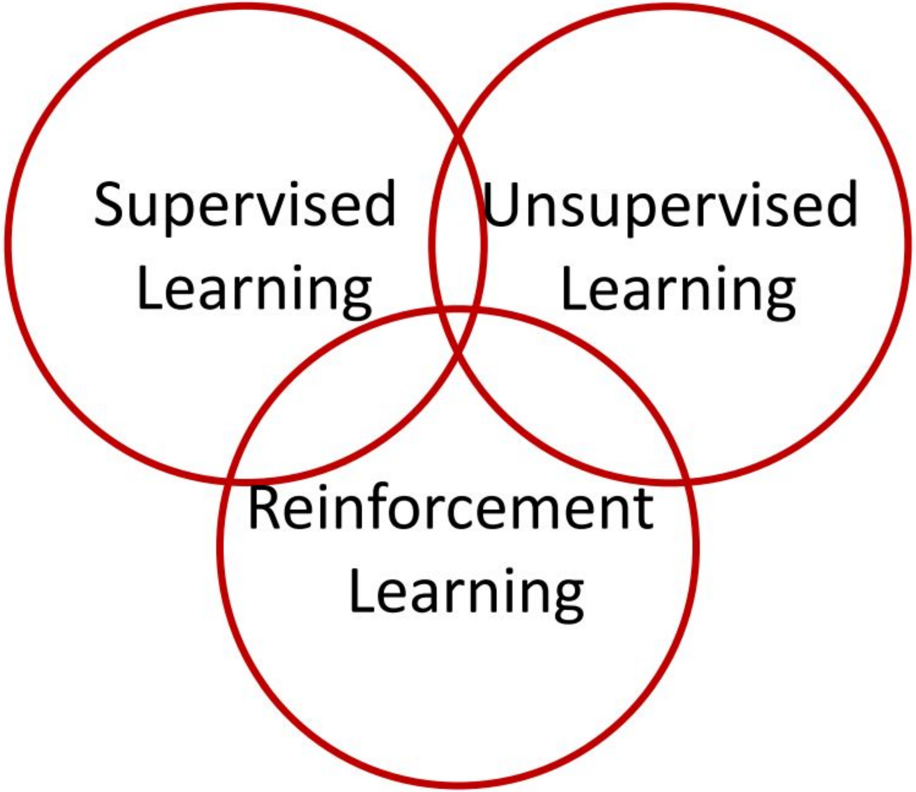
Supervised Learning

Prof. Rosa Paccotacya Yanque

Recap - Tipos de ML



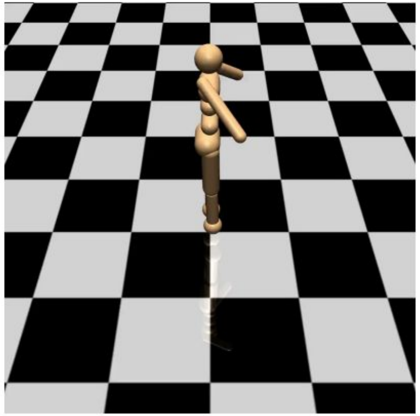
Recap: Reinforcement Learning

El aprendizaje por refuerzo se ocupa de agentes que aprenden a tomar decisiones interactuando con un entorno. El agente recibe retroalimentación en forma de recompensas o sanciones.

Objetivo: el objetivo es que el agente aprenda una política que maximice la recompensa acumulada a lo largo del tiempo.

Ejemplos: Las aplicaciones incluyen juegos (por ejemplo, AlphaGo), control robótico y sistemas autónomos.

learning to walk to the right

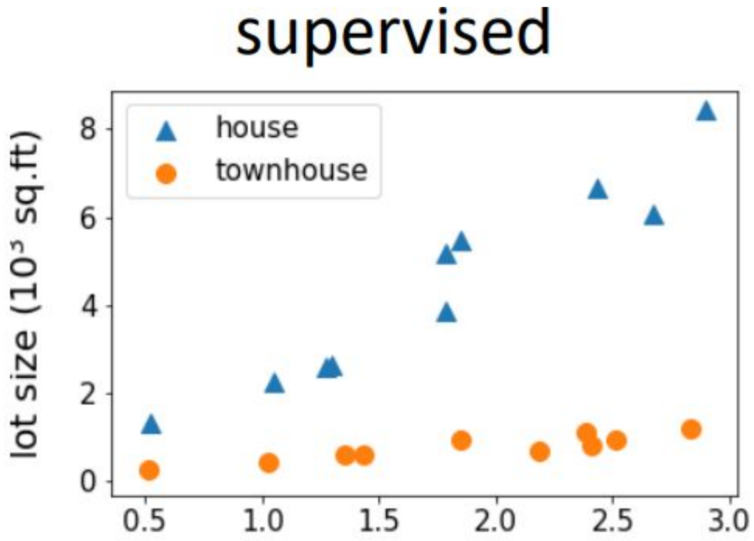


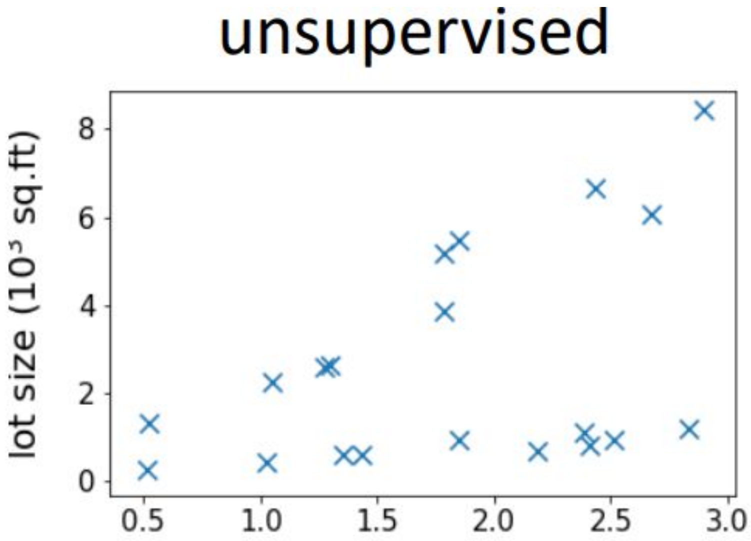
Iteration 10

Recap: Unsupervised Learning

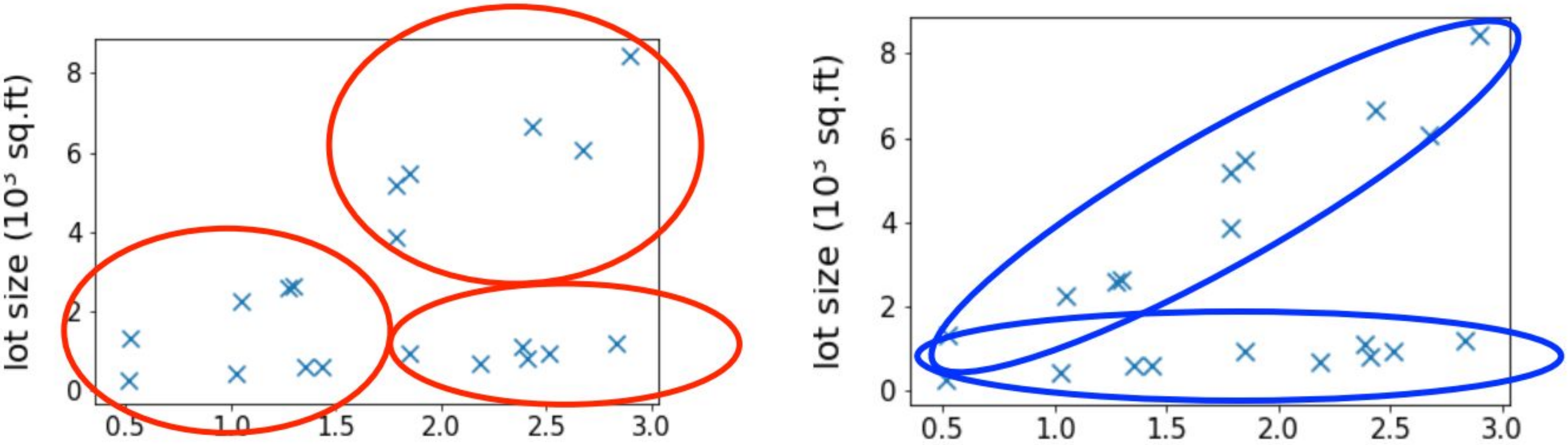
El aprendizaje no supervisado implica algoritmos que trabajan con datos sin etiquetar, con el objetivo de descubrir patrones o estructuras inherentes dentro de los datos.

- Objetivo: El modelo identifica relaciones, grupos o asociaciones sin orientación explícita sobre los resultados correctos.
- Ejemplos: agrupación (agrupar puntos de datos similares), reducción de dimensionalidad (simplificar datos preservando la información esencial)


supervised
house
townhouse
lot size (10^3 sq.ft)
0
2
4
6
8
0.5
1.0
1.5
2.0
2.5
3.0


unsupervised
lot size (10^3 sq.ft)
0
2
4
6
8
0.5
1.0
1.5
2.0
2.5
3.0

Clustering



K-Means - Método del cubo

Reducción de dimensionalidad

tract	T9_est1	T9_est2	T9_est3	T9_est4	T9_est5	T9_est6	T9_est7	T9_est8	T9_est9	T9_est10	T9_est11	T9_est12
20100	690	500	470	415	40	15	0	25	25	0	0	0
20200	795	500	225	200	4	20	0	240	165	50	25	0
20300	1210	905	695	605	30	60	0	170	140	25	4	0
20400	1705	1375	1235	1020	120	90	0	40	40	0	0	0
20500	4165	2395	2210	1870	210	130	0	145	85	60	0	0
20600	1255	935	845	695	90	60	0	40	15	0	20	0
20700	1015	715	635	575	15	50	0	55	15	10	30	0
20801	1180	995	810	720	75	0	15	70	55	0	10	4
20802	3715	3155	2815	2110	400	290	15	250	130	40	75	0
20900	2080	1680	1445	1000	205	225	15	235	65	100	65	0
21000	1200	1040	795	695	85	20	0	230	170	4	60	0
21100	1295	1050	465	420	10	35	0	585	365	155	60	0
10100	1280	1215	870	695	100	40	40	255	135	95	25	0
10200	1100	885	795	555	115	120	4	30	30	0	0	0
10300	2460	2030	1780	1460	220	55	50	240	50	140	50	0
10400	1665	1365	1335	1015	165	130	25	15	0	15	0	0
10500	1780	1165	1125	945	115	55	10	20	4	0	0	10
10600	1225	705	285	215	30	40	0	415	275	65	75	0
10701	3045	2330	2130	1745	140	230	15	90	40	50	0	0
10703	5660	4075	3745	3010	375	360	0	135	135	0	0	0
10704	1775	1375	1285	995	200	85	4	50	25	15	10	0
10705	3720	1985	1740	1260	240	210	30	155	120	20	20	0
10800	2705	1730	1250	1030	95	125	0	465	245	85	125	10
10903	1860	1235	1095	870	120	105	0	70	60	10	0	0
10904	2300	1995	1940	1560	215	120	45	4	4	0	0	0
10905	3085	1995	1805	1430	160	205	4	130	130	0	0	0
10906	1595	1235	1210	1050	140	20	0	0	0	0	0	0
11000	1660	1275	1150	760	190	195	4	65	25	40	4	0
11101	3600	2775	2710	2060	405	250	0	30	30	0	0	0

