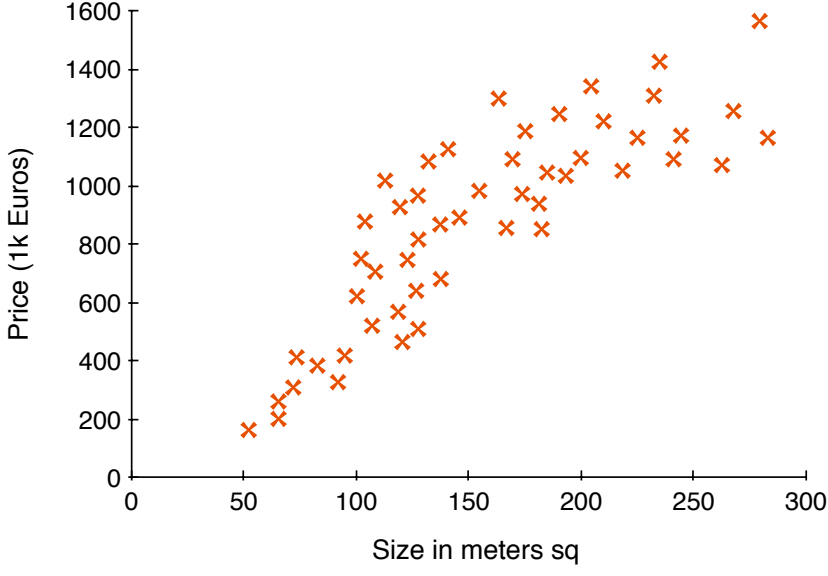
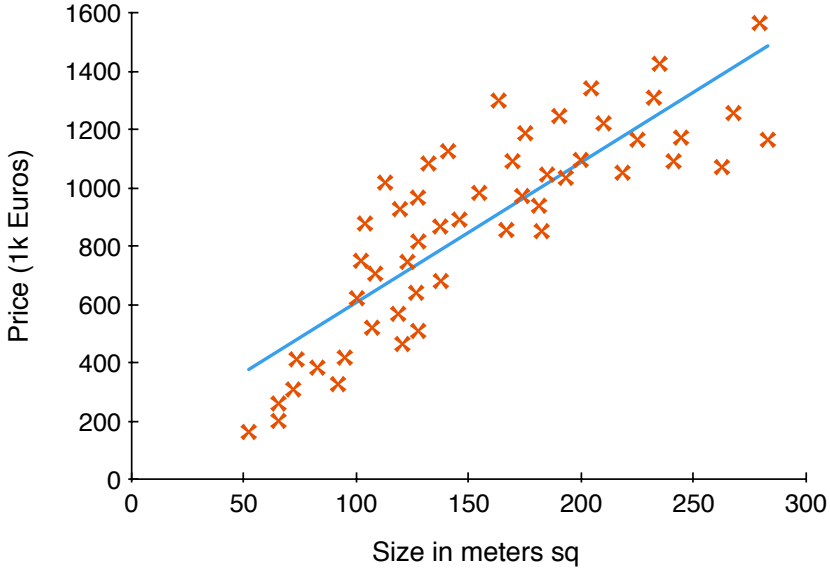
Linear Regression & Gradient Descent

Lecture 2 - DAMLF | ML1

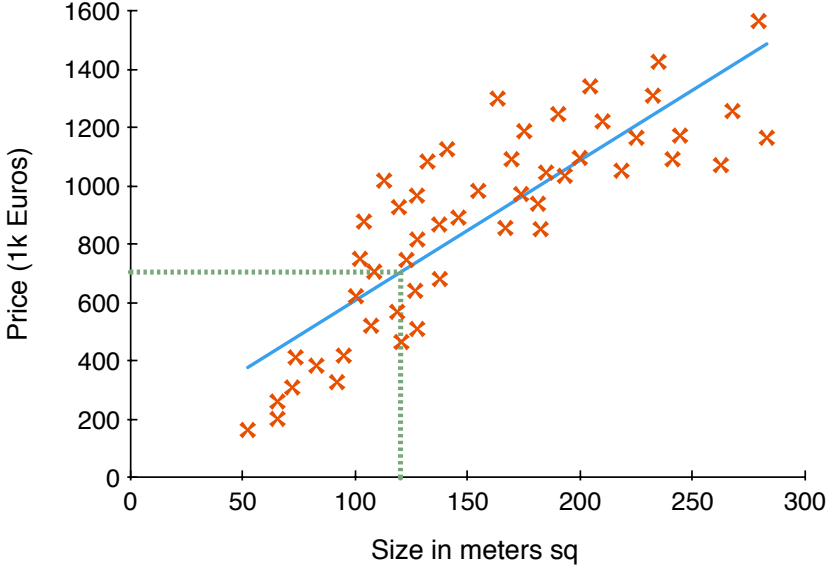
Univariate Linear Regression



Univariate Linear Regression



Univariate Linear Regression



Learning Task: Predict Price

Training set of house sizes in meters sq

	Size in meters sq (x)	Price in 1k Euros (y)
1)	52.14	164.21
2)	82.67	384.50
3)	120.47	465.41
4)	122.73	746.54
⋮	⋮	⋮
m)	241.03	1091.10

Notation:

Learning Task: Predict Price

Training set of house sizes in meters sq

	Size in meters sq (x)	**Price in 1k Euros** (y)
1)	52.14	164.21
2)	82.67	384.50
3)	120.47	465.41
4)	122.73	746.54
⋮	⋮	⋮
m)	241.03	1091.10

Notation:

m = Number of training examples

Learning Task: Predict Price

Training set of house sizes in meters sq

	Size in meters sq (x)	Price in 1k Euros (y)
1)	52.14	164.21
2)	82.67	384.50
3)	120.47	465.41
4)	122.73	746.54
⋮	⋮	⋮
m)	241.03	1091.10

Notation:

m	=	Number of training examples
x	=	Input variable(s) / feature(s)
y	=	Output variable / target variable

Learning Task: Predict Price

Training set of house sizes in meters sq

	Size in meters sq (x)	Price in 1k Euros (y)
1)	52.14	164.21
2)	82.67	384.50
3)	120.47	465.41
4)	122.73	746.54
⋮	⋮	⋮
m)	241.03	1091.10

Notation:

m	=	Number of training examples
x	=	Input variable(s) / feature(s)
y	=	Output variable / target variable
(x, y)	=	A single training example

Learning Task: Predict Price

Training set of house sizes in meters sq

	Size in meters sq (x)	Price in 1k Euros (y)
1)	52.14	164.21
2)	82.67	384.50
3)	120.47	465.41
4)	122.73	746.54
⋮	⋮	⋮
m)	241.03	1091.10

Notation:

m	=	Number of training examples
x	=	Input variable(s) / feature(s)
y	=	Output variable / target variable
(x, y)	=	A single training example
$(x^{(i)}, y^{(i)})$	=	The i^{th} training example

Examples:

$x^{(2)} = 82.67$

$(x^{(3)}, y^{(3)}) = (120.47, 465.41)$

What is a Machine Learning Problem?

"A computer program is said to learn from **experience** *E* with respect to some class of **tasks** *T* and **performance measure** *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."

–Thomas Mitchell (1997)

What is a Machine Learning Problem?

1. Data

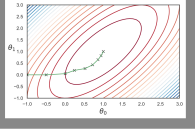
loan_number	amount_borrowed	term	borrower_rate	installment	grade	origination_date	listing_title	principal_balance	principal_paid	interest_paid
1014904	21563	36	0.1099	685.46	B	2013-12-01T00:00	debt_consolidation	0.00	21563.00	4702.33
1014948	4800	36	0.1099	157.13	B	2013-12-01T00:00	home_improvement	0.00	4800.00	397.92
1014972	12000	36	0.0762	373.84	A	2013-12-01T00:00	debt_consolidation	0.00	12000.00	1397.54
1015023	12000	36	0.1199	398.32	B	2013-12-01T00:00	debt_consolidation	0.00	12000.00	2348.48
1015048	12000	36	0.0662	368.45	A	2013-12-01T00:00	debt_consolidation	0.00	12000.00	1203.95
1015590	11550	60	0.2299	323.54	E	2013-12-01T00:00	debt_consolidation	0.00	11506.00	7206.25
1015648	15000	36	0.0899	476.30	A	2013-12-01T00:00	debt_consolidation	0.00	15000.00	2148.73
1017781	24000	36	0.1053	814.80	B	2013-12-	credit_card	0.00	24000.00	4852.21

2. Select/Build Model

Define: $h(x;\theta)$

3. Optimization

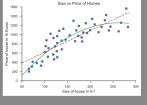
$$\min_{\theta} J(\boldsymbol{\theta})$$

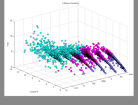


4. Prediction or Exploration

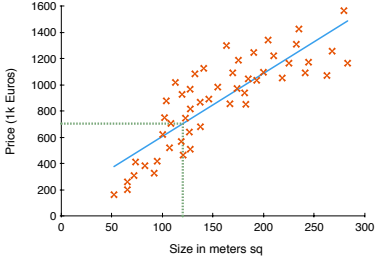
supervised learning

unsupervised learning





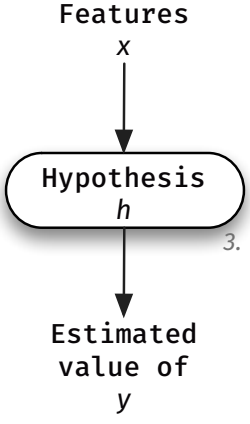
Hypothesis Representation



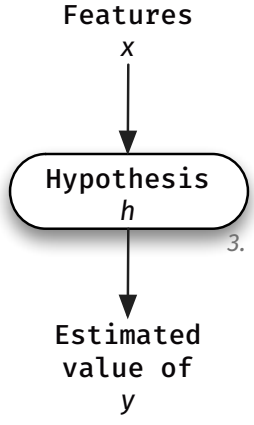
When selecting a learning algorithm or designing a new one, the next step is to decide how to **represent** *h*.

