Indian Institute of Information Technology, Allahabad

Introduction to Machine Learning

Gradient Descent

By

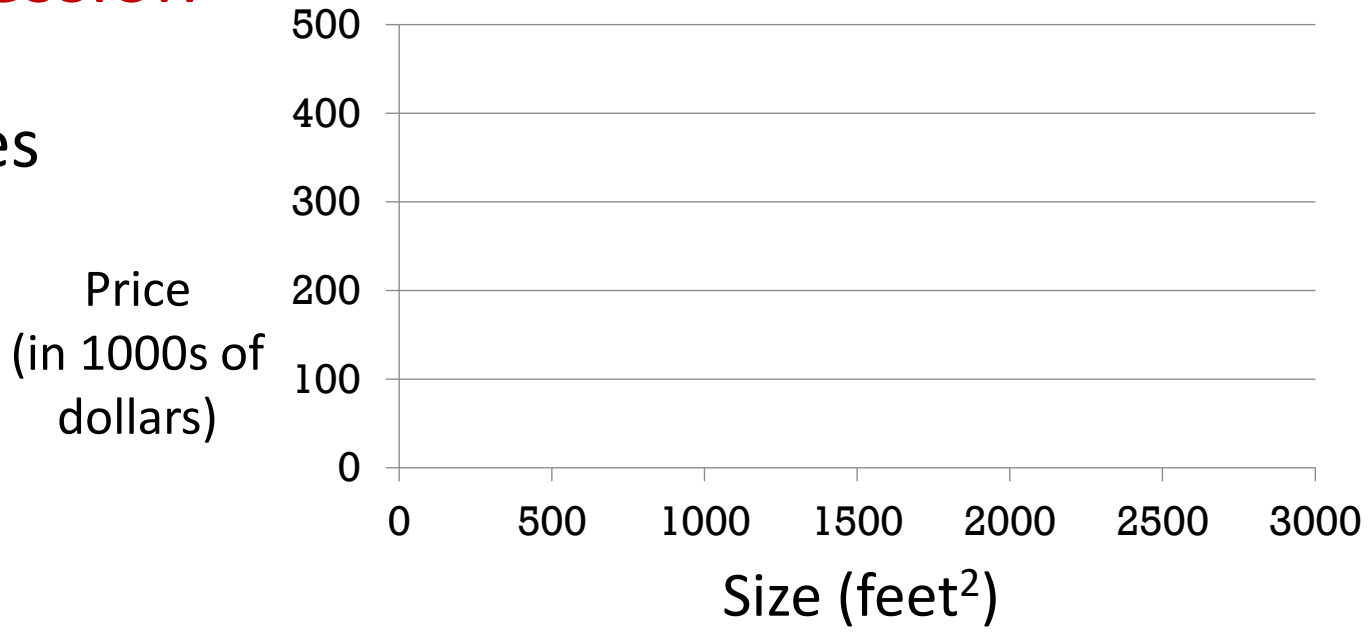
Dr. Shiv Ram Dubey
Associate Professor
Computer Vision and Biometrics Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad
Web: https://profile.iiita.ac.in/srdubey/

DISCLAIMER

The content (text, image, and graphics) used in this slide are adopted from many sources for academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.

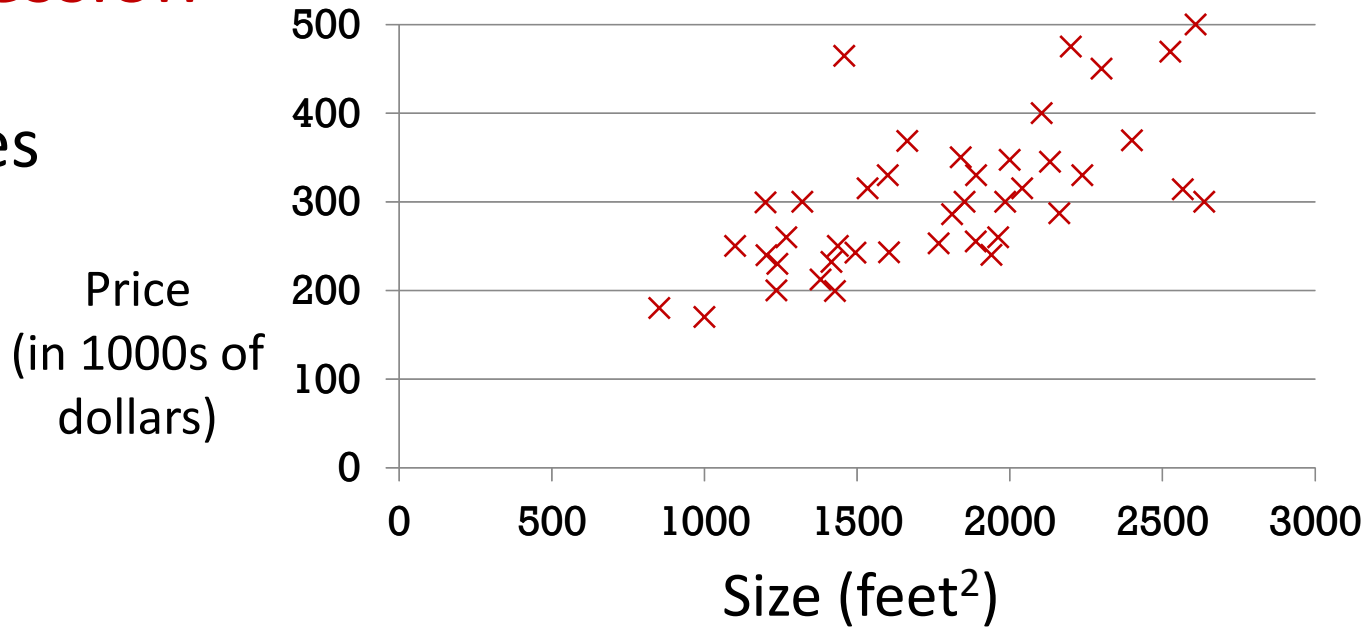
Linear Regression

Housing Prices



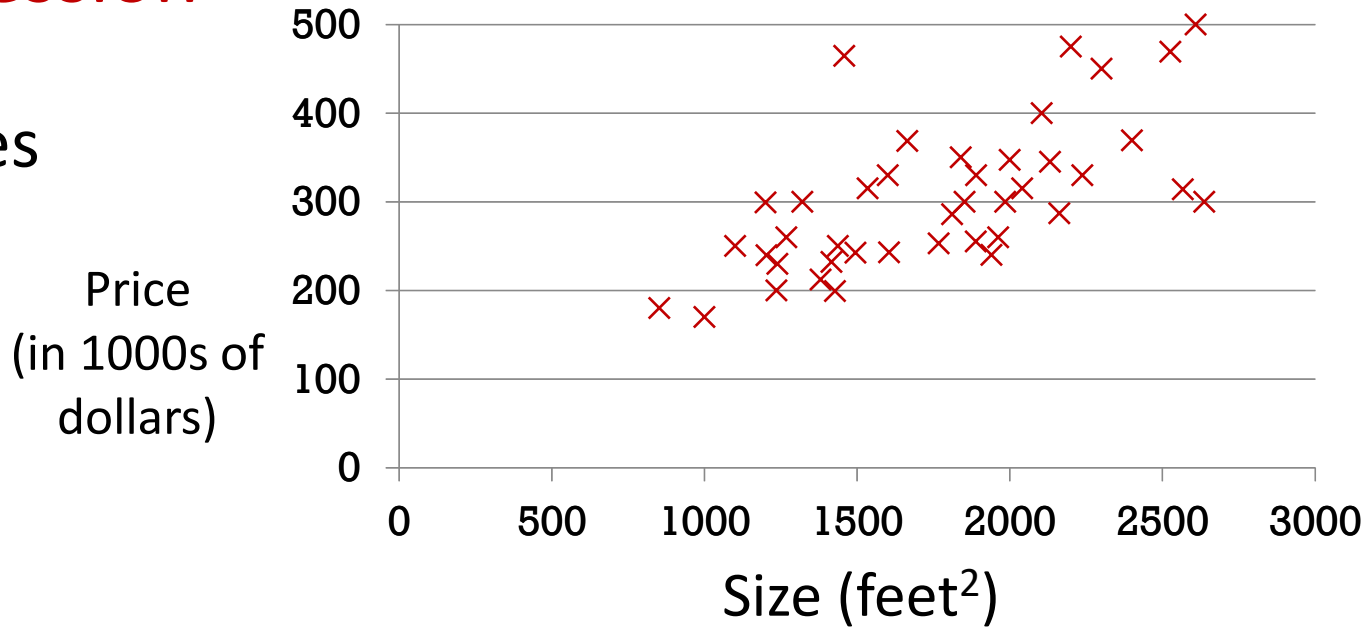
Linear Regression

Housing Prices



Linear Regression

Housing Prices

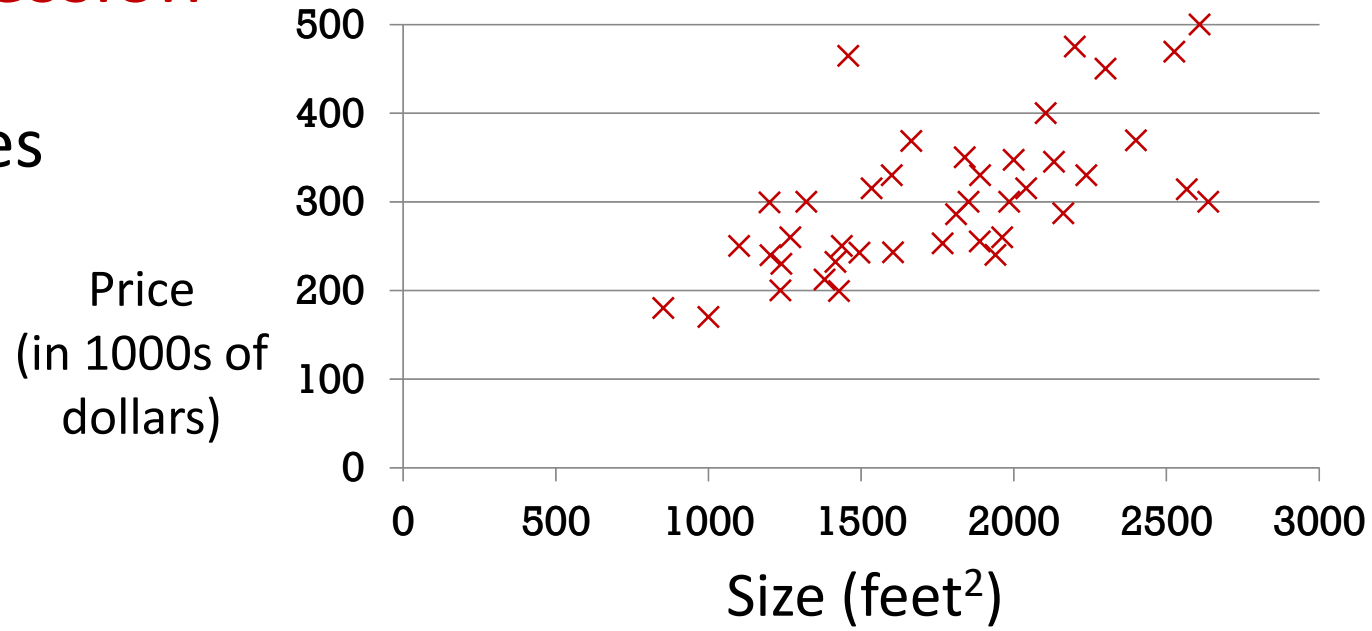


Supervised Learning

Given the “right answer” for each example in the data.