CL101	CL102	CL103	CL104
343	4556	243	9766
7676	7567	4676	4443
686	8766	6656	6777
4768	3445	76876	2445
9809	4556	785	3456
9806	4577	588	3566
3889	243	443	6776
9766	24344	2567	3356
887	356	7889	7555
5633	6678	7894	899
45667	8655	865	6546
2343	47899	5688	2344
4556	57899	54336	2656
3567	90887	36740	14631
96556	99776	20625	11892
4677	97335	7676	97878
4663	4567	30347	15305
235	4578	54505	19670
456	5466	356	44967
79799	4567	90808	9909
9877	8986	7987	8990
8667	9809	6980	4677
24980	19318	16600	8184
18409	16818	11277	7033
9756	60980	7090	98654
26424	14511	14891	9111
29054	26944	24868	10722

PCA - TSNE

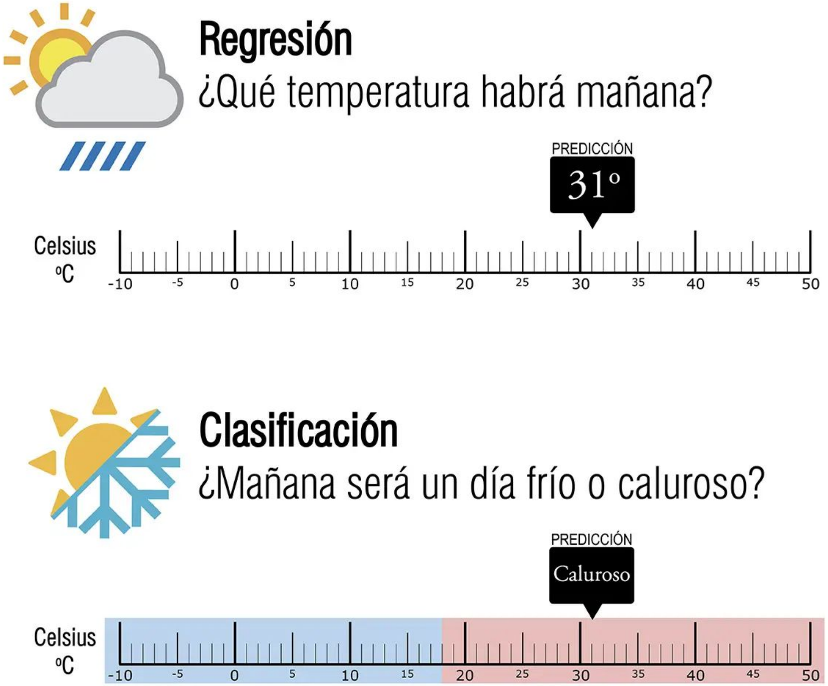
Supervised Learning

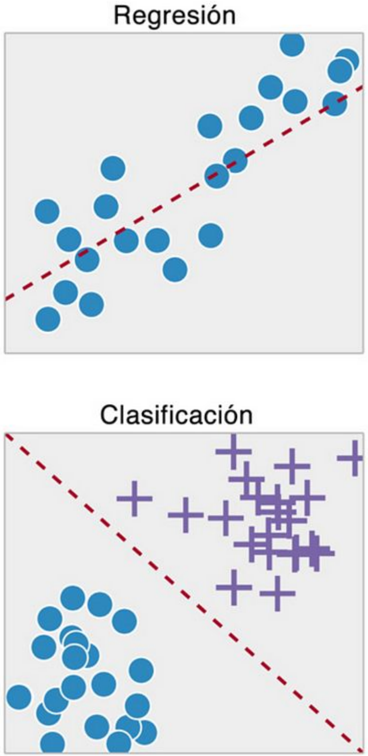
En el aprendizaje supervisado, el algoritmo se entrena en un conjunto de datos etiquetados, donde cada entrada tiene una salida correspondiente.

- Objetivo: el modelo aprende la relación entre las características de entrada y las etiquetas de destino correspondientes, lo que le permite hacer predicciones sobre datos nuevos e invisibles.

- Ejemplos: La clasificación (asignar entradas a clases predefinidas) y la regresión (predecir valores continuos) son tareas comunes en el aprendizaje supervisado.

Supervised Learning





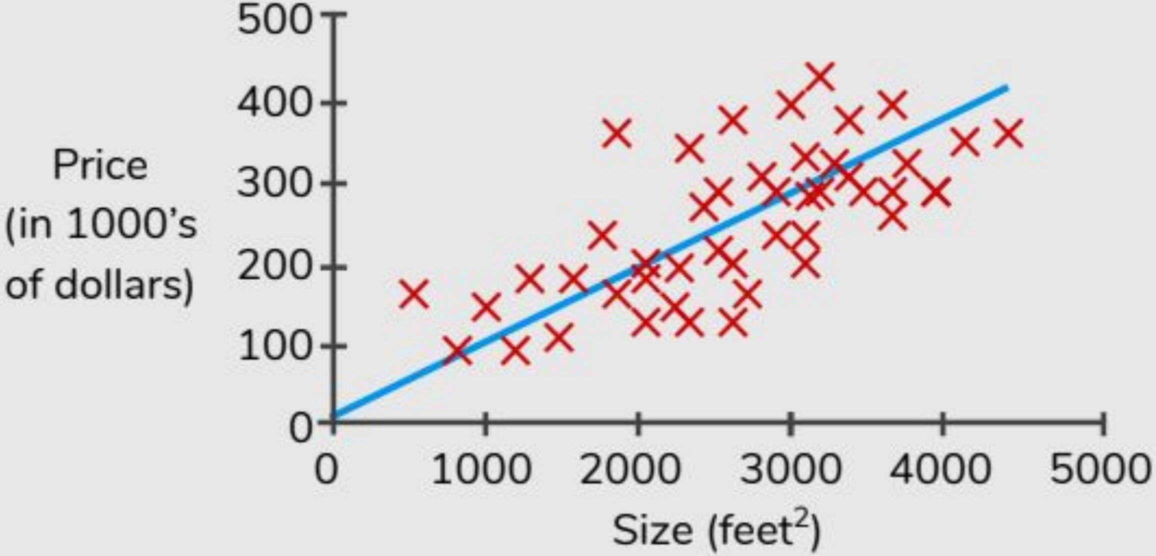
Linear Regression

$ 70 000

$ 160 000

???

Housing Prices



$$
\begin{aligned}
y_1 &= w_0 + w_1x_{11} + w_2x_{12} + w_3x_{13} + \dots \\
y_2 &= w_0 + w_1x_{21} + w_2x_{22} + w_3x_{23} + \dots \\
y_3 &= w_0 + w_1x_{31} + w_2x_{32} + w_3x_{33} + \dots \\
y_4 &= w_0 + w_1x_{41} + w_2x_{42} + w_3x_{43} + \dots \\
y_5 &= w_0 + w_1x_{51} + w_2x_{52} + w_3x_{53} + \dots \\
&\vdots
\end{aligned}
$$

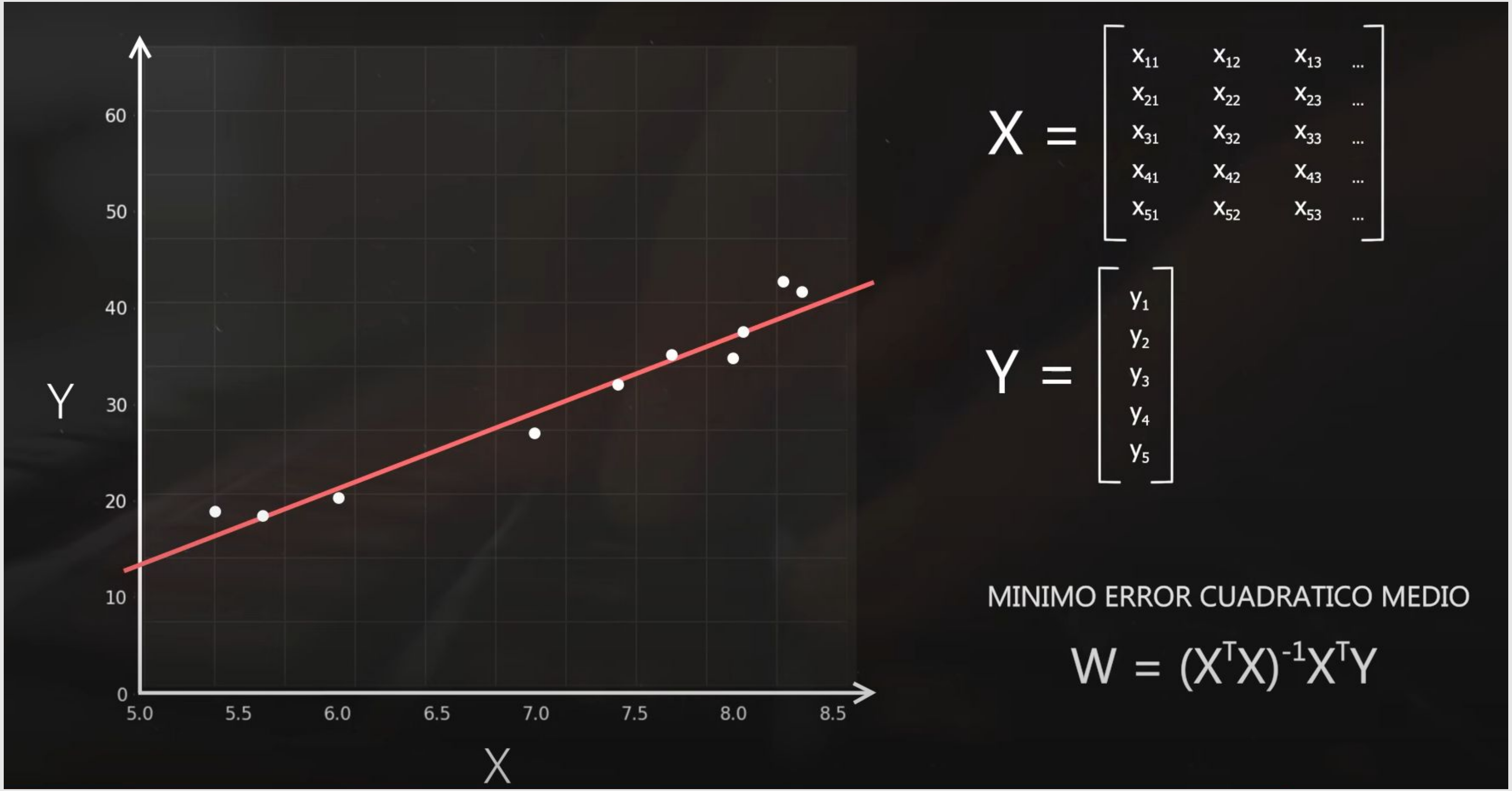
$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots \\ x_{21} & x_{22} & x_{23} & \dots \\ x_{31} & x_{32} & x_{33} & \dots \\ x_{41} & x_{42} & x_{43} & \dots \\ x_{51} & x_{52} & x_{53} & \dots \end{bmatrix}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