Hypothesis Representation

Univariate Linear Regression:



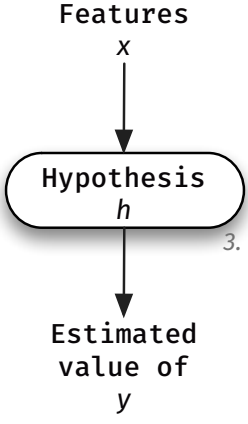
Hypothesis Representation



Univariate Linear Regression:

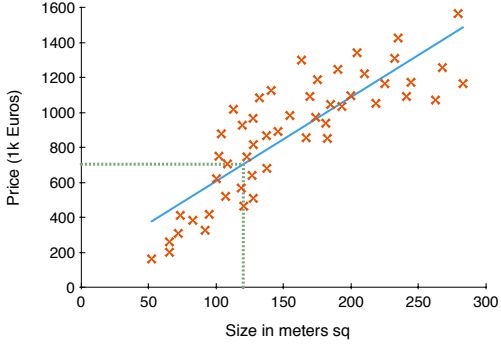
$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$

Hypothesis Representation

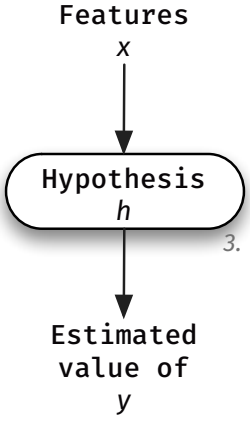


Univariate Linear Regression:

$h(x;\boldsymbol{\theta}) = \theta_0 + \theta_1 x$



Hypothesis Representation



Once we decide how to represent *h*, the next step is to **implement** the model.

Implementing Univariate Linear Regression

Training set:

	Size in m^2 (x)	Price (1000s) (y)
1)	52.14	164.21
2)	82.67	384.50
3)	120.47	465.41
4)	122.73	746.54
⋮	⋮	⋮
m)	241.03	1091.10

Learning Algorithm: Univariate Linear Regression

Hypothesis: $h(x;\boldsymbol{\theta}) = \theta_0 + \theta_1(x)$

Implementation

Hypothesis: $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1(x)$

Notation: $\boldsymbol{\theta} = (\theta_0, \theta_1)$

θ_i are the parameters

Implementation

Hypothesis: $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1(x)$

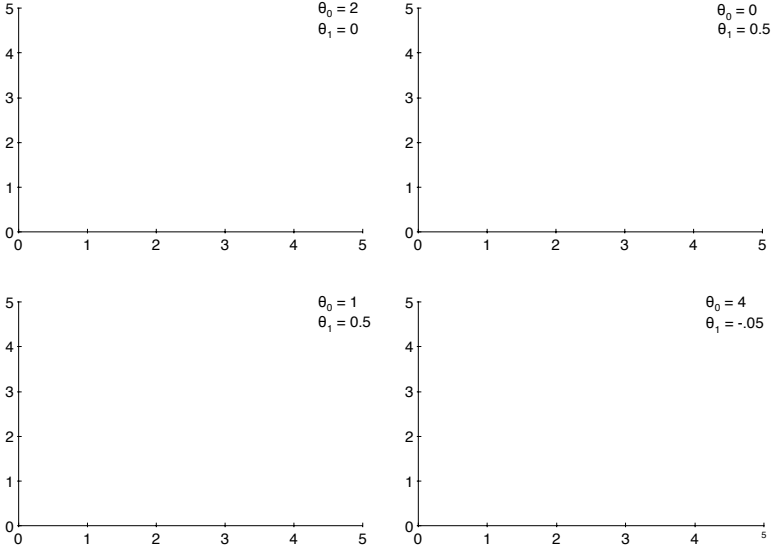
Notation: $\boldsymbol{\theta} = (\theta_0, \theta_1)$

θ_i are the parameters

How are θ_i's chosen?

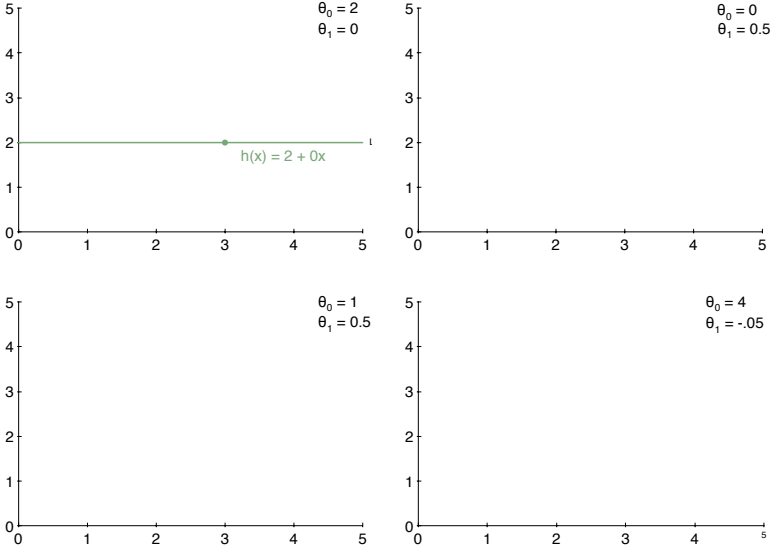
Choosing θ_i's

$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1(x)$



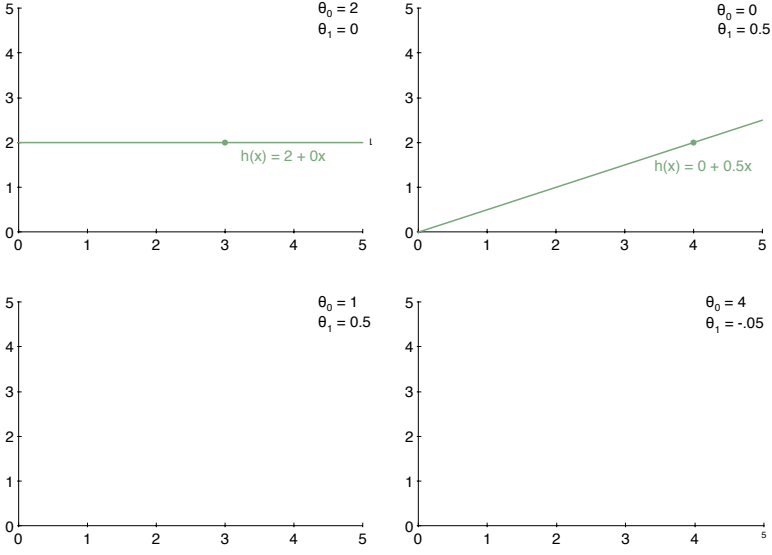
Choosing θ_i's

$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1(x)$



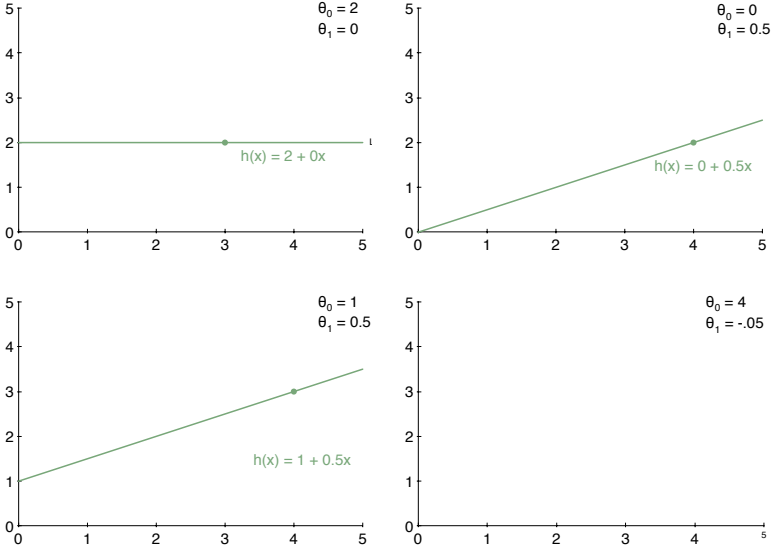
Choosing θ_i's

$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1(x)$



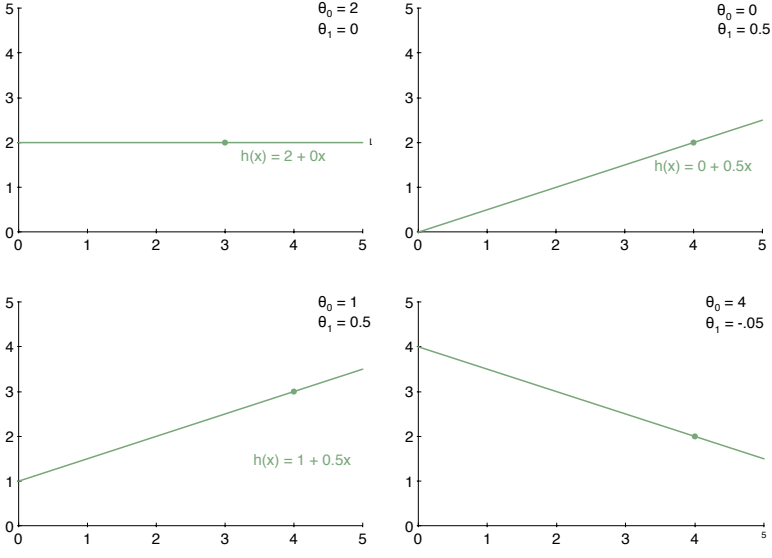
Choosing θ_i's

$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1(x)$

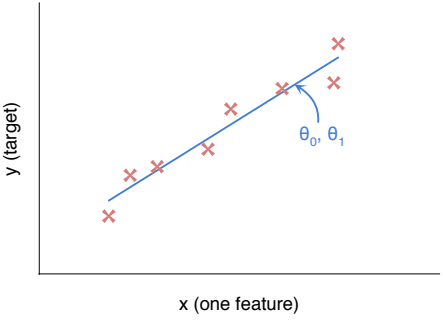


Choosing θ_i's

$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1(x)$

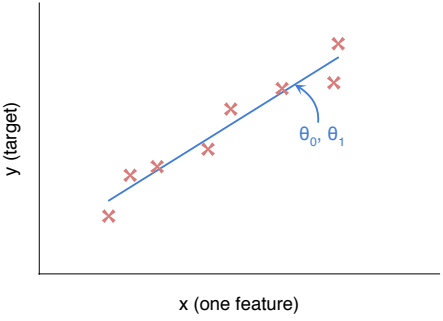


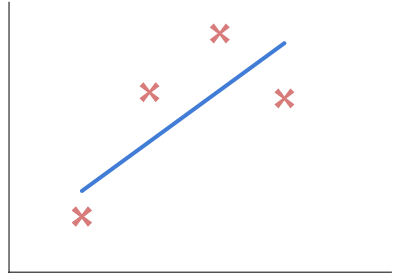
Cost Function: Motivation



Goal: Select θ_0, θ_1 such that $h(x; \boldsymbol{\theta})$ is 'close to' y for each (x, y) in the training set

