Machine Learning

MCs. Rayner Harold Montes Condori (ICMC - USP)
MCs (c). Jhosimar Arias Figueroa (IC - UNICAMP)

Github del curso: https://github.com/raynerhmc/CursoMachineLearning
ingresen aquí: https://goo.gl/3Gkejk
ray.montes@gmail.com

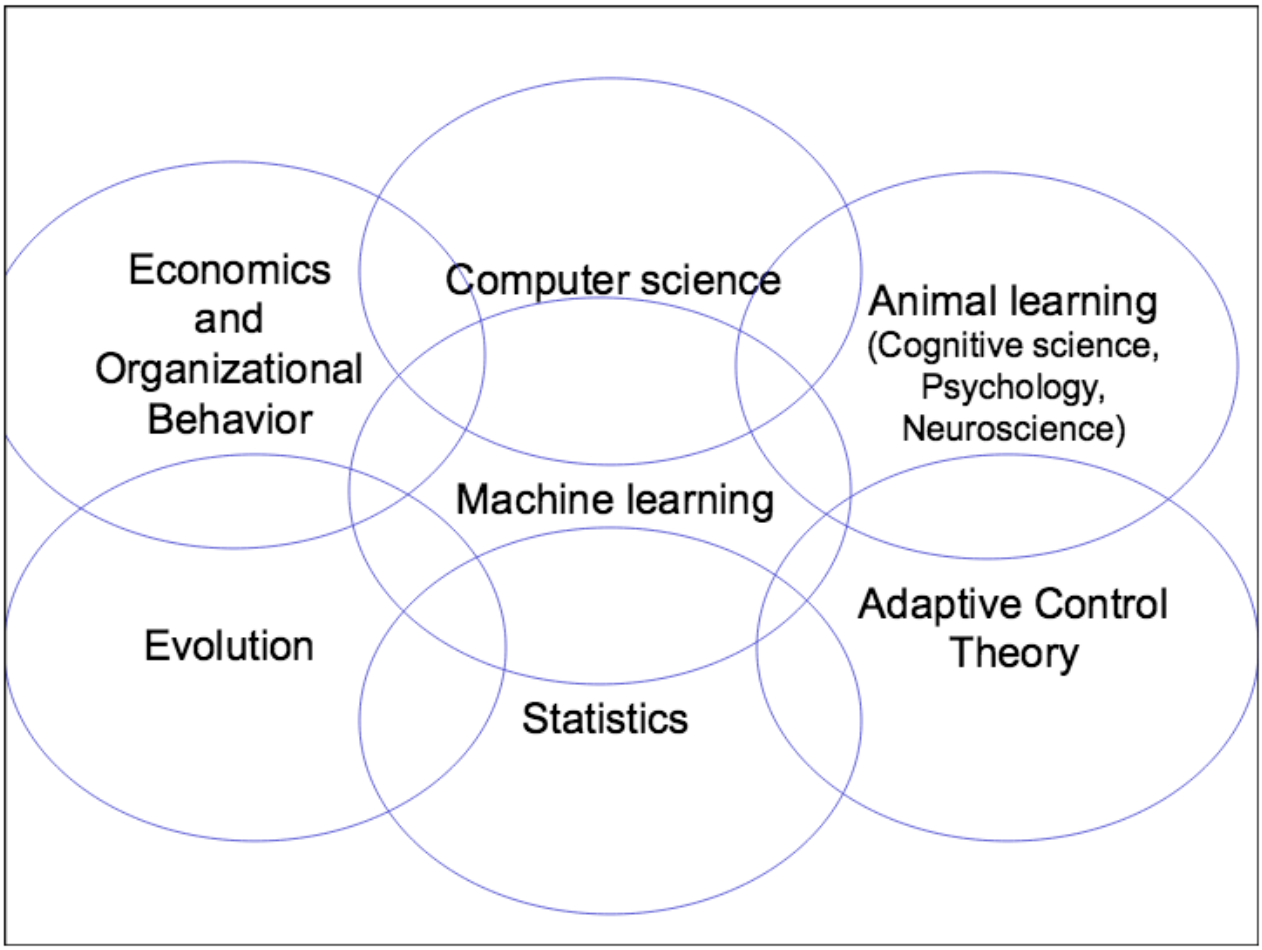
Primera Semana (6 horas de clase)

1. Introducción
2. Regresión Lineal (una variable, múltiples variables)
3. Clasificación y regresión logística
4. Regularización.

[QUIZ 1]

1) Introducción

Surge con la intención de explorar posibilidades de que las computadoras consigan "aprender" sobre diversión asuntos y contextos. El objetivo es resolver problemas específicos sin que hayan sido explícitamente programadas para ello.



Una definición mas formal es dada por Tom Mitchell (Profesor de la Universidad Carnegie Mellon)

> " Se dice que un programa de computador aprende de cierta experiencia ***E***, en relación para alguna clase de tarea**s** ***T*** y una medida de desempeño ***P***, si su éxito en realizar las tareas en ***T***, medidas por ***P***, mejoran con la experiencia ***E***. "

Ejemplo: *Juego de ajedrez*

- ***E:*** La experiencia adquirida a través de varios juegos de ajedrez jugados.
- ***T:*** La tarea de jugar ajedrez.
- ***P:*** La probabilidad que en el programa de computador gane el siguiente juego.

Para pensar: Cuáles serían los parámetros E, T, P en un programa que dirige automáticamente un auto (vehículos autónomos)?

La experiencia ***E*** esta constituida de un conjunto de ejemplos de entrenamiento, cada ejemplo a su vez esta compuesto de una serie de variables (conocidos también como atributos, características o descriptores) que describen alguna propiedad útil para mejorar el desempeño de la tarea ***T***.

Ejemplo: Predicción del cáncer de mama (benigno, maligno) (***T***)
Variables:

- Edad del paciente
- Tamaño del tumor
- Uniformidad del tamaño de las células

- Espesor

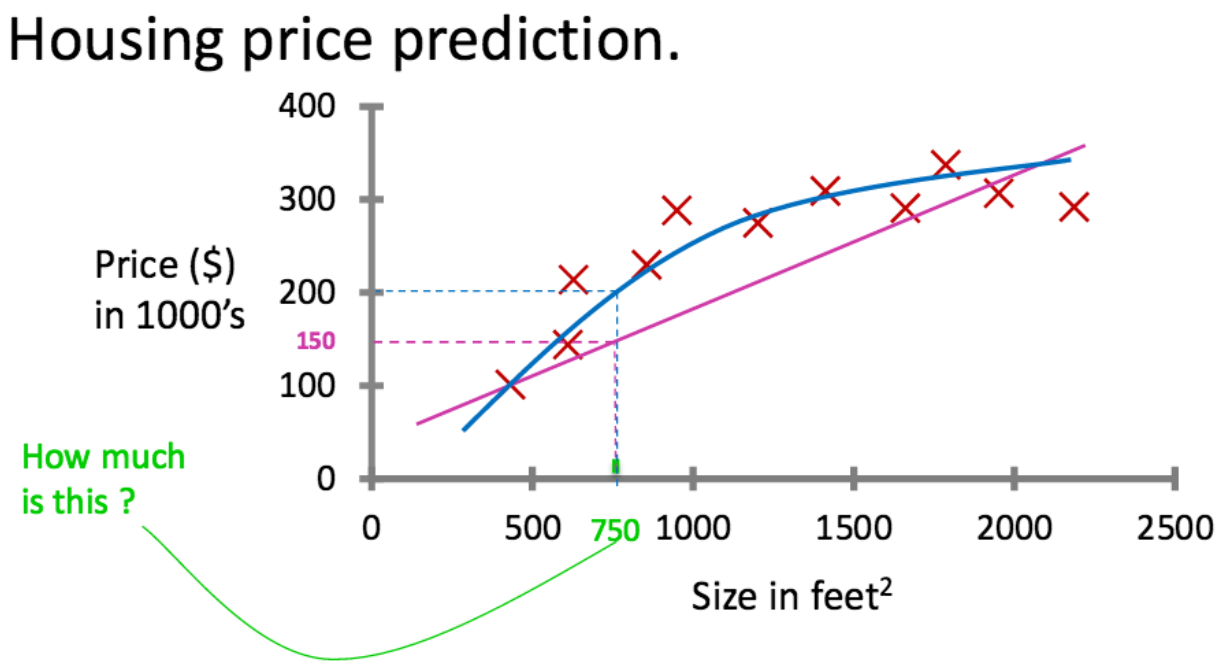
1.1) Tipos de Aprendizaje

Aprendizaje Supervisado

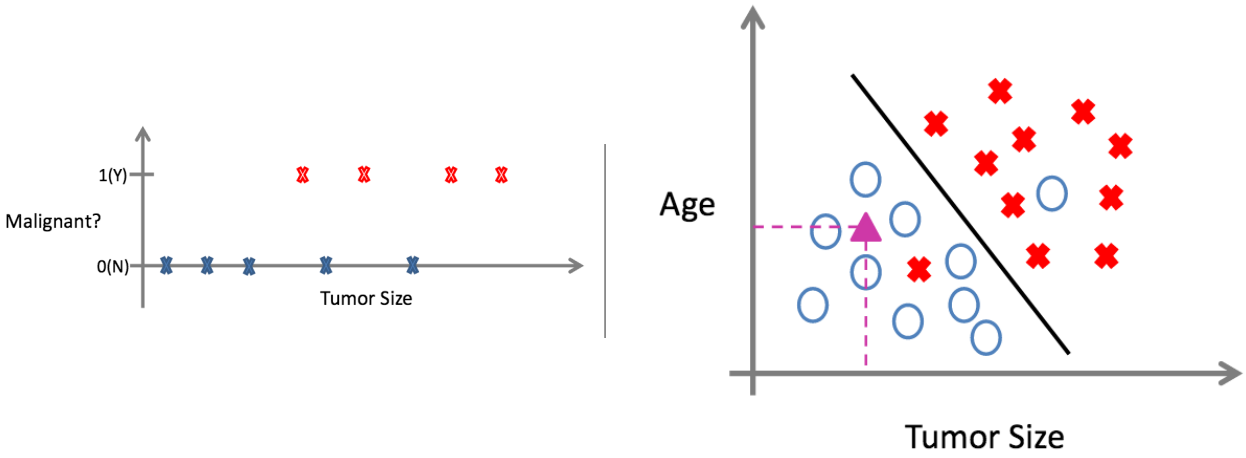
En el aprendizaje supervisado, nosotros tenemos un conjunto de entrenamiento y sabemos el resultado correcto que este debería producir. Dejando en claro la relación directa entre la entrada y la salida. En otras palabras existe un profesor. Dos tipos de problemas en el aprendizaje supervisado son:

Regresión (salidas continuas)	Clasificación (salidas discretas)
• En problemas de regresión, tratamos de predecir los resultados dentro de una salida continua. • Por lo tanto se hace un mapeo de las variables de entrada para una función continua.	• En un problema de clasificación, tratamos de predecir los resultados en una salida discreta. • En otras palabras, estamos tratando de asignar variables de entrada en categorías discretas.

Ejemplo 1.1.1: Predicción del precio de casas.



Ejemplo 1.1.2: Predicción del cáncer de mama. (benigno, maligno)



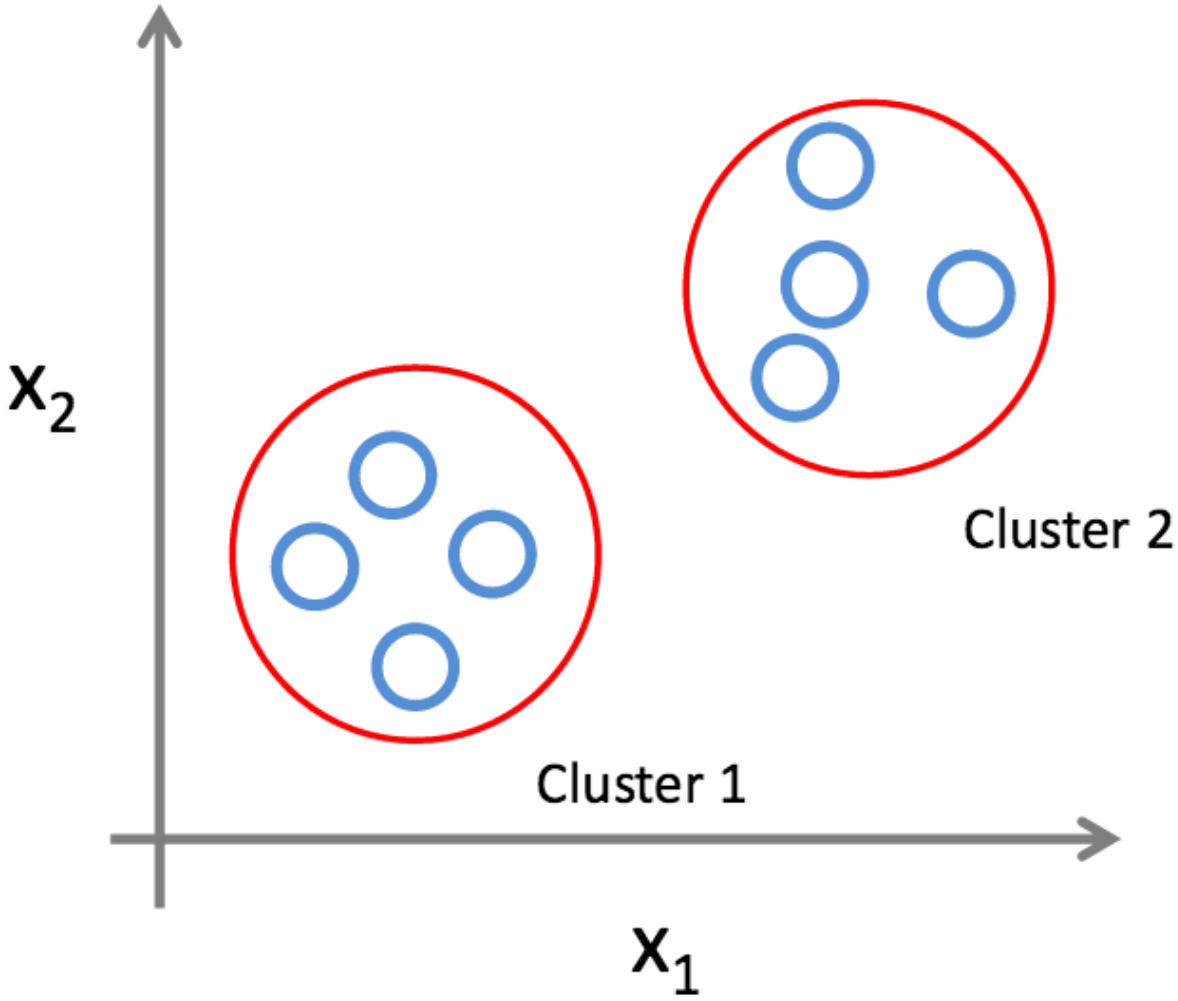
Para pensar: Regresión o clasificación?