Linear Regression

Housing Prices



Supervised Learning

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

Regression Problem

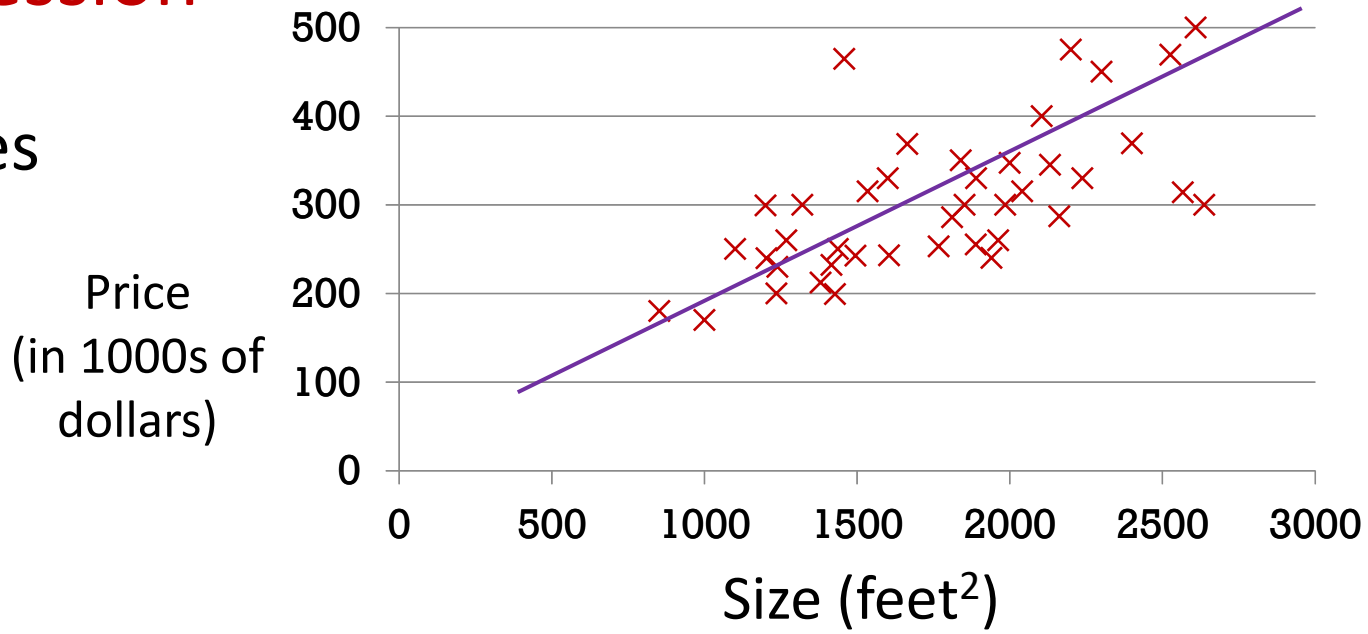
Predict real-valued output

Classification Problem

Discrete real-valued output

Linear Regression

Housing Prices



Supervised Learning

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

Regression Problem

Predict real-valued output

Classification Problem

Discrete real-valued output

Linear Regression with One Variable

Training set of housing prices

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

Notation:

n = Number of training examples

x's = "input" variable / features

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

(x, y) – One Training Example

$(x^{(i)}, y^{(i)})$ – i^{th} Training Example

Cost Function

Training Set

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

Hypothesis: $h_w(x) = w_0 + w_1 x$

w_i's: Parameters

Cost Function

Training Set

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

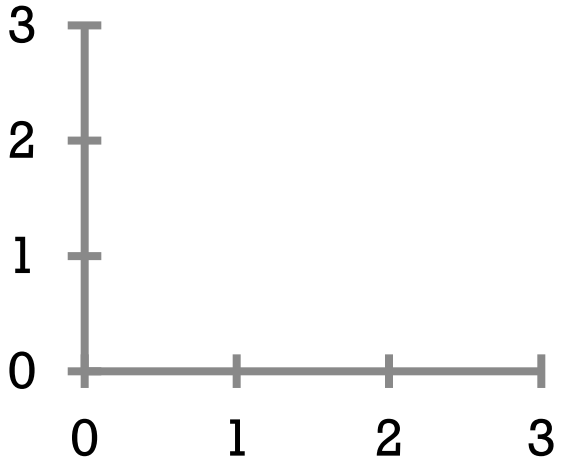
Hypothesis: $h_w(x) = w_0 + w_1 x$

w_i's: Parameters

How to choose w_i's ?

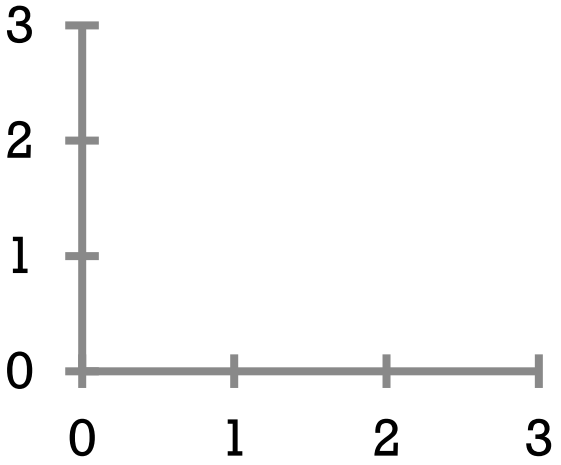
Cost Function

$h_w(x) = w_0 + w_1 x$



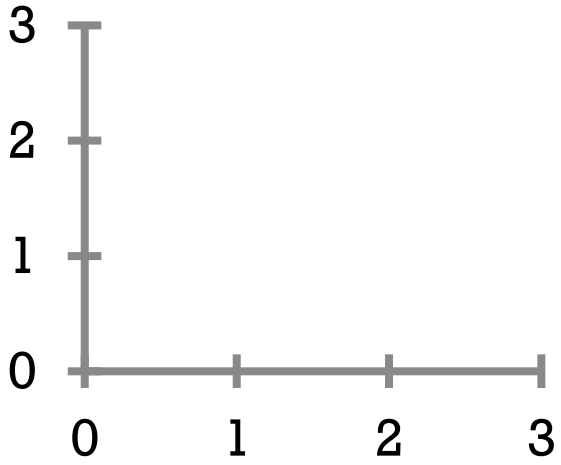
$w_0 = 1.5$

$w_1 = 0$



$w_0 = 0$

$w_1 = 0.5$

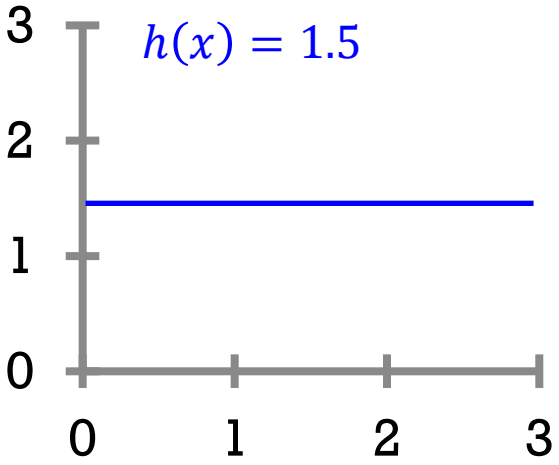


$w_0 = 1$

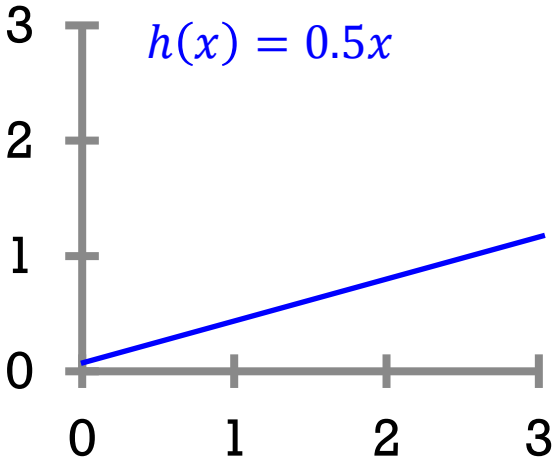
$w_1 = 0.5$

Cost Function

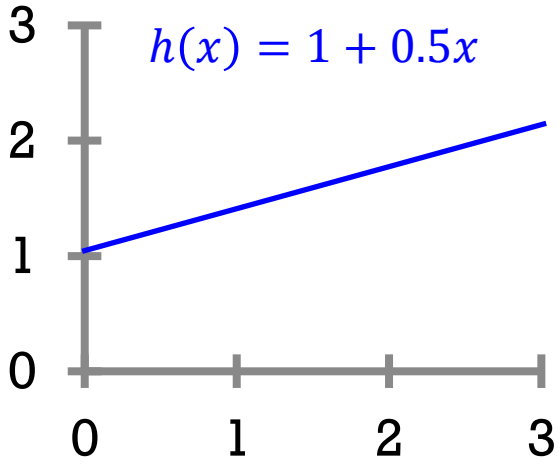
$$h_w(x) = w_0 + w_1 x$$



$w_0 = 1.5$
$w_1 = 0$

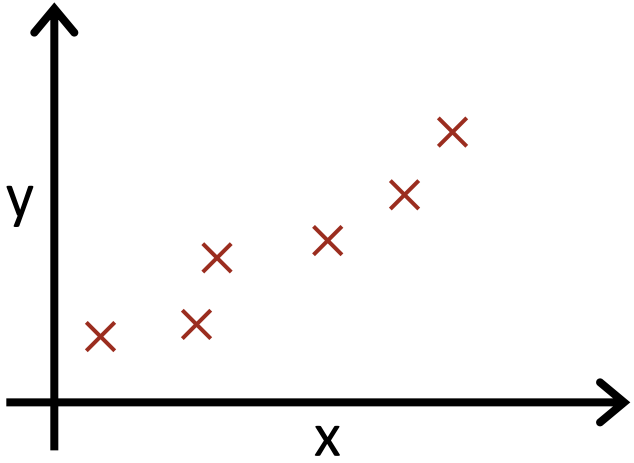


$w_0 = 0$
$w_1 = 0.5$



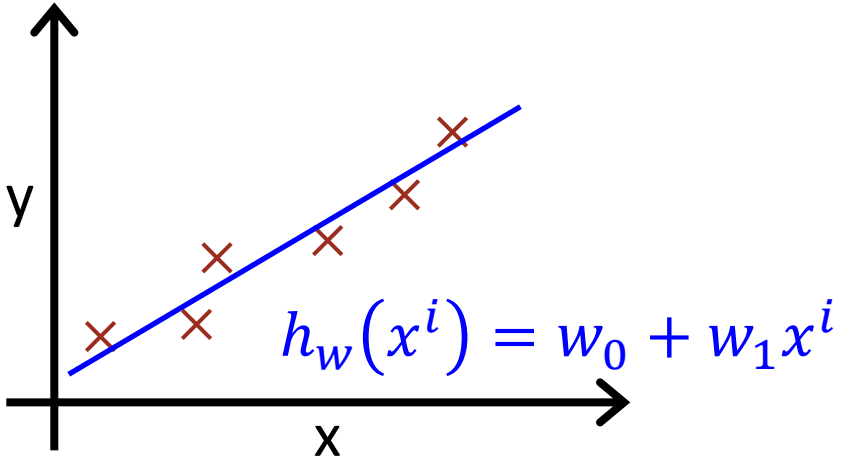
$w_0 = 1$
$w_1 = 0.5$

Cost Function



Idea: Choose w_0 and w_1 so that $h_w(x)$ is close to y for our training examples (x, y)

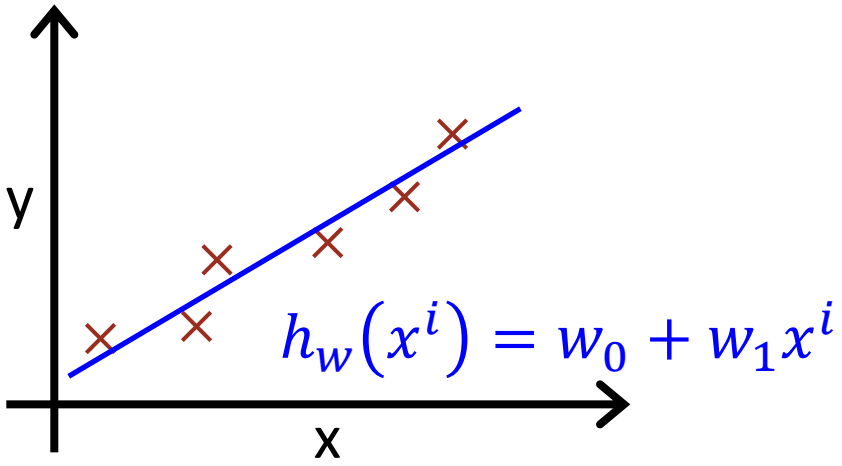
Cost Function



Idea: Choose w_0 and w_1 so that $h_w(x)$ is close to y for our training examples (x, y)

Cost Function

n -> # Training Examples

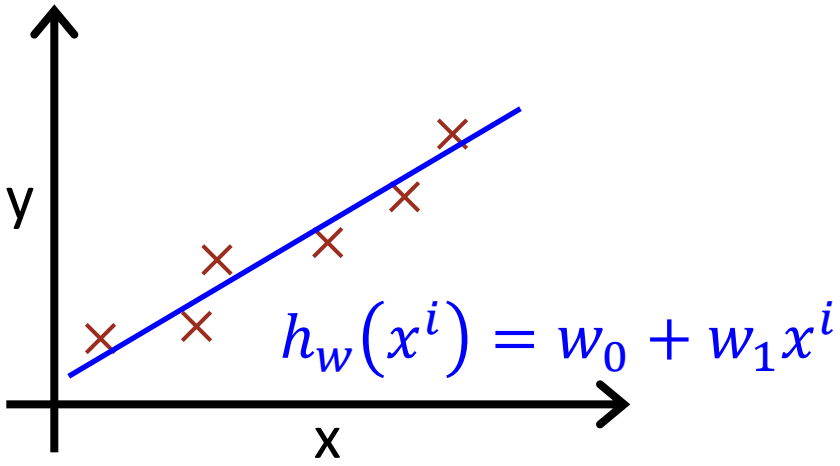


$$\text{minimize} \frac{1}{2n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$