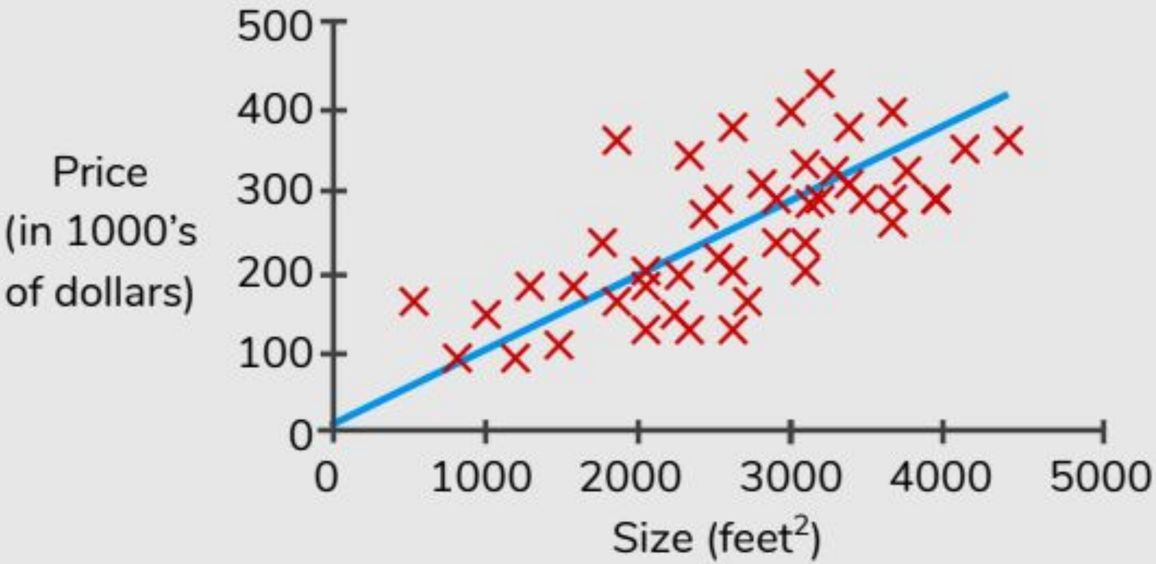
$$Y = XW$$

$$W = \begin{bmatrix} w_0 & w_1 & w_2 & w_3 \end{bmatrix}$$

Mínimos cuadrados ordinarios (Ordinary Least Squares)



Housing Prices



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Training set of housing prices

Size in feet^2 (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

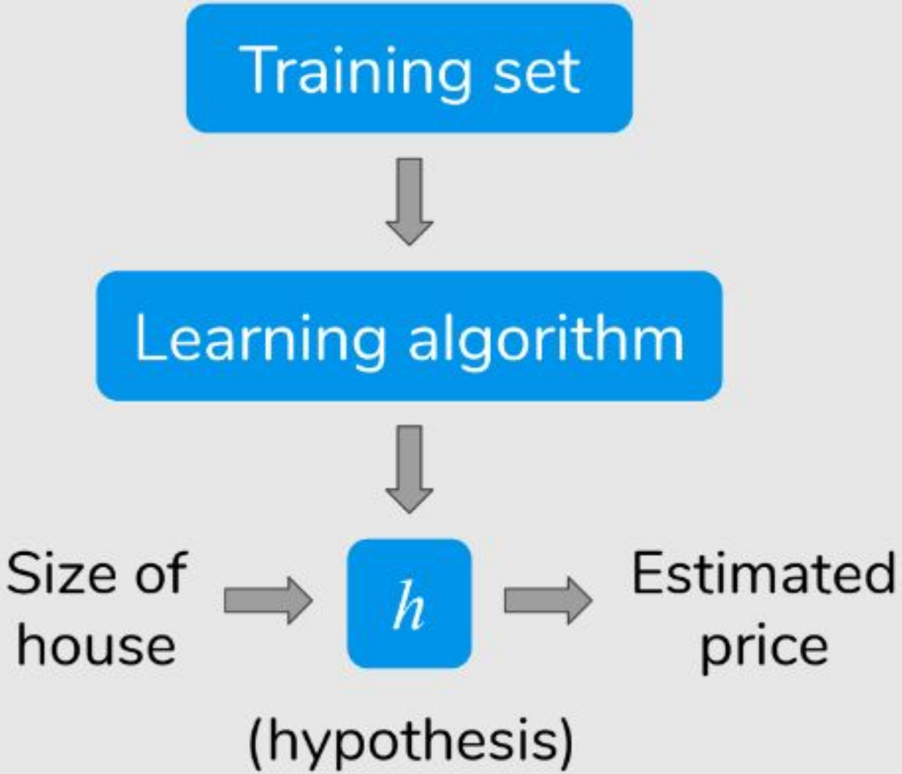
Notation:

m = Number of training examples

x's = "input" variable / features

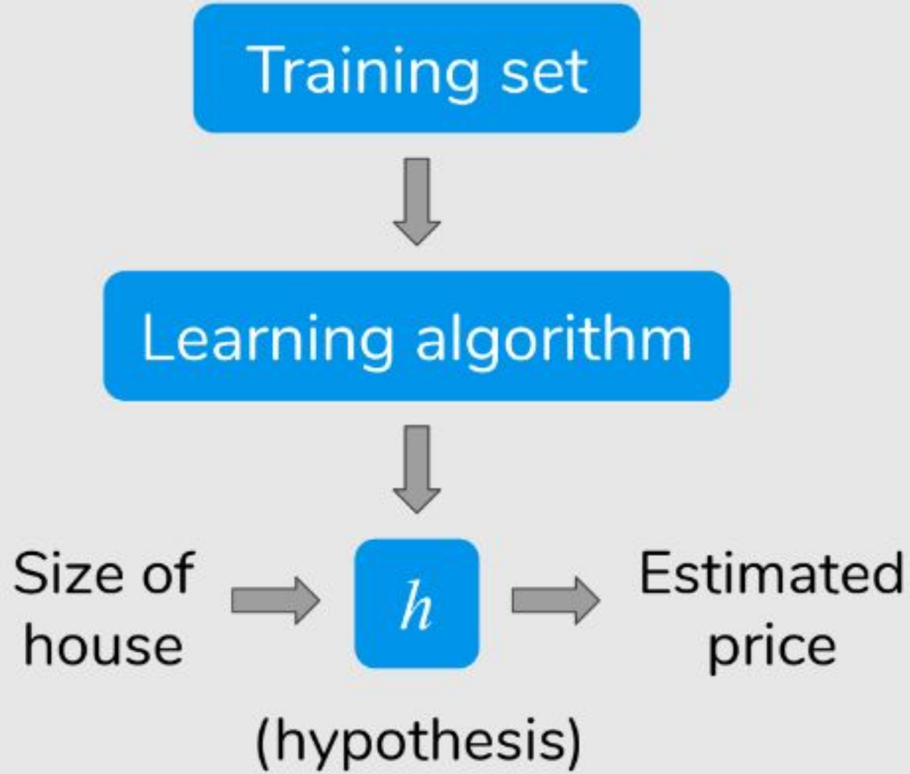
y's = "output" variable / "target" variable

How do we represent h ?



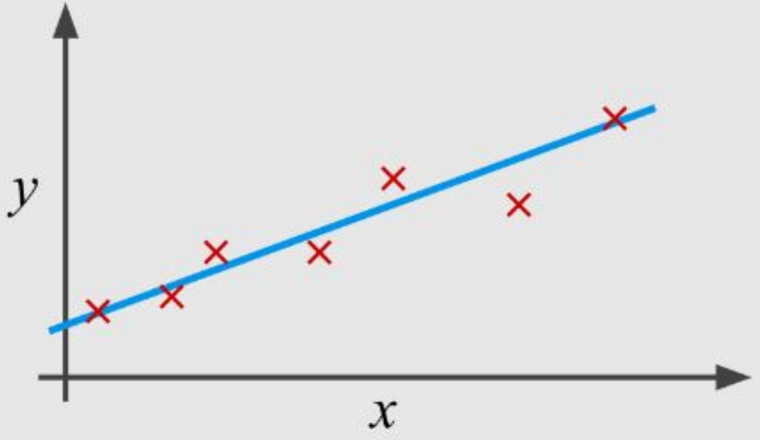
h **maps** x**'s to** y**'s**

How do we represent h ?



h maps x's to y's

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



Linear regression with one variable.
Univariate linear regression.

