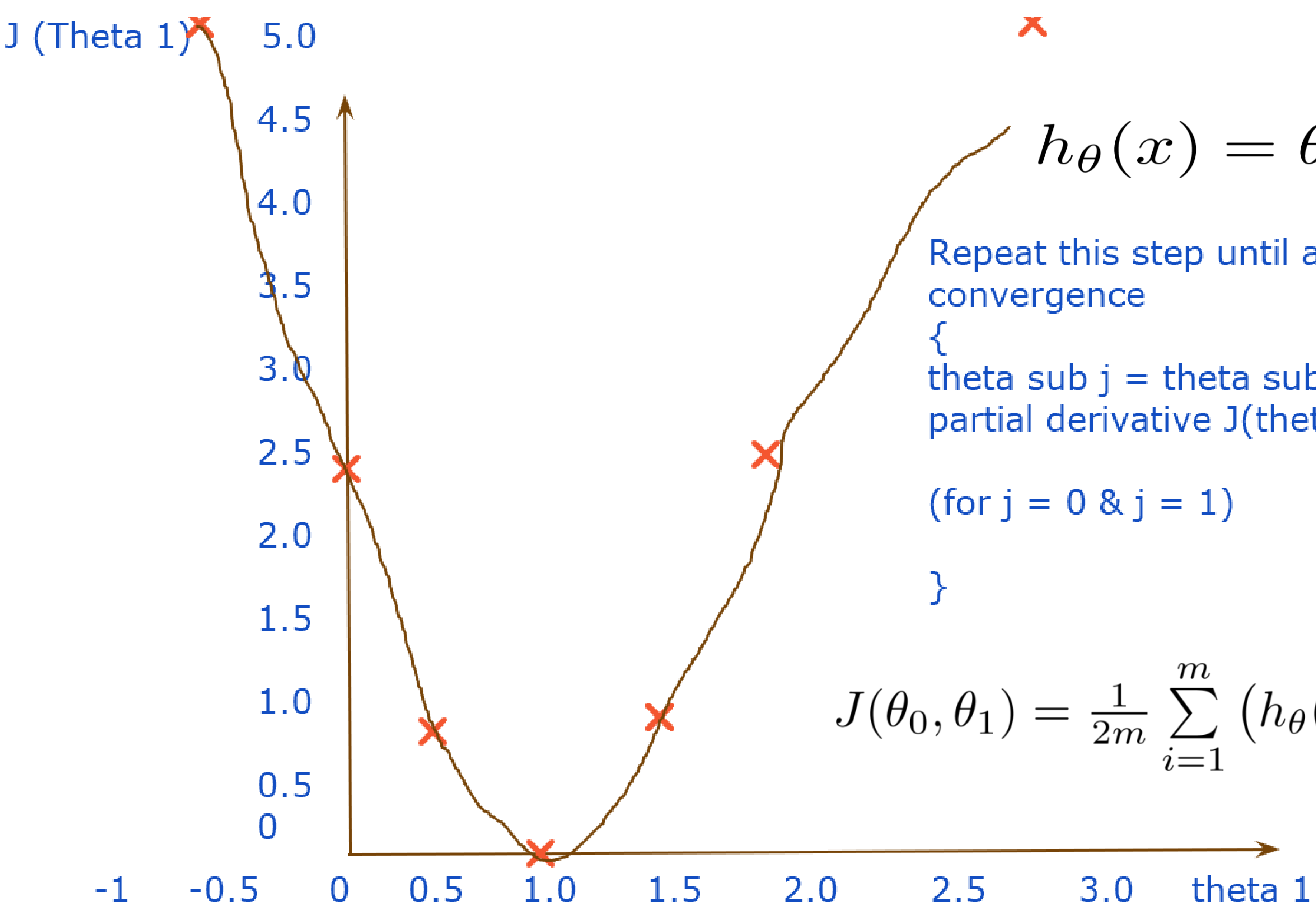


$$h_\theta(x) = \theta_1 x$$

Repeat this step until achieve convergence

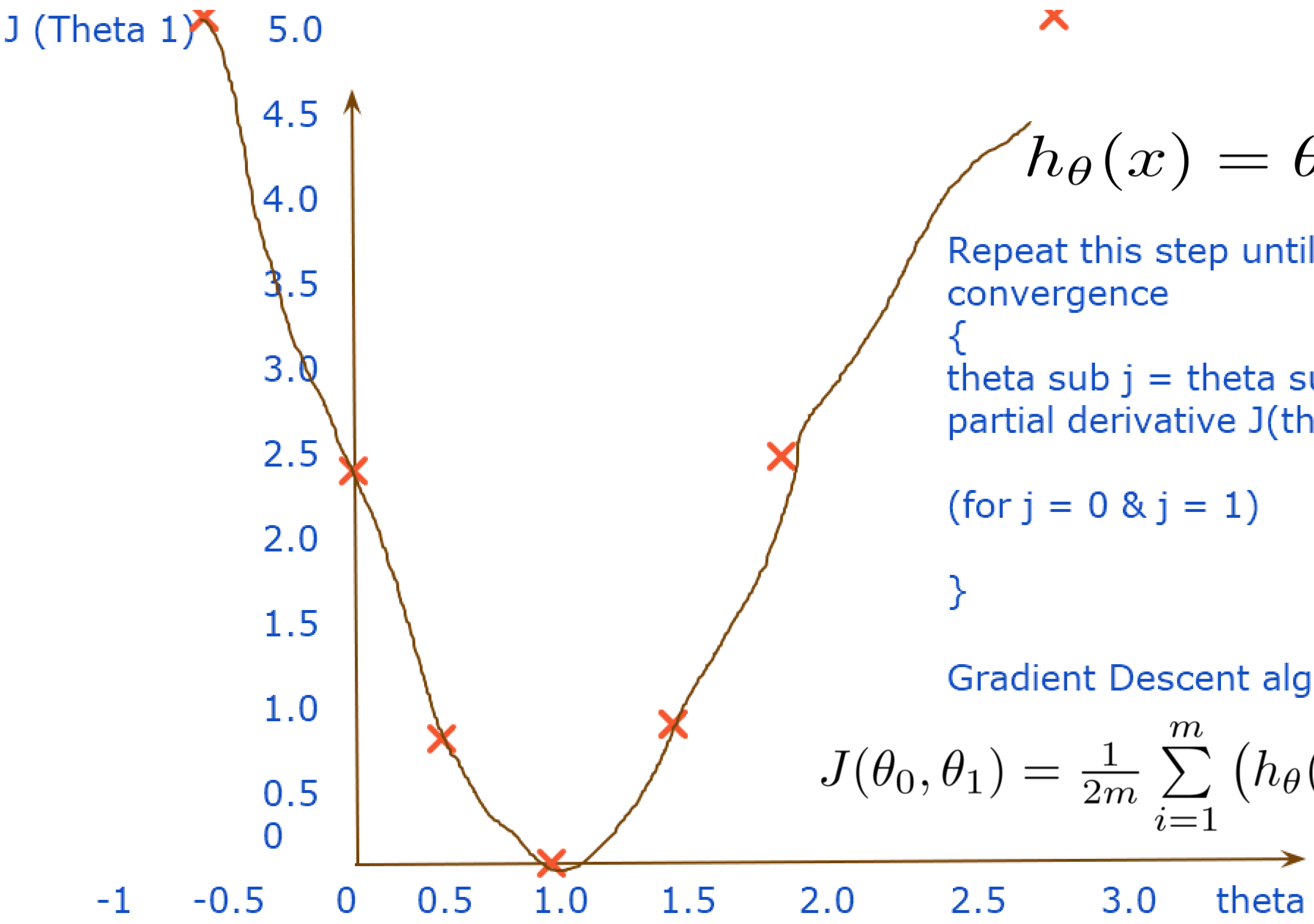
{

theta sub j = theta sub j - alpha partial derivative J(theta's)/j

(for j = 0 & j = 1)

}

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



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

Repeat this step until achieve convergence

{

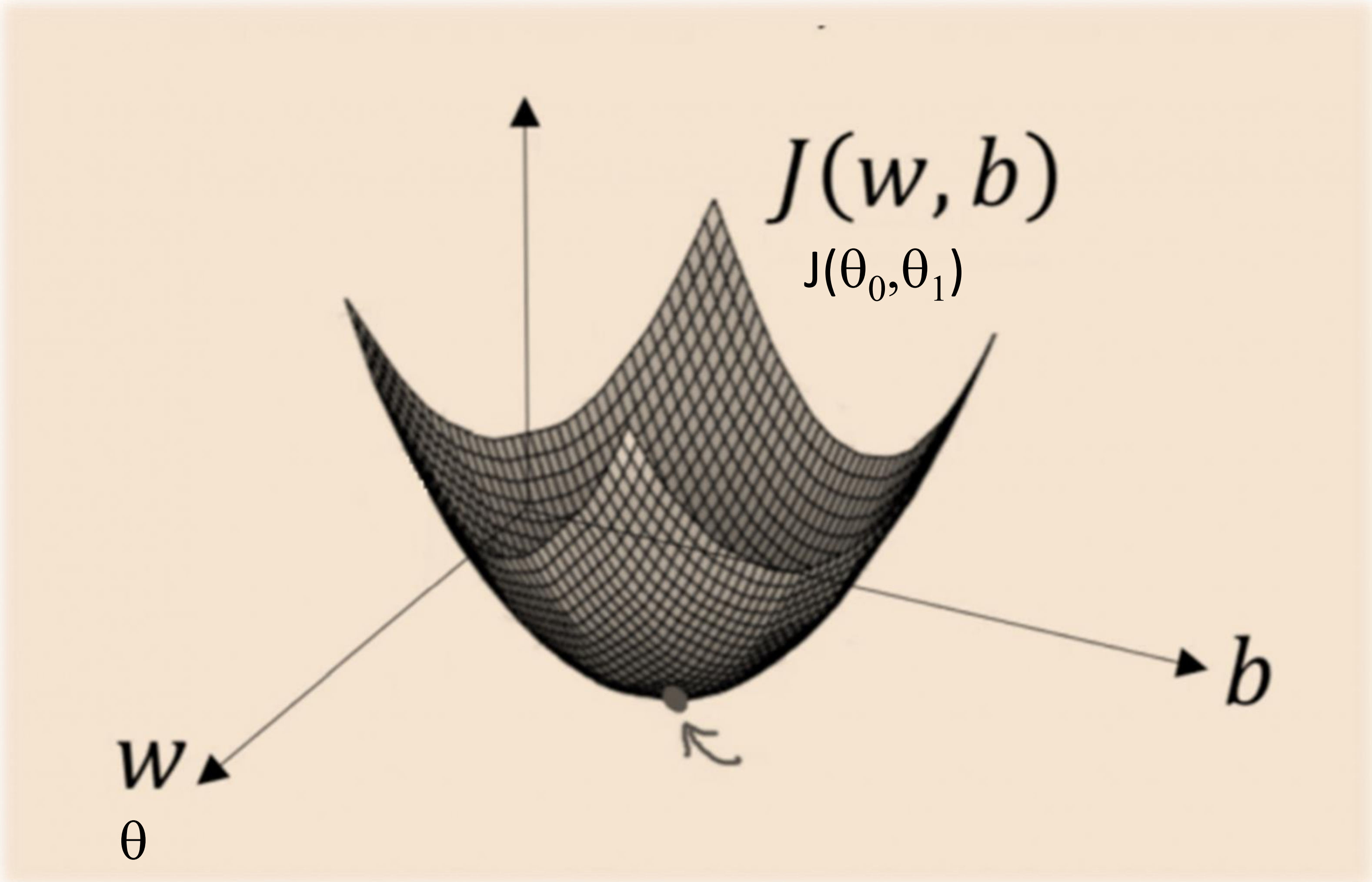
theta sub j = theta sub j - alpha partial derivative J(theta's)/j