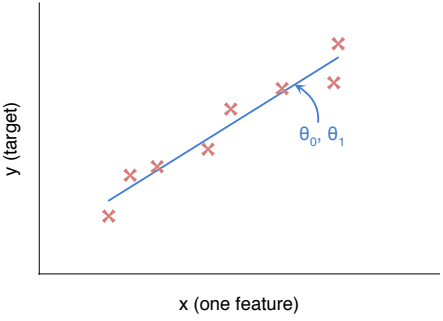
Cost Function: Motivation

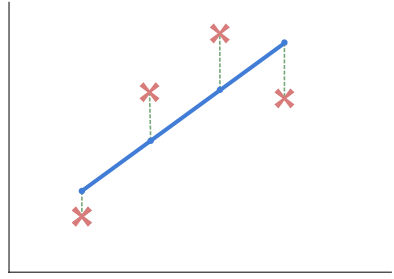




Goal: Select θ_0, θ_1 such that $h(x; \boldsymbol{\theta})$ is 'close to' y for each (x, y) in the training set

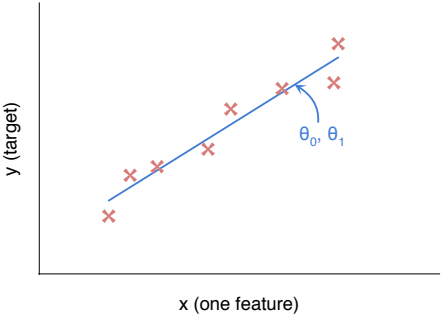
Cost Function: Motivation

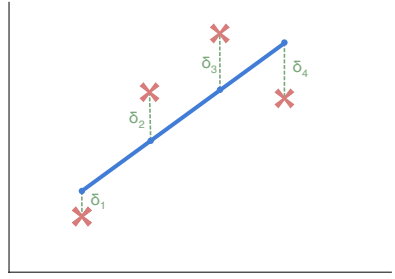




Goal: Select θ_0, θ_1 such that $h(x; \boldsymbol{\theta})$ is 'close to' y for each (x, y) in the training set

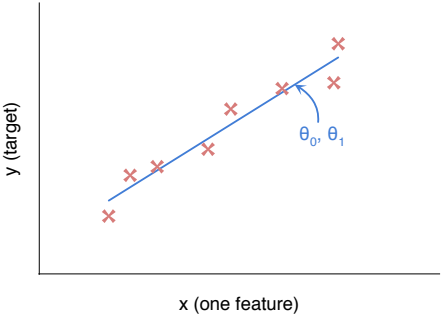
Cost Function: Motivation

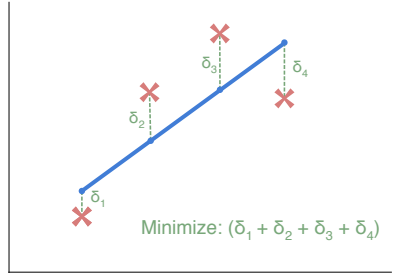




Goal: Select θ_0, θ_1 such that $h(x; \boldsymbol{\theta})$ is 'close to' y for each (x, y) in the training set

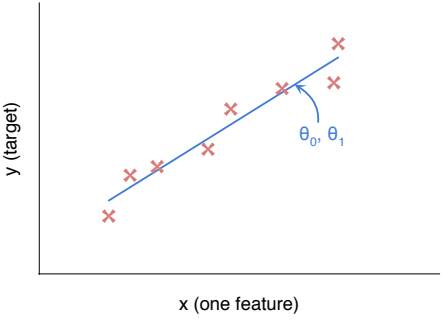
Cost Function: Motivation

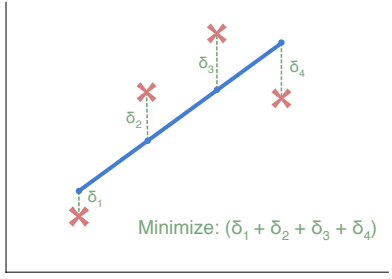




Goal: Select θ_0, θ_1 such that $h(x; \boldsymbol{\theta})$ is 'close to' y for each (x, y) in the training set

Cost Function: Motivation





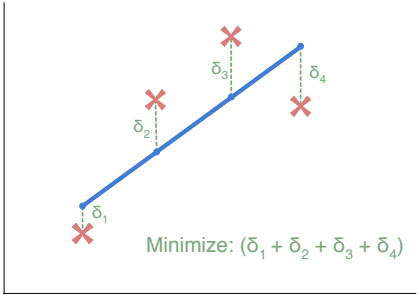
Goal: Select θ_0, θ_1 such that $h(x; \boldsymbol{\theta})$ is 'close to' y for each (x, y) in the training set

$\min_{\theta_0,\theta_1} \frac{1}{2m} \sum_i^m (h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)})^2$,

where:

$$h(x^{(i)}; \boldsymbol{\theta}) = \theta_0 + \theta_1 x^{(i)}$$

Cost Function $J(\theta_0, \theta_1)$



Cost Function:

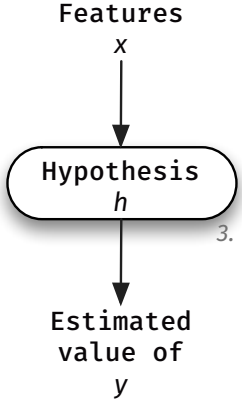
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_i^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

Goal: $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

minimize squared-error loss

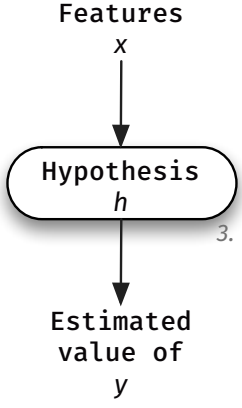
minimize the Brier Score

Univariate Linear Regression



Univariate Linear Regression

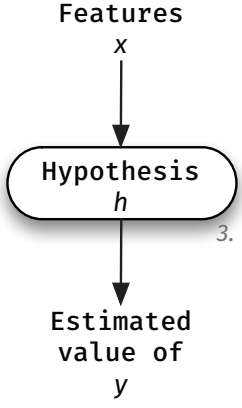
Univariate Linear Regression



Univariate Linear Regression

1. **Hypothesis**
   $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$

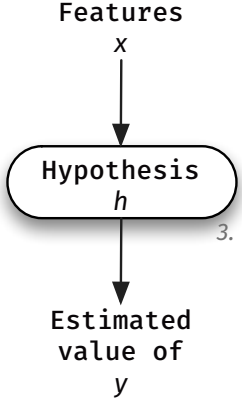
Univariate Linear Regression



Univariate Linear Regression

1. **Hypothesis**
   $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$
2. **Parameters**
   θ_0, θ_1 (each θ_i are in $\mathbb{R}$)

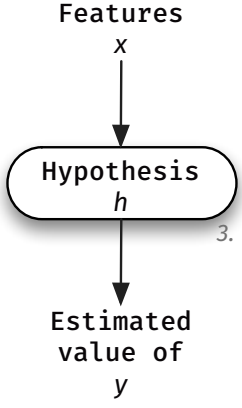
Univariate Linear Regression



Univariate Linear Regression

1. **Hypothesis**
   $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$
2. **Parameters**
   θ_0, θ_1 (each θ_i are in $\mathbb{R}$)
3. **Cost Function**
   $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)})^2$

Univariate Linear Regression



Univariate Linear Regression

1. **Hypothesis**
   $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$
2. **Parameters**
   θ_0, θ_1 (each θ_i are in $\mathbb{R}$)
3. **Cost Function**
   $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)})^2$
4. **Goal**
   $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Univariate Linear Regression

Example:

Univariate Linear Regression

1. **Hypothesis**
   $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$
2. **Parameters**
   θ_0, θ_1 (each are in $\mathbb{R}$)
3. **Cost Function**
   $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)})^2$
4. **Goal**
   $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Univariate Linear Regression

Example:

Parameters
Suppose that $\theta_0 = 0$.

Univariate Linear Regression

1. **Hypothesis**
   $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$
2. **Parameters**
   θ_0, θ_1 (each are in $\mathbb{R}$)
3. **Cost Function**
   $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)})^2$
4. **Goal**
   $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Univariate Linear Regression

Example:

Parameters
Suppose that $\theta_0 = 0$.

Hypothesis
$h(x; \boldsymbol{\theta}) = 0 + \theta_1 x$
$= \theta_1 x$

Univariate Linear Regression

1. **Hypothesis**
   $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$
2. **Parameters**
   θ_0, θ_1 (each are in $\mathbb{R}$)
3. **Cost Function**
   $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)})^2$
4. **Goal**
   $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Univariate Linear Regression

Example:

Parameters
Suppose that $\theta_0 = 0$.

Hypothesis
$$h(x; \boldsymbol{\theta}) = 0 + \theta_1 x$$
$$= \theta_1 x$$

Cost Function
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)})^2$$
$$= \frac{1}{2m} \sum_{i=1}^{m} (\theta_1 x^{(i)} - y^{(i)})^2$$

Univariate Linear Regression

1. **Hypothesis**
   $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$
2. **Parameters**
   θ_0, θ_1 (each are in $\mathbb{R}$)
3. **Cost Function**
   $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)})^2$
4. **Goal**
   $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Univariate Linear Regression

Example:

Parameters

Suppose that $\theta_0 = 0$.

Hypothesis

$$h(x; \boldsymbol{\theta}) = 0 + \theta_1 x$$
$$= \theta_1 x$$

Cost Function

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

Goal

$$\min_{\theta_1} J(\theta_1)$$

Univariate Linear Regression

1. **Hypothesis**
   $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$
2. **Parameters**
   θ_0, θ_1 (each are in $\mathbb{R}$)
3. **Cost Function**
   $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)})^2$
4. **Goal**
   $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Univariate Linear Regression

Example:

Parameters

Suppose that $\theta_0 = 0$.