Training Set

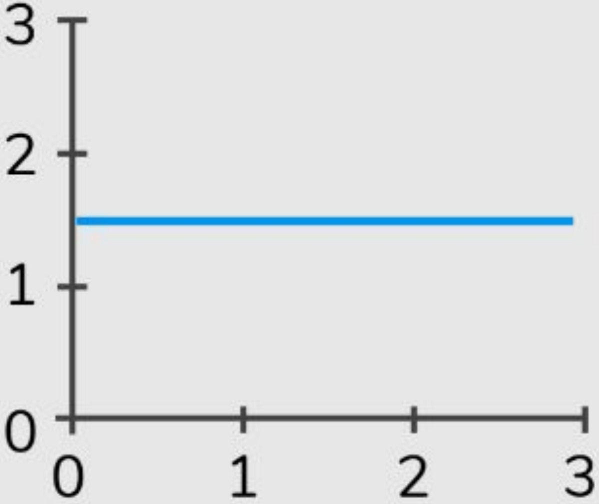
Size in feet^2 (x)	Price $(\$)$ in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

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

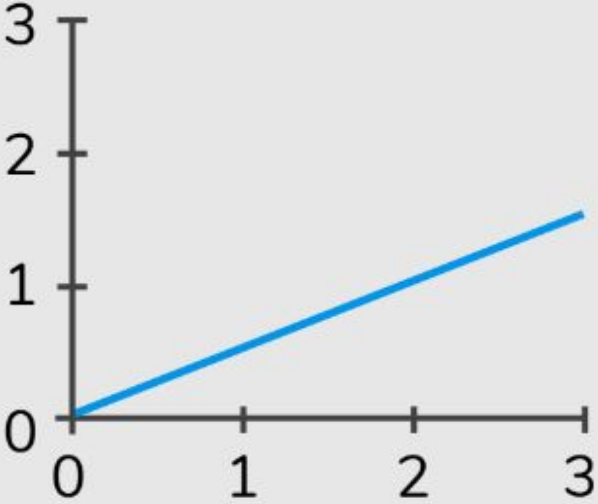
θi's: Parameters

How to choose θi's ?

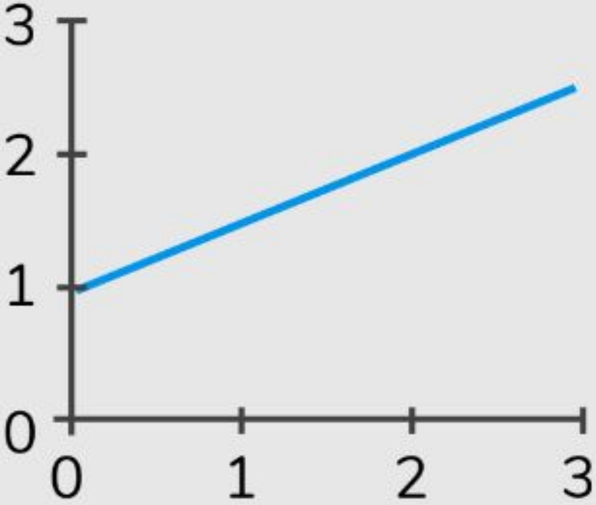
$$h_\theta(x) = \theta_0 + \theta_1 x$$



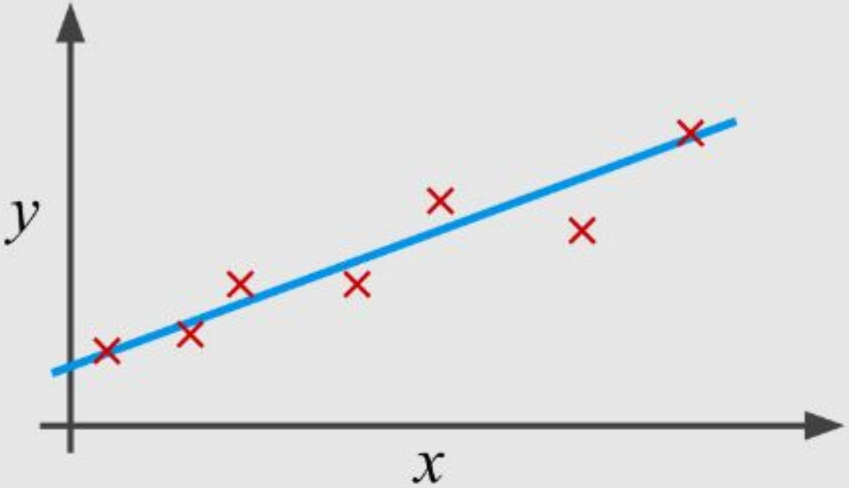
$\theta_0 = 1.5$
$\theta_1 = 0$



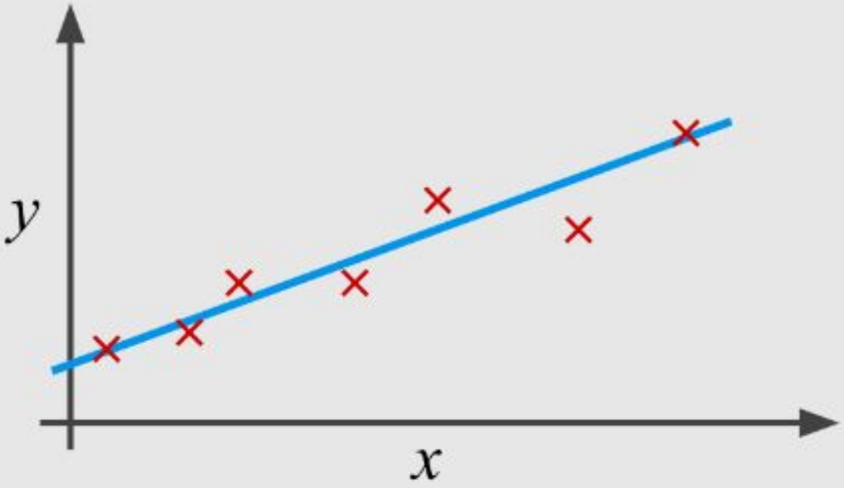
$\theta_0 = 0$
$\theta_1 = 0.5$



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

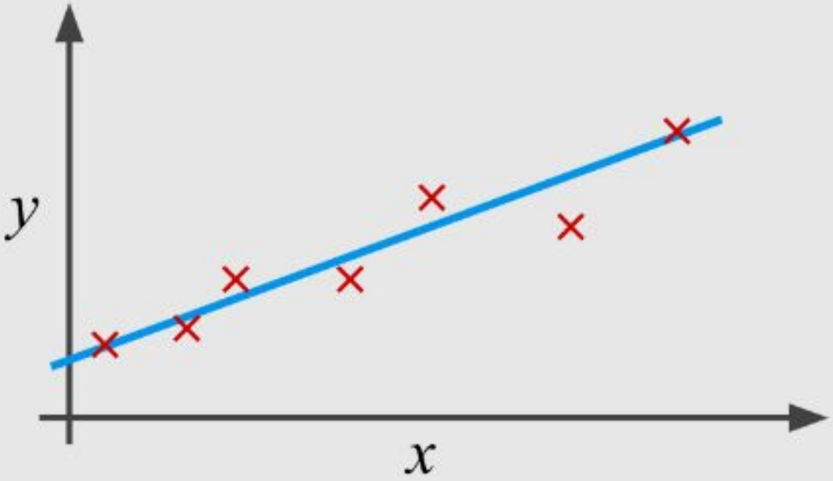


Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ close to y for our training examples (x, y)



$$\underset{\theta_0,\theta_1}{\text{minimize}} \qquad \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ close to y for our training examples (x,y)



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

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

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ close to y for our training examples (x,y)

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

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

Cost function
(Squared error function)