1. Tienes un histórico de varios años del volumen (en m^3) del caudal del rio Chili, predecir el volumen por los próximos 3 meses.
2. Dado el texto de una noticia, decidir si ella es una noticia política, social, deportiva o de espectáculos.

Aprendizaje no supervisado

Aprendizaje no supervisado, por otro lado, nos permite acercarnos a los problemas con poca o ninguna idea de lo que nuestros resultados deberían parecer. No existe profesor, por lo tanto, no existe un feedback basado en los resultados predichos.

Podemos obtener esta estructura agrupando los datos en función de las relaciones entre sus variables.



Para pensar: Aprendizaje supervisado o no supervisado.

1. En varias tiendas virtuales, automáticamente agrupar opiniones similares sobre un determinado producto.
2. Agrupar las noticias similares publicadas en diferentes sitios web.
3. Un programa que detecte automáticamente si un email es spam o no.
4. Dada una base de datos de información de clientes, descubrir nuevos segmentos de mercado provechosos.

---

2) Regresión Lineal

En regresión, hablamos de valores de salida continuas.

2.1) Regresión Lineal con una variable

2.1.1) Representación del modelo

Se habla de regresión lineal con una variable cuando *un valor de salida* corresponde a un *valor de entrada (o variable de entrada).*

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

M=47

Notación:

- ***M:*** Número de ejemplos de entrenamiento.
- ***x's:*** Variable de entrada.
- ***y's:*** Variable de salida o variable objetivo.

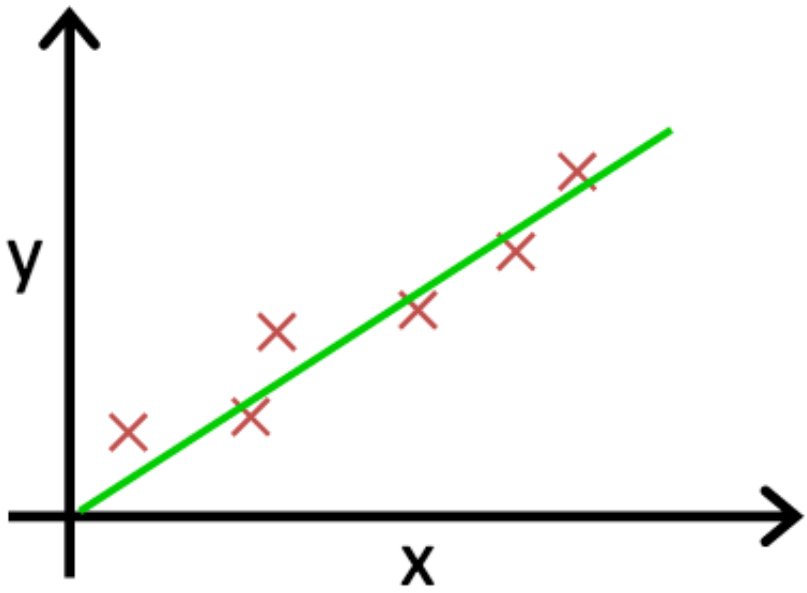
$(x^{(i)}, y^{(i)})$: i-ésimo ejemplo de entrenamiento, comenzando en ***i = 1.***

$(x^{(1)}, y^{(1)})=(2104,460)$

2.1.2 Función hipótesis

Como su propio nombre lo dice, en la regresión linear, la función hipótesis tiende a ajustar los datos para la ecuación de una linea. Esta función predice un valor de salida para cada valor de entrada (*x's*) dada.

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



2.1.3) Función costo (*J*)

Para medir que tan bien los parámetros de la función hipótesis han sido asignados, se calcula el ***error cuadrático promedio,*** el cual, resulta del sumatorio de las diferencias al cuadrado entro los valores ***y's*** de los ejemplos de entrenamiento y el resultado de la función hipótesis de los valores ***x's***.

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

Cuanto menor sea ***J***, mejor ajustado estará la curva a los datos.

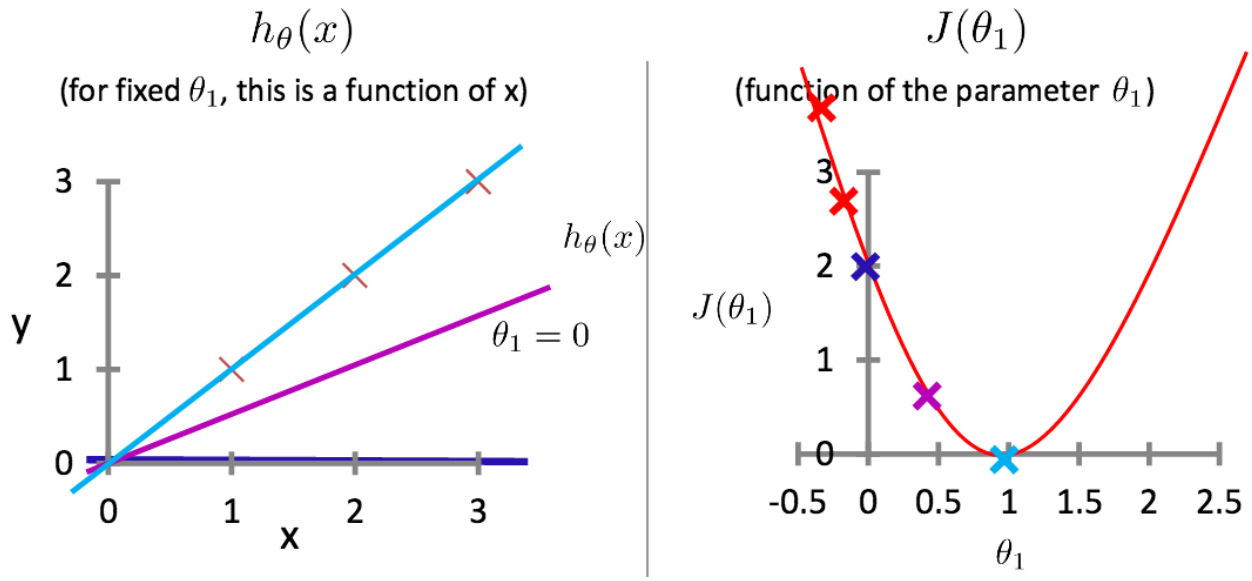
Por lo tanto, el objetivo es minimizar la función costo

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

2.1.4) Intuición de la función *J* para una función hipótesis simplificada

Acontece cuando dejamos que la función hipótesis parta desde el origen de coordenadas.

