado  $x^{(i)}$

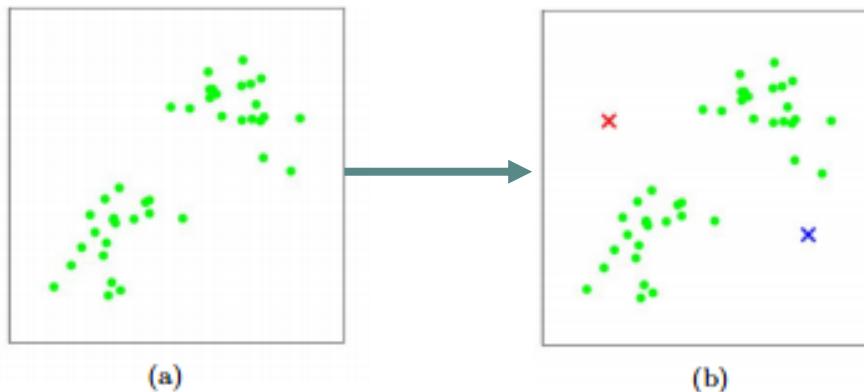


# K-Means: Representación del modelo

- Algoritmo:

- Ejemplo: K=2
  - **Paso 1:**

Inicializar de forma random K centroides

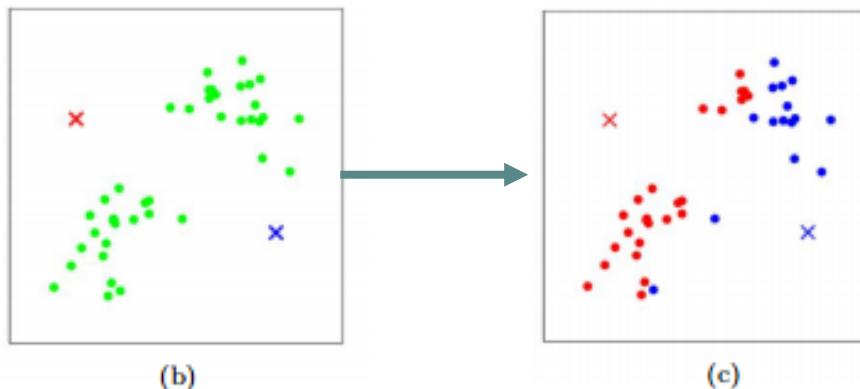




# K-Means: Representación del modelo

- Algoritmo:

- Ejemplo: K=2
  - **Paso 2:**



Inicializar de forma random K centroides

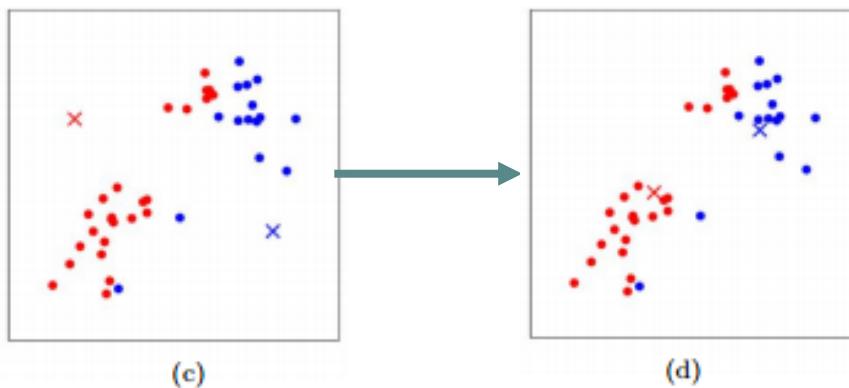
Asignar cada instancia del training a su centroide más cercano



# K-Means: Representación del modelo

- Algoritmo:

- Ejemplo: K=2
  - **Paso 3:**



Inicializar de forma random K centroides

Asignar cada instancia del training a su centroide más cercano

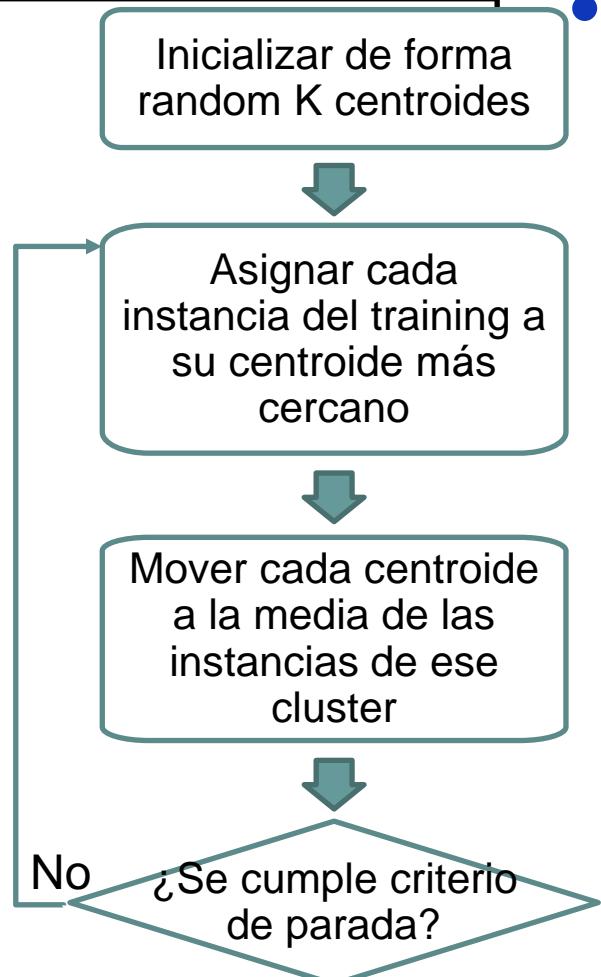
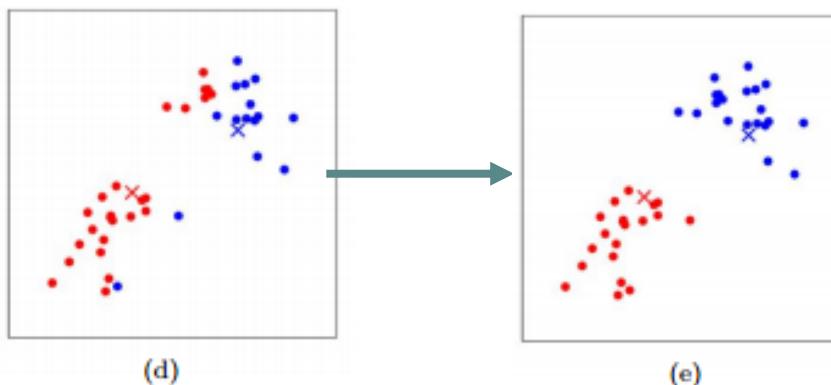
Mover cada centroide a la media de las instancias de ese cluster



# K-Means: Representación del modelo

- Algoritmo:

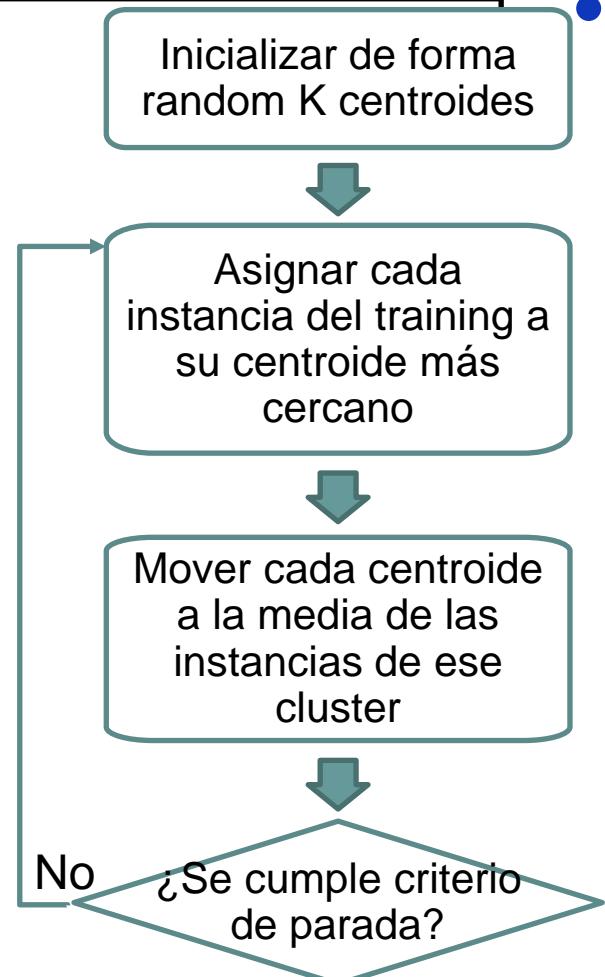
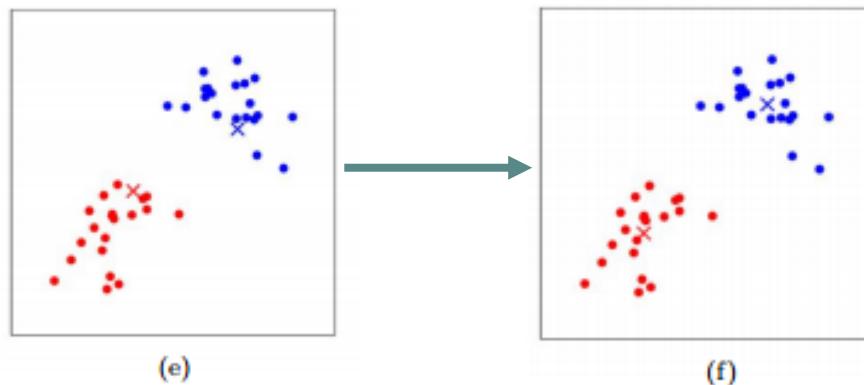
- Ejemplo: K=2
  - **Paso 4:** Suponemos que no se cumple el criterio de parada y volvemos al **Paso 2**