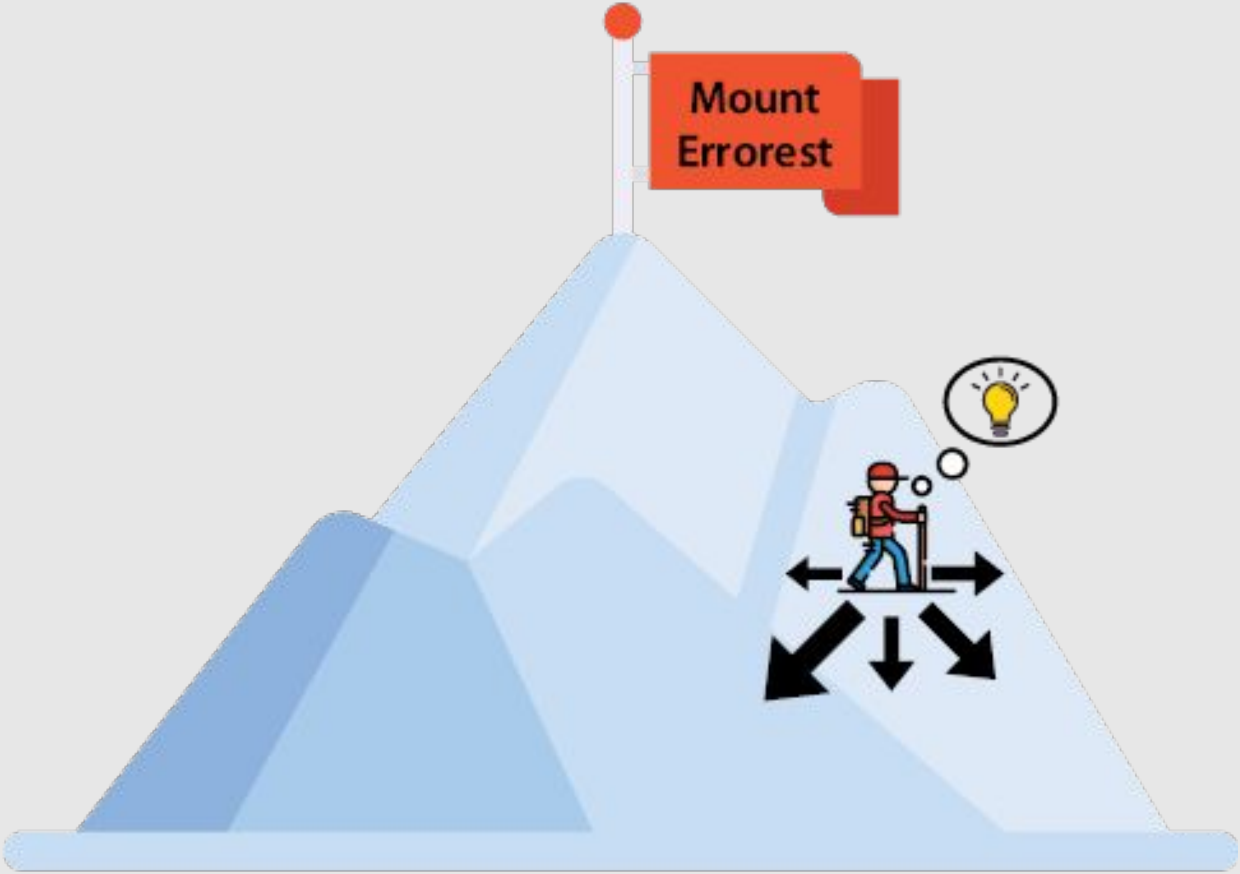
Gradient Descent

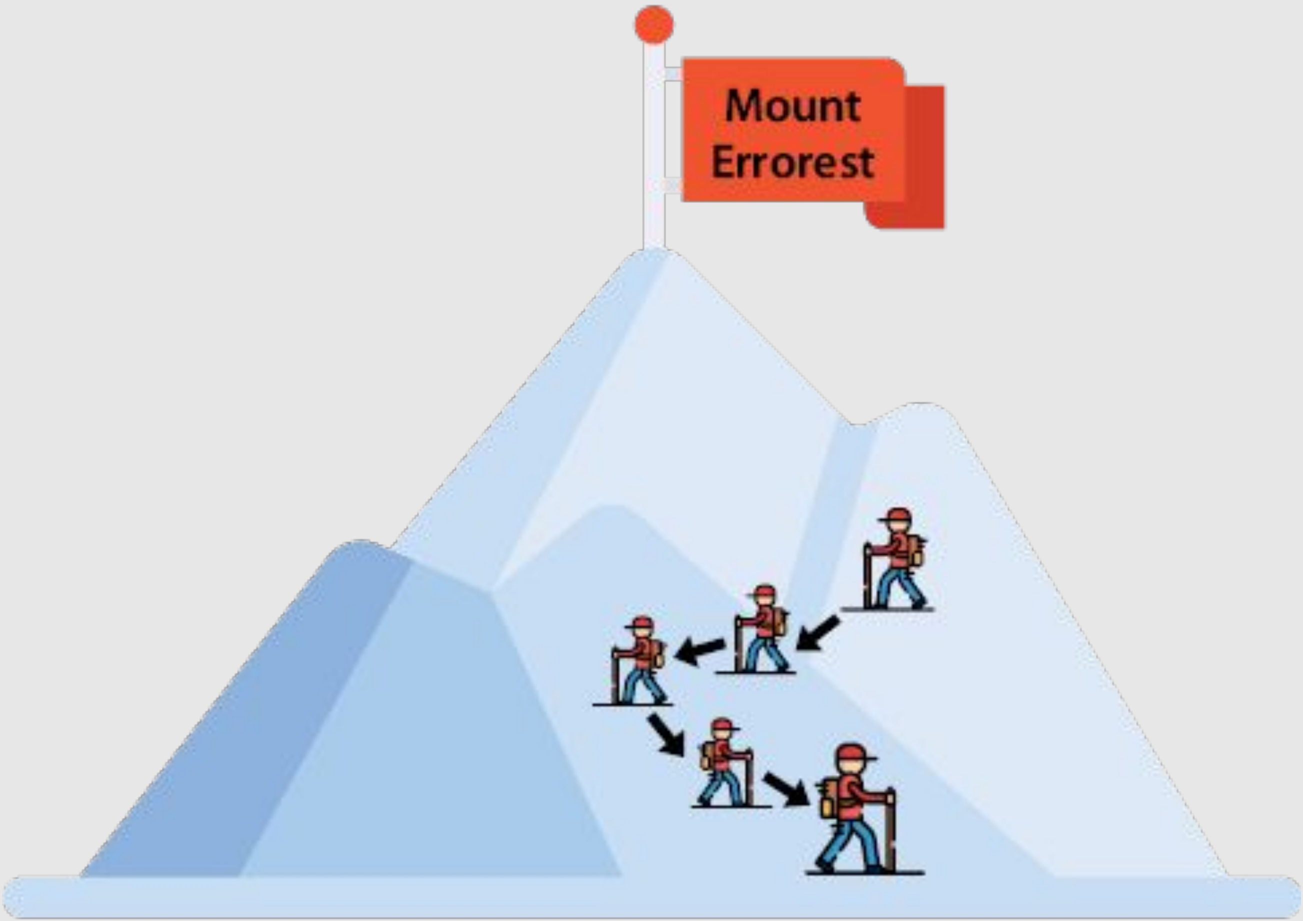
Have some function $J(\theta_0,\theta_1)$

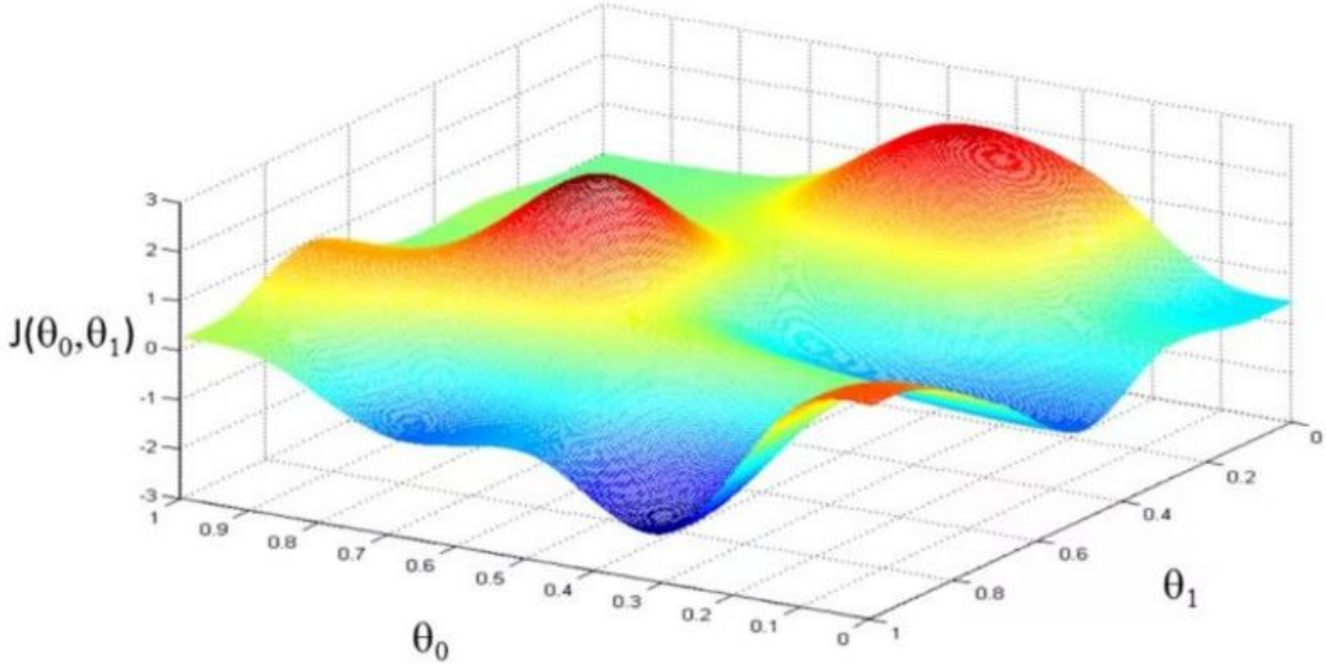
Want $\underset{\theta_0,\theta_1}{\text{minimize}}\, J(\theta_0,\theta_1)$

Outline:

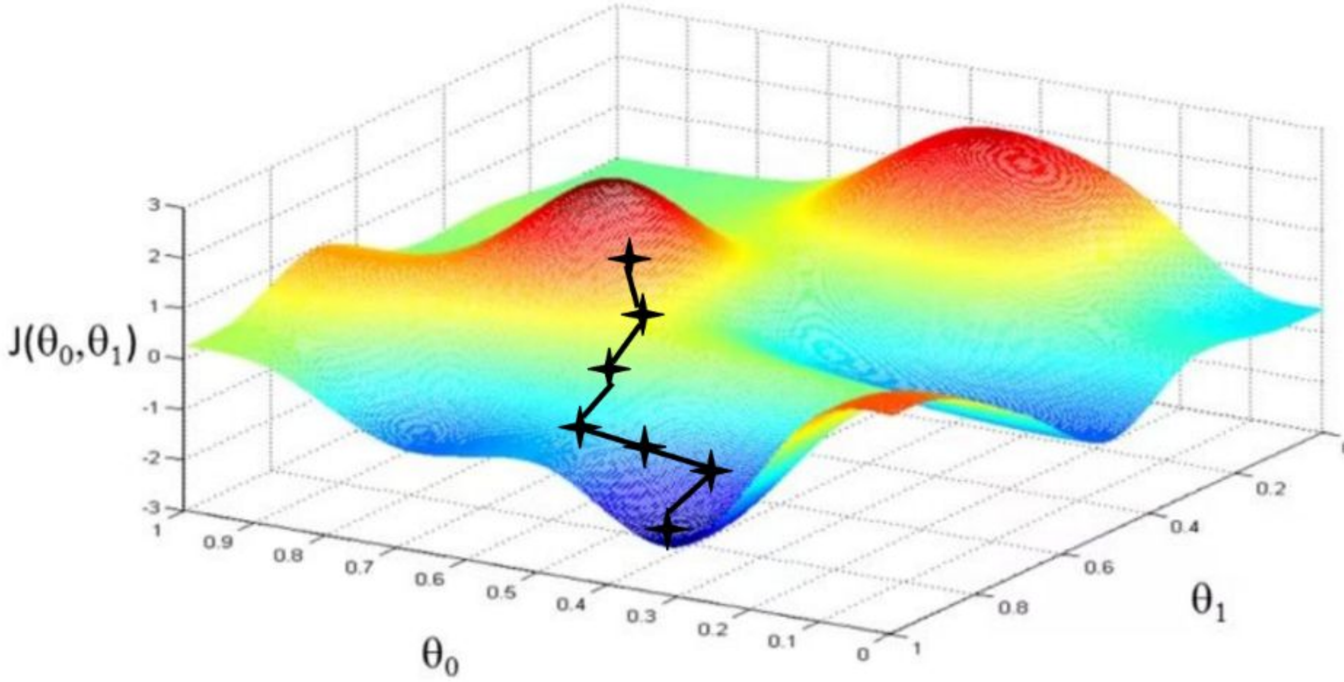
- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0,\theta_1)$ until we hopefully end up at a minimum


