Inteligencia Artificial

Aprendizaje basado en ejemplos.

In whitch we describe agents that can improve their behavior through diligent study of their own experiences.

¿Porqué queremos que un agente aprenda?

Si es posible un mejor diseño, **¿porqué no lo diseñamos mejor desde el principio?**

Aprendizaje basado en ejemplos.

In whitch we describe agents that can improve their behavior through diligent study of their own experiences.

¿Porqué queremos que un agente aprenda?

Si es posible un mejor diseño, **¿porqué no lo diseñamos mejor desde el principio?**

1. No es posible anticipar todas las situaciones
2. No se puede anticipar todos los cambios
3. El programador no tiene ni idea de como programar una solución

Componentes a aprender.

1. Mapeo de condiciones de un estado a acciones
2. Inferir propiedades relevantes del mundo desde la secuencia de percepciones
3. Como impactan las acciones
4. Utilidad de los estados
5. Información acerca de las preferencias de las acciones

Representación del conocimiento.

- **Representación factorizada** como entradas (un vector de atributos)
- **Aprendizaje inductivo** es aprender una función general desde casos específicos.

Feedback. ¿es necesario?

Feedback. ¿es necesario?

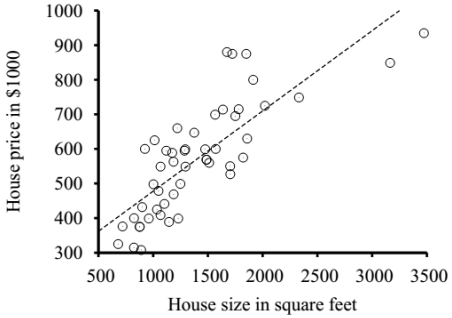
3 tipos de feedback:

1. **Aprendizaje no supervisado:** aprende sin feedback. Lo más común son algoritmos de **clustering**
2. **Aprendizaje por refuerzo:** el agente aprende a través de una serie de refuerzos (recompensas o castigos)
3. **Aprendizaje supervisado:** el agente cuenta con entradas y las salidas esperadas y aprende una función para realizar el mapeo.

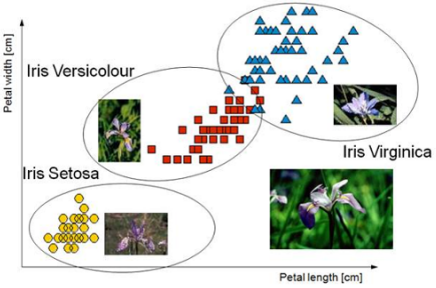
Aprendizaje semi-supervisado es un gris entre 1 y 3. Se debe al ruido o a la falta de etiquetas.

Aprendizaje supervisado.

- Dado un **Conjunto de entrenamiento** *(x1, y1), (x2, y2),..., (xn, yn)* donde cada *yj* es generado mediante **y = f(x)**, donde *f* es desconocida; se trata de encontrar un *h(x)* que aproxime a *f*.



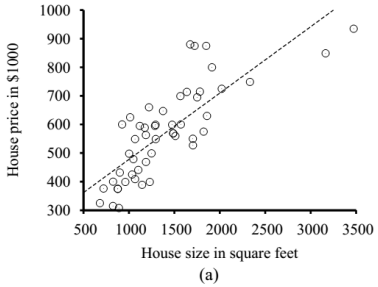
(a)



(b)

- *x* e *y* pueden adoptar cualquier tipo de datos, *xj* es un vector de atributos
- **h** es una **hipótesis**. El aprendizaje consiste en buscar una hipótesis que se ajuste bien a los datos.
- Un **conjunto de test** es usado para medir la **precisión** de una hipótesis.
- Decimos que una hipótesis **generaliza** bien, cuando predice correctamente con entradas no conocidas.
- Cuando la salida es un valor de un conjunto finito, el problema es llamado **clasificación**.
- Cuando la salida es un número el problema es llamado **regresión**.

Regresión lineal univariada.



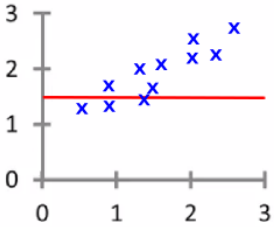
- Tiene la forma de una recta: $y = w1\ x + w0$
- usamos *w* porque vamos a pensar los coeficientes como pesos (weights)
- vamos a usar *w* y Theta indistintamente
- **w** es el vector definido por [w0, w1]
- hw(x) = w1 x + w0

- Se conoce como regresión a la tarea de encontrar un h que minimice el **error** o **costo**

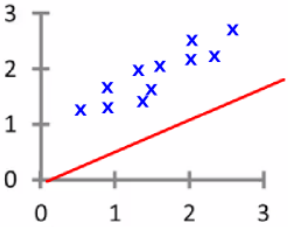
Función de costo.