# K-Means: Representación del modelo

- Algoritmo:
  - Ejemplo: K=2
    - Repetimos el **Paso 3**

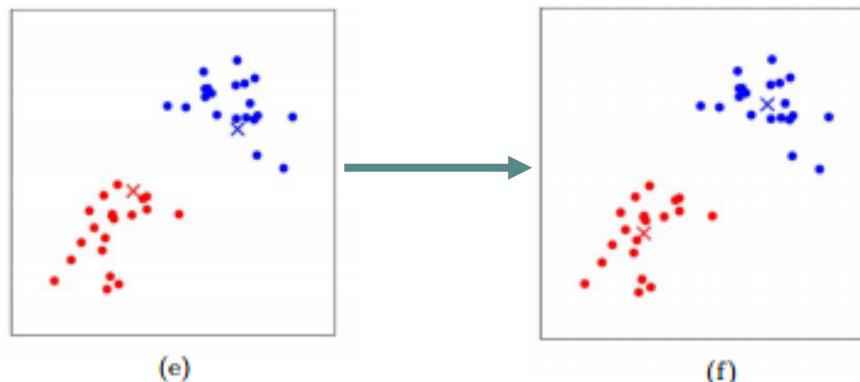




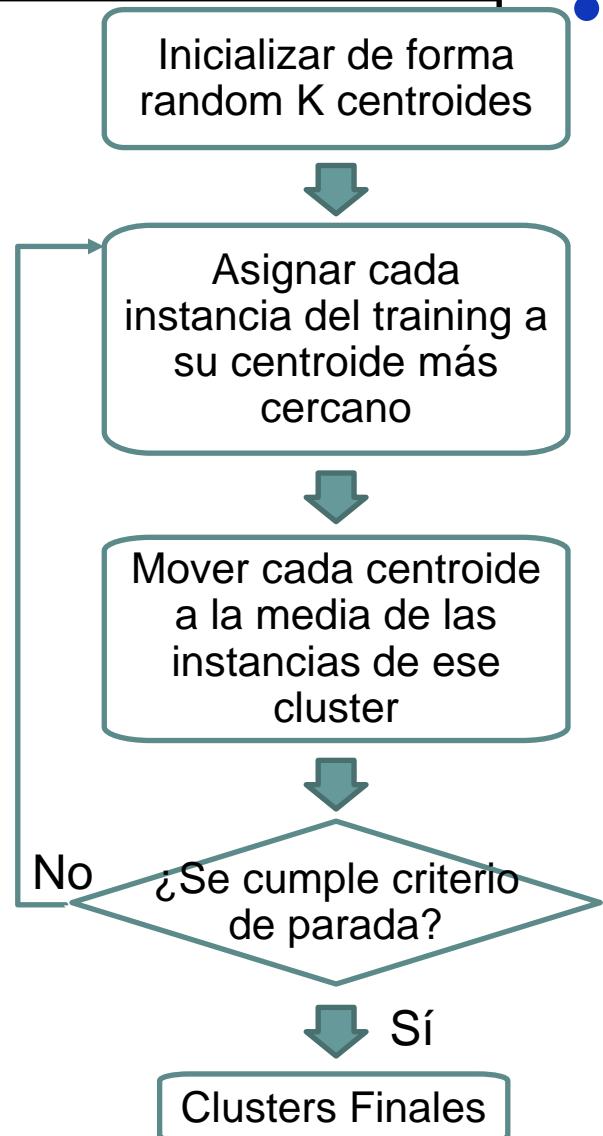
# K-Means: Representación del modelo

- Algoritmo:

- Ejemplo: K=2
  - Repetimos el **Paso 4**: Suponemos que se cumple el criterio de parada y finaliza el algoritmo.



- Los clusters de la gráfica (f) son los clusters finales.

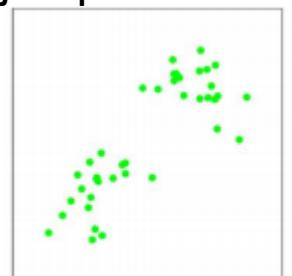




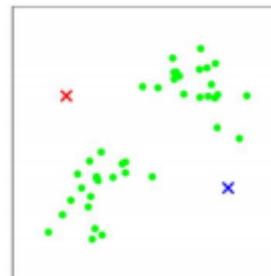
# K-Means: Representación del modelo

## Algoritmo:

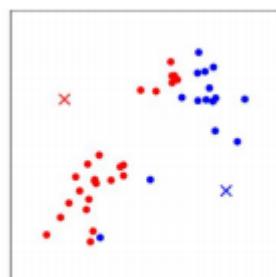
- Ejemplo: K=2 → Compactado



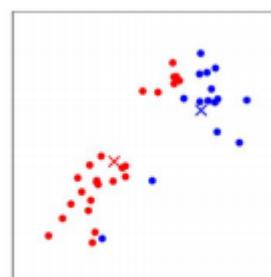
(a)



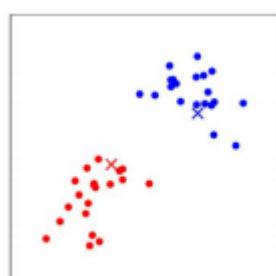
(b)



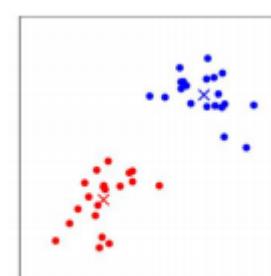
(c)



(d)



(e)



(f)

Inicializar de forma random K centroides



Asignar cada instancia del training a su centroide más cercano



Mover cada centroide a la media de las instancias de ese cluster



No

¿Se cumple criterio de parada?



Clusters Finales



# K-Means: Representación del modelo

- Pseudocódigo:

Iniciar de forma random K centroides  $\mu_1, \mu_2, \dots, \mu_k \in \mathbb{R}^n$

Repetir hasta que se cumpla el criterio de parada (se alcanza la “convergencia”) {

Para cada  $i$  in range(1, m):

$c^{(i)} :=$  Índice (de 1 a K) del centroide  
más cercano a  $x^{(i)}$

Para cada  $k$  in range(1, K):

$\mu_k :=$  Mover centroide a la media de  
los datos del cluster k

}

}

Paso: Asignar cada  
instancia del training a su  
centroide más cercano

}

Paso: Mover cada  
centroide a la media de  
las instancias de ese  
cluster

Criterio de parada: [ Opción 1: se alcanzan un número de iteraciones indicado.