Mount
Errorest

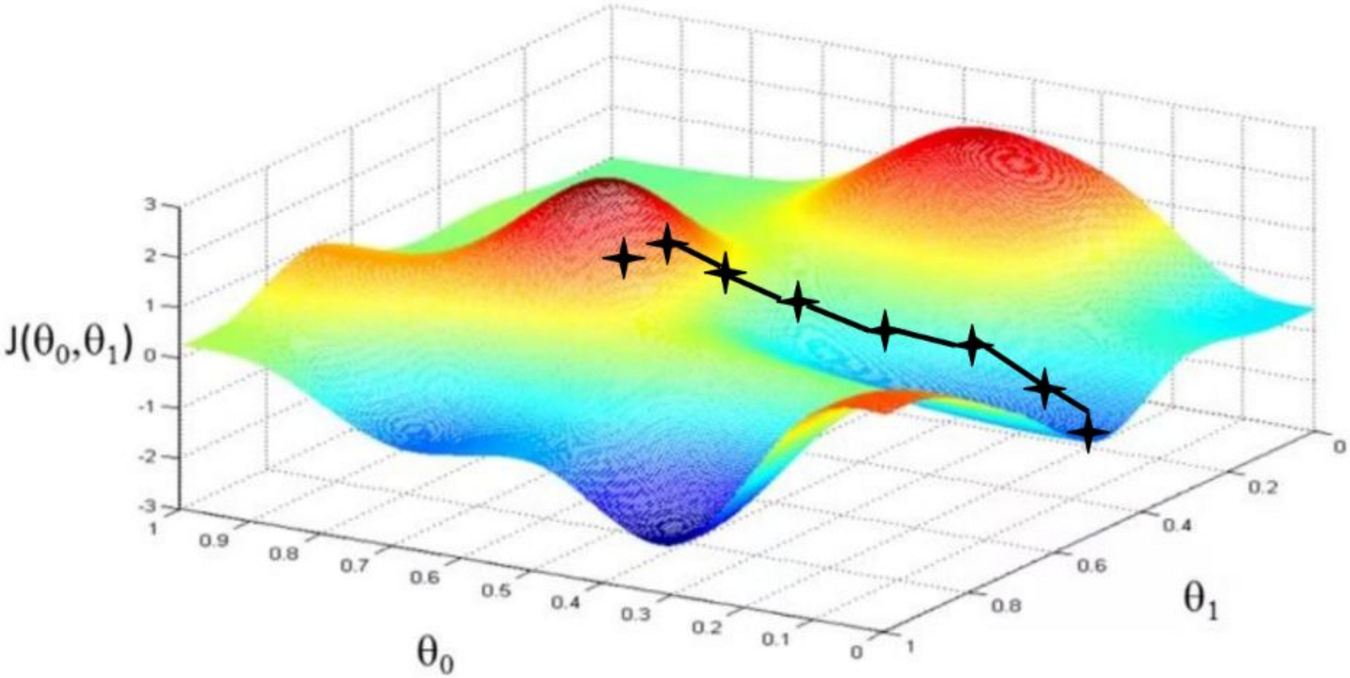

Mount
Errorest



$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1



$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1



J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1

Gradient Descent algorithm

repeat until convergence {

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

(simultaneously update $j = 0$ and $j = 1$)

}

Learning rate

Derivative term

Gradient Descent algorithm

repeat until convergence {

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

}

Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

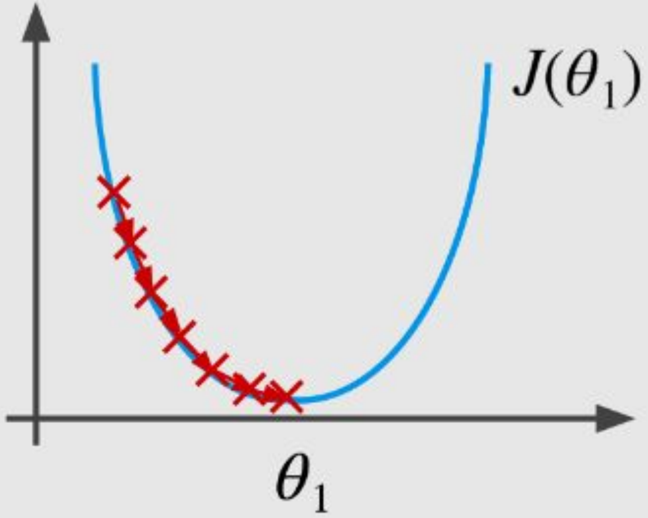
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := \text{temp0}$$

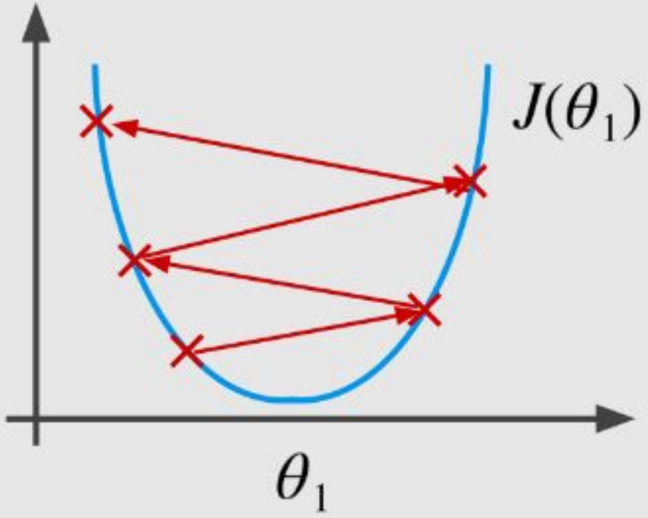
$$\theta_1 := \text{temp1}$$

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

If α is too small, gradient descent can be slow.



If α is too large, gradient descent can be overshoot the minimum. It may fail to converge, or even diverge.



Gradient Descent algorithm

repeat until convergence {

$$\left.\begin{aligned}
\theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \\
\theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}
\end{aligned}\right\} \text{update } \theta_0 \text{ and } \theta_1 \text{ simultaneously}$$

}

“Batch” Gradient Descent

“Batch”: Each step of gradient descent uses **all the training examples**.

- Stochastic Gradient Descent
- Mini-batch Gradient Descent

“Batch” Gradient Descent

repeat until convergence {

$$\left.\begin{aligned} \theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \\ \theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)} \end{aligned}\right\} \begin{array}{l}\text{update } \theta_0 \text{ and } \theta_1 \\ \text{simultaneously}\end{array}$$

}

Stochastic Gradient Descent

Each step of gradient descent uses **one training example**.

repeat until convergence {

for $i = 1, \ldots, m$ {

$$\theta_0 := \theta_0 - \alpha(h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

}

}

Mini-batch Gradient Descent

Each step of gradient descent uses ***b* training examples**.

Say $b = 10, m = 1000.$
repeat until convergence {

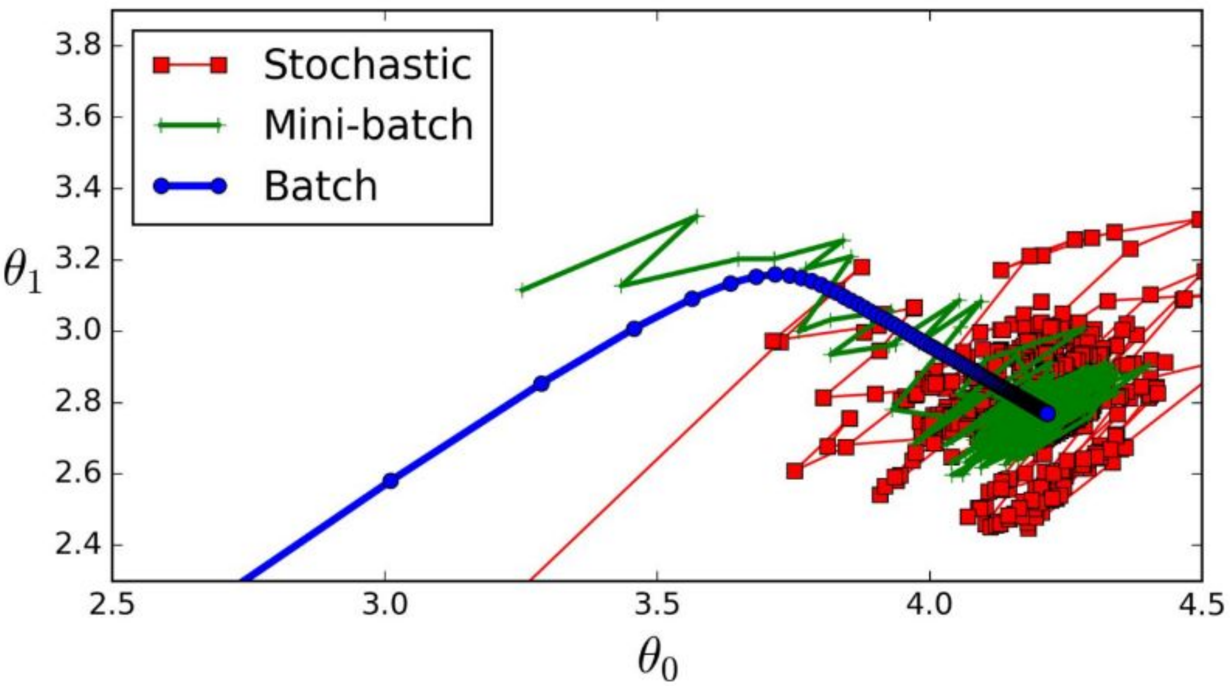
for $i = 1, 11, 21 \ldots, 991$ {

$$\theta_0 := \theta_0 - \alpha \frac{1}{10} \sum_{i=k}^{i+9} (h_\theta(x^{(k)}) - y^{(k)})$$