- ¿Como elegir **W**?

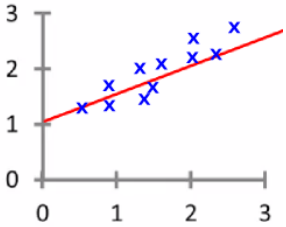
$$h_\theta(x) = \theta_0 + \theta_1 x$$



$\theta_0 = 1.5$
$\theta_1 = 0$



$\theta_0 = 0$
$\theta_1 = 0.5$



$\theta_0 = 1$
$\theta_1 = 0.5$

Función de costo.

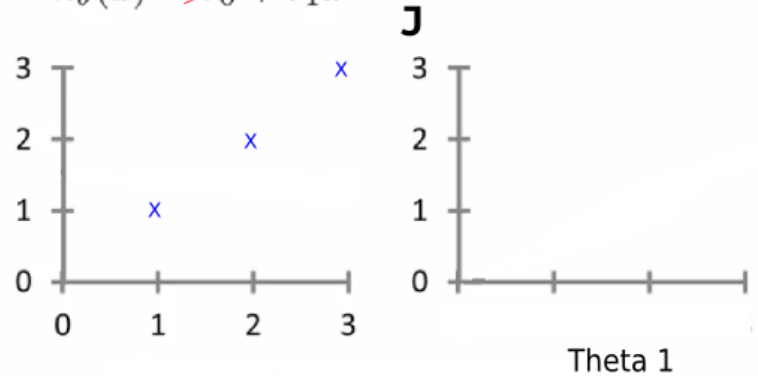
Error cuadrático medio

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

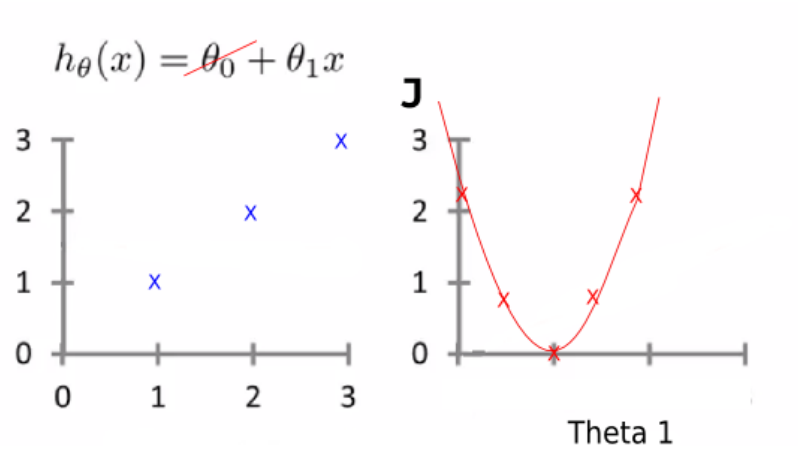
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Función de costo.

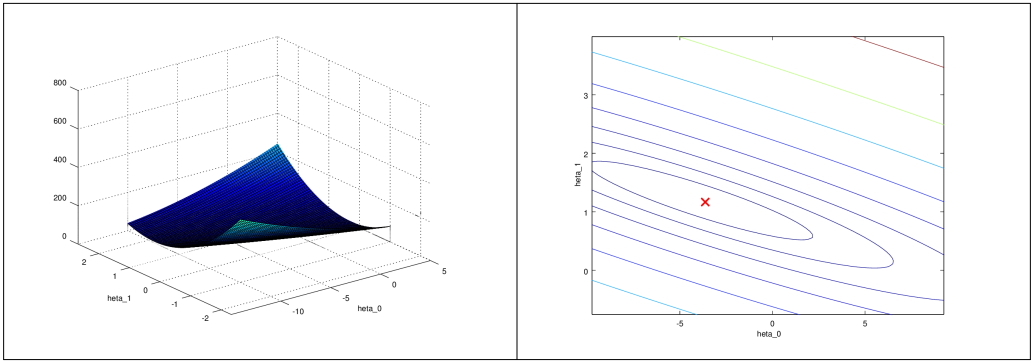
$$h_\theta(x) = \theta_0 + \theta_1 x$$



Función de costo.