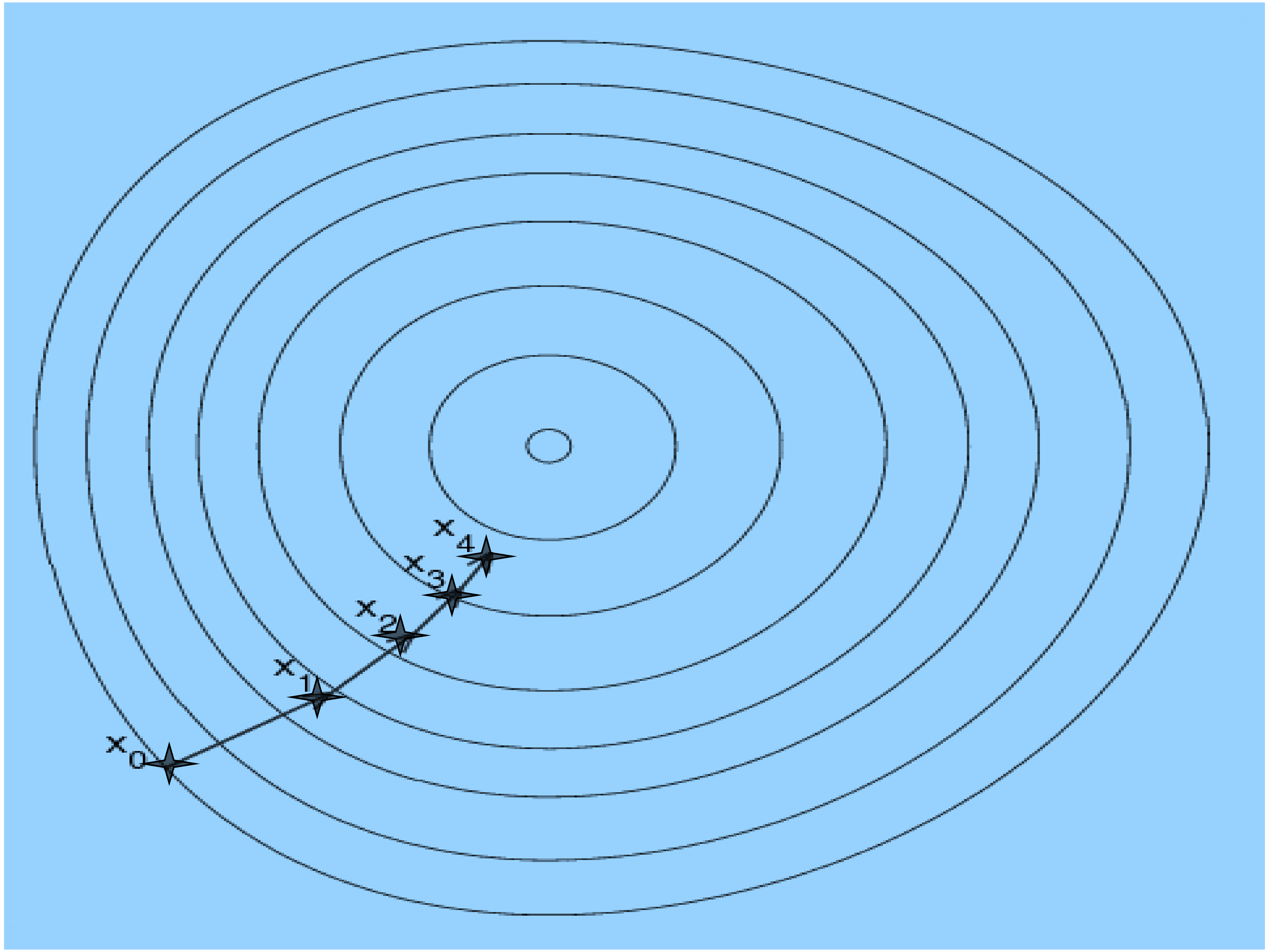
(for j = 0 & j = 1)

}

Gradient Descent algorithm

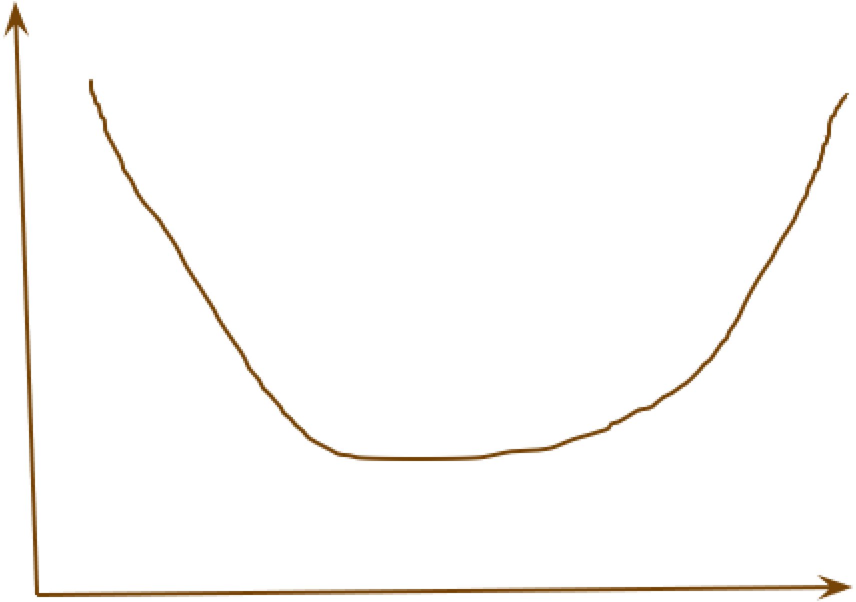
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$