[ Opción 2: los datos pertenecientes a cada cluster (y por tanto sus  
centroides) no cambian.



# K-Means: Función coste

- Buscar los parámetros  $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$  que minimizan la función coste:

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$$\begin{aligned} & \min_{\substack{c^{(1)}, \dots, c^{(m)} \\ \mu_1, \dots, \mu_K}} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) \end{aligned}$$



# K-Means: Función coste

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

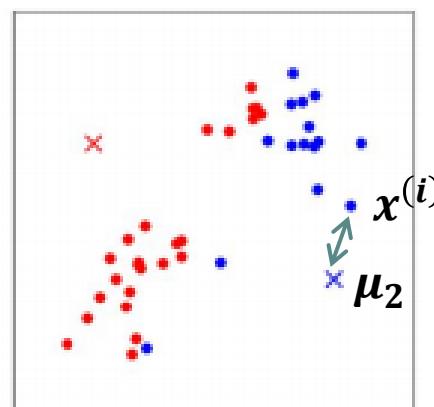
$$\min_{\substack{c^{(1)}, \dots, c^{(m)} \\ \mu_1, \dots, \mu_K}} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

- Ejemplo gráfico:

$x^{(i)} \rightarrow$  Asociado al cluster 2

$$c^{(i)} = 2$$

$$\mu_{c^{(i)}} = \mu_2$$



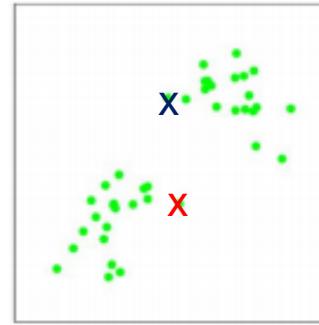
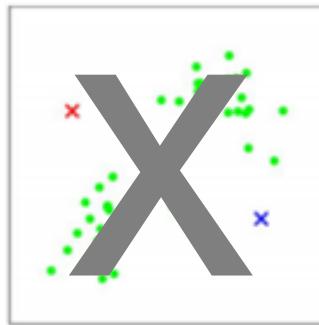
**Minimizar  
la distancia**

(c)

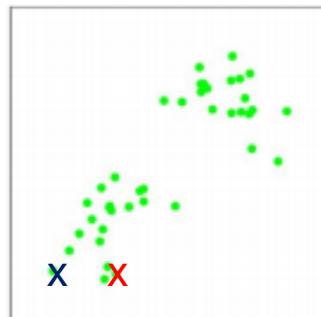


# K-Means: Inicialización de los centroides

- Se inicializan los centroides ( $\mu_1, \mu_2, \dots, \mu_k$ ) a  $K$  instancias random del conjunto de training.



- Esa inicialización puede ser más o menos afortunada, incluso llegando a alcanzar óptimos locales, es decir, no se conseguiría minimizar correctamente la función coste.





# K-Means: Inicialización de los centroides

- Para evitar encontrar mínimos locales:
  - 1) Se ejecuta *varias veces* el algoritmo de inicialización y el propio algoritmo K-means para intentar encontrar el óptimo global.
  - 2) Para cada una de esas ejecuciones se calcula la función coste.
  - 3) Al finalizar se seleccionará la opción de clustering con menor coste.



# K-Means: Inicialización de los centroides

- Pseudocódigo:

Para  $i$  in range(1, 100):

Iniciar los K centroides con instancias random del training

Ejecutar K-Means: Obtener  $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$

Calcular la función coste:  $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

Seleccionar el clustering con menor coste :  $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$



# K-Means: Elegir K

- Manualmente: visualización de datos, observando la salida de ejecuciones de K-Means con distintos K,.... En algunas ocasiones será una selección muy ambigua.
- Algoritmo Elbow: Encontrar el punto a partir del cual el coste disminuye muy poco al aumentar K. En algunas ocasiones será una selección muy ambigua.