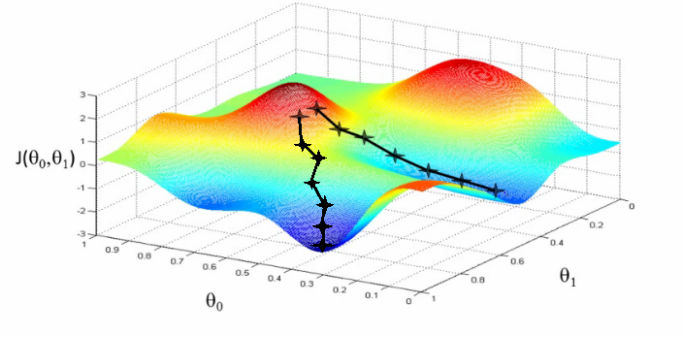
Función de costo.



- La función de costo es estrictamente convexa, con lo cual, solo tiene un mínimo global

Descenso por el gradiente (GD).

- Empezamos con valores aleatorios para los parámetros
- Vamos adaptando los parámetros tratando de reducir hasta encontrar un mínimo.



Descenso por el gradiente (GD).

repeat until convergence {

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

}

- esta regla de actualización se aplica sobre todos los thetas simultáneamente
- alpha se denomina **learning rate**.
  - Si es demasiado chico, GD puede ser lento
  - Si es demasiado grande, GD puede no converger o diverger

Desarrollo:

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

Desarrollo:

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

$$\text{repeat until convergence } \{$$
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$
$$\}$$

- Este algoritmo también se conoce como **Batch gradient descent**: en cada paso se usan **todos** los ejemplos de entrenamiento

Regresión lineal multivariada.

Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Regresión lineal multivariada.

Es la misma idea pero **x** es un vector

$$h_{sw}(\mathbf{x}_j) = w_0 + w_1 x_{j,1} + \cdots + w_n x_{j,n} = w_0 + \sum_i w_i x_{j,i} \, .$$

Si creamos una entrada *xj, 0* que siempre valga 1 podemos expresar **h** como el producto punto o producto de matrices.

$$h_{sw}(\mathbf{x}_j) = \mathbf{w} \cdot \mathbf{x}_j = \mathbf{w}^\top \mathbf{x}_j = \sum_i w_i x_{j,i} \, .$$

La regla de actualización es:

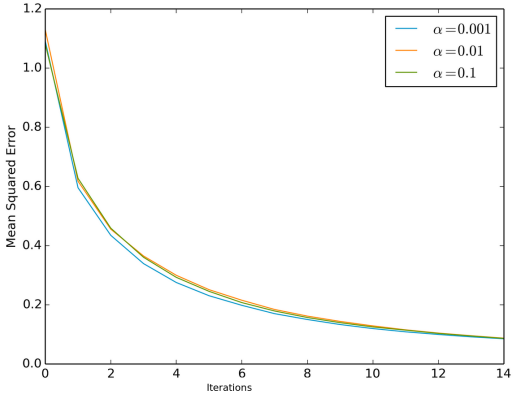
$$w_i \leftarrow w_i + \alpha \sum_j x_{j,i}(y_j - h_{\mathbf{w}}(\mathbf{x}_j)) \, .$$

Feature scaling o normalización de datos.

- Se aplica cuando tenemos multiples atributos de distintas magnitudes
- Hace que gradient descent (y otros algoritmos) converja más rápido
- Una posibilidad es **(Xi - media) / desvio**
- Otra posibilidad para cuando trabajamos en papel es **(Xi - min) / (max - min)**

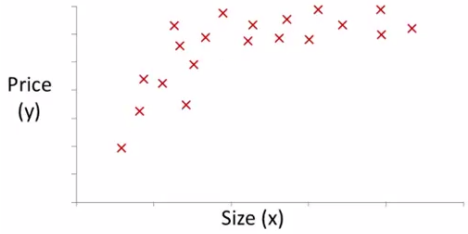
Eligiendo un valor de alpha adecuado.

- Una alternativa es dibujar **Jw** respecto de la cantidad de iteraciones y elegir



Regresion polinómica.

- Puede ser deseable usar como hipótesis un polinomio de grado mayor a 1.



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

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 \\ &= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 \end{aligned}$$

$$\begin{aligned} x_1 &= (size) \\ x_2 &= (size)^2 \end{aligned}$$

Bibliografía y enlaces útiles.

- Russell S., Norvig P.: Artificial Intelligence: A modern Approach. Third Edition. Chapter 18.
- Curso de Machine Learning dictado por Andrew Ng - https://www.coursera.org/learn/machine-learning/