Idea: Choose w_0 and w_1 so that $h_w(x)$ is close to y for our training examples (x, y)

Cost Function

n -> # Training Examples



$$\text{minimize} \frac{1}{2n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$

$$J(w_0, w_1) = \frac{1}{2n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$

Idea: Choose w_0 and w_1 so that $h_w(x)$ is close to y for our training examples (x, y)

$$\underset{w_0, w_1}{\text{minimize}}\, J(w_0, w_1)$$ Cost Function

Squared Error Function

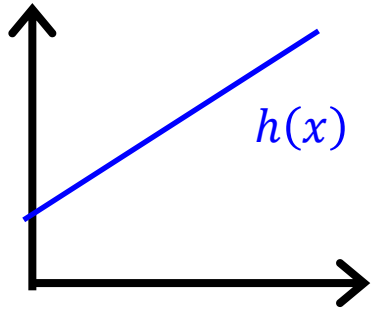
Cost Function

Hypothesis:

$$h_w(x) = w_0 + w_1 x$$

Parameters:

$$w_0, w_1$$



Cost Function:

$$J(w_0, w_1) = \frac{1}{2n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$

Goal: $\underset{w_0, w_1}{\text{minimize}}\, J(w_0, w_1)$

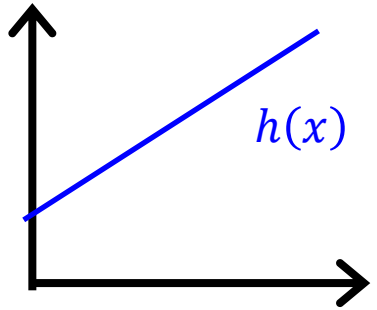
Cost Function

Hypothesis:

$$h_w(x) = w_0 + w_1 x$$

Parameters:

$$w_0, w_1$$



Cost Function:

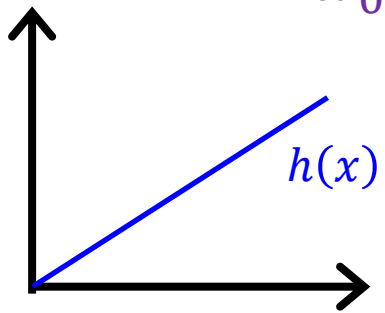
$$J(w_0, w_1) = \frac{1}{2n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$

Goal: $\underset{w_0, w_1}{\text{minimize}}\, J(w_0, w_1)$

Simplified

$$h_w(x) = w_1 x$$

$w_0 = 0$

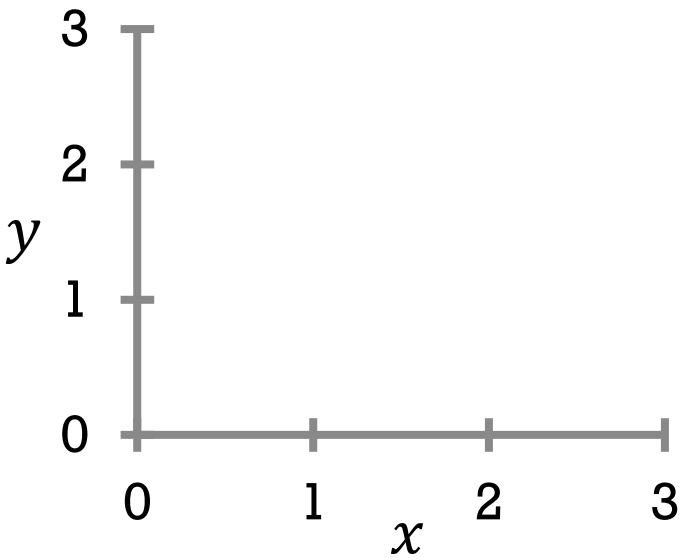


$$J(w_1) = \frac{1}{2n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$

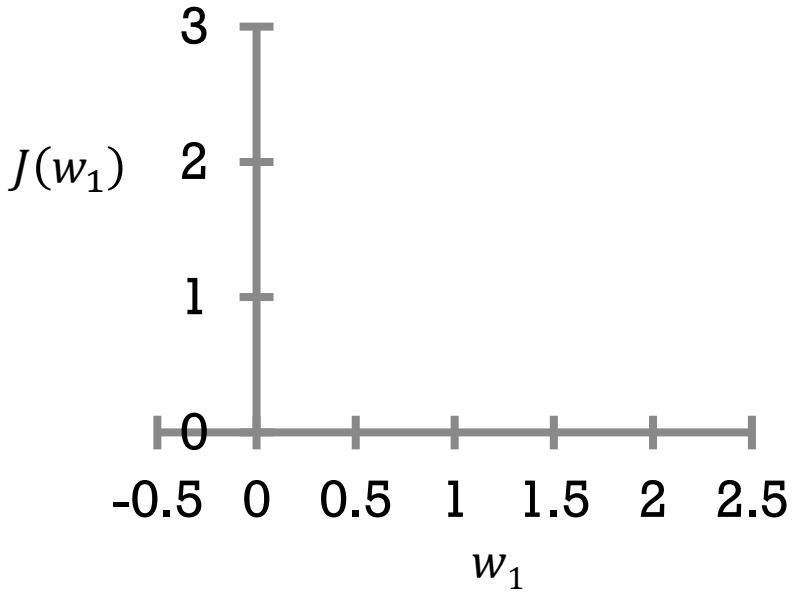
$\underset{w_1}{\text{minimize}}\, J(w_1)$

Cost Function

$h_w(x)$ (for fixed w_1, this is a function of x)

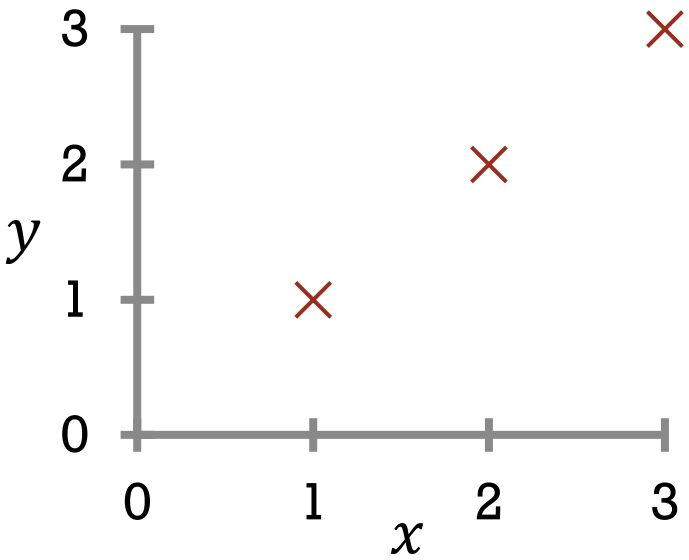


$J(w_1)$ (function of the parameter w_1)

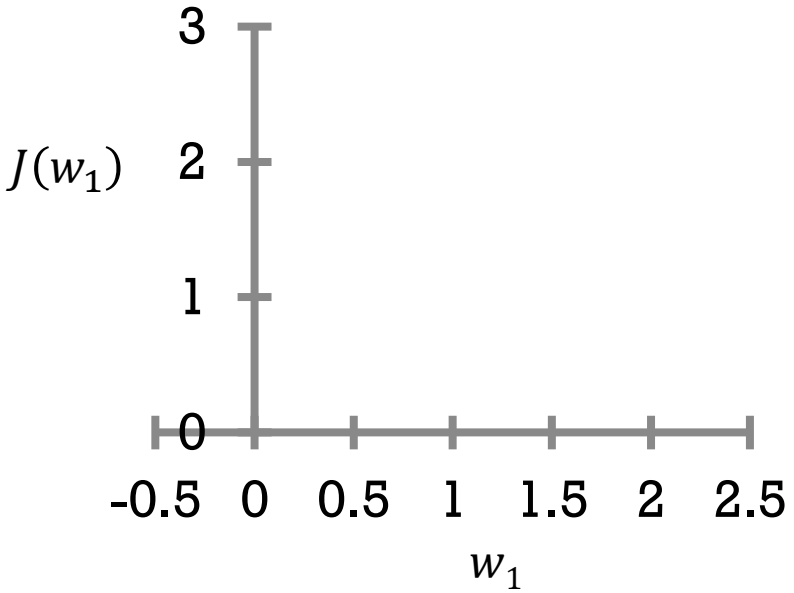


Cost Function

$h_w(x)$ (for fixed w_1, this is a function of x)

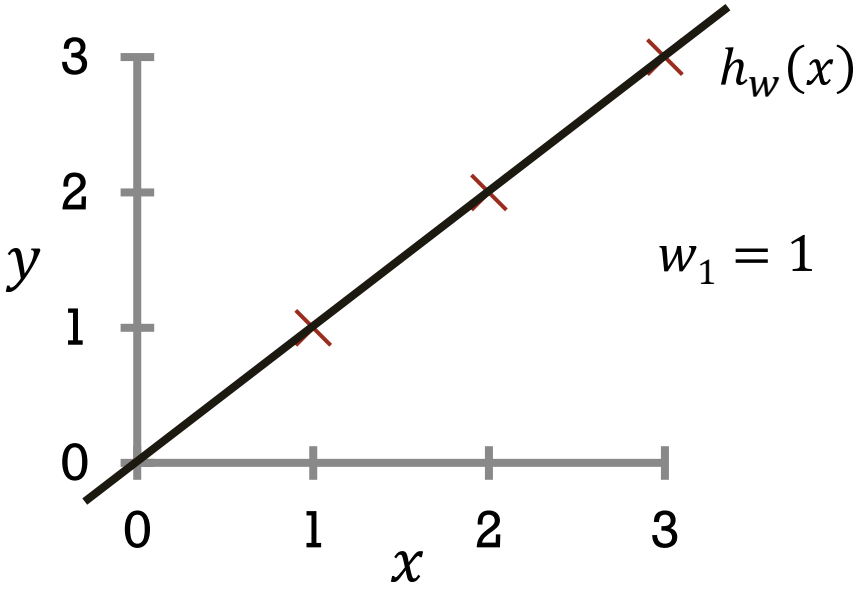


$J(w_1)$ (function of the parameter w_1)

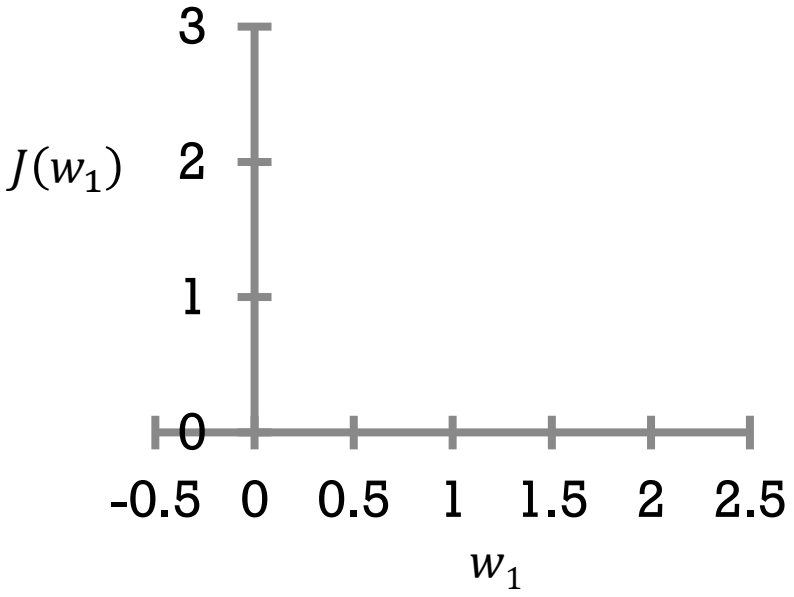


Cost Function

$h_w(x)$ (for fixed w_1, this is a function of x)