Hypothesis

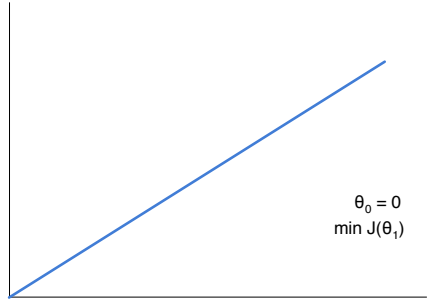
$$h(x; \boldsymbol{\theta}) = 0 + \theta_1 x$$
$$= \theta_1 x$$

Cost Function

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

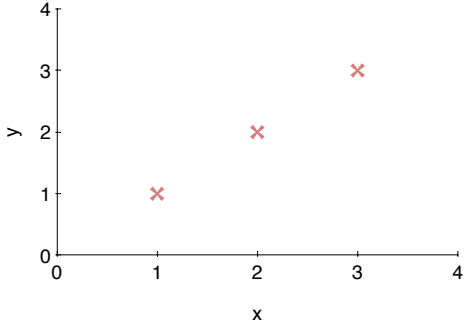
Goal

$$\min_{\theta_1} J(\theta_1)$$



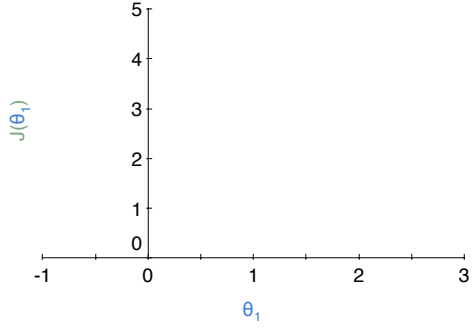
Example

Hypothesis: $h(x; \theta)$



$$h(x; \theta) = 0 + \theta_1 x$$
$$= \theta_1 x$$

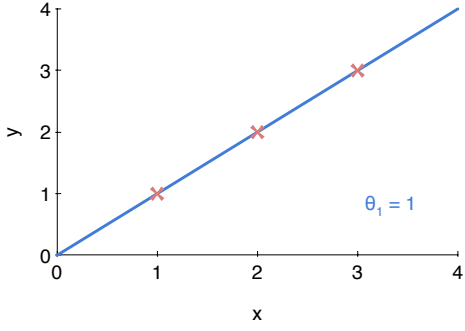
Cost Function: $J(\theta_1)$



$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \theta) - y^{(i)})^2$$
$$= \frac{1}{2m} \cdot \big((\theta_1 x^{(1)} - y^{(1)})^2 + (\theta_1 x^{(2)} - y^{(2)})^2 + (\theta_1 x^{(3)} - y^{(3)})^2 \big)$$
$$=$$

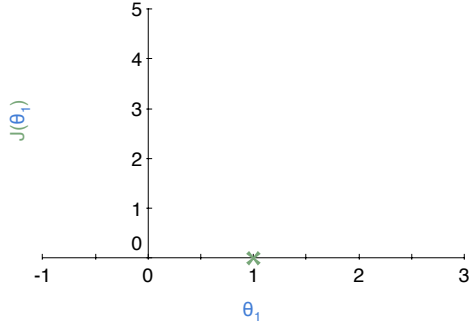
Example

Hypothesis: $h(x;\theta)$



$$h(x;\theta) = 0 + \theta_1 x$$
$$= 1x$$

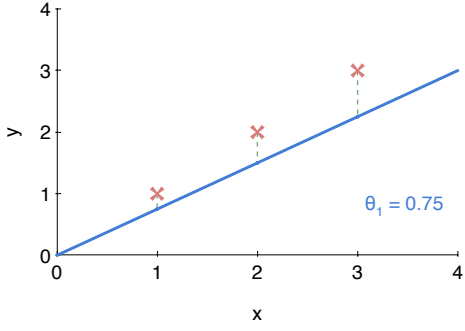
Cost Function: $J(\theta_1)$



$$J(1) = \frac{1}{2m}\sum_{i=1}^{m}(h(x^{(i)};\theta) - y^{(i)})^2$$
$$= \frac{1}{2m}\cdot\big((1-1)^2 +$$
$$(2-2)^2 +$$
$$(3-3)^2\big)$$
$$= 0$$

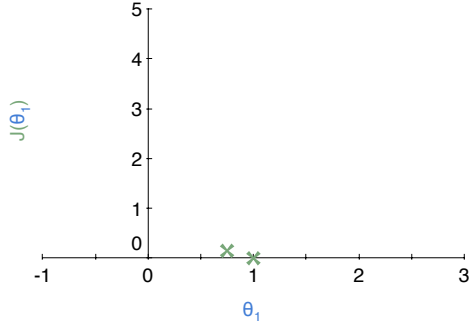
Example

Hypothesis: $h(x;\theta)$



$$h(x;\theta) = 0 + \theta_1 x$$
$$= 0.75x$$

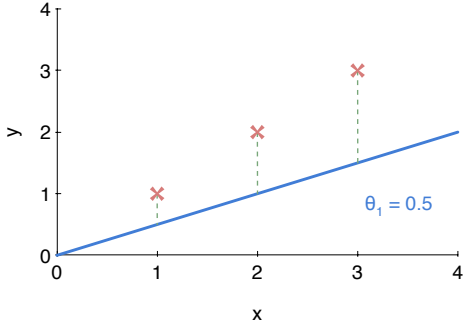
Cost Function: $J(\theta_1)$



$$J(0.75) = \frac{1}{2m}\sum_{i=1}^{m}(h(x^{(i)};\theta) - y^{(i)})^2$$
$$= \frac{1}{2m}\cdot\big((0.75-1)^2 +$$
$$(1.5-2)^2 +$$
$$(2.25-3)^2\big)$$
$$\approx 0.15$$

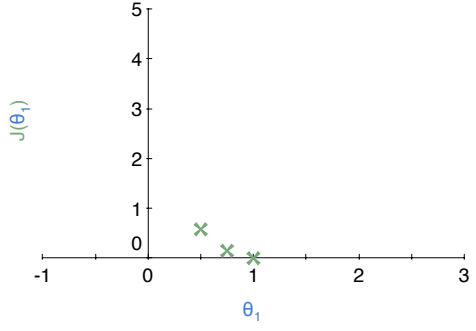
Example

Hypothesis: $h(x;\theta)$



$$h(x;\theta) = 0 + \theta_1 x$$
$$= 0.5x$$

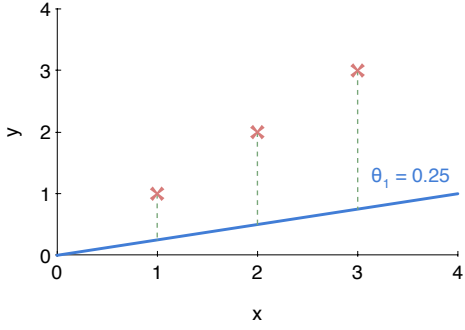
Cost Function: $J(\theta_1)$



$$J(0.5) = \frac{1}{2m}\sum_{i=1}^{m}(h(x^{(i)};\theta) - y^{(i)})^2$$
$$= \frac{1}{2m}\cdot((0.5-1)^2 +$$
$$(1-2)^2 +$$
$$(1.5-3)^2)$$
$$\approx 0.58$$

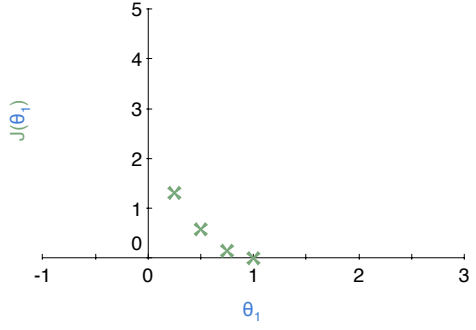
Example

Hypothesis: $h(x;\theta)$



$$h(x;\theta) = 0 + \theta_1 x$$
$$= 0.25x$$

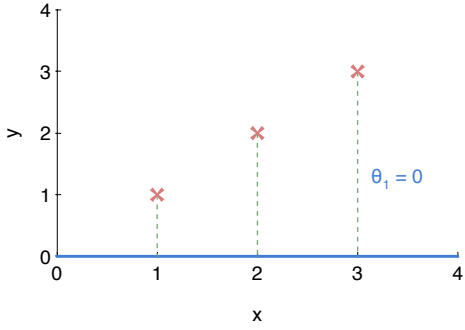
Cost Function: $J(\theta_1)$



$$J(0.25) = \frac{1}{2m}\sum_{i=1}^{m}(h(x^{(i)};\theta) - y^{(i)})^2$$
$$= \frac{1}{2m}\cdot\big((0.25-1)^2 + (0.5-2)^2 + (0.75-3)^2\big)$$
$$\approx 1.31$$

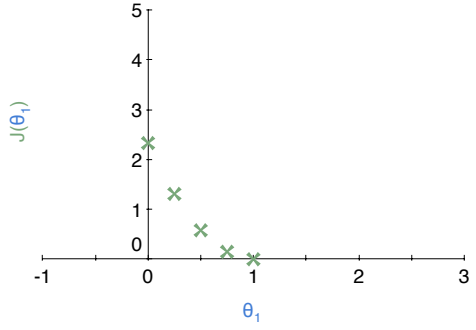
Example

Hypothesis: $h(x;\theta)$



$$h(x;\theta) = 0 + \theta_1 x$$
$$= 0x$$

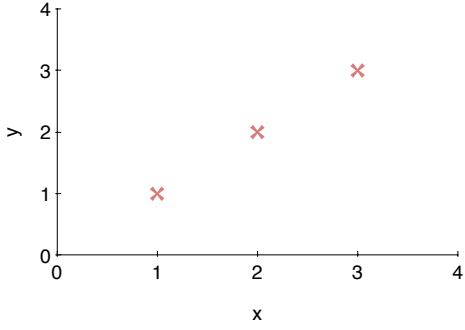
Cost Function: $J(\theta_1)$



$$J(0) = \frac{1}{2m}\sum_{i=1}^{m}(h(x^{(i)};\theta) - y^{(i)})^2$$
$$= \frac{1}{2m}\cdot\big((0-1)^2 +$$
$$(0-2)^2 +$$
$$(0-3)^2\big)$$
$$\approx 2.33$$