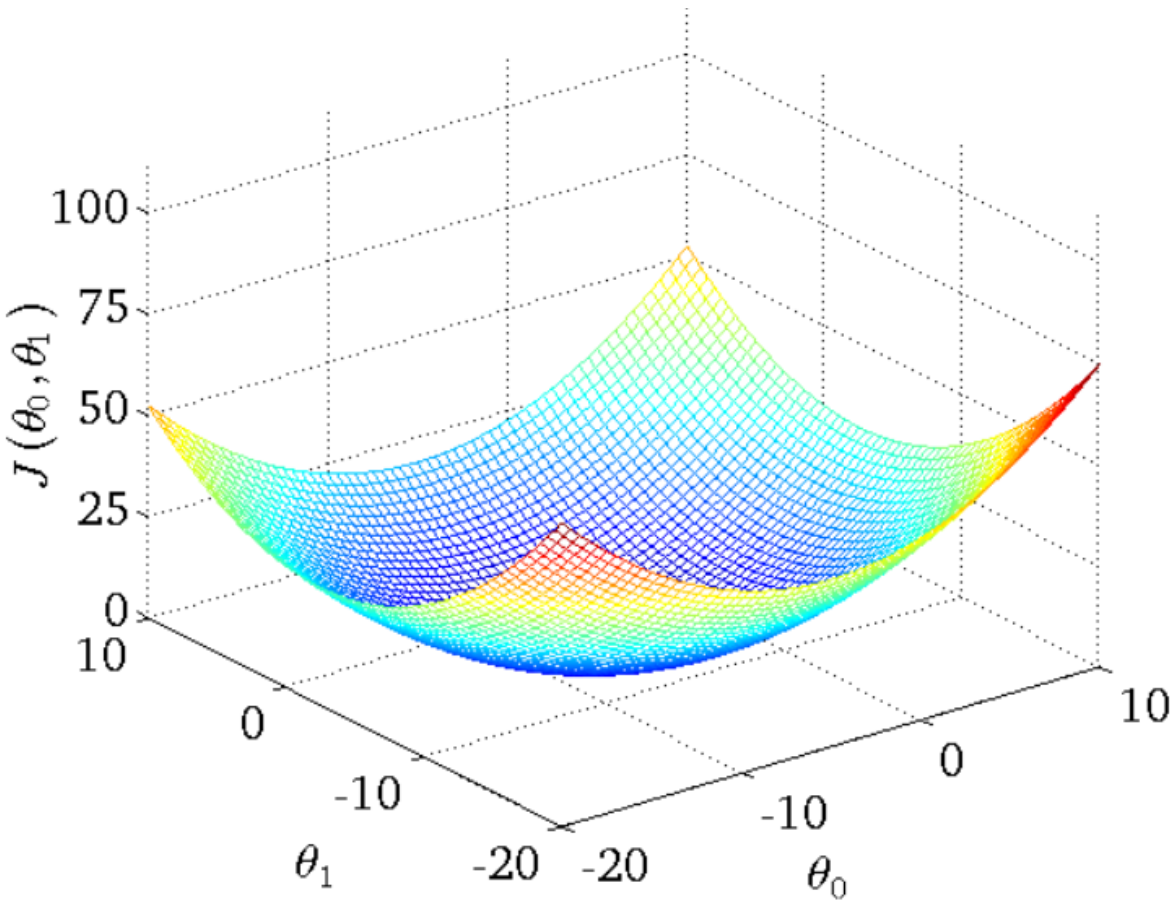
$$h_\theta(x) = \theta_1 x \quad \text{Set } \theta_0 = 0$$



Vemos claramente que la función costo tiene un mínimo global

2.1.4) Intuición de la función *J* para una función hipótesis completa

Como sucedió en el caso incompleto, aquí también se puede notar un mínimo global.



Dependiendo del conjunto de datos de entrenamiento, puede que el gráfico de la función costo ***J*** tenga varios mínimos locales.

Por consiguiente, es necesario un algoritmo que pueda encontrar el mínimo local o global de una

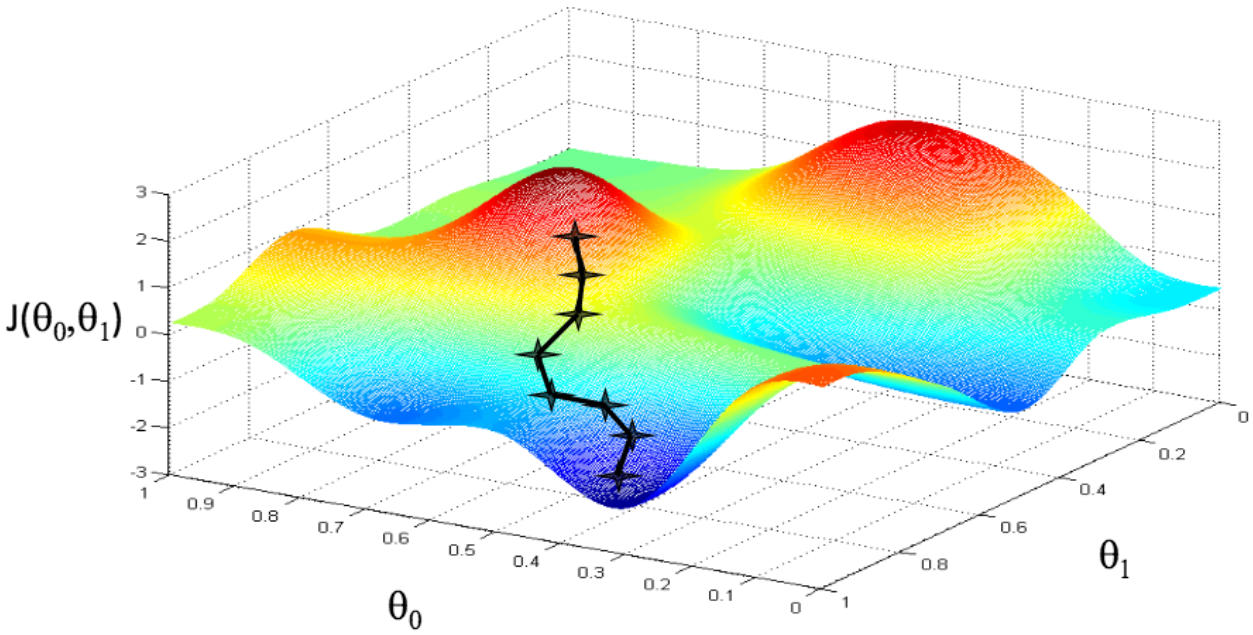
función ***J*** inicializada con valores aleatorios.

2.1.5 Algoritmo Gradient Descent

Resumén del algoritmo

Dada una función costo $J(\theta_0, \theta_1)$ o $J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$ Con el objetivo de minimizar esa función se siguen los siguientes pasos genéricos:

1. Comenzar con algún valor de θ_0 y θ_1, por ejemplo $\theta_0 = 0, \theta_1 = 0$.
2. Iterar, minimizando a cada paso el valor de $J(\theta_0, \theta_1)$ hasta converger en un mínimo local o global.



repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$

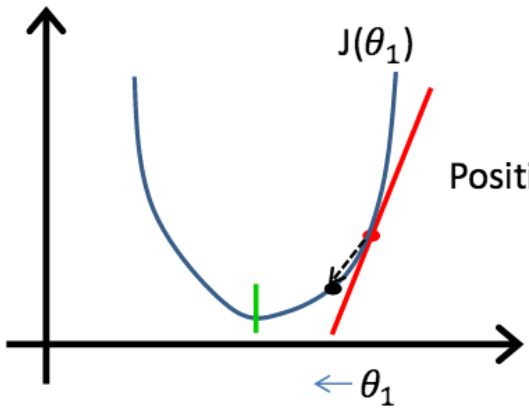
}

Learning rate

Simultaneously update $\theta_0 and\ \theta_1$

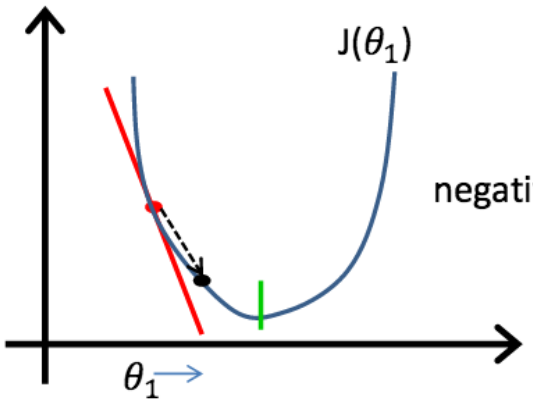
La formula de arriba nos dice que cada parámetro es actualizado simultáneamente, a través de sus derivadas parciales y un ratio de aprendizaje (learning rate), los cuales controlan cuan largos son los pasos que sigue el algoritmo.