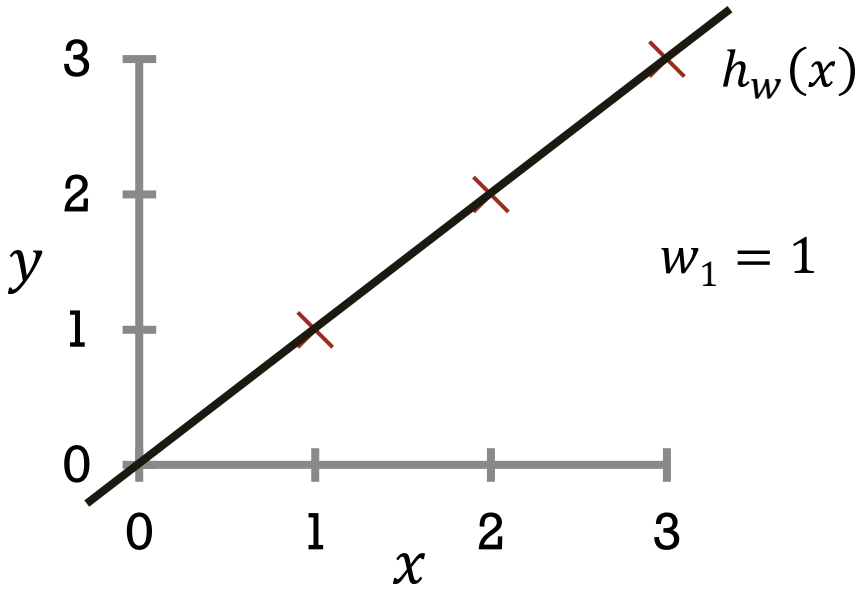


$J(w_1)$ (function of the parameter w_1)

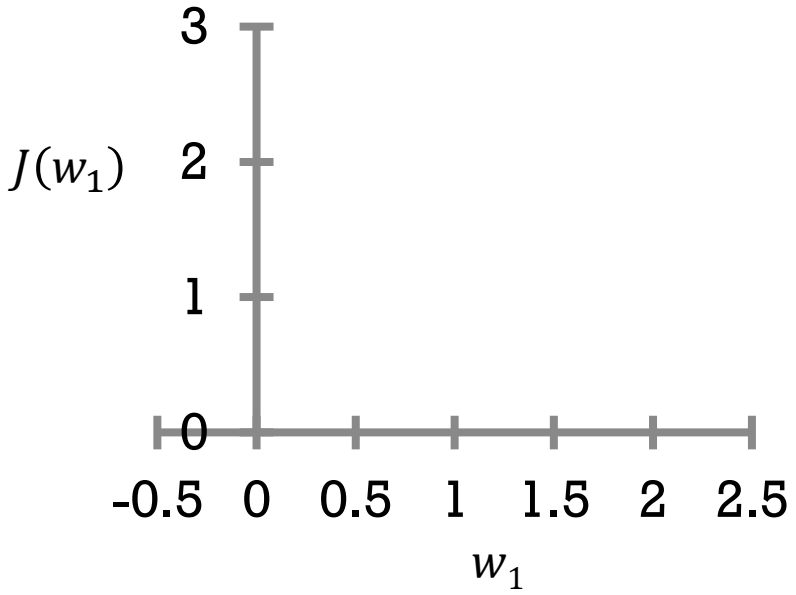


Cost Function

$h_w(x)$ (for fixed w_1, this is a function of x)



$J(w_1)$ (function of the parameter w_1)

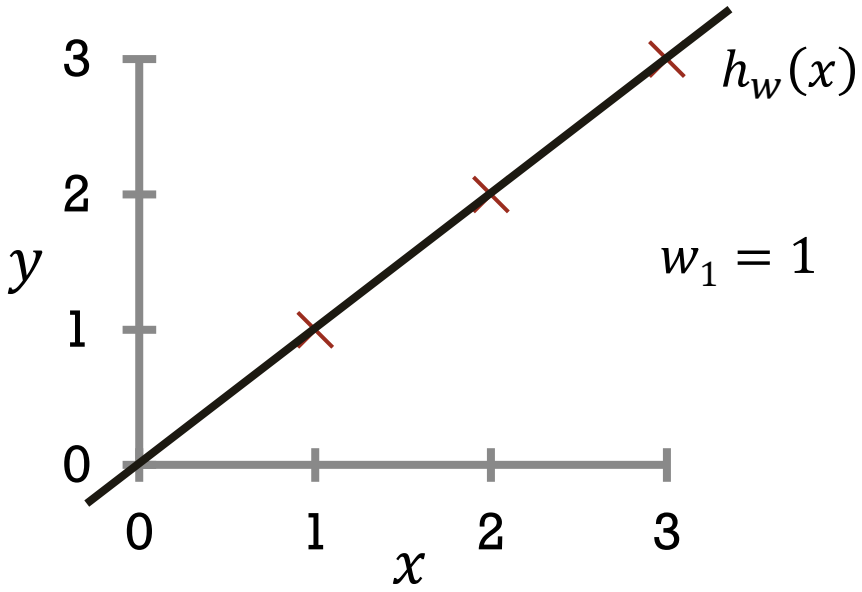


$$J(w_1) = \frac{1}{2n}\sum_{i=1}^{n}(h_w(x^i) - y^i)^2$$

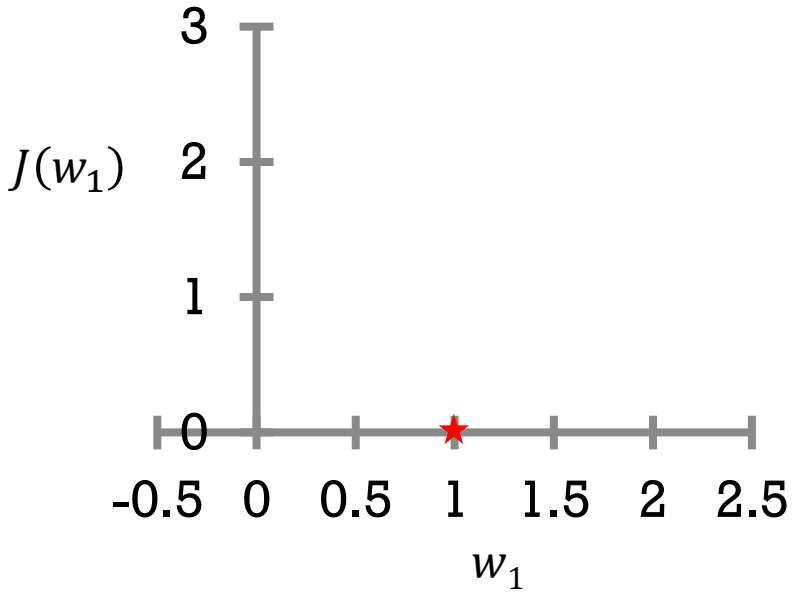
$$h_w(x^i) = w_1 x^i$$

Cost Function

$h_w(x)$ (for fixed w_1, this is a function of x)



$J(w_1)$ (function of the parameter w_1)



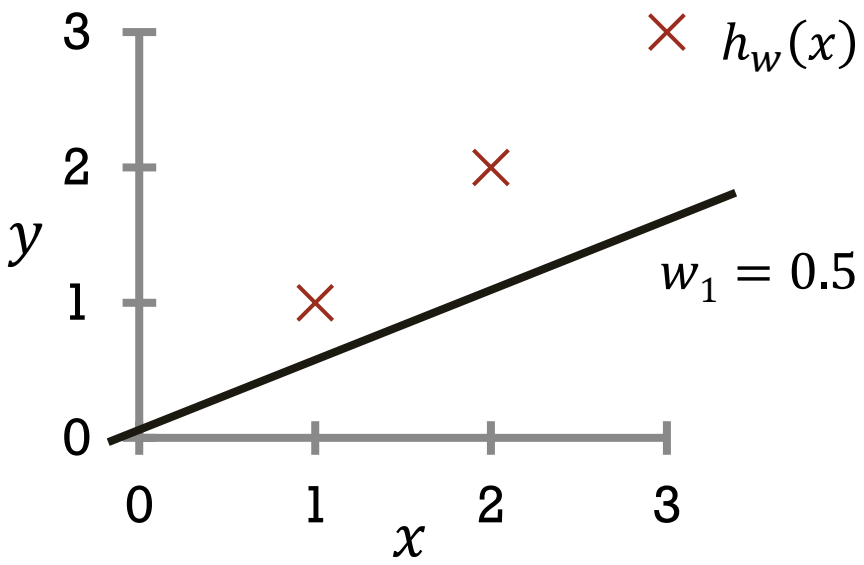
$$J(w_1) = \frac{1}{2n}\sum_{i=1}^{n}(h_w(x^i) - y^i)^2$$

$$h_w(x^i) = w_1 x^i$$

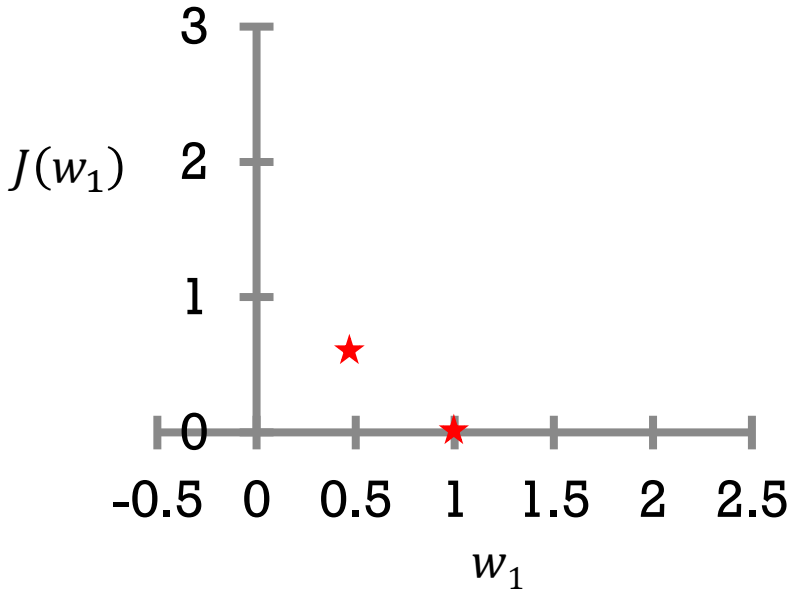
$$J(1) = \frac{1}{2*3}(0^2 + 0^2 + 0^2) = 0$$

Cost Function

$h_w(x)$ (for fixed w_1, this is a function of x)



$J(w_1)$ (function of the parameter w_1)



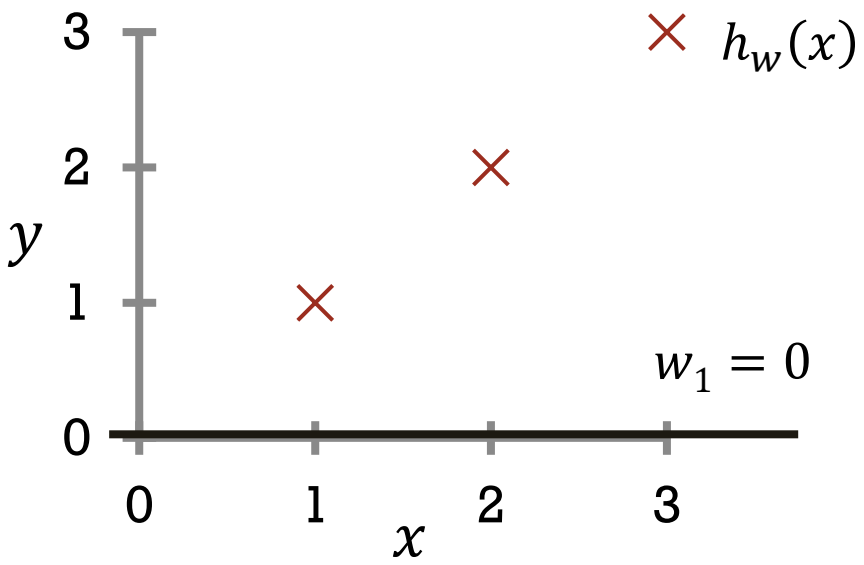
$$J(w_1) = \frac{1}{2n}\sum_{i=1}^{n}(h_w(x^i) - y^i)^2$$

$$h_w(x^i) = w_1 x^i$$

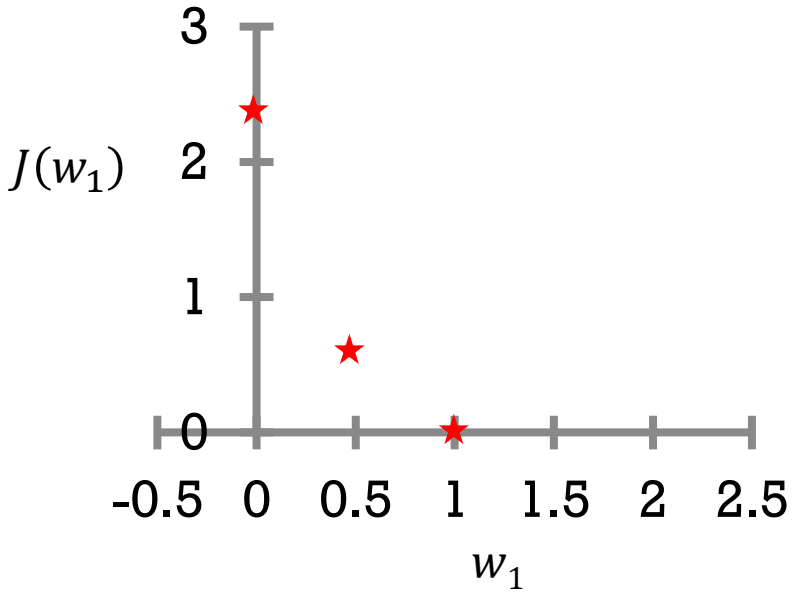
$$J(0.5) = \frac{1}{2*3}\left((0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right) = 0.58$$