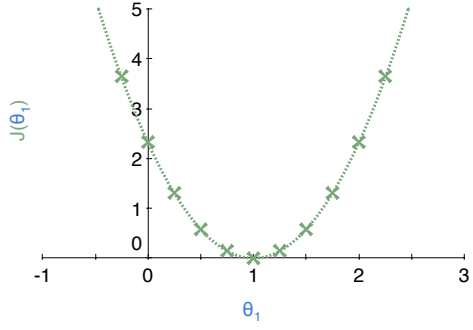
Example

Hypothesis: $h(x; \theta)$



$$h(x; \theta) = 0 + \theta_1 x$$
$$= \theta_1 x$$

Cost Function: $J(\theta_1)$

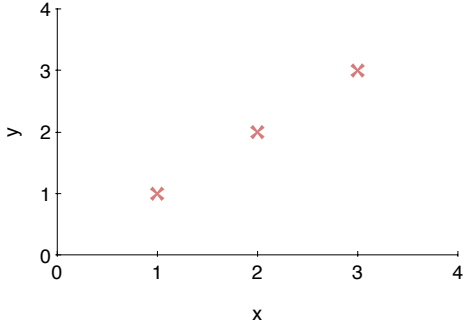


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \theta) - y^{(i)})^2$$
$$= \frac{1}{2m} \cdot \big((\theta_1 x^{(1)} - y^{(1)})^2 + (\theta_1 x^{(2)} - y^{(2)})^2 + (\theta_1 x^{(3)} - y^{(3)})^2 \big)$$

Goal: minimize $J(\theta_1)$.

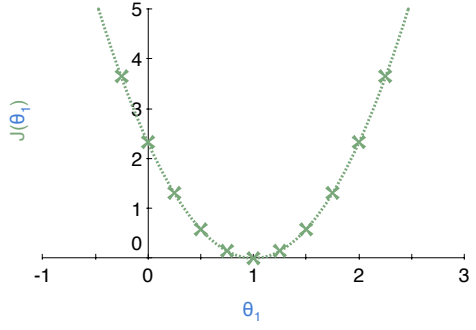
Example

Hypothesis: $h(x; \theta)$



$$h(x; \theta) = 0 + \theta_1 x$$
$$= \theta_1 x$$

Cost Function: $J(\theta_1)$

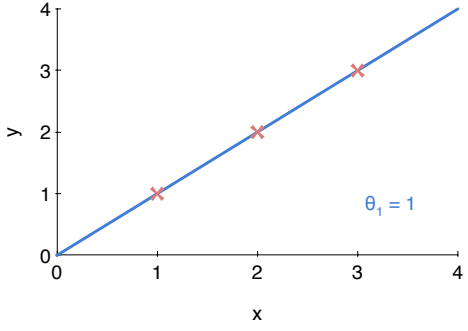


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \theta) - y^{(i)})^2$$
$$= \frac{1}{2m} \cdot \big((\theta_1 x^{(1)} - y^{(1)})^2 + (\theta_1 x^{(2)} - y^{(2)})^2 + (\theta_1 x^{(3)} - y^{(3)})^2 \big)$$

Goal: minimize $J(\theta_1)$. **Answer:** $\theta_1 = 1$

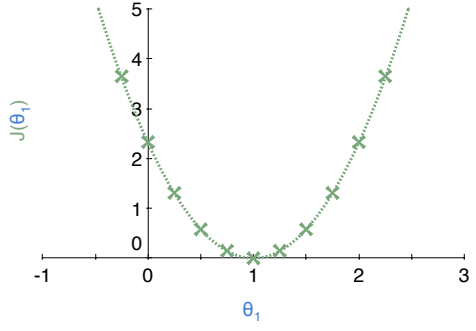
Visualizing the Cost Function $J(\theta)$

Hypothesis: $h(x; \theta)$



$$h(x; \theta) = 0 + \theta_1 x$$
$$= \theta_1 x$$

Cost Function: $J(\theta_1)$

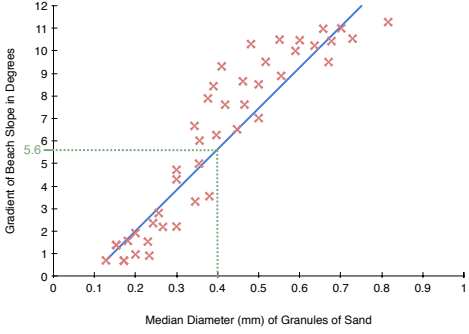


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \theta) - y^{(i)})^2$$
$$= \frac{1}{2m} \cdot \big((\theta_1 x^{(1)} - y^{(1)})^2 + (\theta_1 x^{(2)} - y^{(2)})^2 + (\theta_1 x^{(3)} - y^{(3)})^2 \big)$$

Goal: minimize $J(\theta_1)$.

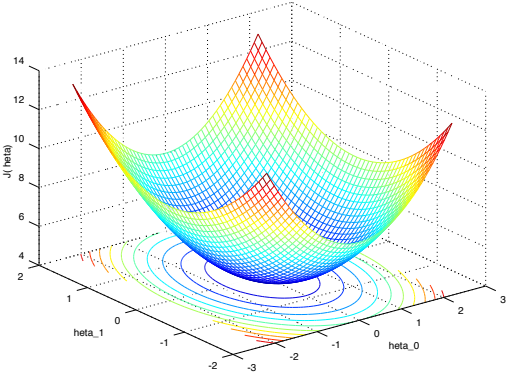
Visualizing the Cost Function $J(\theta)$

Hypothesis: $h(x; \theta)$



$$h(x; \theta) = \theta_0 + \theta_1 x$$

Cost Function: $J(\theta_1)$



Computing an Answer

The Gradient Descent Algorithm

- A very general optimization algorithm
- Variants of Gradient Descent are used often

Gradient Descent: The Basic Strategy

A Function: $J(\theta_0, \theta_1)$
Goal: minimize$_{\theta_0, \theta_1}$ $J(\theta_0, \theta_1)$

Gradient Descent: The Basic Strategy

A Function: $J(\theta_0, \theta_1)$
Goal: minimize$_{\theta_0, \theta_1}$ $J(\theta_0, \theta_1)$
Gradient Descent (outline)

Gradient Descent: The Basic Strategy

A Function: $J(\theta_0, \theta_1)$
Goal: minimize$_{\theta_0, \theta_1}$ $J(\theta_0, \theta_1)$
Gradient Descent (outline)

Initialize values for θ_0, θ_1

Choose both $\theta_0 \in \mathbb{R}$ and $\theta_1 \in \mathbb{R}$

Gradient Descent: The Basic Strategy

A Function: $J(\theta_0, \theta_1)$
Goal: minimize$_{\theta_0, \theta_1}$ $J(\theta_0, \theta_1)$
Gradient Descent (outline)

Initialize values for θ_0, θ_1

Choose both $\theta_0 \in \mathbb{R}$ and $\theta_1 \in \mathbb{R}$

Compute $J(\theta_0, \theta_1)$ for those values.

Just as we did in the example before

Gradient Descent: The Basic Strategy

A Function: $J(\theta_0, \theta_1)$
Goal: minimize$_{\theta_0, \theta_1}$ $J(\theta_0, \theta_1)$
Gradient Descent (outline)

Initialize values for θ_0, θ_1

Choose both $\theta_0 \in \mathbb{R}$ and $\theta_1 \in \mathbb{R}$

Compute $J(\theta_0, \theta_1)$ for those values.

Just as we did in the example before

Change θ_0 and θ_1 to **reduce** $J(\theta_0, \theta_1)$

NEW: This 'updating' part needs to be explained

Gradient Descent: The Basic Strategy

A Function: $J(\theta_0, \theta_1)$
Goal: minimize$_{\theta_0, \theta_1}$ $J(\theta_0, \theta_1)$
Gradient Descent (outline)

- Initialize values for θ_0, θ_1
  - Choose both $\theta_0 \in \mathbb{R}$ and $\theta_1 \in \mathbb{R}$
- Compute $J(\theta_0, \theta_1)$ for those values.
  - Just as we did in the example before
- **Change** θ_0 and θ_1 to **reduce** $J(\theta_0, \theta_1)$
  - *NEW:* This 'updating' part needs to be explained
- Compute $J(\theta_0, \theta_1)$ for *updated* values.

Gradient Descent: The Basic Strategy

A Function: $J(\theta_0, \theta_1)$
Goal: minimize$_{\theta_0,\theta_1}$ $J(\theta_0, \theta_1)$
Gradient Descent (outline)

- Initialize values for θ_0, θ_1
  - Choose both $\theta_0 \in \mathbb{R}$ and $\theta_1 \in \mathbb{R}$
- Compute $J(\theta_0, \theta_1)$ for those values.
  - Just as we did in the example before
- **Change** θ_0 and θ_1 to **reduce** $J(\theta_0, \theta_1)$
  - *NEW:* This 'updating' part needs to be explained
- Compute $J(\theta_0, \theta_1)$ for *updated* values.
- Repeat: Change, compute; change, compute; ...

End (hopefully) at a minimum value for $J(\theta_0, \theta_1)$

Gradient Descent Algorithm for Two Parameters

» Enter loop

Repeat:

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

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

Stop at convergence

» Exit loop

Gradient Descent Algorithm for Two Parameters

» Enter loop
Repeat:

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

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

Stop at convergence
» Exit loop

Assignment Function

Gradient Descent Algorithm for Two Parameters

» Enter loop

Repeat:

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

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

Stop at convergence

» Exit loop

Assignment Function

In Python, '=' is used:

```
In [1]: a = 1        # assign a the value 1
        a
Out[1]: 1

In [2]: a = a + 1    # update a to the value 2
        a
Out[2]: 2
```

Compare with identity '=='

```
In [3]: a == a + 1  # assert "a equals a + 1"
Out[3]: False

In [4]: a
Out[4]: 2
```

Gradient Descent Algorithm for Two Parameters

» Enter loop
Repeat:

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

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

Stop at convergence
» Exit loop

Derivative Term

Gradient Descent Algorithm for Two Parameters

» Enter loop
Repeat:

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

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

Stop at convergence
» Exit loop

Derivative Term

The partial derivative $\frac{\partial}{\partial \theta_0}$ of the cost function $J(\theta_0, \theta_1)$ determines the 'direction' of the 'steps' to take (**positive**, **negative**) within the θ_0 component of $J(\theta_0, \theta_1)$, or to take no steps at all in θ_0 (when **zero**).