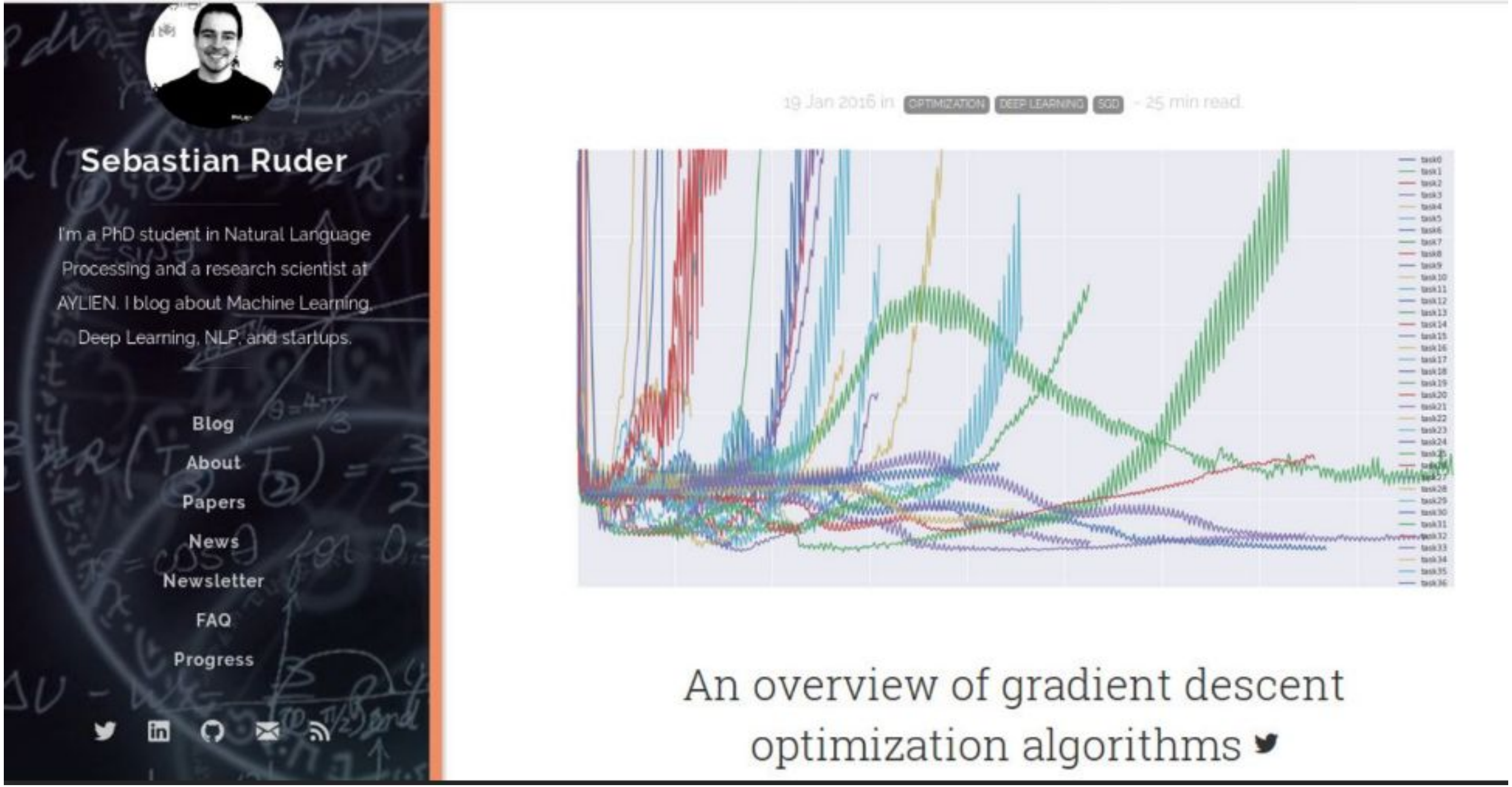
$$\theta_1 := \theta_1 - \alpha \frac{1}{10} \sum_{i=k}^{i+9} (h_\theta(x^{(k)}) - y^{(k)})\, x^{(k)}$$

} }

Batch vs. Stochastic vs. Mini-batch



http://ruder.io/optimizing-gradient-descent



Referencias

Machine Learning Books

- Python for Data Science, Yuli Vasiliev, Chap 12
- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 2 & 4
- Pattern Recognition and Machine Learning, Chap. 3

Machine Learning Courses

- https://www.coursera.org/learn/machine-learning, Week 1 & 2
- https://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html
- https://serrano.academy/espanol/ Minicurso de ML en español

Las diapositivas están basadas parcialmente en el curso de Machine Learning de la Prof. Sandra Ávila

Linear Regression with Multiple Variables

Prof. Rosa Paccotacya Yanque

“Batch” Gradient Descent

“Batch”: Each step of gradient descent uses **all the training examples**.

- Stochastic Gradient Descent
- Mini-batch Gradient Descent

Epochs: One epoch is usually defined to be **ONE** complete run through **ALL** of the training data.

Batch Size: Total number of training examples present in a **SINGLE** batch.

Iterations: The number of batches needed to complete **ONE** epoch.

Epochs & Batch size & Iterations

Let's say we have 10,000 training examples that we are going to use.

> We can divide the dataset of 10,000 examples into **batches of 16** then it will take **625 iterations** to complete **1 epoch**.

Multiple Linear Regression

Dependent variable (DV) Independent variables (IVs)

$$y = b_0 + b_1{*}x_1 + b_2{*}x_2 + \ldots + b_n{*}x_n$$

Multiple ~~Variables~~ Features

Size in feet^2 x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price (\$) in 1000's y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	2	36	178
...	...	...	...	...

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example

$x_j^{(i)}$ = value of features j in i^{th} training example

Hypothesis

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

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

Hypothesis

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

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

$$h_\theta(x) = 80 + 0.1 x_1 + 10 x_2 + 3 x_3 - 2 x_4$$

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

For convenience of notation, define $x_0 = 1$.

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

$$\begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$h_\theta(x) = \theta^T x$$

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

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

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

Gradient Descent:

repeat {

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

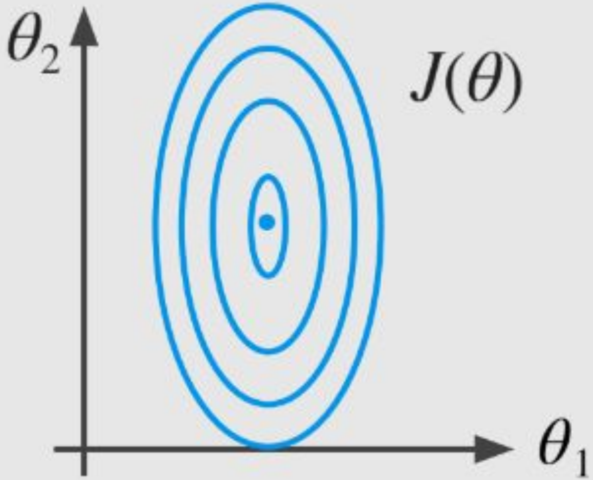
} (simultaneously update for every $j = 0, 1, \ldots, n$)

Feature Scaling

Idea: Make sure features are on similar scale.

E.g. x_1= size ($0-2000$ feet^2)

x_2= number of bedrooms ($1-5$)

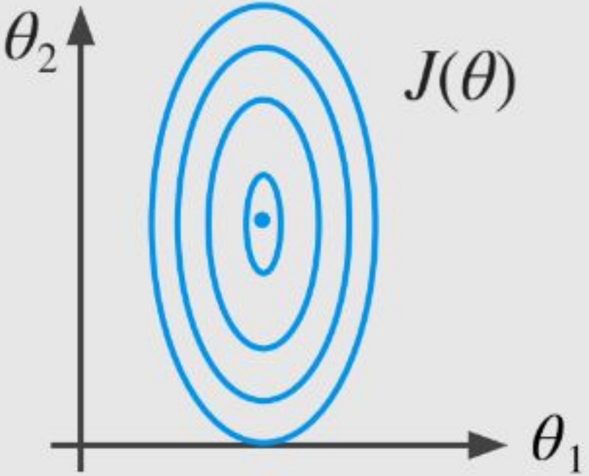


Feature Scaling

Idea: Make sure features are on similar scale.

E.g. x_1 = size ($0-2000$ feet^2)

x_2 = number of bedrooms ($1-5$)



$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$

Learning rate

Gradient Descent

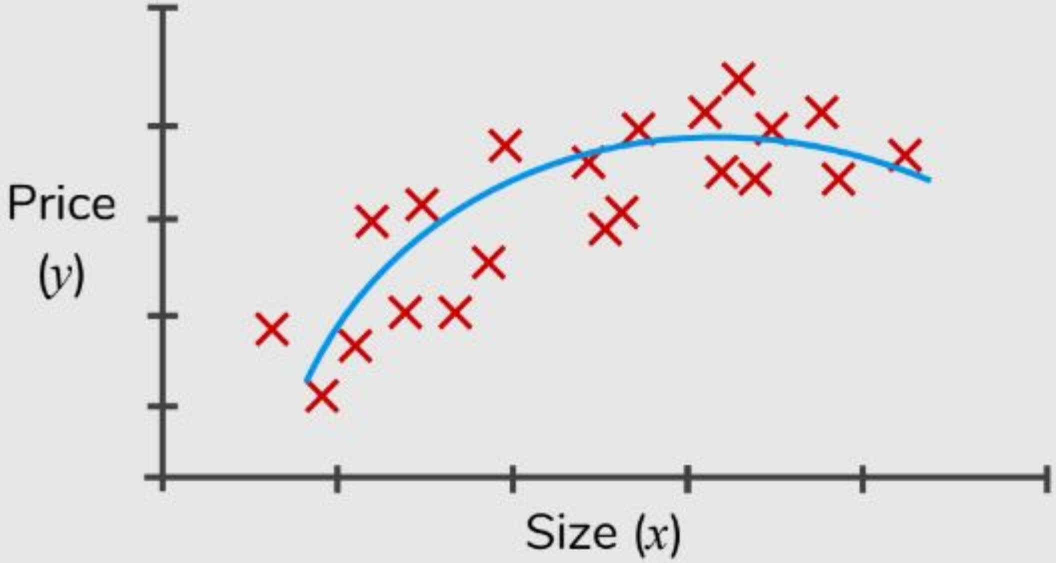
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge.