Cost Function

$h_w(x)$ (for fixed w_1, this is a function of x)



$J(w_1)$ (function of the parameter w_1)



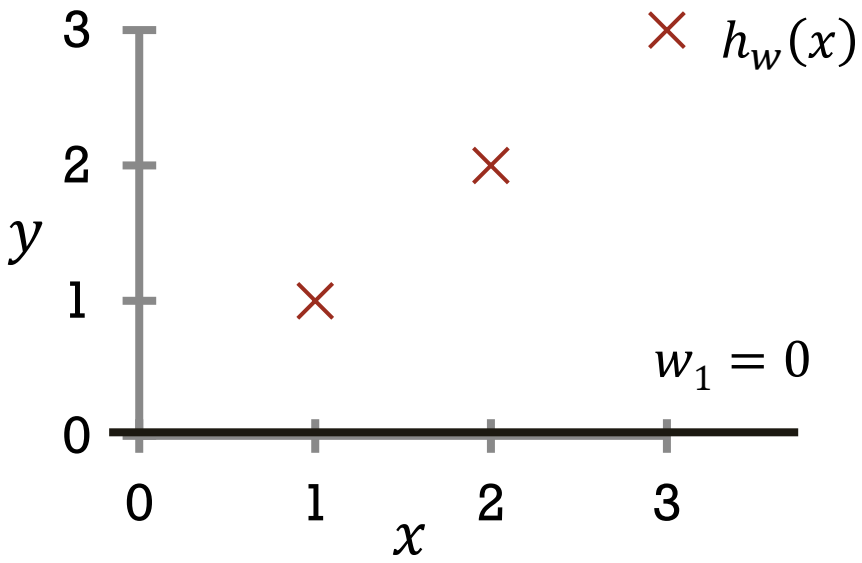
$$J(w_1) = \frac{1}{2n}\sum_{i=1}^{n}(h_w(x^i) - y^i)^2$$

$$h_w(x^i) = w_1 x^i$$

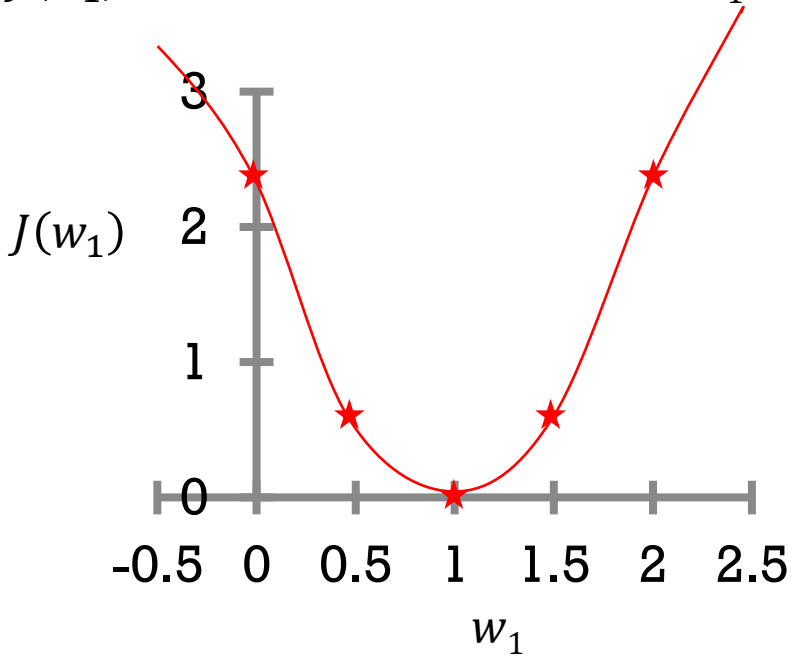
$$J(0) = \frac{1}{2*3}\left((0-1)^2 + (0-2)^2 + (0-3)^2\right) = 2.33$$

Cost Function

$h_w(x)$ (for fixed w_1, this is a function of x)



$J(w_1)$ (function of the parameter w_1)



$$J(w_1) = \frac{1}{2n}\sum_{i=1}^{n}(h_w(x^i) - y^i)^2$$

$$h_w(x^i) = w_1 x^i$$

Cost Function

Hypothesis: $h_w(x) = w_0 + w_1 x$

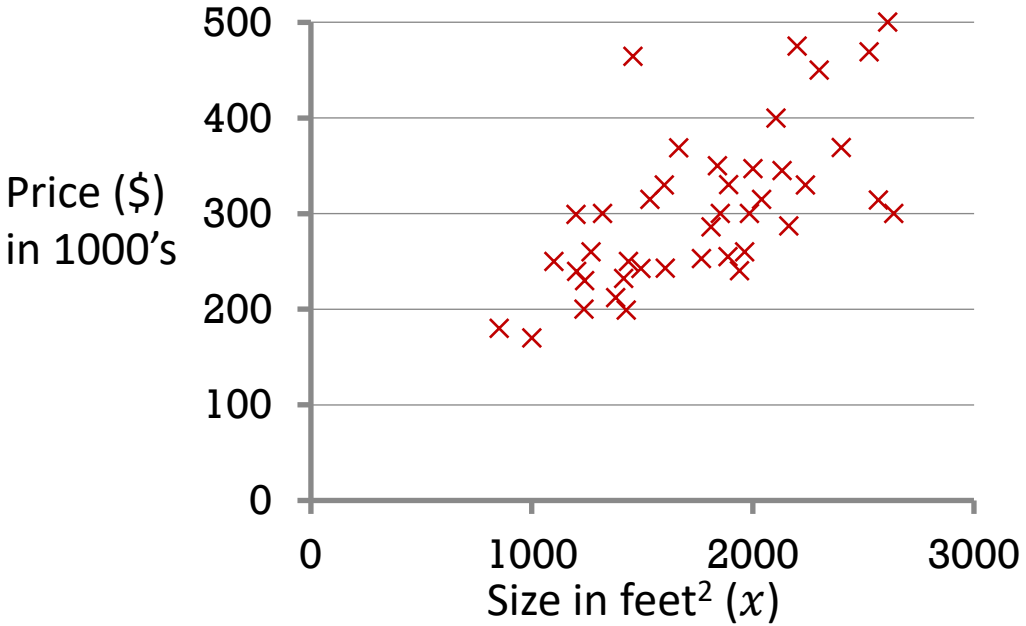
Parameters: w_0, w_1

Cost Function: $J(w_0, w_1) = \frac{1}{2n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$

Goal: $\underset{w_0, w_1}{\text{minimize}}\, J(w_0, w_1)$

Cost Function

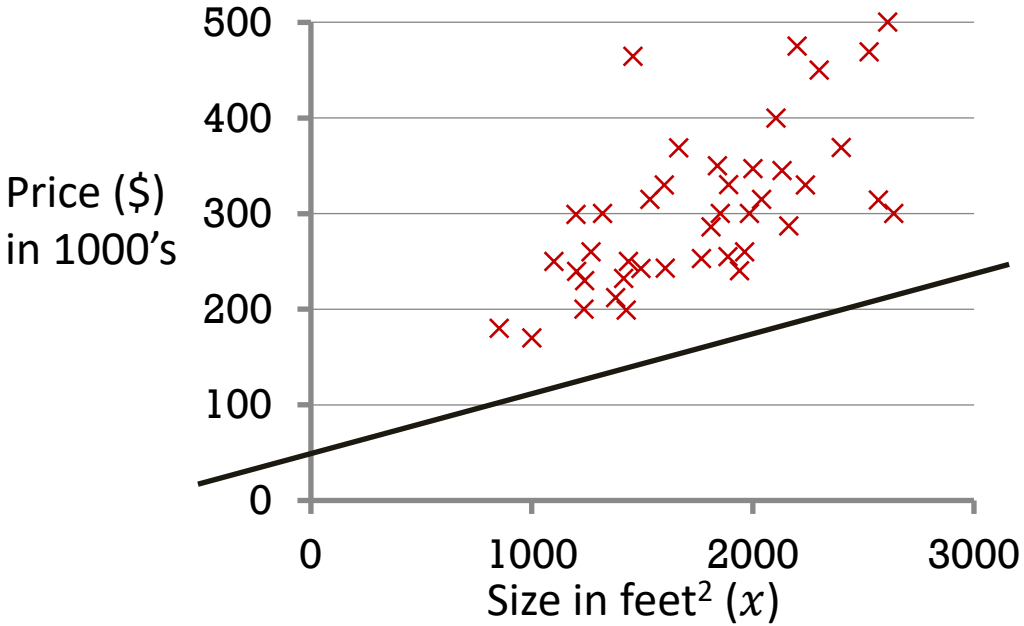
$h_w(x)$ (for fixed w_0, w_1, this is a function of x)



$J(w_0, w_1)$ (function of the parameters) w_0, w_1

Cost Function

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

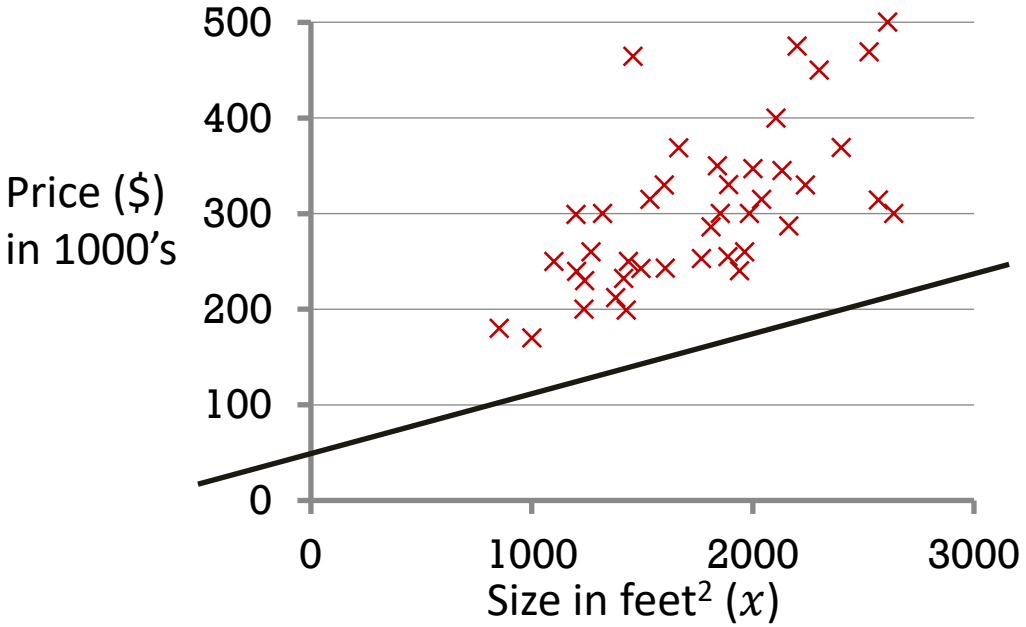


$h_w(x) = 50 + 0.06x$

$J(w_0, w_1)$ (function of the parameters) w_0, w_1

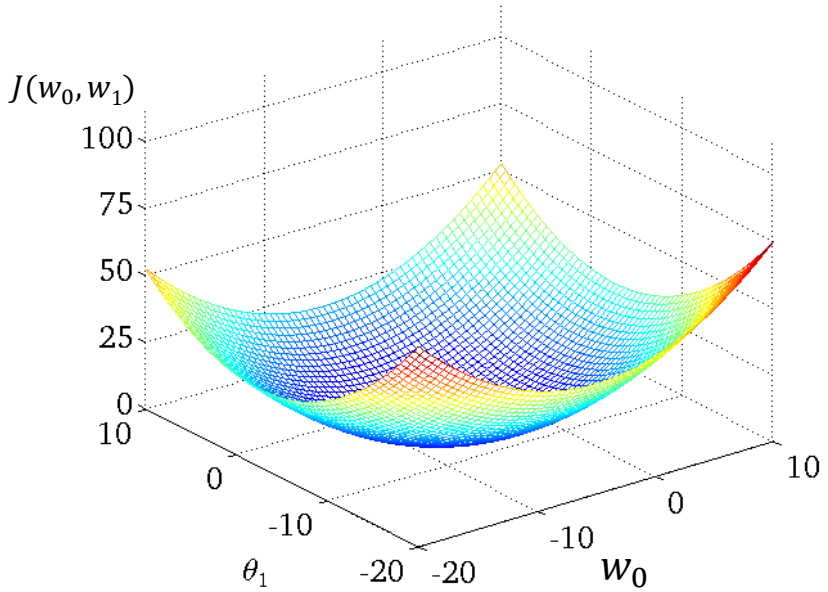
Cost Function

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)