La derivada parcial controla la dirección hacia donde los parámetros de la función hipótesis van.



$$\frac{\partial}{\partial \theta_1} J(\theta_1) > 0$$

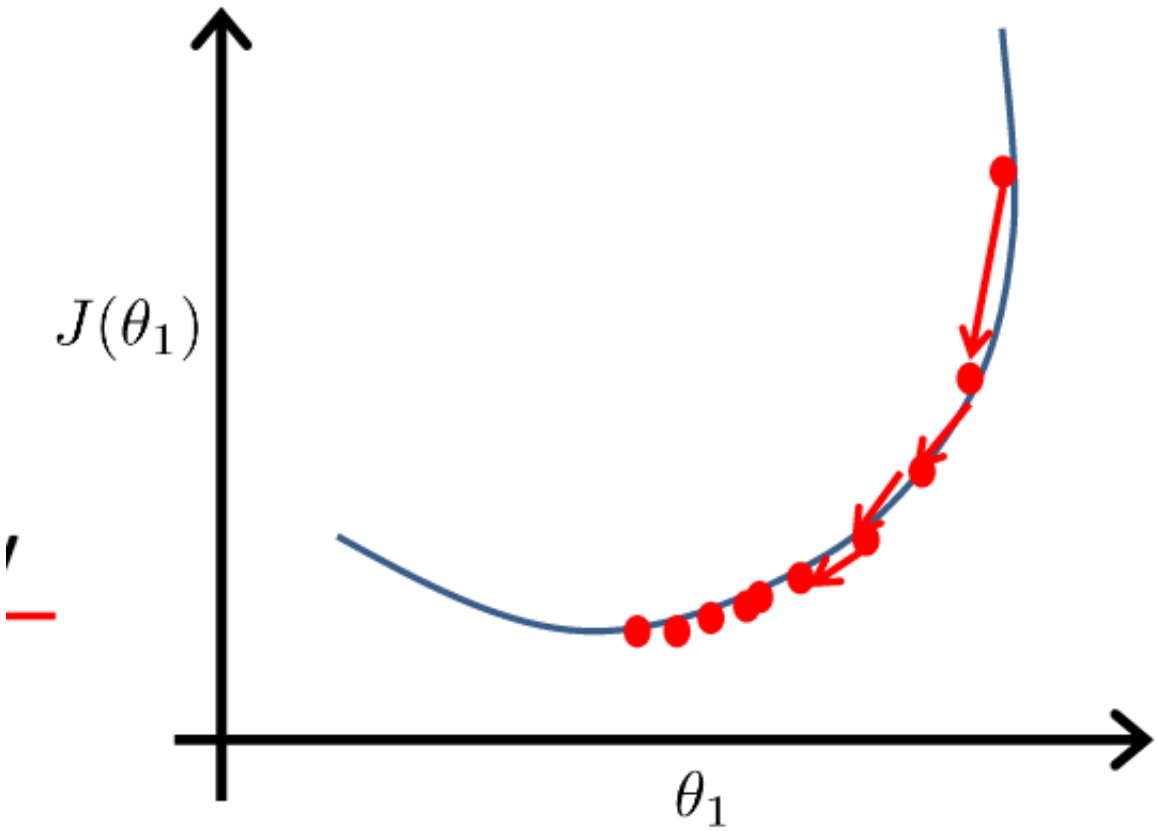
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$



$$\frac{\partial}{\partial \theta_1} J(\theta_1) <= 0$$

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

El algoritmo se detiene cuando la derivada parcial es igual a cero. Note que a medida que el algoritmo va llegando a un mínimo local, el valor de las derivadas parciales se tornan menores. Por lo tanto, no es necesario en actualizar el valor del ratio de aprendizaje en ningún paso.



2.1.5 Algoritmo Gradient Descent para regresión linear

Calculando las derivadas parciales para la función hipótesis linear tenemos:

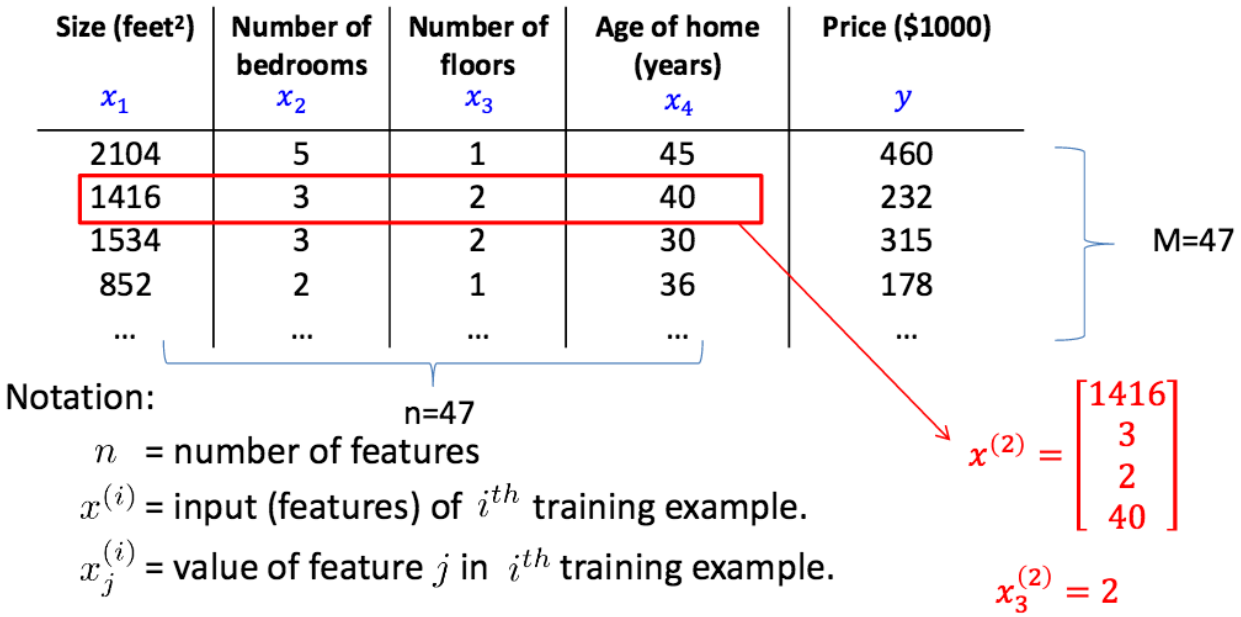
$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \left.
\begin{aligned}
\theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \\
\theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}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

```
% MATLAB CODE
% Usar el archivo ex1data1.txt, utilizando la funcion load(ex1data1.txt) para cargar x, y.
% Gradient descent para regresion lineal de una variable.
% la variable de entrada x, asi como la variable de salida y deben estar normalizadas en el rango de [0 - 1].
% El parametro alpha debe ser positivo y es aconsejable que sea menor a 1.

function gradient_descent_one_var(x, y, alpha)
m = length(x); % numero de ejemplos de entrenamiento

%Para la hipotesis: h_theta = theta_0 + theta_1 * x.
theta_0 = 0;
theta_1 = 0;

% Variables auxiliares donde se almacena el valor de las derivadas parciales de la funcion costo.
sum_0 = 1;
sum_1 = 1;
eps = 0.001; % condicion de termino de programa.

while abs(sum_0 + sum_1) > eps
    sum_0 = 0;
    sum_1 = 0;
    for i = 1:m
```

```
        h_theta = theta_0  + theta_1 * x(i);
        sum_0 = sum_0 + ( h_theta - y(i) );
        sum_1 = sum_1 + ( ( h_theta - y(i) ) * x(i) );
    end
    % Actualizacion simultanea de los parametros theta_0 y theta_1
    theta_0 = theta_0 - alpha * ( 1.0 / m ) * sum_0;
    theta_1 = theta_1 - alpha * ( 1.0 / m ) * sum_1;
end

h_y = theta_0 + theta_1 * x;
plot(x, h_y);
```