$J(w,b)$
$J(\theta_0,\theta_1)$
b
w
θ

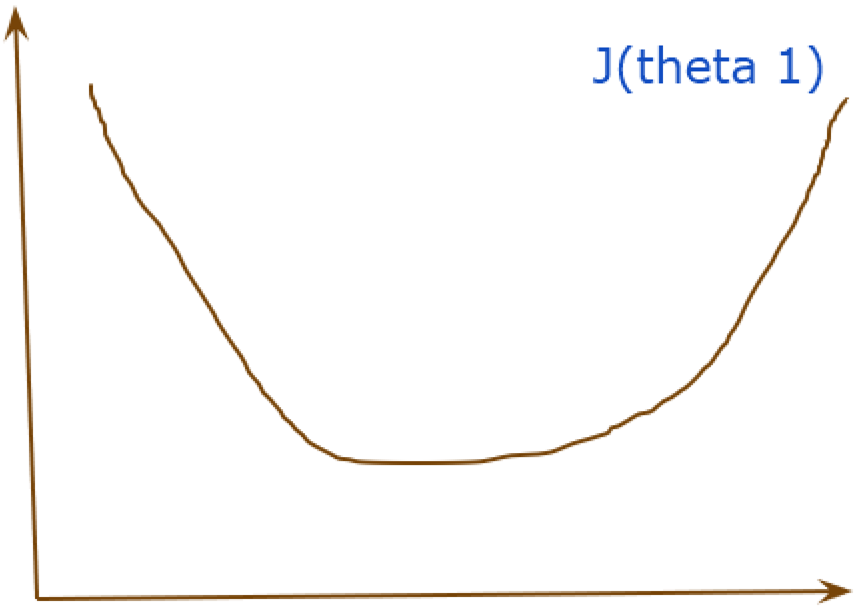

x_0
x_1
x_2
x_3
x_4

Gradient Descent algorithm

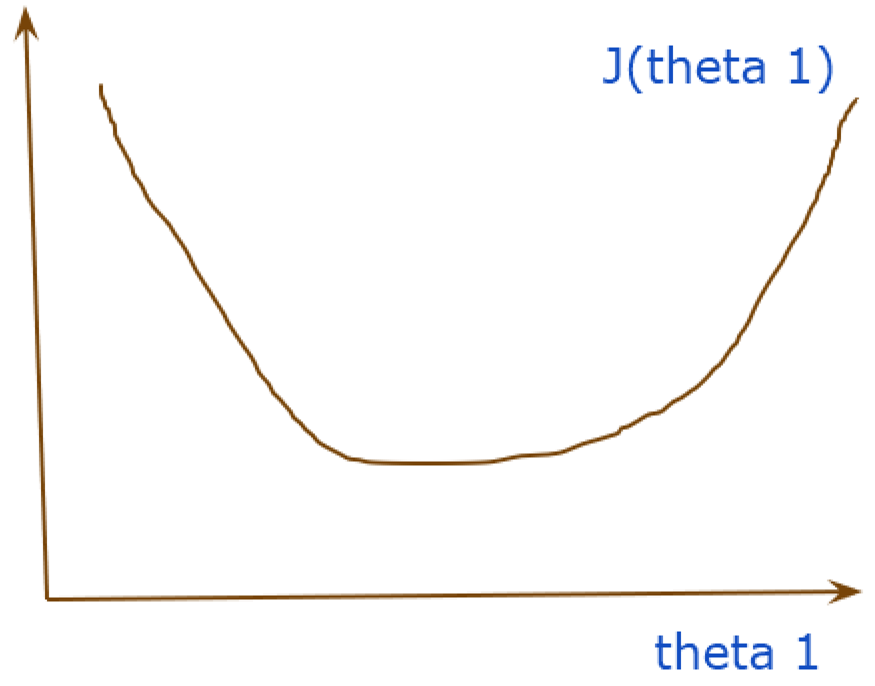
Gradient Descent algorithm



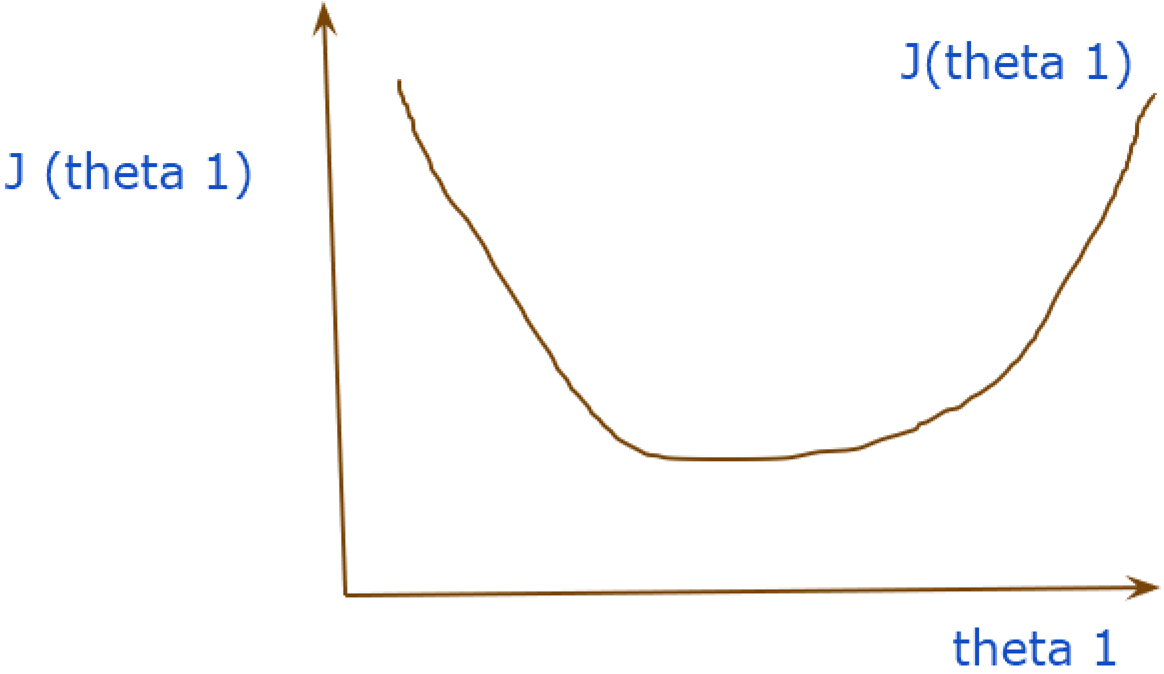
Gradient Descent algorithm



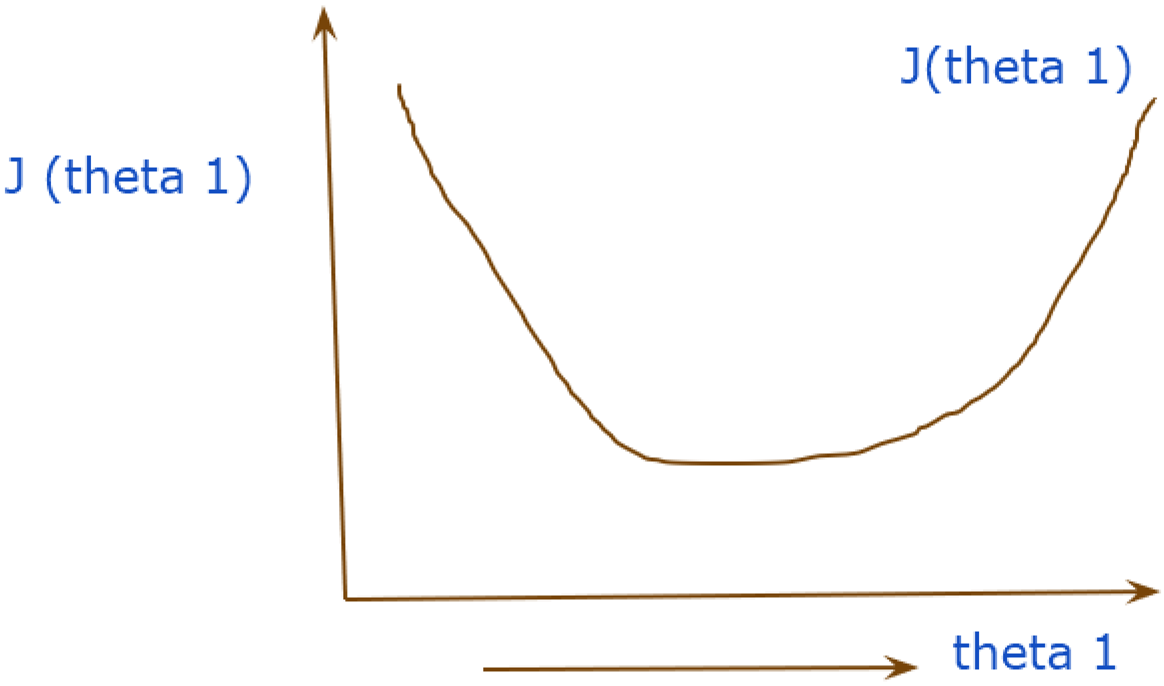
Gradient Descent algorithm



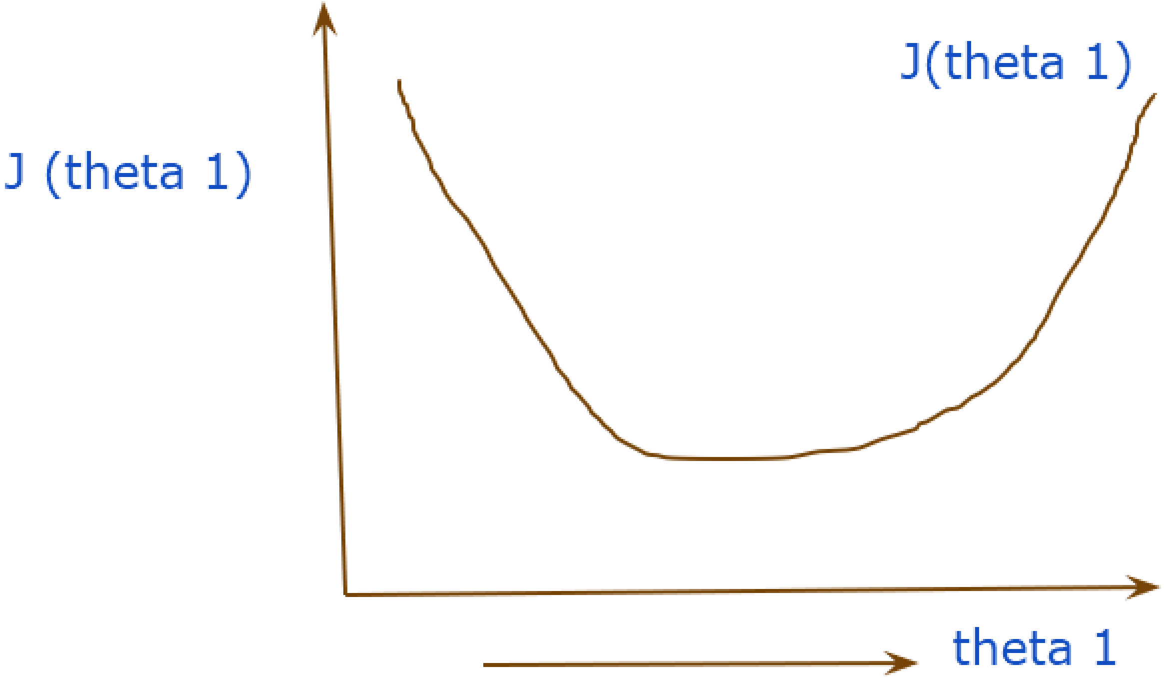
Gradient Descent algorithm



Gradient Descent algorithm

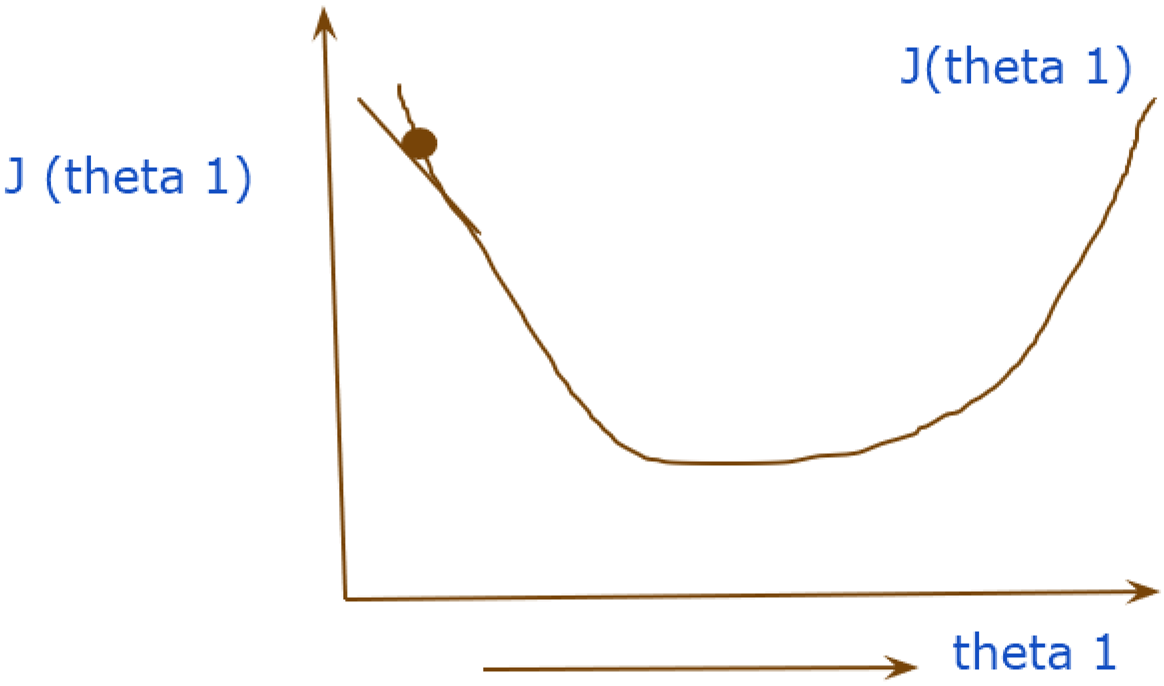


Gradient Descent algorithm



theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

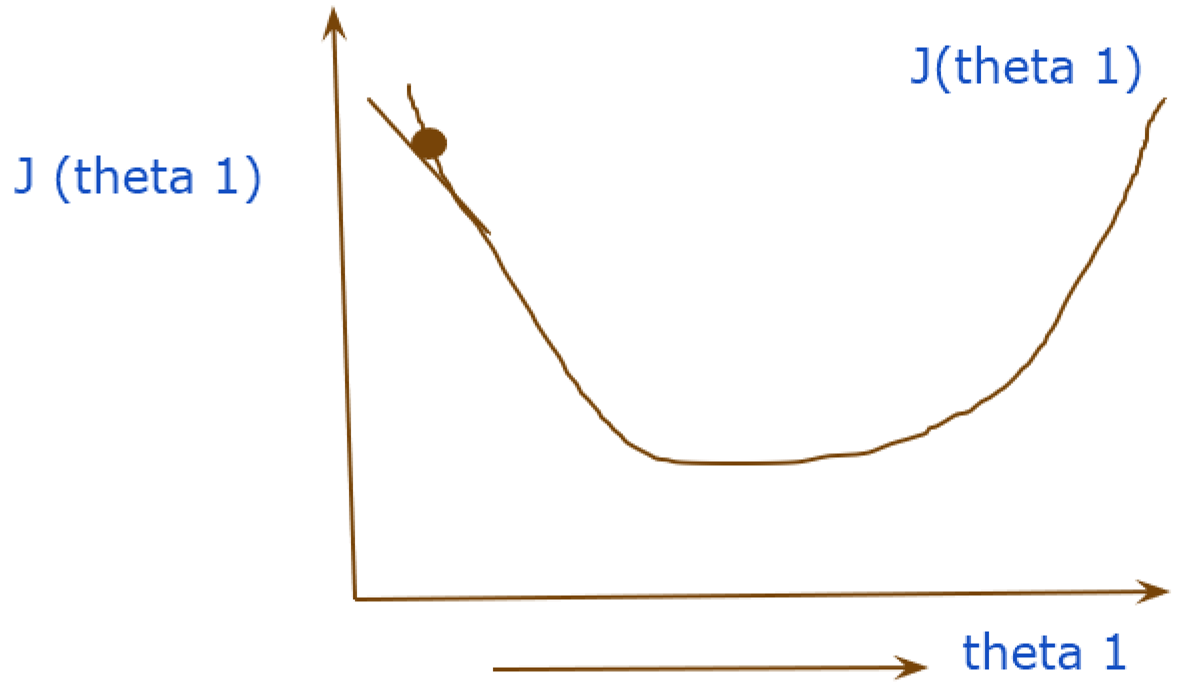
Gradient Descent algorithm



theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

Gradient Descent algorithm

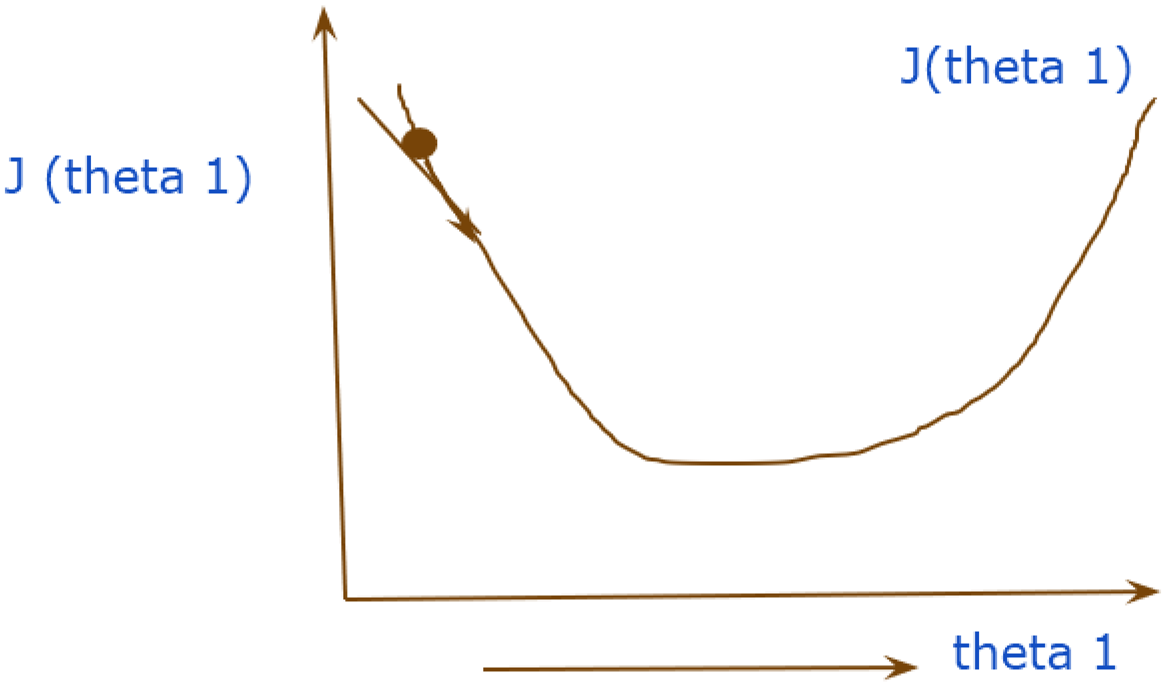


alpha: learning rate

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

Gradient Descent algorithm



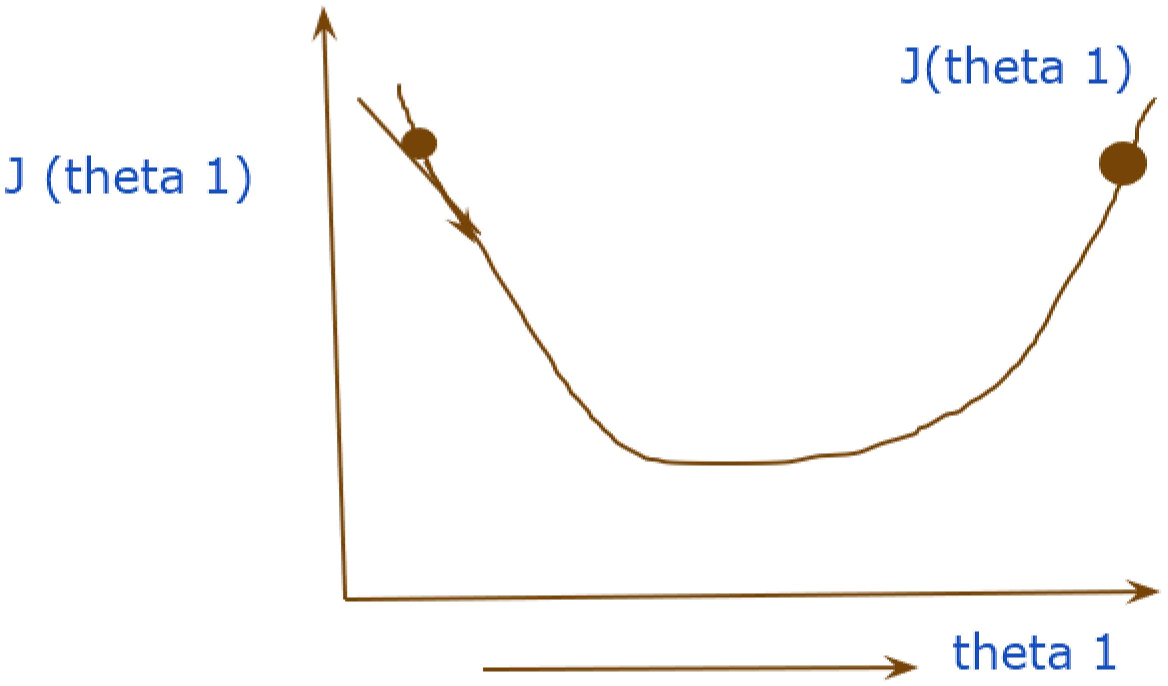
alpha: learning rate

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



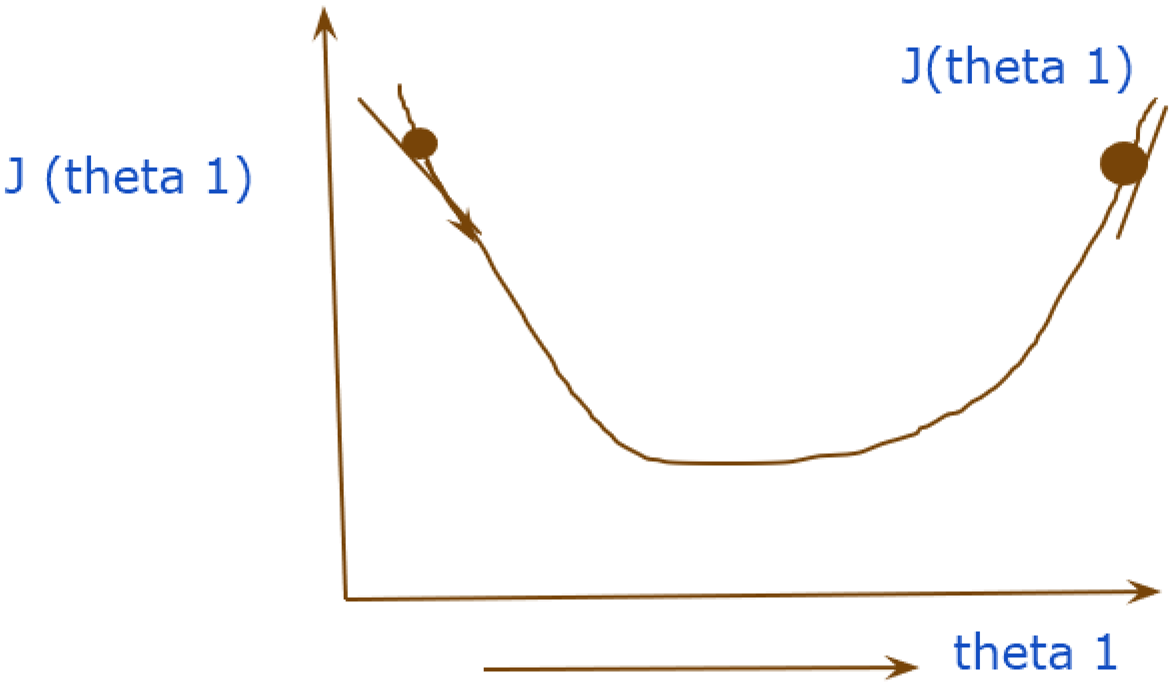
alpha: learning rate