$h_w(x) = 50 + 0.06x$

$J(w_0, w_1)$ (function of the parameters) w_0, w_1

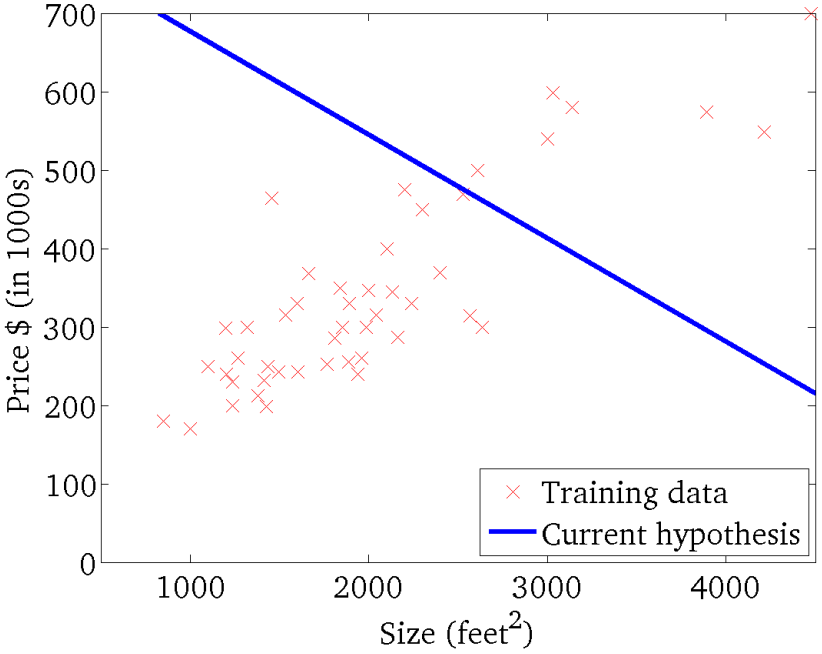


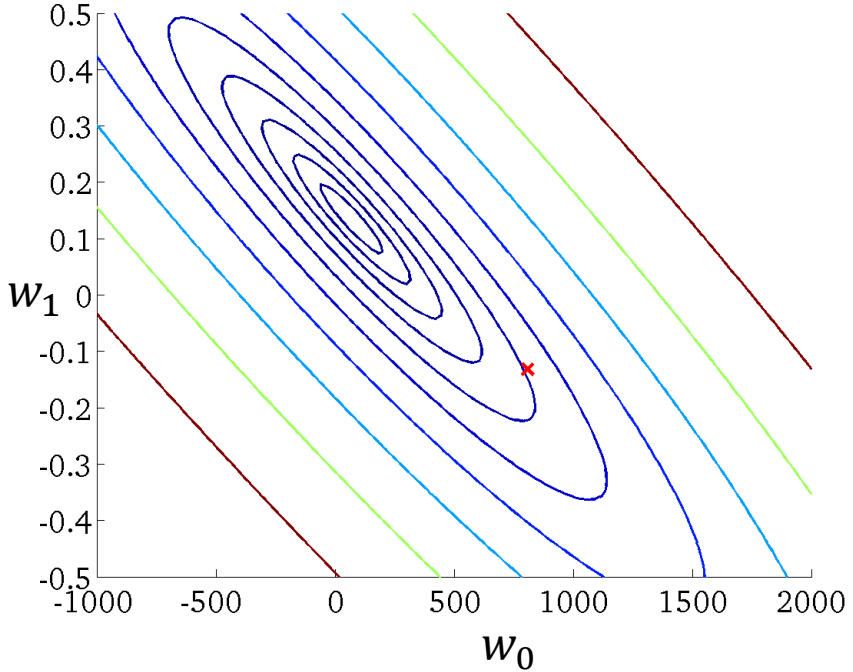
Contour Plot
Contour Figure

Cost Function

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)



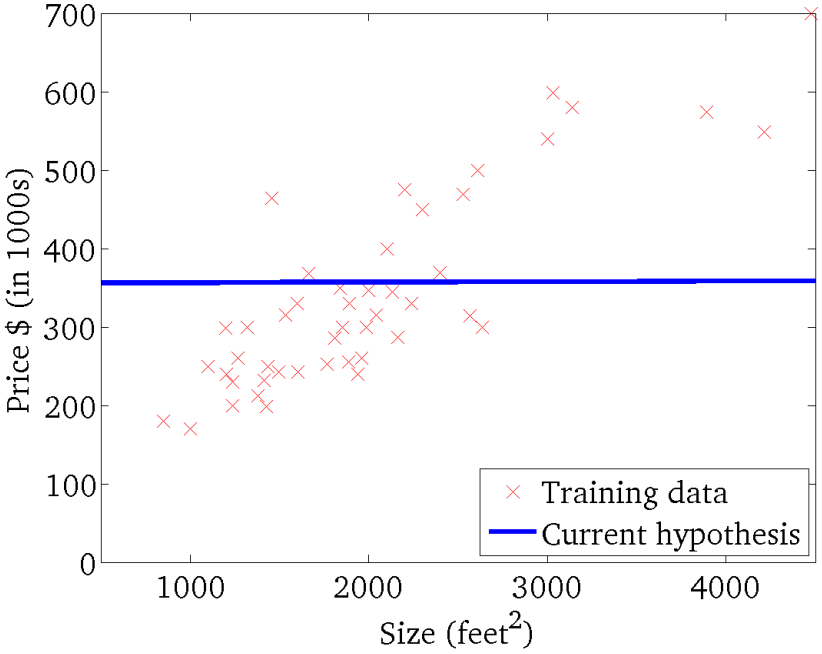


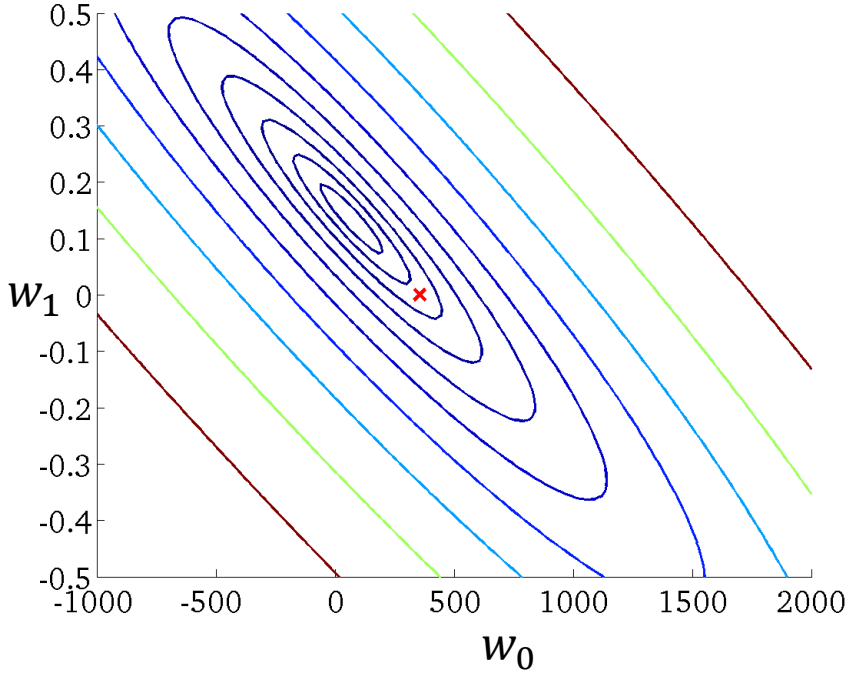
$h(x) = 800 - 0.15x$

Cost Function

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)



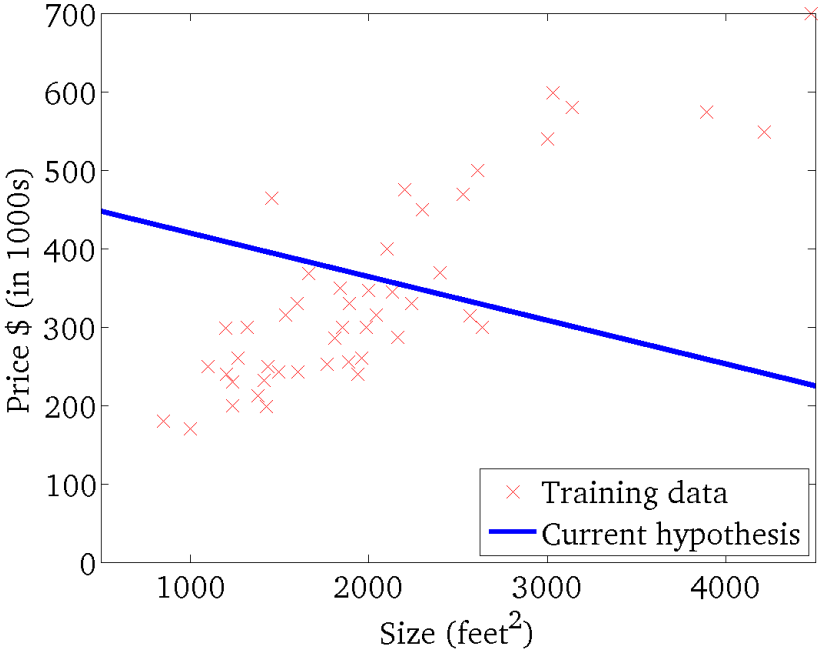


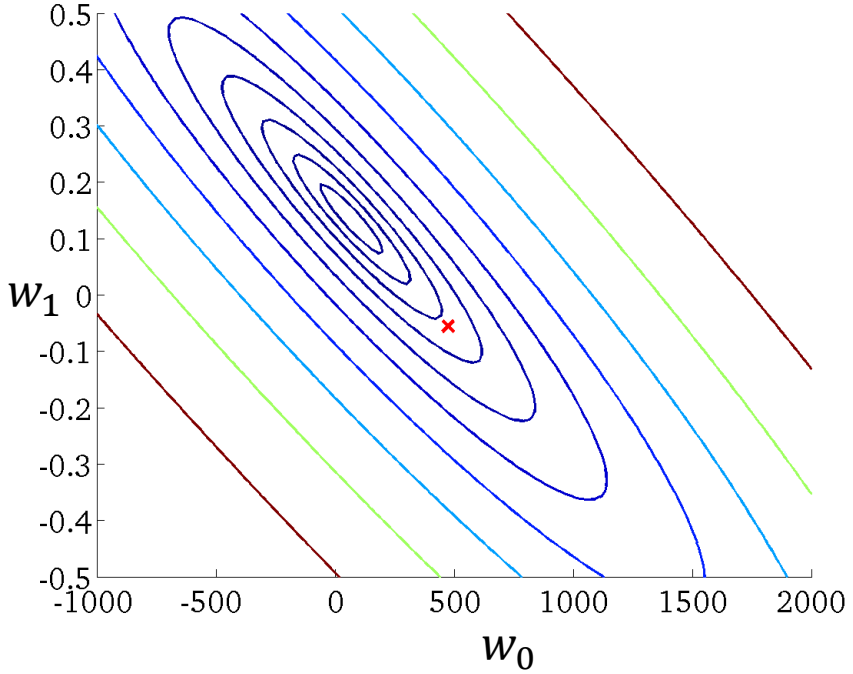
$h(x) = 360 - 0\,.x$

Cost Function

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)