2.2 Regresión lineal con múltiples variables

Trabaja con múltiples variables, por lo tanto, es de alguna forma mas poderoso y completo que la regresión lineal con una variable. Se requiere un entendimiento básico de álgebra lineal para entender esta sección.



2.2.1 Función hipótesis

Considerando el ejemplo anterior, nos damos cuenta que existen 4 variables de entrada (o características) por lo tanto la función hipótesis queda actualizada de la siguiente forma:

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

En general la fórmula es la siguiente:

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

Por conveniencia:

$$x_0 = 1. \longrightarrow x_0^{(i)} = 1$$

De esa forma se consigue utilizar operaciones de vectores e matrices. Donde finalmente obtenemos una función hipótesis corta e elegante.

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in R^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in R^{n+1}$$

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

$$= \boldsymbol{\theta^T X}$$

$$\underbrace{[\theta_0, \theta_1, \ldots, \theta_n]}_{\boldsymbol{\theta^T}} \quad \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}}_{\boldsymbol{X}}$$

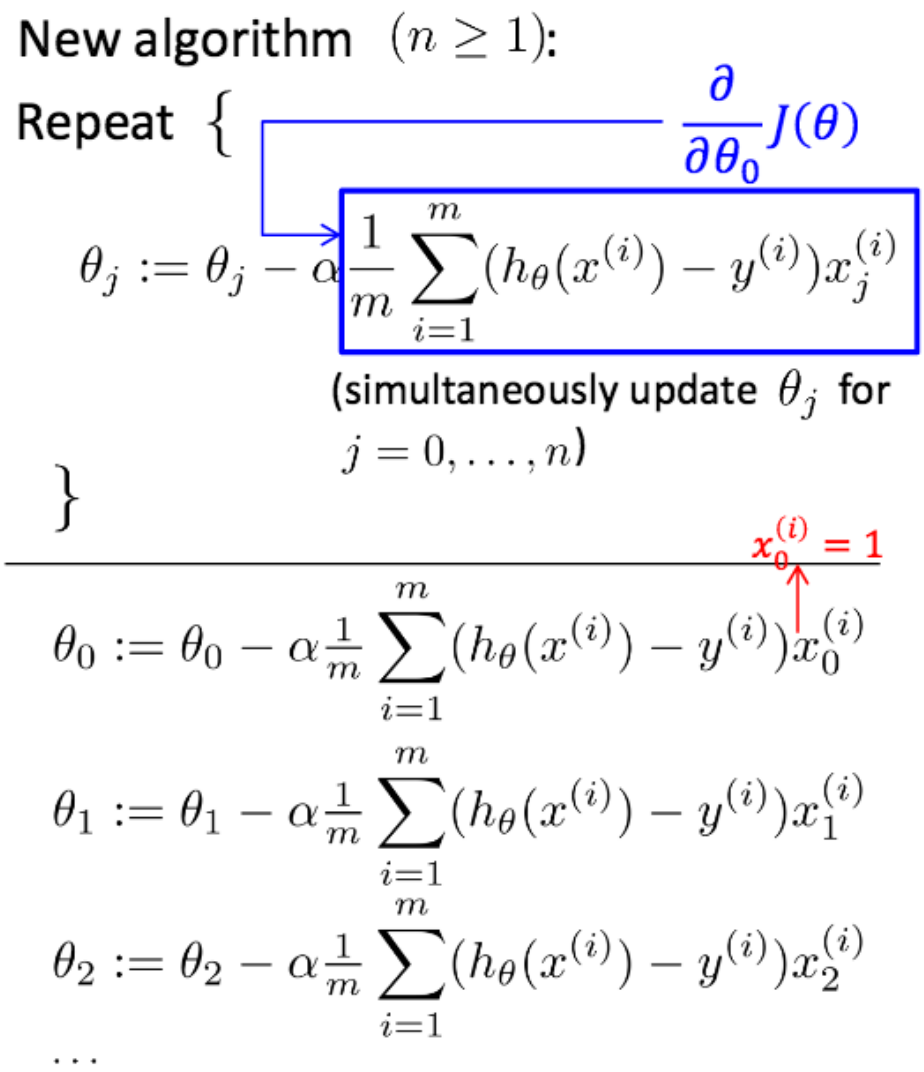
2.2.2 Función costo

Al igual que la función hipótesis, en este caso J también es adaptada para trabajar con múltiples variables.

Cost function:

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

2.2.3 Algoritmo gradient descent para regresión con múltiples variables

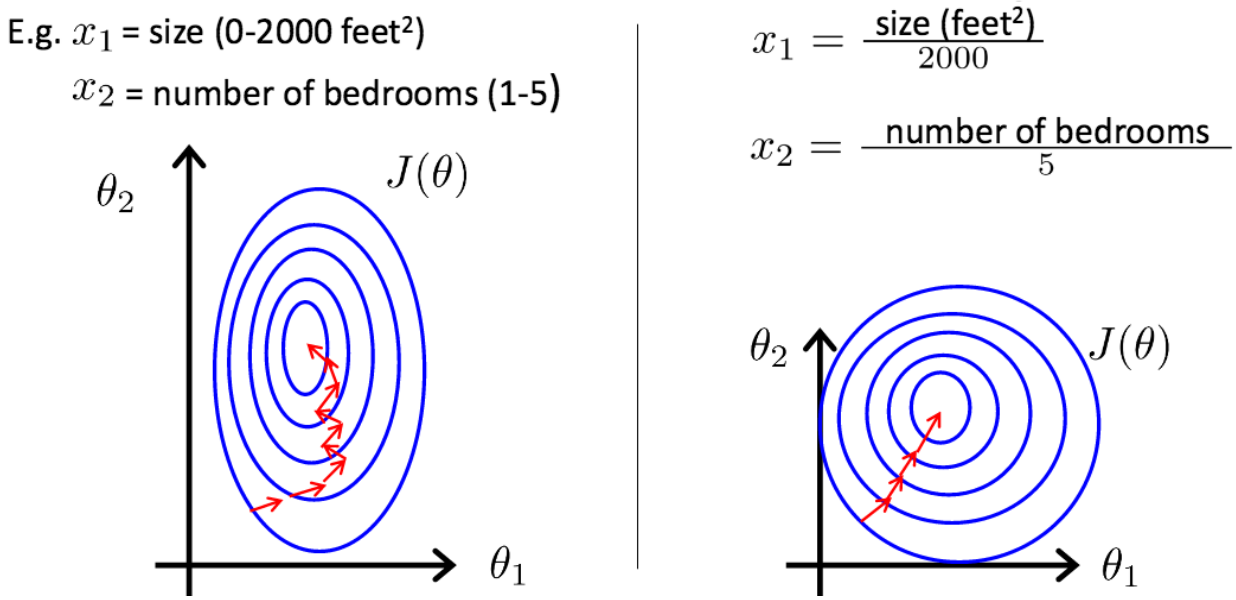



```
function Gradient_descent_mul_var(X, y, alpha, num_iters)
m = length(y)
Theta = zeros(size(X, 2), 1);
J_history = zeros(1, num_iters);
for i = 1:num_iters
    J_history(i) = ( 1 / ( 2 * m ) ) * ( X * Theta - y )' * ( X * Th
eta - y );
    Theta = Theta - alpha * (1 / m ) * (( X * Theta - y )' * X)';
    disp(Theta);
end
```