The partial derivative $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$ does the same thing but for θ_1

Gradient Descent Algorithm for Two Parameters

» Enter loop

Repeat:

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

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

Stop at convergence

» Exit loop

Learning Rate

The learning rate α controls the size of the 'steps' you take at each pass of the loop while the loop is running:

BIG $\alpha \Rightarrow$ BIG steps

small $\alpha \Rightarrow$ small steps

Gradient Descent Algorithm for Two Parameters

» Enter loop

Repeat:

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

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

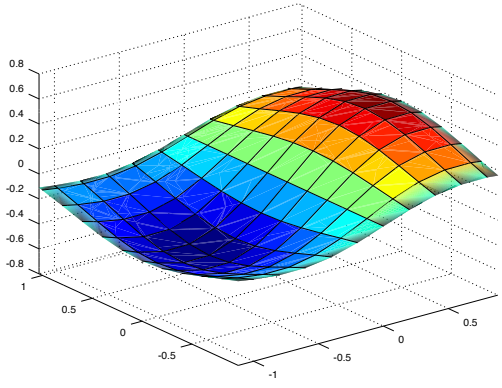
Stop at convergence

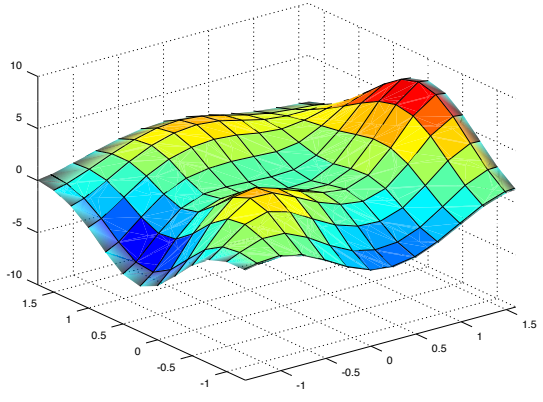
» Exit loop

Negative Gradient

Our aim is to pick values θ_0, θ_1 to **minimize** the cost function $J(\boldsymbol{\theta})$ so we compute the **negative gradient** for each $\theta_i \in \boldsymbol{\theta}$.

Example


0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5


10
5
0
-5
-10
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1.5

Gradient Descent Algorithm for Two Parameters

» Enter loop

Repeat:

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

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

Stop at convergence

» Exit loop

Simultaneous Update

Gradient Descent Algorithm for Two Parameters

» Enter loop

Repeat:

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

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

Stop at convergence

» Exit loop

Simultaneous Update

Each step of the loop updates *both* parameters θ_0 and θ_1 *simultaneously*

Correct	Incorrect
Assign θ_0	Assign θ_0
Assign θ_1	Update θ_0
Update θ_0	Assign θ_1
Update θ_1	Update θ_1

Gradient Descent Algorithm for Two Parameters

» Enter loop

Repeat:

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

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

Stop at convergence

» Exit loop

Simultaneous Update

For each iteration of the loop:

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

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

Gradient Descent Algorithm for Two Parameters

» Enter loop

Repeat:

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

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

Stop at convergence

» Exit loop

Simultaneous Update

For each iteration of the loop:

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

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

$$\text{updated-}\theta_0 := \text{interim-}\theta_0$$

$$\text{updated-}\theta_1 := \text{interim-}\theta_1$$

Gradient Descent for One Parameter

Let's look at Gradient Descent for **one** parameter, θ_1.

$$\theta_0 := 0$$

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

Gradient Descent for One Parameter

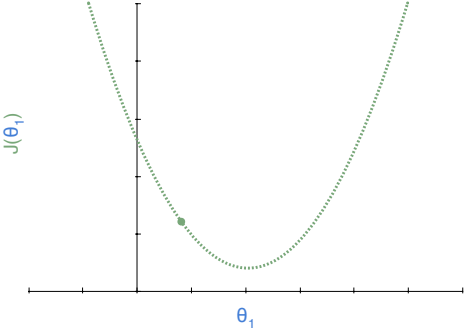
Let's look at Gradient Descent for **one** parameter, θ_1.

$$\theta_0 := 0$$

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

This will allow us to see how the learning rate and the derivative term work.

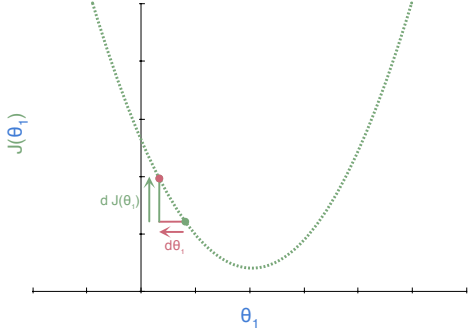
Gradient Descent for One Parameter



Role of the Derivative Term

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

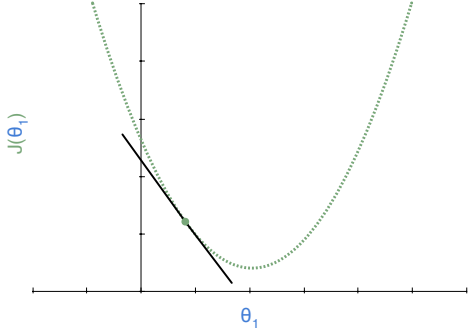
Gradient Descent for One Parameter



Role of the Derivative Term

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

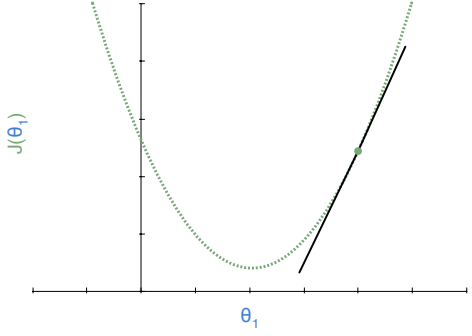
Gradient Descent for One Parameter



Role of the Derivative Term

$$\begin{aligned}\theta_1 &:= \theta_1 - \alpha \cdot \frac{d}{d\theta_1} J(\theta_1) \\ &:= \theta_1 - \alpha \cdot \text{negative} \\ &:= \theta_1 + \text{positive number}\end{aligned}$$

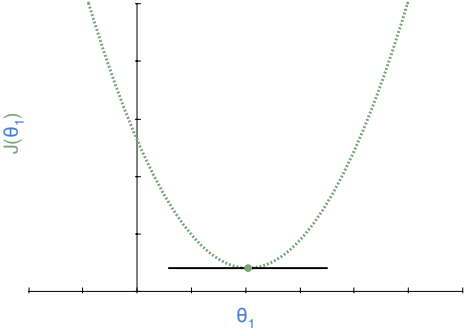
Gradient Descent for One Parameter



Role of the Derivative Term

$$\begin{aligned}\theta_1 &:= \theta_1 - \alpha \cdot \frac{d}{d\theta_1} J(\theta_1) \\ &:= \theta_1 - \alpha \cdot \text{positive} \\ &:= \theta_1 - \text{positive number}\end{aligned}$$

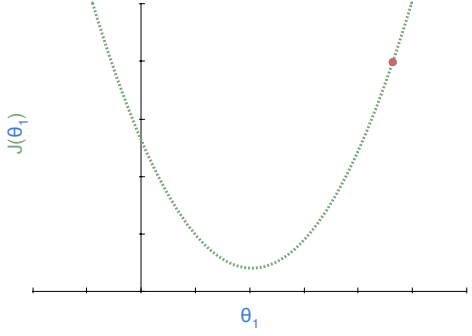
Gradient Descent for One Parameter



Role of the Derivative Term

$$\begin{aligned}\theta_1 &:= \theta_1 - \alpha \cdot \frac{d}{d\theta_1} J(\theta_1) \\ &:= \theta_1 - \alpha \cdot 0 \\ &:= \theta_1 - 0\end{aligned}$$

Gradient Descent for One Parameter

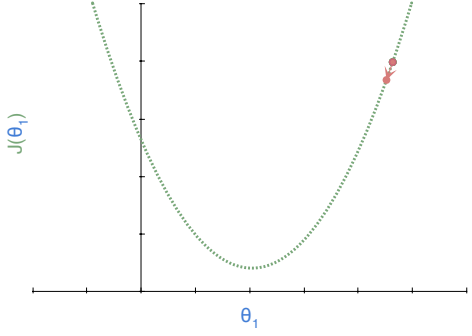


Role of the Learning Rate

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

$$:= \theta_1 - \text{small} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent for One Parameter

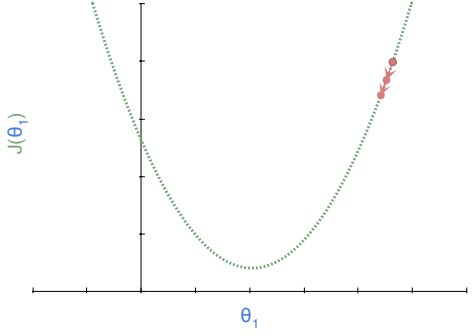


Role of the Learning Rate

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

$$:= \theta_1 - \text{small} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent for One Parameter

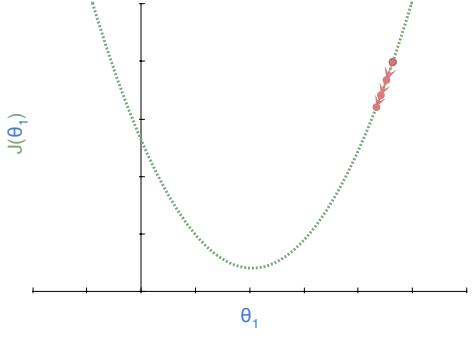


Role of the Learning Rate

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

$$:= \theta_1 - \text{small} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent for One Parameter