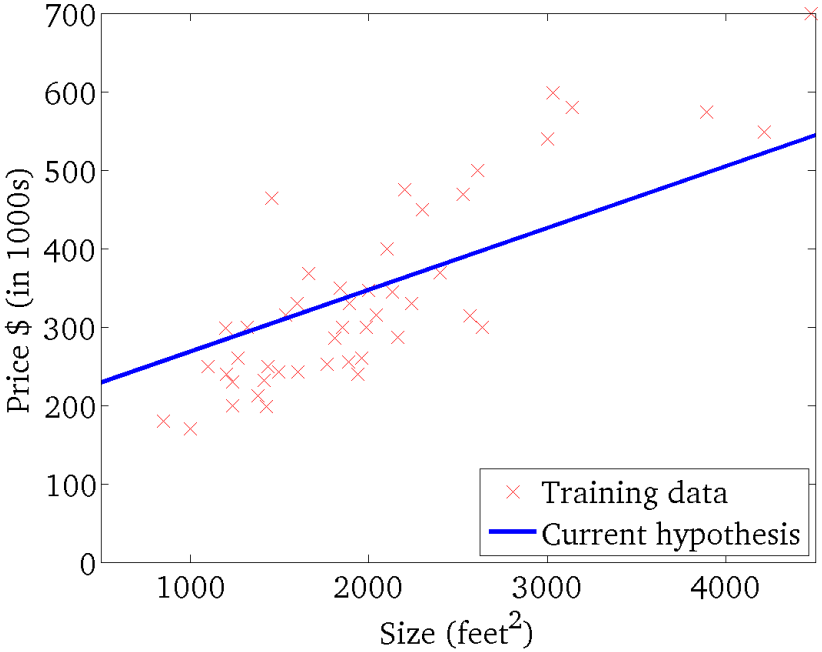


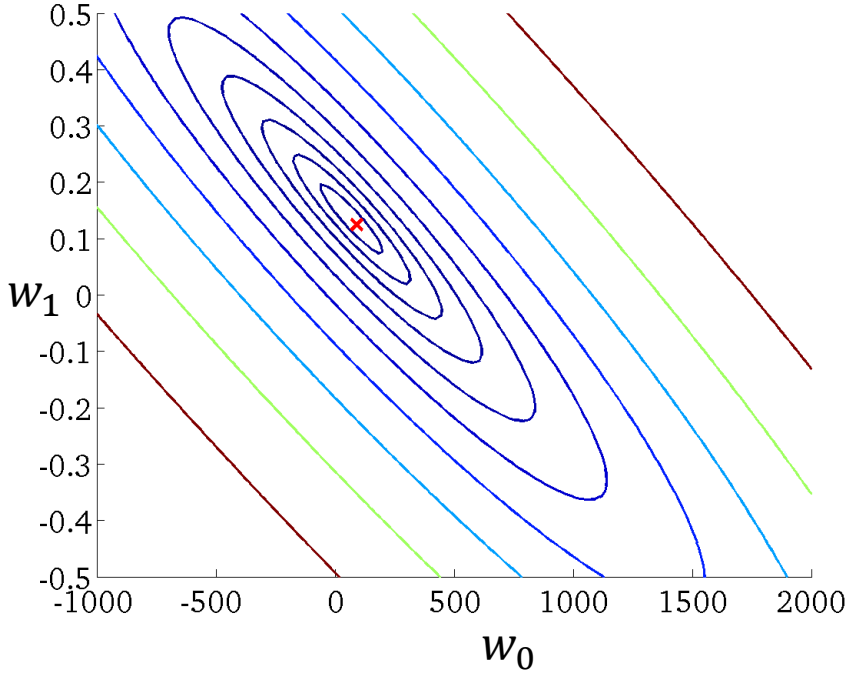


Cost Function

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)





Gradient Descent

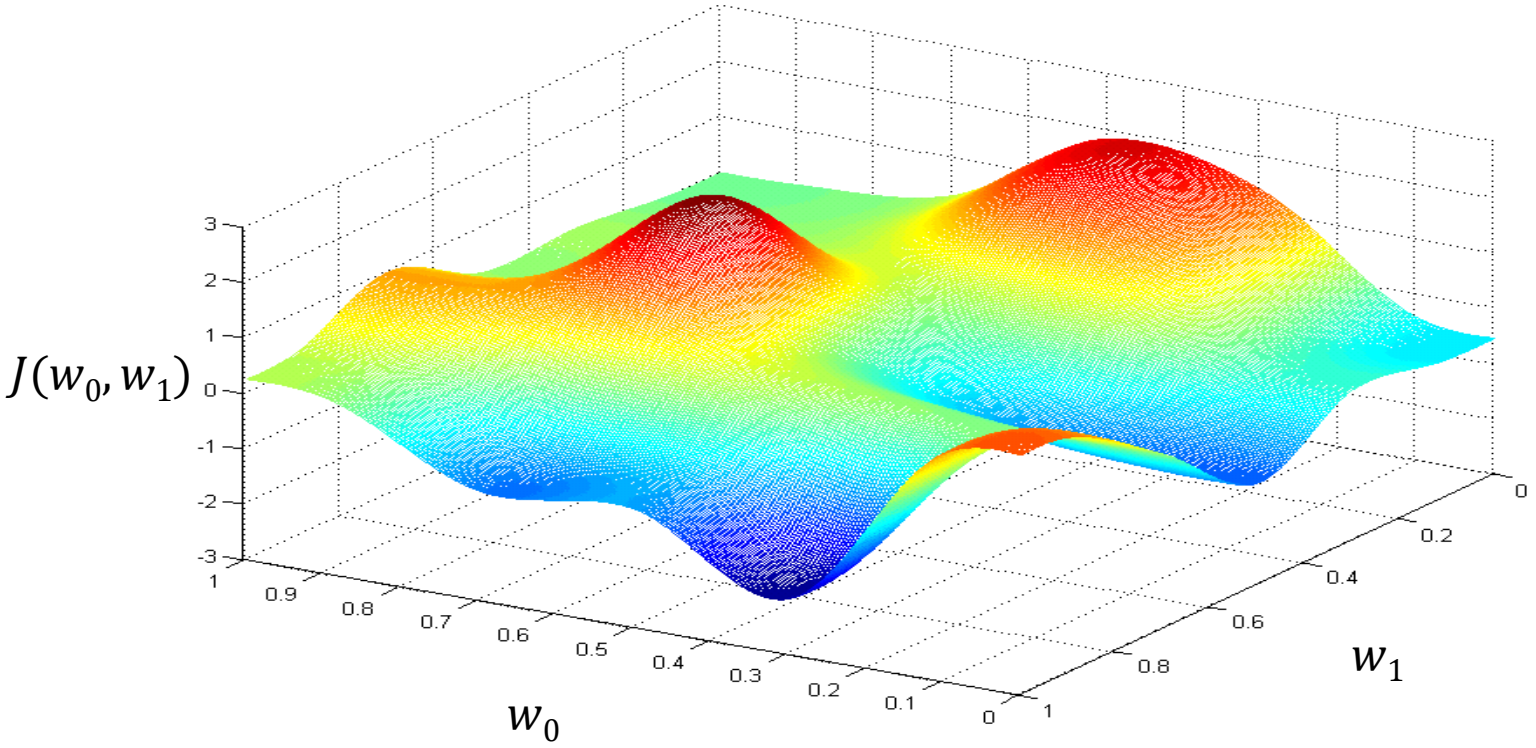
Why gradient descent?

- Standard ML paradigm: Define loss, then optimize

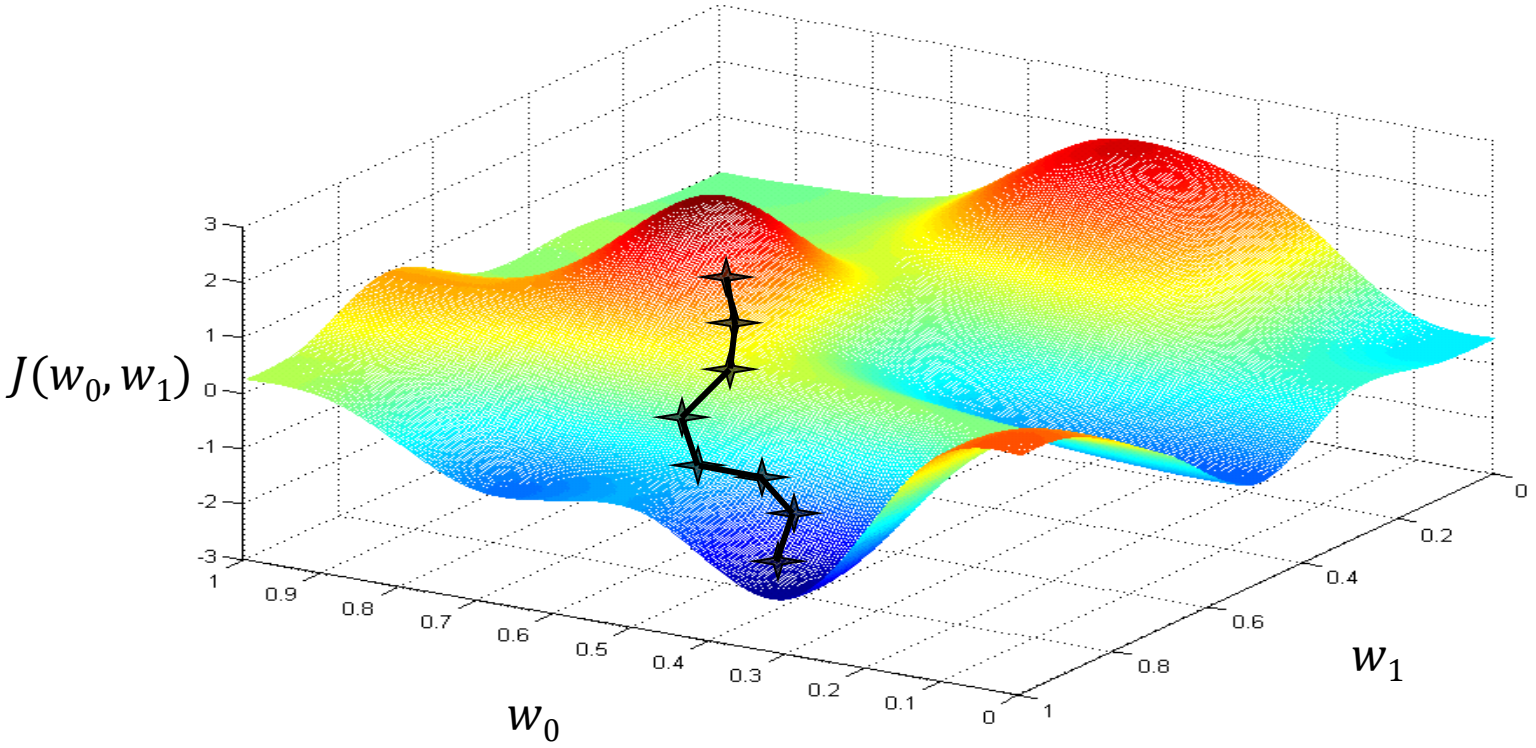
$$\widehat{w}_{LS} = \arg\min_{w} ||\mathbf{y} - \mathbf{X}w||_2^2$$

- But, no closed-form solutions for most losses we use in practice.
- Key idea: Iterative methods
- Used everywhere!

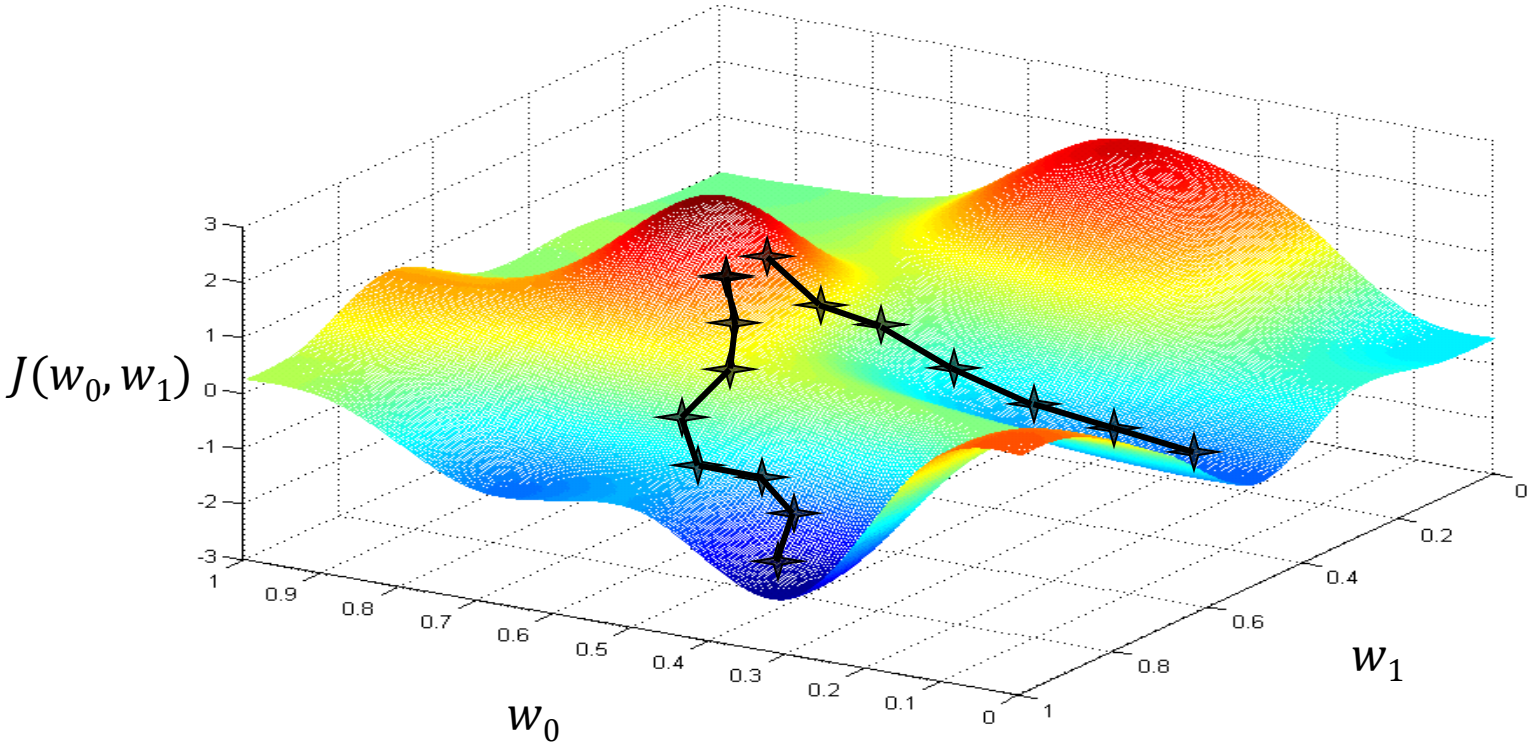
Gradient Descent



Gradient Descent



Gradient Descent



Gradient Descent Algorithm

repeat until convergence {

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

}

Correct: Simultaneous update

$$\text{temp0} \leftarrow w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$\text{temp1} \leftarrow w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$