To choose α, try

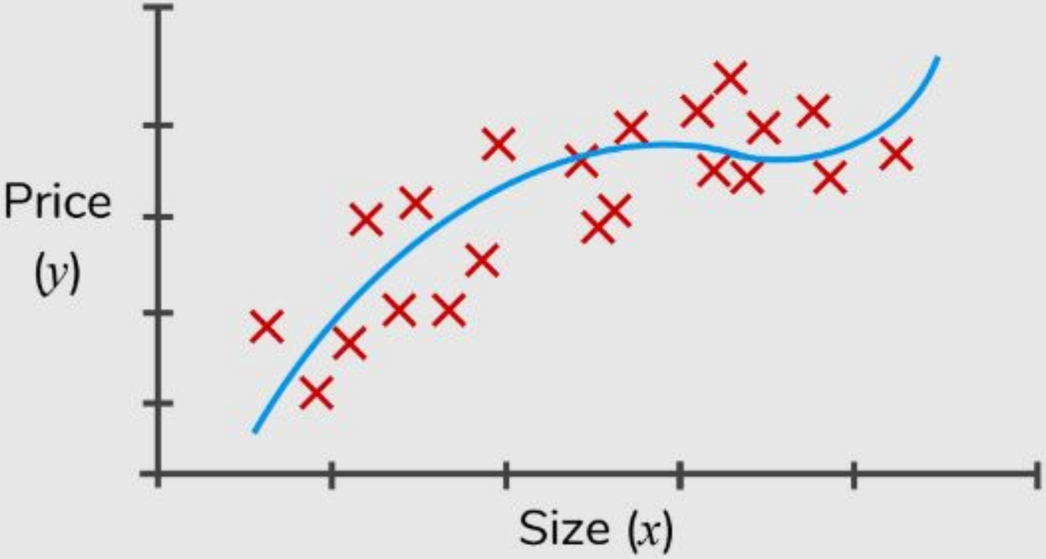
$$\ldots, 0.001, \ldots, 0.01, \ldots, 0.1, \ldots, 1, \ldots$$

Polynomial Regression



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

Polynomial Regression

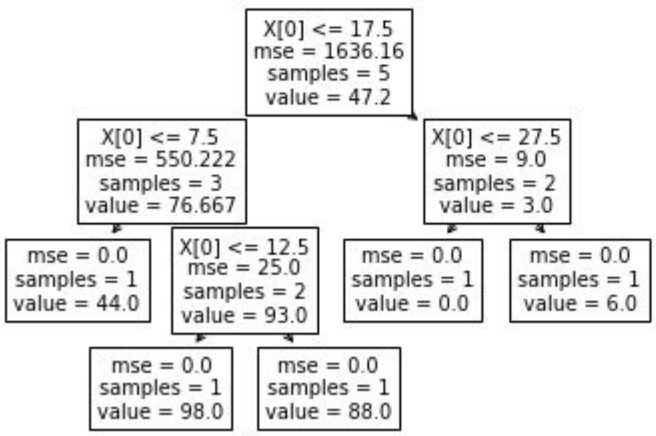


$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

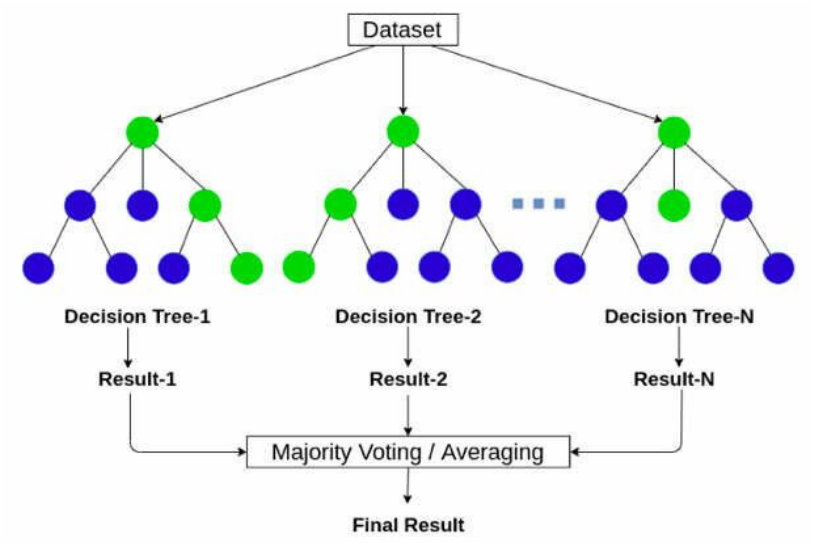
Other Regression Algorithms

- Decision Tree Regressor: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.DecisionTreeRegressor



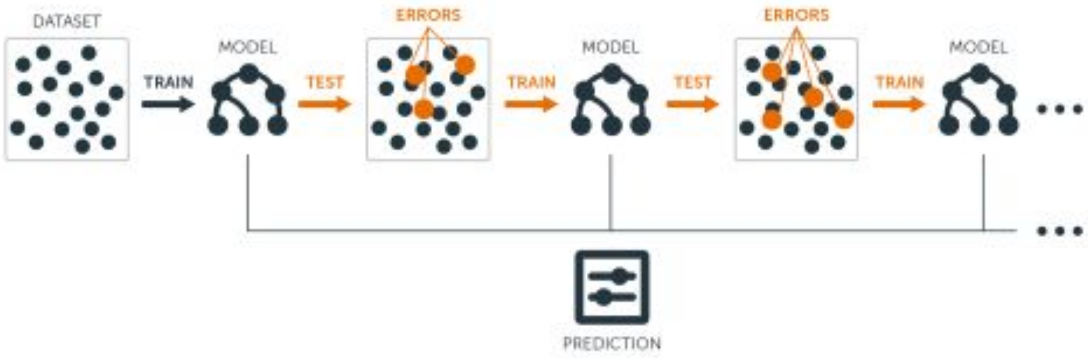
Other Regression Algorithms

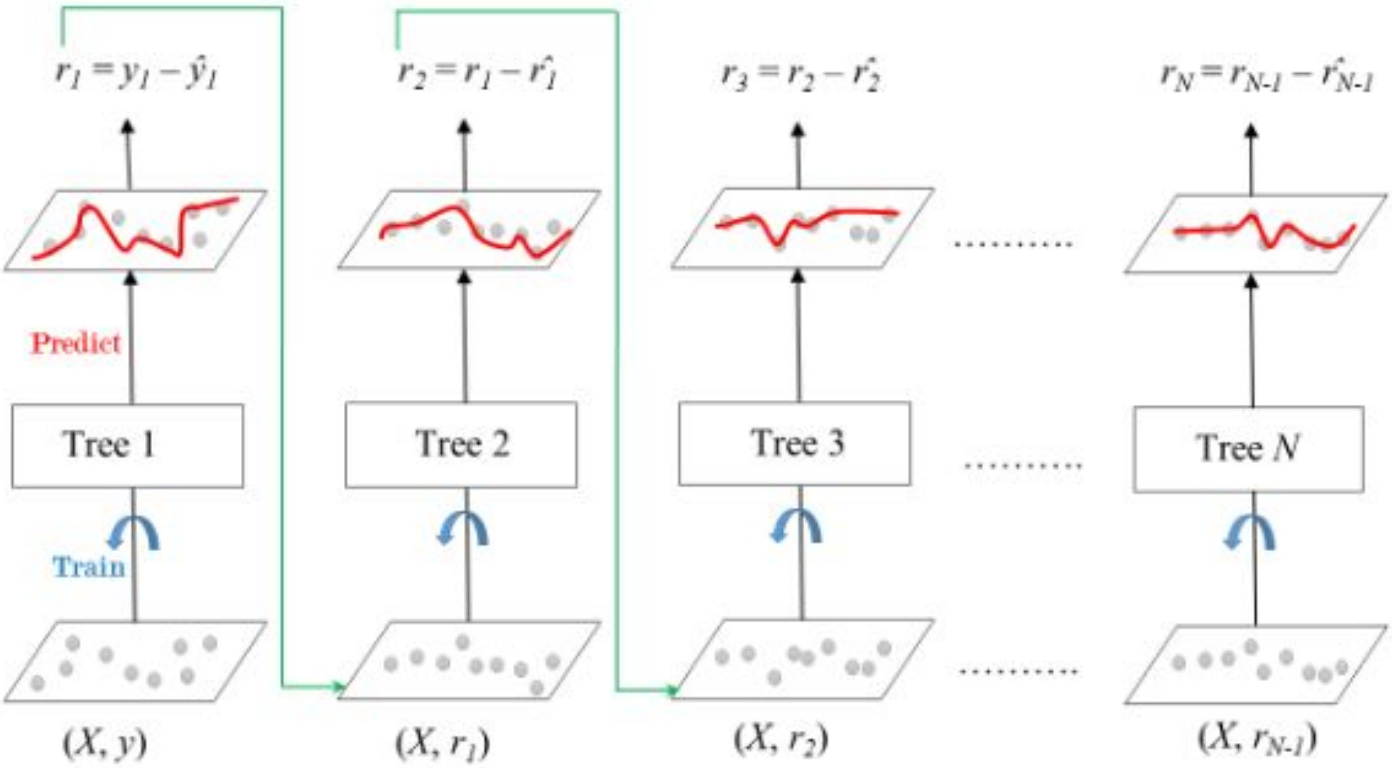
- Random Forest Regressor: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html



Other Regression Algorithms

- Gradient Boosting Regressor: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html





y(pred) = y1 + (eta * r1) + (eta * r2) + ... + (eta * rN)

Notebook:

Regresion.ipynb

References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Geron, 3rd-edition, Chap. 4, 6 & 7
- Pattern Recognition and Machine Learning, Bishop, Chap. 3 , 14.3, 14.4, 14.5

Machine Learning Courses

- https://www.coursera.org/learn/machine-learning, Week 1 & 2
- https://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html

Slides are partially based on Prof. Sandra Ávila’s Machine Learning course