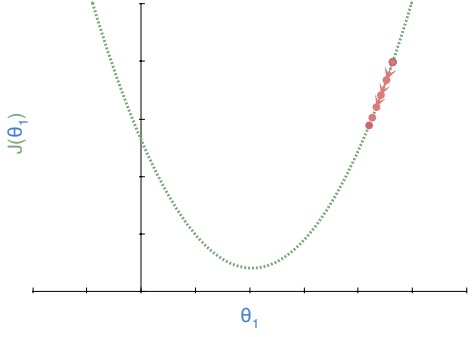


Role of the Learning Rate

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

$$:= \theta_1 - \text{small} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

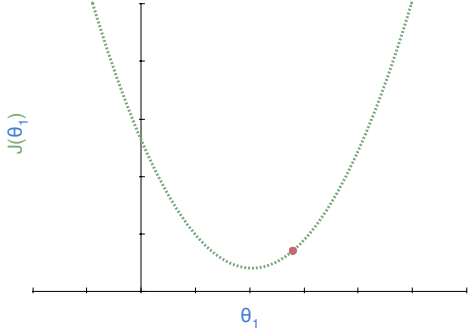
Gradient Descent for One Parameter



Role of the Learning Rate

$$\begin{aligned}\theta_1 &:= \theta_1 - \alpha \cdot \frac{d}{d\theta_1} J(\theta_1) \\ &:= \theta_1 - \text{small} \cdot \frac{d}{d\theta_1} J(\theta_1) \\ &\Rightarrow \text{too slow!}\end{aligned}$$

Gradient Descent for One Parameter

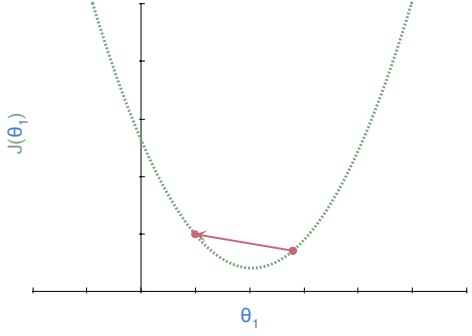


Role of the Learning Rate

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

$$:= \theta_1 - \text{LARGE} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent for One Parameter

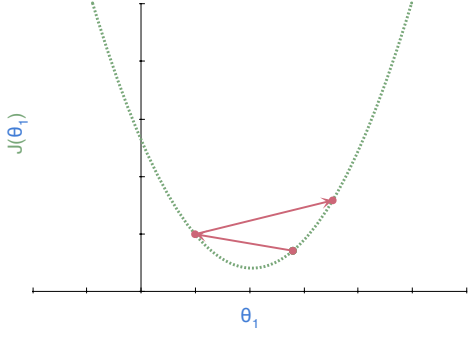


Role of the Learning Rate

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

$$:= \theta_1 - \text{LARGE} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent for One Parameter

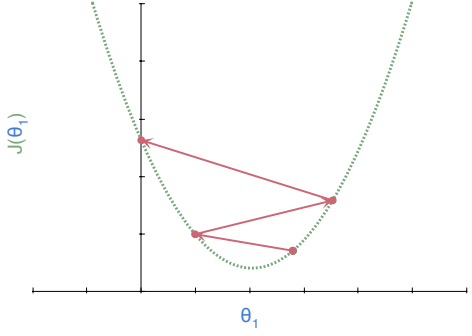


Role of the Learning Rate

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

$$:= \theta_1 - \text{LARGE} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent for One Parameter

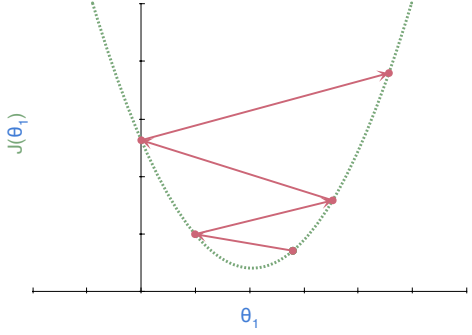


Role of the Learning Rate

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

$$:= \theta_1 - \text{LARGE} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent for One Parameter

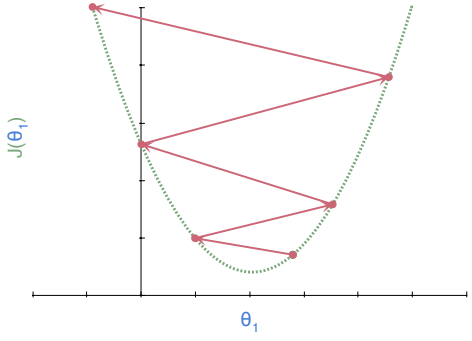


Role of the Learning Rate

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

$$:= \theta_1 - \text{LARGE} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent for One Parameter



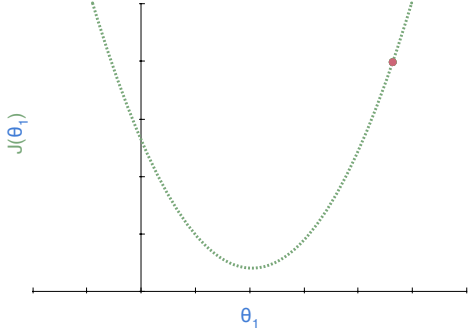
Role of the Learning Rate

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

$$:= \theta_1 - \text{LARGE} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

$$\Rightarrow \text{overshoots!}$$

Gradient Descent for One Parameter

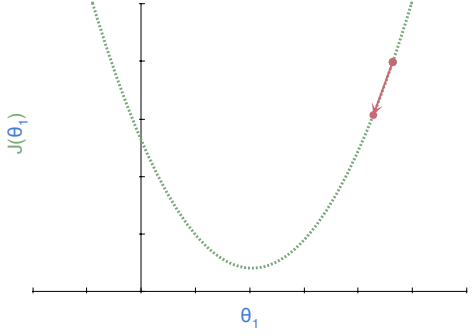


Role of the Learning Rate

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

$$:= \theta_1 - \text{just right!} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent for One Parameter

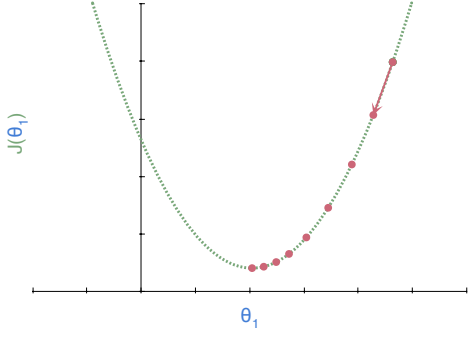


Role of the Learning Rate

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

$$:= \theta_1 - \text{just right!} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent for One Parameter



Role of the Learning Rate

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

$$:= \theta_1 - \text{just right!} \cdot \frac{d}{d\theta_1} J(\theta_1)$$

$$\Rightarrow \text{converges to } 0$$

Choosing a learning rate α

A summary of the effects of **different learning rates**:

If α is **too small**, gradient descent is slow to converge

If α is **too large**, gradient descent may not decrease (and even may increase!) on every iteration, and thus fail to converge.

Choosing a learning rate α

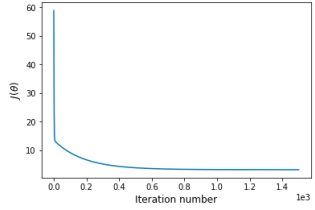
Pick, wiggle and skip to find a good value for α:

1. **Pick** a value, say $\alpha = 0.03$ and run gradient descent
2. **Wiggle** by a factor of 3, so try $\alpha = 0.09$ next
3. **Skip** by a factor of 10, so try $\alpha = 0.9$
4. **Repeat** until you've found a value of α that is too small and a value of α that is too large

Choosing a learning rate α

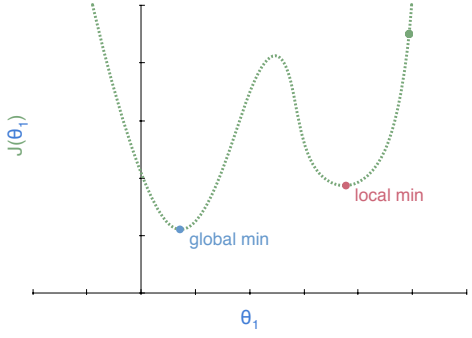
To judge the performance of gradient descent for a particular α:

Plot $J(\theta)$ against number of iterations (e.g., `iterations = 1500`)



This allows you to inspect whether $J(\theta)$ is converging too slow or converging at all for each choice of α

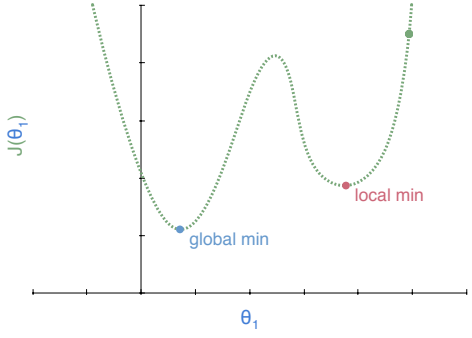
Varieties of Gradient Descent



Local vs Global Minima

Gradient Descent is a *local optimization* method

Varieties of Gradient Descent



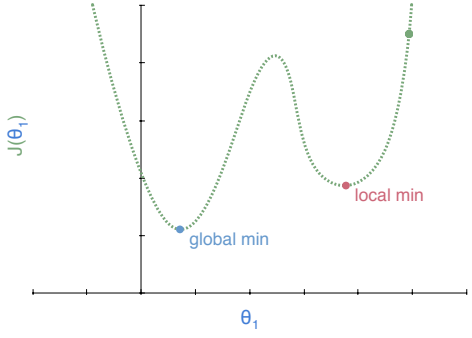
Local vs Global Minima

Gradient Descent is a *local optimization* method

Solution:

Stochastic Gradient Descent

Varieties of Gradient Descent



Local vs Global Minima

Gradient Descent is a *local optimization* method

Solution:

Stochastic Gradient Descent

Varieties of Gradient Descent

- Statistical Gradient Descent
  (sample size M = 10 to 100s)
- Minibatch Gradient Descent
  (sample size M = 10 to 100s)
- Batch Gradient Descent
  (full training set M = *m*)

Recap: Gradient Descent

» Enter loop

Repeat:

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

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

Stop at convergence

» Exit loop

Recap: Gradient Descent

» Enter loop
Repeat:

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

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

Stop at convergence
» Exit loop

Assignment Function

Learning Rate

Derivative Term

Negative Gradient

Simultaneous Update

Gradient Descent and Univariate Linear Regression

Algorithm:
Gradient Descent

» Enter loop
Repeat:

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

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

Stop at convergence
» Exit loop

Model:
Linear Regression

Hypothesis:

$$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$$

Cost Function:

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

Idea: Use gradient descent to minimize the cost function $J(\theta)$.