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



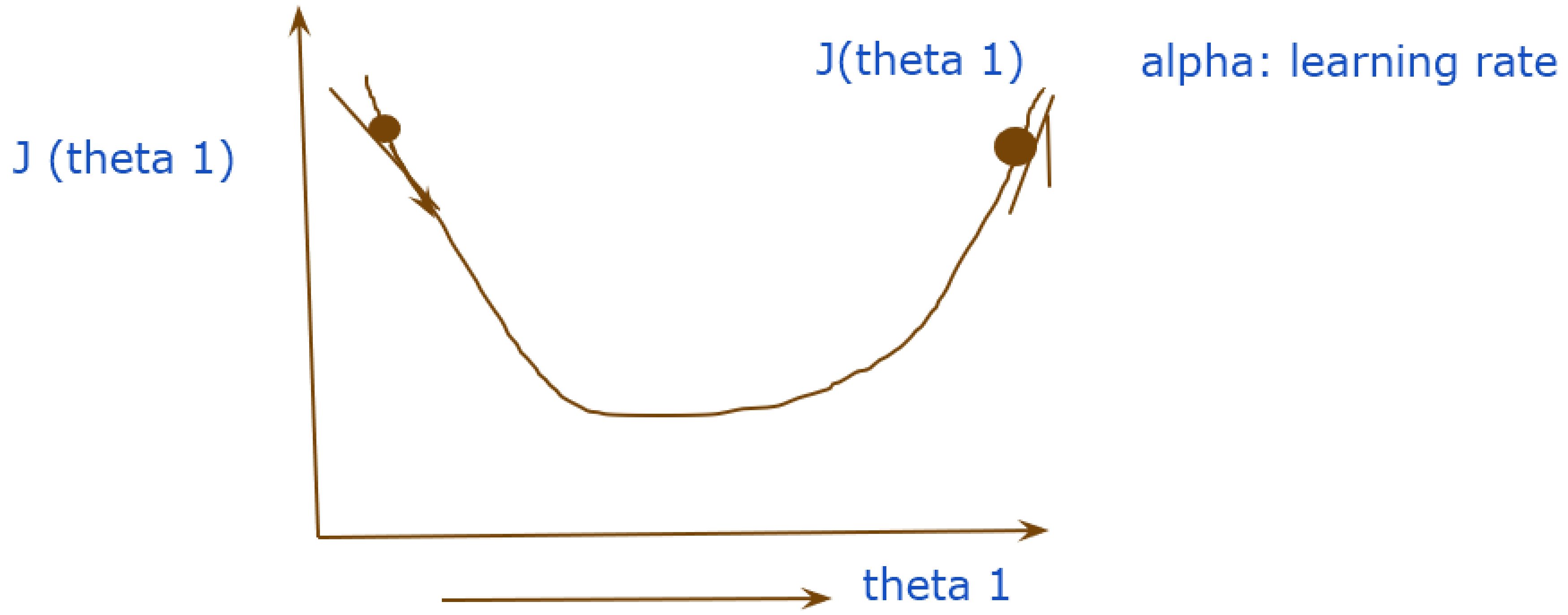
alpha: learning rate

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm

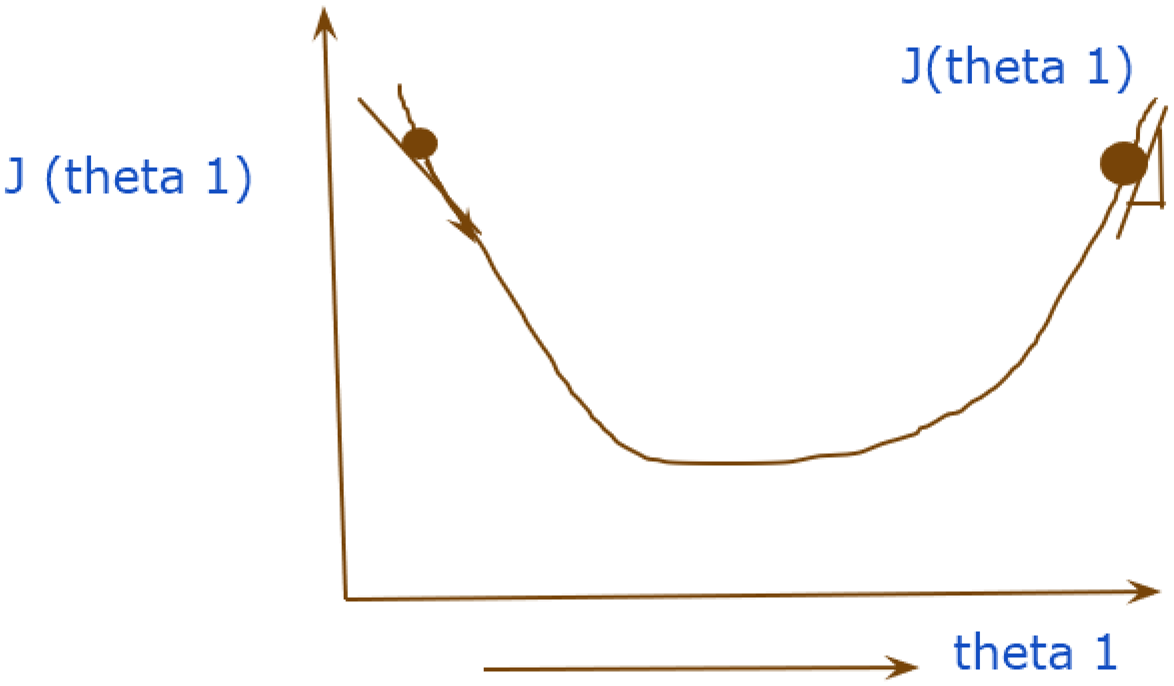


theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



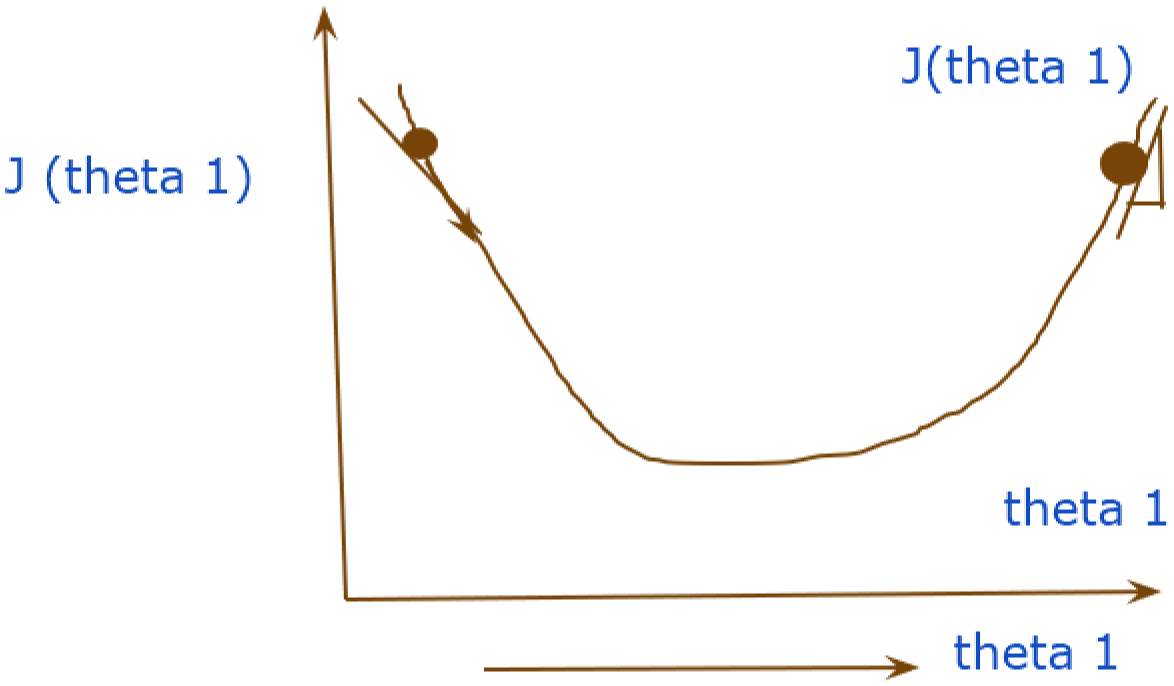
alpha: learning rate

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



alpha: learning rate

theta 1:= theta 1 - alpha (positive slope)

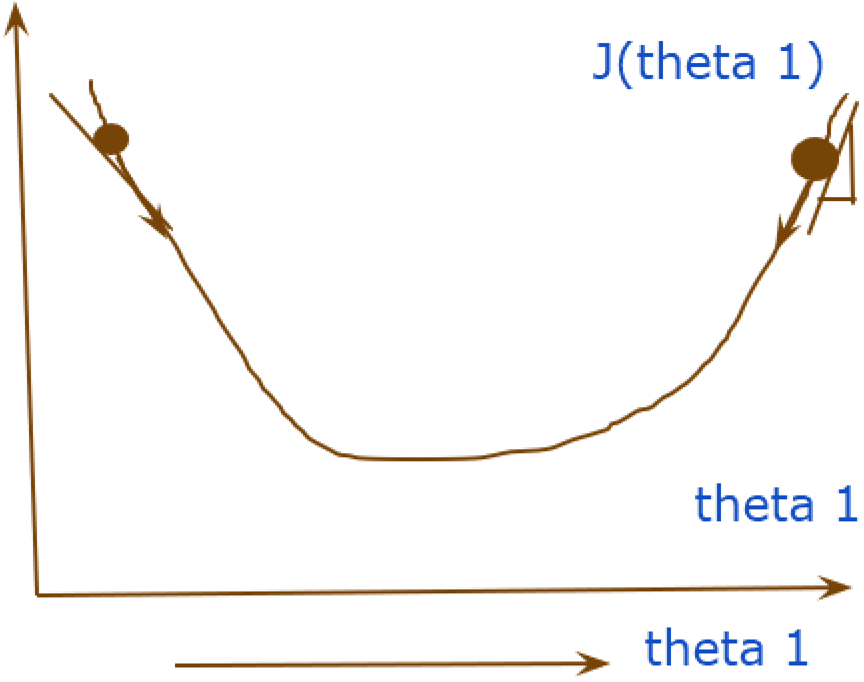
theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm

J (theta 1)



J(theta 1)

alpha: learning rate

theta 1:= theta 1 - alpha (positive slope)

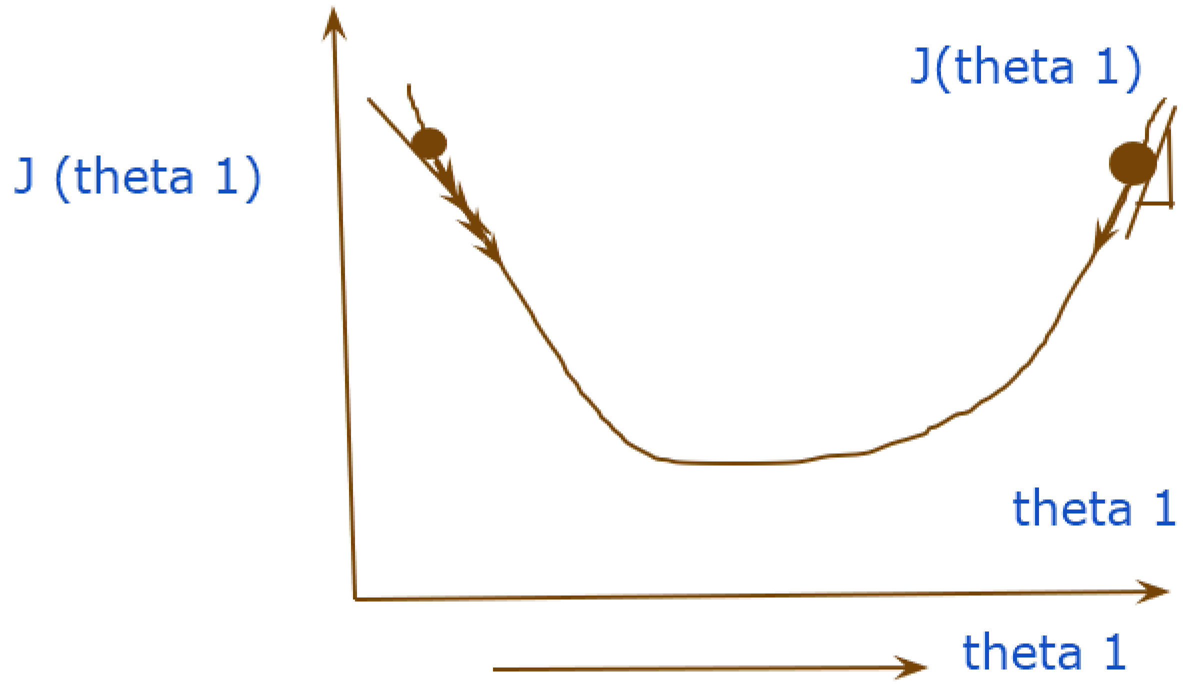
theta 1

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



alpha: learning rate

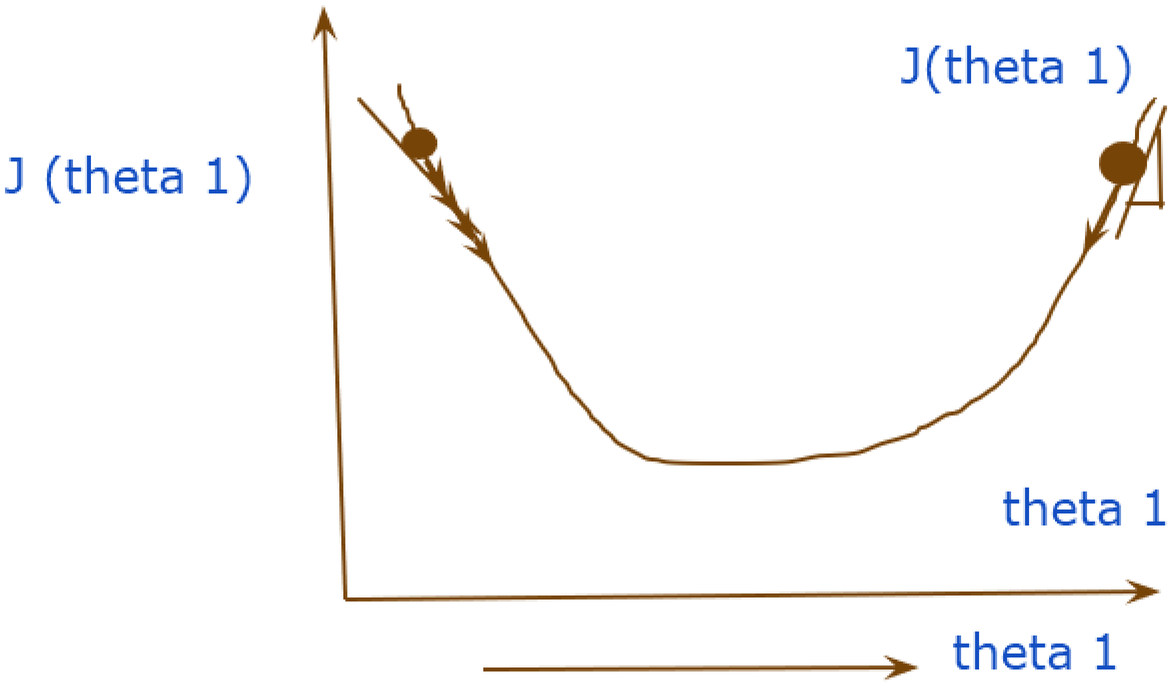
theta 1:= theta 1 - alpha (positive slope)

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



alpha: learning rate

alpha: very small

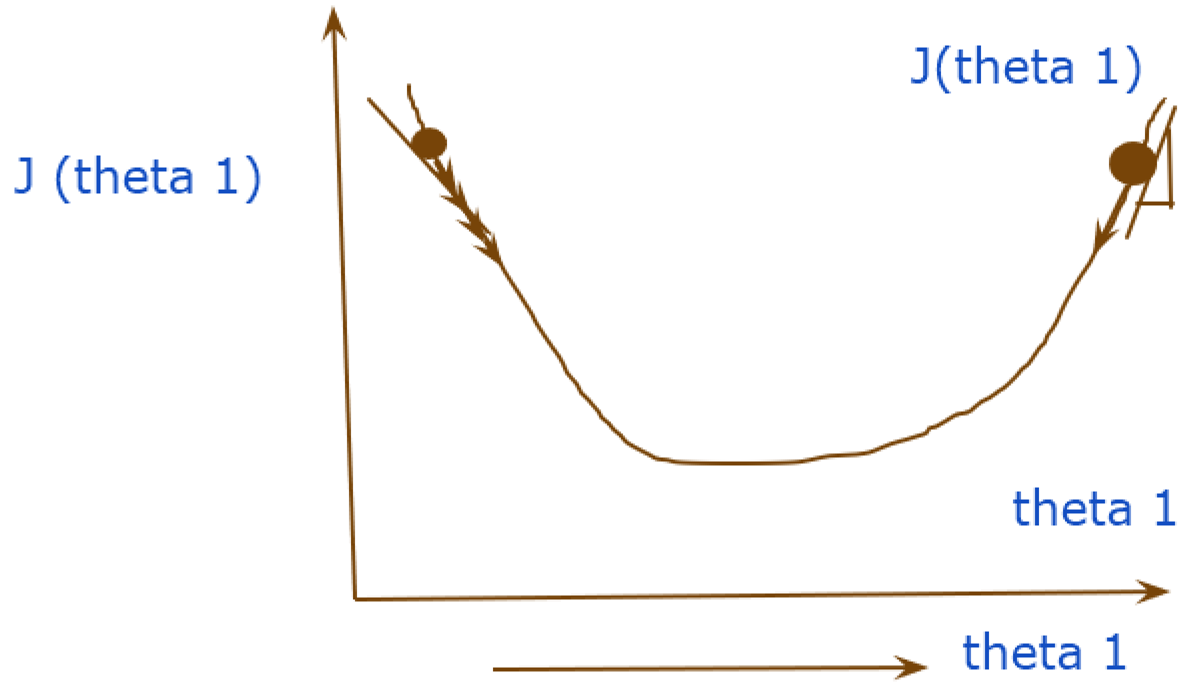
theta 1:= theta 1 - alpha (positive slope)

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



alpha: learning rate

alpha: very small

it will be very slow

theta 1:= theta 1 - alpha (positive slope)

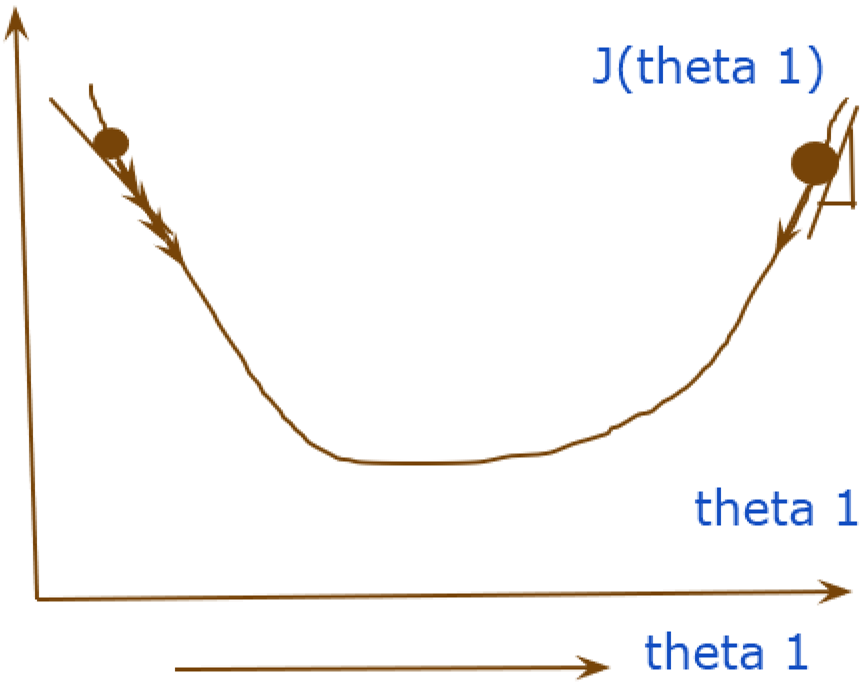
theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm

J (theta 1)



theta 1

alpha: learning rate

alpha: very small

it will be very slow

alpha: very high

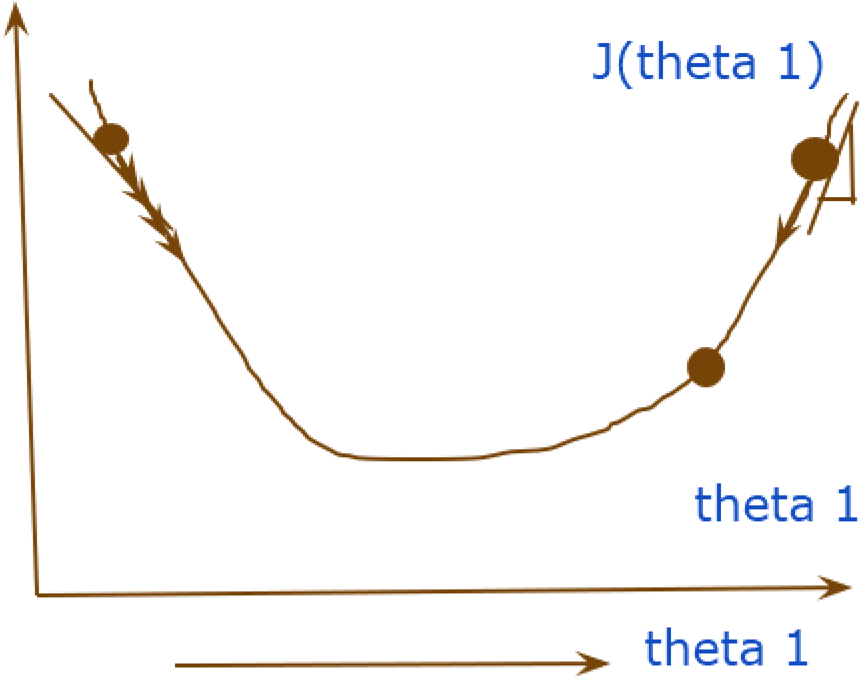
theta 1:= theta 1 - alpha (positive slope)

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



alpha: learning rate

alpha: very small

it will be very slow

alpha: very high

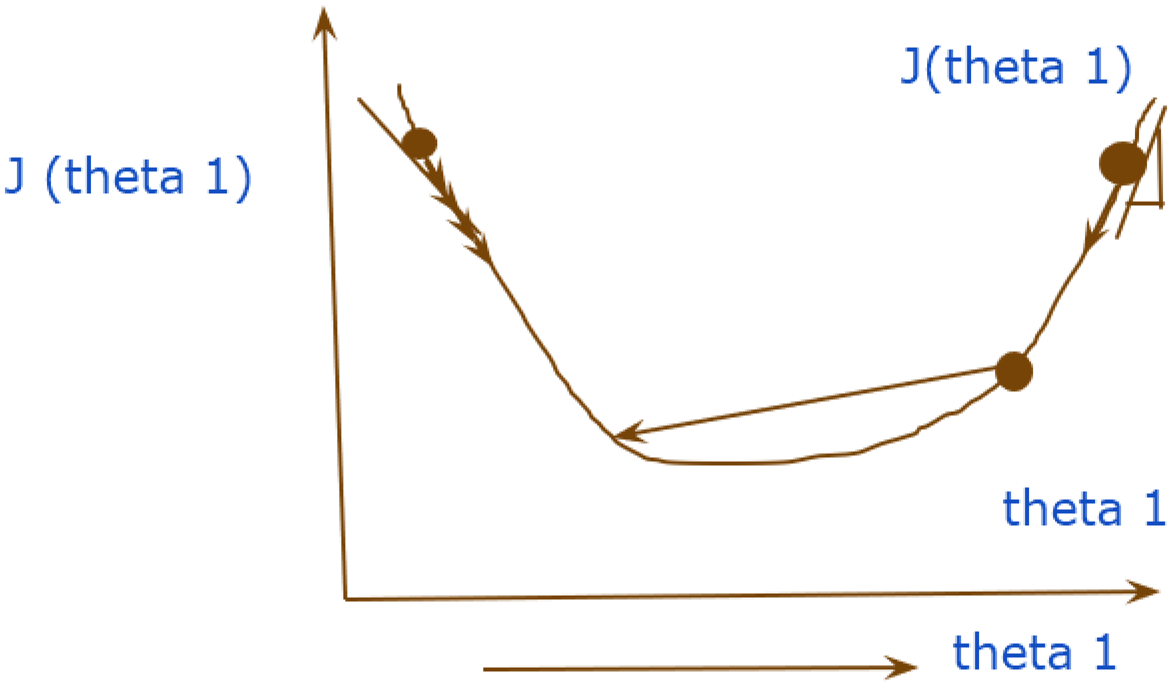
theta 1:= theta 1 - alpha (positive slope)

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



alpha: learning rate

alpha: very small

it will be very slow

alpha: very high

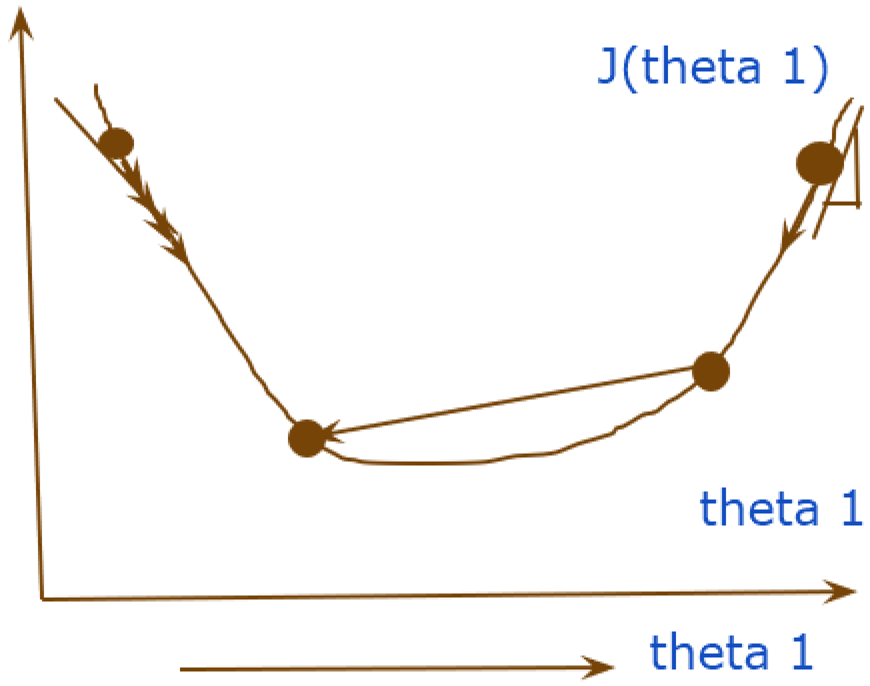
theta 1:= theta 1 - alpha (positive slope)

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



alpha: learning rate

alpha: very small

it will be very slow

alpha: very high

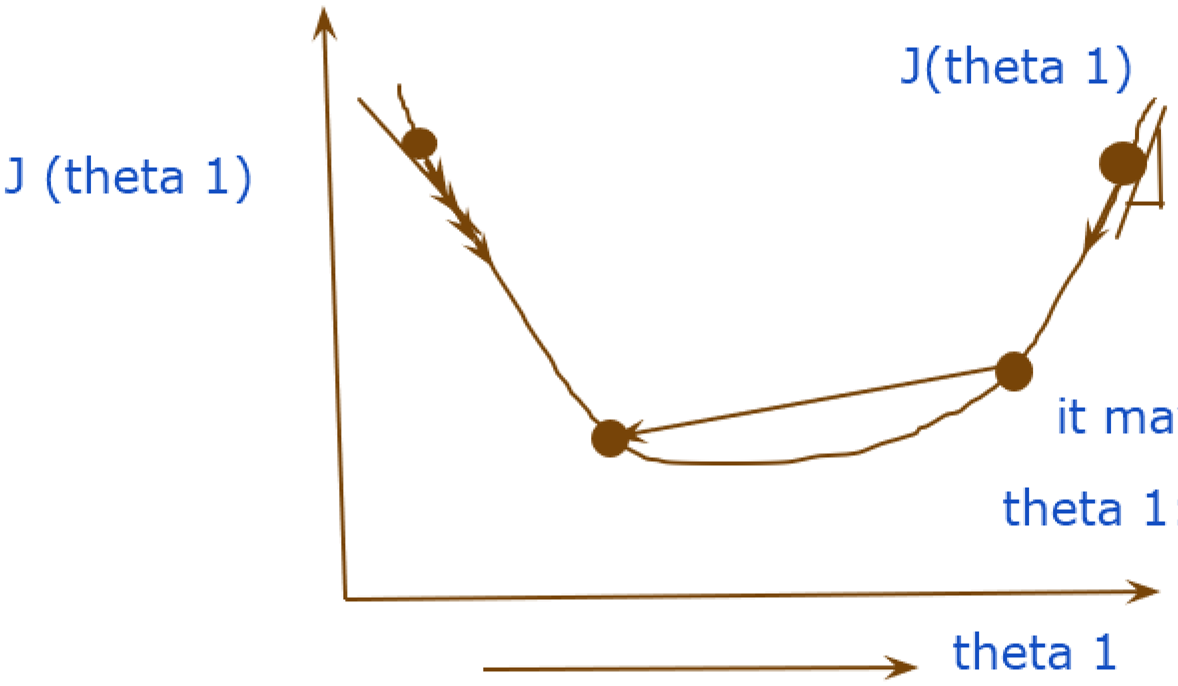
theta 1:= theta 1 - alpha (positive slope)