2.3 Escalamiento de características

Es una buena idea escalar las variables o características envueltas en la regresión. El motivo de hacer esto recae en que muchas veces el rango de valores posibles de una variable es notablemente mayor al rango de valores de las otras variables. Eso afecta el tiempo de convergencia del algoritmo Gradient Descent y hasta puede llegar a nunca encontrar la solución (divergir).

En el gráfico de la izquierda vemos que la variable ***x1*** puede tomar valores de [0 - 2000] mientras que la variable ***x2*** esta en el rango [1-5]. Por lo tanto, la ***función costo*** resulta ser mas larga en un eje, lo que dificulta el proceso de convergencia del algoritmo. En tanto, en el gráfico de la derecha, las variables están normalizadas, por consiguiente, tenemos un tiempo de convergencia menor.



En la práctica esta demostrado que una variable ***X*** debe estar aproximadamente en un rango de [-1, 1]. Por ejemplo:

Correcto	Incorrecto
• -1 <= x1 <= 1 • 0 <= x2 <= 5 • -2 <= x3 <= 0.5	• -0.00001 <= x1 <= 0.00001 • 0 <= x2 <= 1000 • -100 <= x3 <= 100

Formas de escalamiento (normalización)

- **Por el máximo y mínimo**

$$x_i = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)}$$

Todas las variables normalizadas con este método van a recaer entre el rango de [0 - 1].

- **Utilizando la media (mean normalization)**

Este método está orientado a conseguir que las variables ***xi*** tengan una media de

aproximadamente 0.

$$x_i = \frac{x_i - u_i}{s_i}$$

Donde:

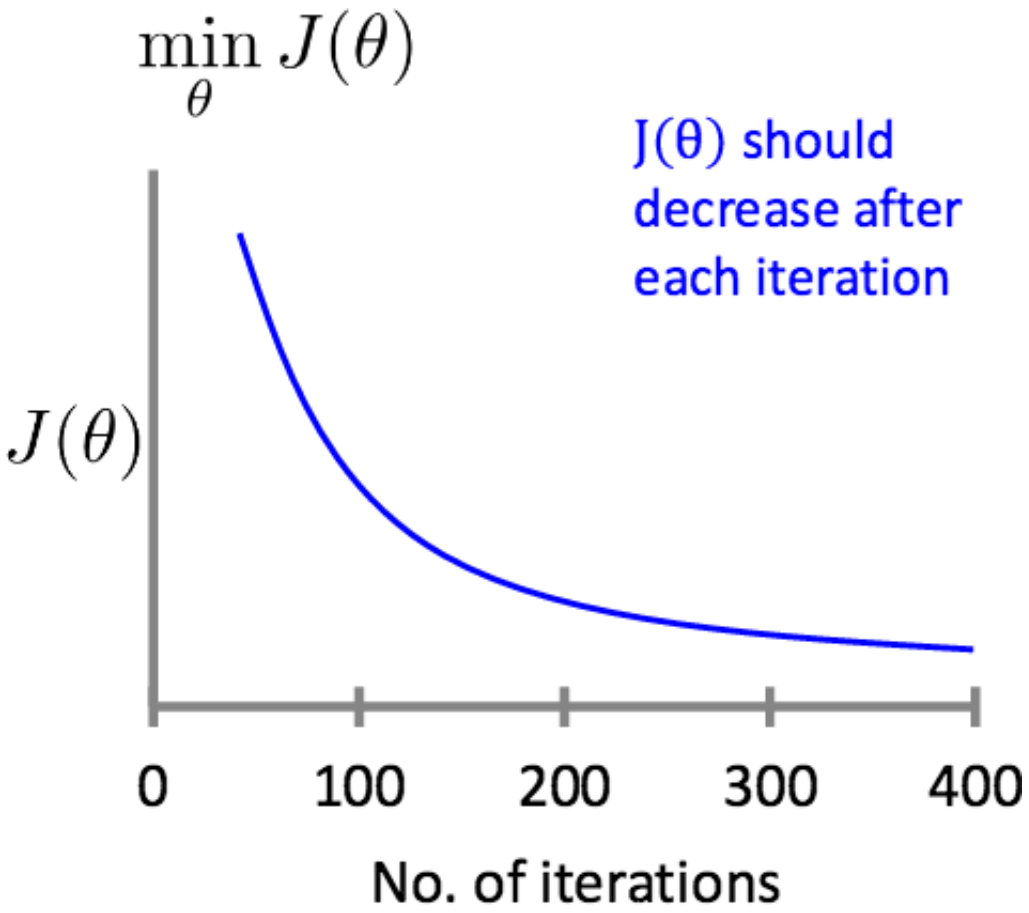
- μ_i = media de la variable x_i
- s_i = puede ser igual a $max(x_i) - min(x_i)$ o la desviación estandar de x_i.

2.4 Mejorando la búsqueda del ratio de aprendizaje.

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

Debugging

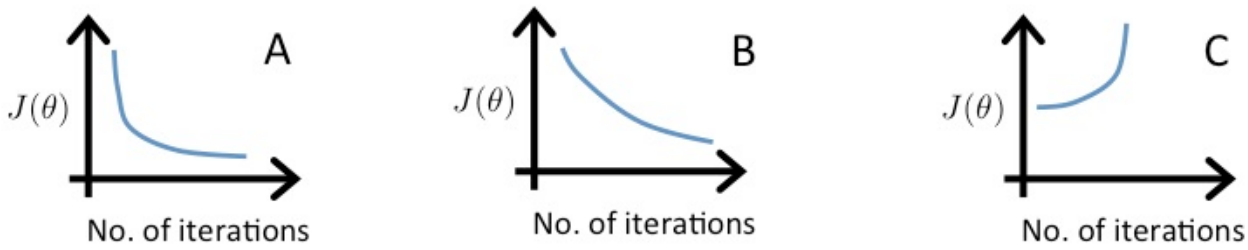
Para entender el funcionamiento de la funcion costo $J(\theta)$ es necesario hacer un gráfico que demuestre su grado de convergencia en relación al número de iteraciones hechas.



Este gráfico ayuda a entender el impacto del ratio de aprendizaje α sobre el desempeño del algoritmo *Gradient Descent*.