Gradient Descent and Univariate Linear Regression

Algorithm:

Gradient Descent

» Enter loop
Repeat:

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

for $j = 0, 1$
Stop at convergence

» Exit loop

Implementation:

1. plug in terms

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

$$= \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$$

2. solve equations for θ_0 and θ_1:

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

$$\text{for } \theta_1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

Gradient Descent and Univariate Linear Regression

Algorithm:

Gradient Descent

» Enter loop
Repeat:

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

for $j = 0, 1$
Stop at convergence

» Exit loop

Implementation:

3. plug in partial derivative terms

» Enter loop
Repeat:

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

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

Stop at convergence

» Exit loop

A Linear Algebra Tip

A vector multiplication trick

There is a **trick** in the implementation to minimize $J(\theta)$ with respect to h_θ involves adding a column of 1's to the single feature vector x. The code for doing this in your assignment looks like this:

```
# initialize data array
X = np.ones((m, 2))
X[:,1] = data[:,0]
```

where `data[:,0]` is the single feature vector x from the data set in ps1 stored in the variable `data`, and m is the number of training examples in `data`.

Let's first see how the logic works.

A vector multiplication trick

Suppose we have a data set of $m = 9$ training examples in $\mathbb{R}^2$ of the form (x, y). This means there is a single column of features, x, and a single column of target values, y. If those 9 examples where stored in `data`, the code block on the prior slide would effect the following transformation:

$$\text{Transform features } x = \begin{bmatrix} 0.17 \\ 0.19 \\ 0.22 \\ 0.24 \\ 0.24 \\ 0.30 \\ 0.35 \\ 0.42 \\ 0.85 \end{bmatrix} \text{ into } X = \begin{bmatrix} 1 & 0.17 \\ 1 & 0.19 \\ 1 & 0.22 \\ 1 & 0.24 \\ 1 & 0.24 \\ 1 & 0.30 \\ 1 & 0.35 \\ 1 & 0.42 \\ 1 & 0.85 \end{bmatrix} \text{ in order to fit a model to } y = \begin{bmatrix} 0.63 \\ 0.70 \\ 0.82 \\ 0.88 \\ 1.15 \\ 1.5 \\ 4.40 \\ 7.30 \\ 11.30 \end{bmatrix}$$

A vector multiplication trick

Suppose we have a data set of $m = 9$ training examples in $\mathbb{R}^2$ of the form (x, y). This means there is a single column of features, x, and a single column of target values, y. If those 9 examples where stored in `data`, the code block on the prior slide would effect the following transformation:

$$\text{Transform features } x = \begin{bmatrix} 0.17 \\ 0.19 \\ 0.22 \\ 0.24 \\ 0.24 \\ 0.30 \\ 0.35 \\ 0.42 \\ 0.85 \end{bmatrix} \text{ into } X = \begin{bmatrix} 1 & 0.17 \\ 1 & 0.19 \\ 1 & 0.22 \\ 1 & 0.24 \\ 1 & 0.24 \\ 1 & 0.30 \\ 1 & 0.35 \\ 1 & 0.42 \\ 1 & 0.85 \end{bmatrix} \text{ in order to fit a model to } y = \begin{bmatrix} 0.63 \\ 0.70 \\ 0.82 \\ 0.88 \\ 1.15 \\ 1.5 \\ 4.40 \\ 7.30 \\ 11.30 \end{bmatrix}$$

Let's explain how this works.

Vectorization

The Moore-Penrose psuedoinverse solution to

$$(X^T X)^{-1} X^T y$$

for $X \in \mathbb{R}^{9 \times 2}$ and $y \in \mathbb{R}^2$ is the vector $\boldsymbol{\theta} \in \mathbb{R}^2$ whose component values (rounded to two decimal places) are:

$\theta_0 = -2.48$

$\theta_1 = 17.16$

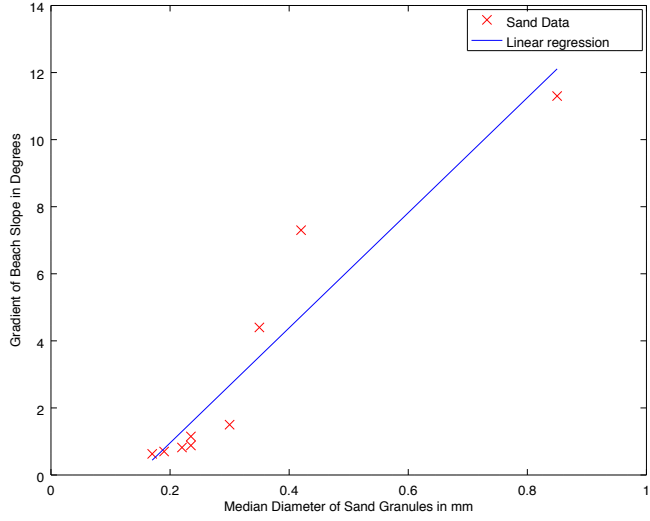
Plugging these two parameter values into the hypothesis $h_\theta(x) = \theta_0 x_0 + \theta_1 x$,

$$h_\theta(x) = -2.48 + 17.16x$$

gives the **least mean squared** (LMS) equation $h_{\boldsymbol{\theta}}$ fit to the 9 training examples.[1]

[1] Recall that $x_0 = 1$, for all 9 examples.

Plot of $h_\theta(x) = -2.48 + 17.16x$



Vectorization

Diameter size (mm)

0.17
0.19
0.22

Hypothesis to predict y

$h(x;\boldsymbol{\theta}) = -2.48 + 17.16x$

Vectorization

Diameter size (mm)

0.17
0.19
0.22

Hypothesis to predict y

$h(x;\theta) = -2.48 + 17.16x$

Problem: How to compute the LMS estimate of (y) using the hypothesis and these three training examples?

Data Matrix

$$X = \begin{bmatrix} 1 & 0.17 \\ 1 & 0.19 \\ 1 & 0.22 \end{bmatrix}$$

Parameter Vector

$$\theta = \begin{bmatrix} -2.48 \\ 17.16 \end{bmatrix}$$

Vectorization

$$h_\theta(x) = -2.48 + 17.16x$$

$$\underset{\text{Matrix } X}{\begin{bmatrix} 1 & 0.17 \\ 1 & 0.19 \\ 1 & 0.22 \end{bmatrix}} \times \underset{\text{Vector } \theta}{\begin{bmatrix} -2.48 \\ 17.16 \end{bmatrix}} = \underset{\text{3 x 1 Vector}}{\begin{bmatrix} -2.48 \cdot 1 + 17.16 \cdot 0.17 \\ \\ \end{bmatrix}}$$

Vectorization

$$h_\theta(x) = -2.48 + 17.16x$$

$$\underset{\textbf{Matrix } X}{\begin{bmatrix} 1 & 0.17 \\ 1 & 0.19 \\ 1 & 0.22 \end{bmatrix}} \times \underset{\textbf{Vector } \theta}{\begin{bmatrix} -2.48 \\ 17.16 \end{bmatrix}} = \underset{\textbf{3 x 1 Vector}}{\begin{bmatrix} -2.48 \cdot 1 + 17.16 \cdot 0.17 \\ -2.48 \cdot 1 + 17.16 \cdot 0.19 \\ \end{bmatrix}}$$

Vectorization

$$h_\theta(x) = -2.48 + 17.16x$$

$$\underset{\text{Matrix } X}{\begin{bmatrix} 1 & 0.17 \\ 1 & 0.19 \\ 1 & 0.22 \end{bmatrix}} \times \underset{\text{Vector } \theta}{\begin{bmatrix} -2.48 \\ 17.16 \end{bmatrix}} = \underset{\text{3 x 1 Vector}}{\begin{bmatrix} -2.48 \cdot 1 + 17.16 \cdot 0.17 \\ -2.48 \cdot 1 + 17.16 \cdot 0.19 \\ -2.48 \cdot 1 + 17.16 \cdot 0.22 \end{bmatrix}}$$

Vectorization

$$h_\theta(x) = -2.48 + 17.16x$$

$$\textbf{Matrix } X \quad \times \quad \textbf{Vector } \theta \; = \qquad \textbf{3 x 1 Vector}$$

$$\begin{bmatrix} 1 & 0.17 \\ 1 & 0.19 \\ 1 & 0.22 \end{bmatrix} \times \begin{bmatrix} -2.48 \\ 17.16 \end{bmatrix} = \begin{bmatrix} 0.4372 \\ 0.7804 \\ 1.2952 \end{bmatrix}$$

Vectorization

$$\underset{\text{Matrix } X}{\begin{bmatrix} 1 & 0.17 \\ 1 & 0.19 \\ 1 & 0.22 \end{bmatrix}} \times \underset{\text{Vector } \theta}{\begin{bmatrix} -2.48 \\ 17.16 \end{bmatrix}} = \underset{\text{3 x 1 Vector}}{\begin{bmatrix} 0.4372 \\ 0.7804 \\ 1.2952 \end{bmatrix}}$$

Idea: predicted values $h_\theta(x)$ = Data matrix X * parameter vector θ.
Benefits:

- A simple line of code (instead of a `for` loop, for example)
- *Much* more computationally efficient
- Will scale to large data sets

Vectorizing Code in Python

Why Least Squares?

Gauss-Markov Theorem

The least squares estimates of the parameter vector θ have the smallest variance among all linear **unbiased** estimates.

Why Least Squares?

Gauss-Markov Theorem

The least squares estimates of the parameter vector θ have the smallest variance among all linear **unbiased** estimates.

However, there are **biased** estimators with lower mean-squared error: such an estimator trades some bias for a reduction in variance.

Examples:

- Subset selection
- Ridge regression

References

Mitchell, T. M. (1997).
Machine Learning.
New York: McGraw-Hill.