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



alpha: learning rate

alpha: very small

it will be very slow

alpha: very high

it may fail to converge or even diverge

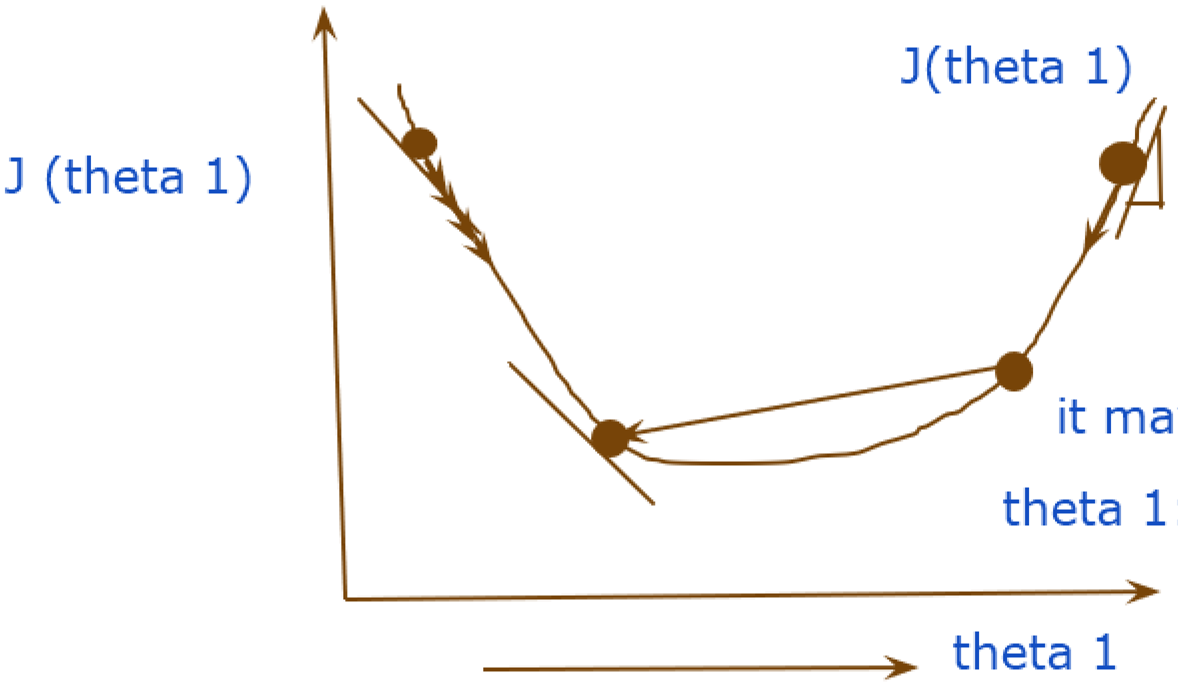
theta 1:= theta 1 - alpha (positive slope)

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent algorithm



alpha: learning rate

alpha: very small

it will be very slow

alpha: very high

it may fail to converge or even diverge

theta 1:= theta 1 - alpha (positive slope)

theta 1:= theta 1 - alpha(partial derivative J(theta 1)/theta 1)

theta 1:= theta 1 - alpha (negative slope)

gradient: measures how much the output of a function changes if we change the input with small values

Gradient Descent Algorithm

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

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ This is true for all θ_i
For i = 0, 1, 2, 3, …., n

Steps:

- 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

Gradient descent algorithm

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

Learning Rate

Correct Update:

$$\left.\begin{aligned}
\text{temp0} &:= \theta_0 - \alpha \tfrac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\
\text{temp1} &:= \theta_1 - \alpha \tfrac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\
\theta_0 &:= \text{temp0} \\
\theta_1 &:= \text{temp1}
\end{aligned}\right\}$$

update

Simultaneously: θ_0 and θ_1

Gradient descent can converge to a local minimum, even with fixed (α) learning rate.

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

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.

"Batch" gradient Descent algorithm:
"Stochastic" gradient descent algorithm

"Batch" gradient Descent algorithm:
"Stochastic" gradient descent algorithm

Each step of gradient descent uses all the training example: batch gradient descent algorithm
Each step of gradient decent uses single training example
Mini-batch: each step of gradient descent uses subset of training example

"Batch" gradient Descent algorithm:
"Stochastic" gradient descent algorithm

Each step of gradient descent uses all the training example: batch gradient descent algorithm
Each step of gradient decent uses single training example
Mini-batch: each step of gradient descent uses subset of training example

“Batch” Gradient Descent

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

“Stochastic” Gradient Descent

“Stochastic”: Each step of gradient descent use only single training example.

“Mini-batch” Gradient Descent

It lies in between of these two extremes, and can use a mini-batch (small portion) of training data examples in each step

Batch Gradient Descent Algorithm

Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i]

Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i]

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i] ^{2}

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i] ^{2}

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i] ^{2}

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

Training data points

Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i] ^{2}

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

Training data points

Vanilla gradient descent algorithm/gradient descent algorithm

Batch Gradient Descent Algorithm

J (theta 1): = 1/2m $\underbrace{\text{summation (i = 1 to m)}}_{\text{Training data points}}$ [h(x)i -(y)i] ^{2}

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

Vanilla gradient descent algorithm/gradient descent algorithm

Stochastic Gradient descent algorithm

Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) $[h(x)i -(y)i]^{2}$

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

Training data points

Vanilla gradient descent algorithm/gradient descent algorithm

Stochastic Gradient descent algorithm

update the parameters for each training example one by one

Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i] ^{2}

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

Training data points

Vanilla gradient descent algorithm/gradient descent algorithm

Stochastic Gradient descent algorithm

update the parameters for each training example one by one

Mini-batch gradient descent algorithm

Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i] ^{2}

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

Training data points

Vanilla gradient descent algorithm/gradient descent algorithm

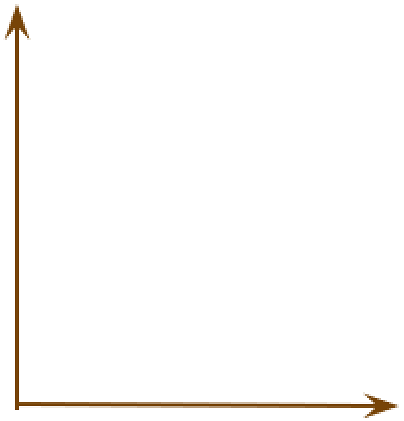
Stochastic Gradient descent algorithm

update the parameters for each training example one by one

Mini-batch gradient descent algorithm : splits the training dataset into small batches & perform an update

Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i]^{2}



theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

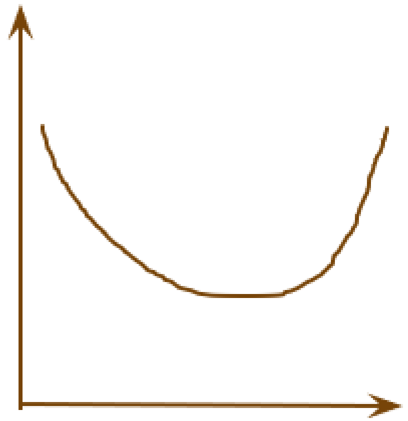
Training data points

Vanilla gradient descent algorithm/gradient descent algorithm

Stochastic Gradient descent algorithm

update the parameters for each training example one by one

Mini-batch gradient descent algorithm : splits the training dataset into small batches & perform an update



Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i]^2

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

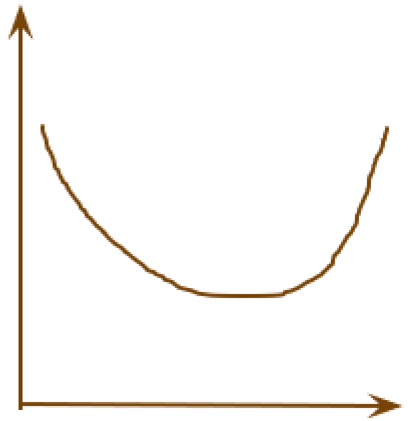
Training data points

Vanilla gradient descent algorithm/gradient descent algorithm

Stochastic Gradient descent algorithm

update the parameters for each training example one by one

Mini-batch gradient descent algorithm : splits the training dataset into small batches & perform an update



Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i] ^{2}

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

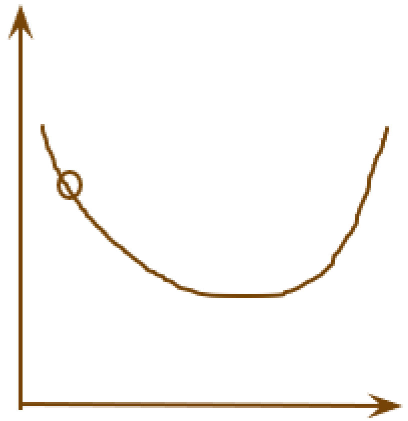
Training data points

Vanilla gradient descent algorithm/gradient descent algorithm

Stochastic Gradient descent algorithm

update the parameters for each training example one by one

Mini-batch gradient descent algorithm : splits the training dataset into small batches & perform an update



Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i] ^{2}

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

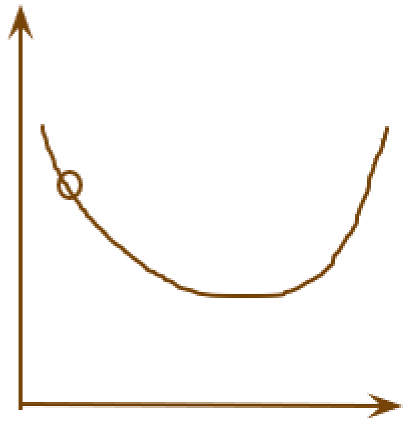
Training data points

Vanilla gradient descent algorithm/gradient descent algorithm

Stochastic Gradient descent algorithm

update the parameters for each training example one by one

Mini-batch gradient descent algorithm : splits the training dataset into small batches & perform an update



Batch Gradient Descent Algorithm

J (theta 1): = 1/2m summation (i = 1 to m) [h(x)i -(y)i]^2

theta 1: theta 1 - (aplha) (partial derivative of cost function with respect to the parameters)

Training data points

Vanilla gradient descent algorithm/gradient descent algorithm

Stochastic Gradient descent algorithm

update the parameters for each training example one by one

Mini-batch gradient descent algorithm : splits the training dataset into small batches & perform an update