- Un buen α logra que $J(\Theta)$ descrezca rapidamente a cada iteración.
- Un α muy grande puede causar problemas de convergencia.
- Un α muy pequeño afecta el tiempo de convergencia del algoritmo

Para pensar: Los 3 gráficos de abajo muestran el grado de convergencia del algoritmo Gradient Descent para un problema en particular. Cada gráfico fue hecho con un α distinto. Los ratios de aprendizaje utilizados fueron: $\alpha = 0.001, \alpha = 0.1$ y $\alpha = 1$. A qué gráfico corresponde cada α?.

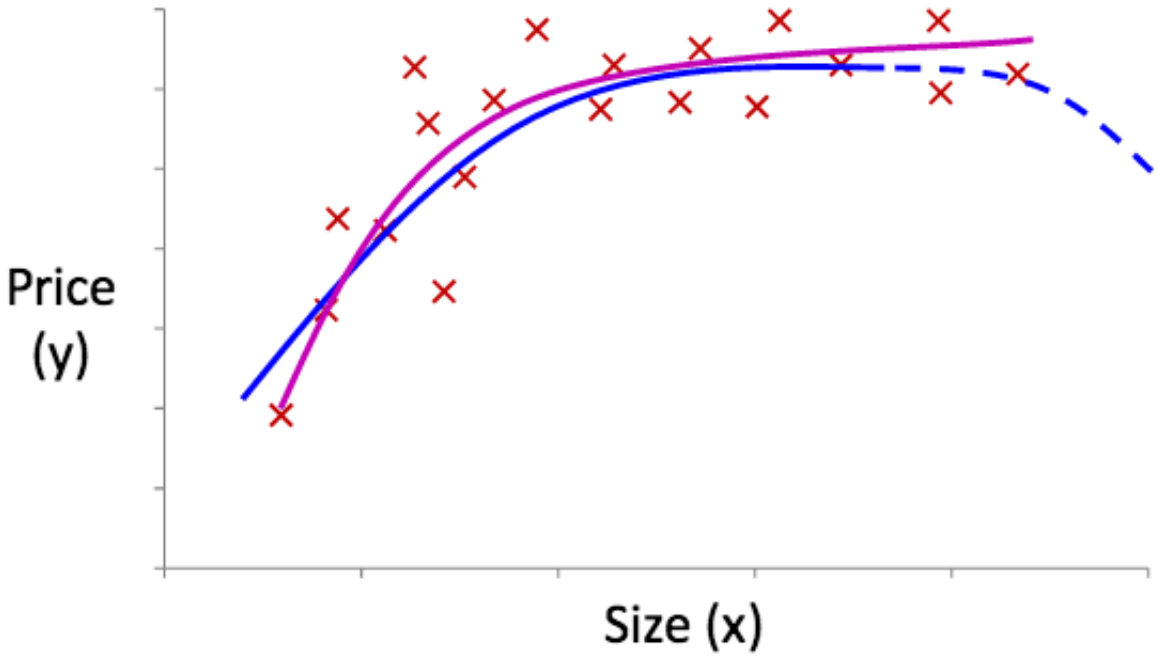


A seguir una lista de ratio de aprendizaje sugeridos.

$$\ldots, 0.001, \mathbf{0.003}, 0.01, \mathbf{0.03}, 0.1, \mathbf{0.3}, 1, \ldots$$

2.5 Regresión Polinomial

Cuando se tiene alguna variable de entrada que afecta de forma no lineal a la variable de salida, es donde se puede utilizar el siguiente truco a la ***función hipótesis.***



$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2$$

$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{(size)}$$

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + \theta_1(size) + \theta_2(size)^2 + \theta_3(size)^3 \end{aligned}$$

$$\begin{aligned} x_1 &= (size) \\ x_2 &= (size)^2 \\ x_3 &= (size)^3 \end{aligned}$$

A partir de ahí se puede aplicar el algoritmo Gradient Descent sin problemas.

IMPORTANTE: En estos casos es muy importante aplicar el escalamiento de características de la

sección 2.3, debido a que una variable elevada al cuadrado tiende a tener un rango de valores mucho mayor.

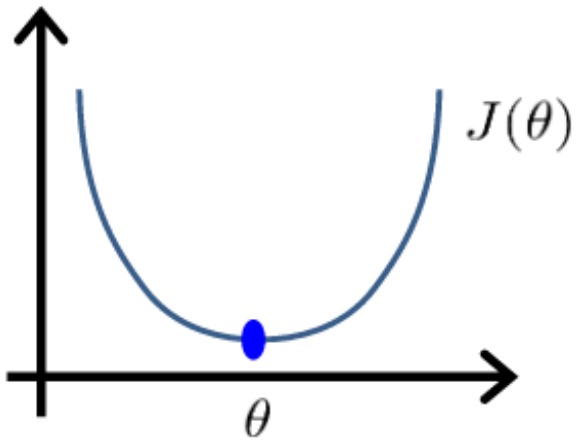
2.6 Ecuación Normal

Es posible determinar analíticamente el mínimo valor de la ***función costo***, a través de algunas operaciones de matrices. La intuición de esto viene del cálculo de mínimos y máximos en funciones a través de su derivada igualada a cero.

- Un ejemplo para el caso 1D es el siguiente:

$$J(\theta) = a\theta^2 + b\theta + c$$

$$\frac{d}{d\theta}J(\theta) = \ldots \overset{set}{=} 0$$



- Para varias variables tenemos:

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

$$\frac{\partial}{\partial \theta_j}J(\theta) = \cdots = 0 \quad \text{(for every } j\text{)}$$

- Resolviendo para:

$$\theta_0, \theta_1, \ldots, \theta_n$$

- Recordando que ***n*** es el número de variables (o características) e ***m*** es el número de muestras de entrenamiento. Tenemos el siguiente ejemplo para ***m*** = 4 e ***n*** = 4.

x_0	Size (feet²) 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

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

- La variable x_0 es un vector de solo 1s, que en la *regresión lineal con múltiples variables* fue utilizada como variable auxiliar de la ***función hipotesis*** ($h_\Theta = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n$), con el objetivo de que se puedan efectuar operaciones de matrices sobre ella.
- La ecuación analítica derivada mediante algebra lineal és: $\Theta = (X^T X)^{-1} X^T y$, donde X^T es la transpuesta de la matriz X.

Gradient Descent VS Ecuación Normal

Gradient Descent	Ecuación Normal
• Se necesita escoger y encontrar el ratio de aprendizaje optimo. • Se necesita un número de iteraciones no conocido, dependiendo del problema a ser resuelto. • Trabaja rápido aun cuando ***n*** es muy grande	• No necesita el parámetro ratio de aprendizaje. • No necesita iteraciones, sin embargo dado que necesita calcular la inversa. Su costo computacional es de ***O(n ^ 3)***. • Es muy lento si ***n*** es grande. • Si la matriz no tiene inversa (por causa de variables linearmente dependientes), entonces, no se puede aplicar.

---

3) Regresión Logística

Slides:
https://docs.google.com/presentation/d/16rGGWWbe3mo4cySjrWW_vzV0VwH5NMIZg90Th9

fyO0s/edit#slide=id.p

4) Regularización

Slides:
https://docs.google.com/presentation/d/12EAsWXemzdIO56JWxGeRFuCB49Hl8OcpjqejGcKFnGQ/edit

5) Redes Neuronales