$$w_0 \leftarrow \text{temp0}$$

$$w_1 \leftarrow \text{temp1}$$

Gradient Descent Algorithm

repeat until convergence {

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

}

Correct: Simultaneous update

$$\text{temp0} \leftarrow w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$\text{temp1} \leftarrow w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$

$$w_0 \leftarrow \text{temp0}$$

$$w_1 \leftarrow \text{temp1}$$

Incorrect:

$$\text{temp0} \leftarrow w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$w_0 \leftarrow \text{temp0}$$

$$\text{temp1} \leftarrow w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$

$$w_1 \leftarrow \text{temp1}$$

Gradient Descent Algorithm

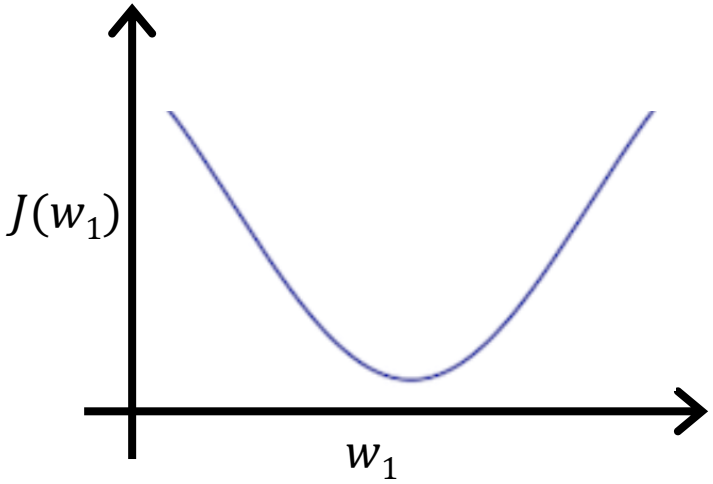
repeat until convergence {

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1)$$

}

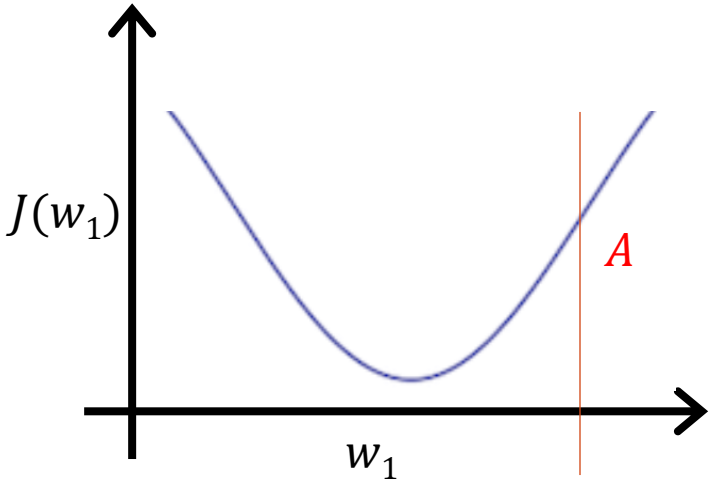
(simultaneously update $j = 0$ and $j = 1$)

Gradient Descent Algorithm: One Parameter



$$w_1 \leftarrow w_1 - \alpha \frac{\partial}{\partial w_1} J(w_1)$$

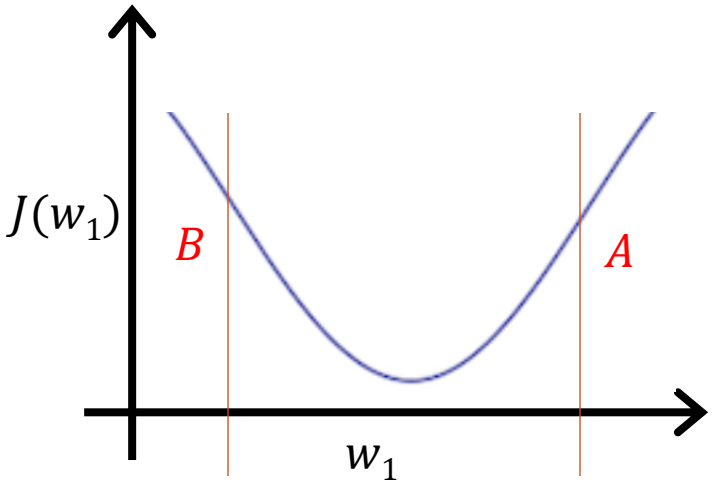
Gradient Descent Algorithm: One Parameter



$$w_1 \leftarrow w_1 - \alpha \frac{\partial}{\partial w_1} J(w_1)$$

A: $w_1 \leftarrow w_1 - \alpha$. (positive number)

Gradient Descent Algorithm: One Parameter



$$w_1 \leftarrow w_1 - \alpha \frac{\partial}{\partial w_1} J(w_1)$$

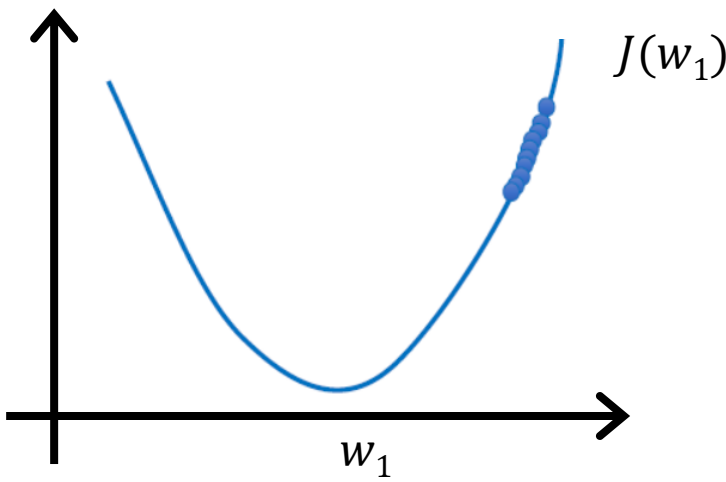
A: $w_1 \leftarrow w_1 - \alpha.$ (positive number)

B: $w_1 \leftarrow w_1 - \alpha.$ (negative number)

Gradient Descent Algorithm

$$w_1 \leftarrow w_1 - \alpha \frac{\partial}{\partial w_1} J(w_1)$$

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

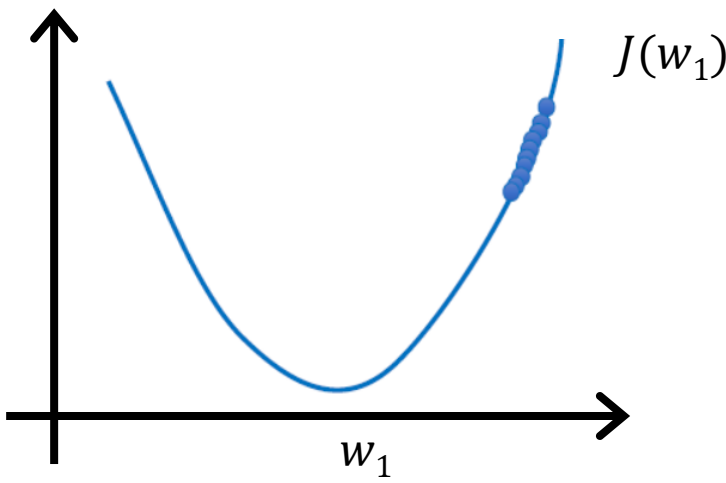


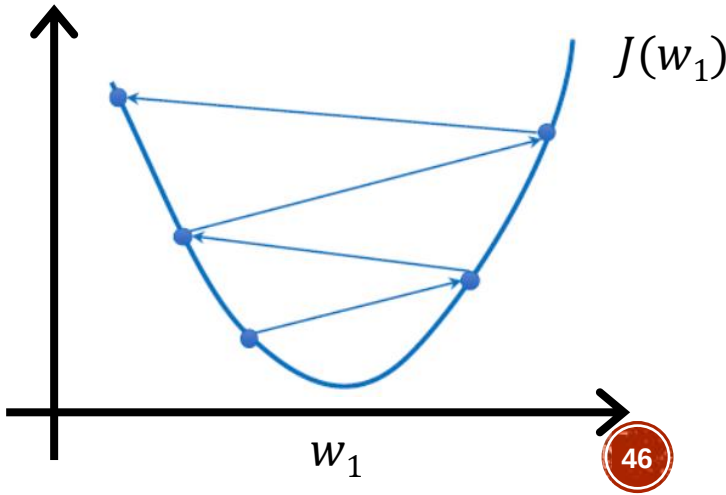
Gradient Descent Algorithm

$$w_1 \leftarrow w_1 - \alpha \frac{\partial}{\partial w_1} J(w_1)$$

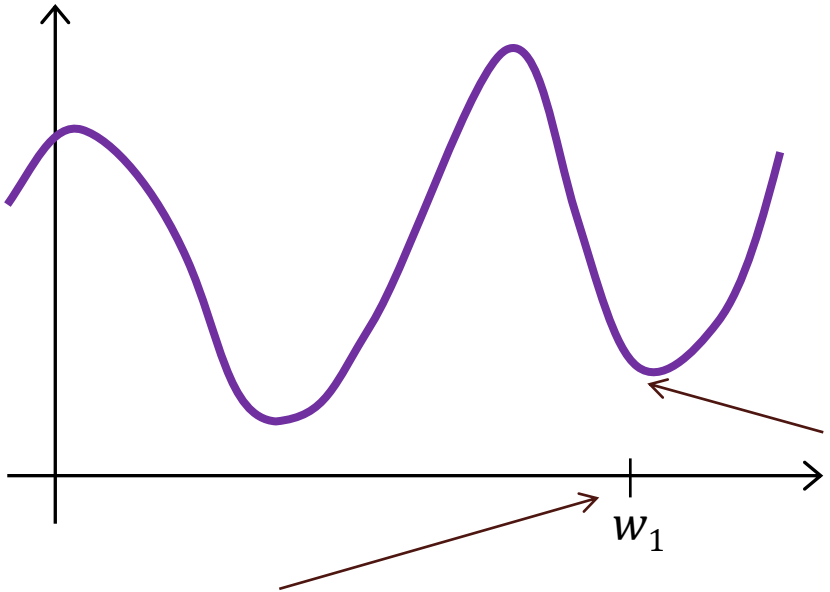
If α is too small, gradient descent can be slow.

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





Gradient Descent Algorithm



$$w_1 \leftarrow w_1 - \alpha \frac{\partial}{\partial w_1} J(w_1)$$

Gradient Descent for Linear Regression

Gradient descent algorithm

repeat until convergence {

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1)$$

}

(for $j = 0$ and $j = 1$)

Linear Regression Model

$$h_w(x) = w_0 + w_1 x$$

$$J(w_0, w_1) = \frac{1}{2n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$

Gradient Descent for Linear Regression

Gradient descent algorithm

repeat until convergence {

$$w_0 \leftarrow w_0 - \alpha \frac{1}{n}\sum_{i=1}^{n}(h_w(x^i) - y^i)$$

$$w_1 \leftarrow w_1 - \alpha \frac{1}{n}\sum_{i=1}^{n}(h_w(x^i) - y^i).x^i$$

}

Gradient Descent for Linear Regression

Gradient descent algorithm

repeat until convergence {

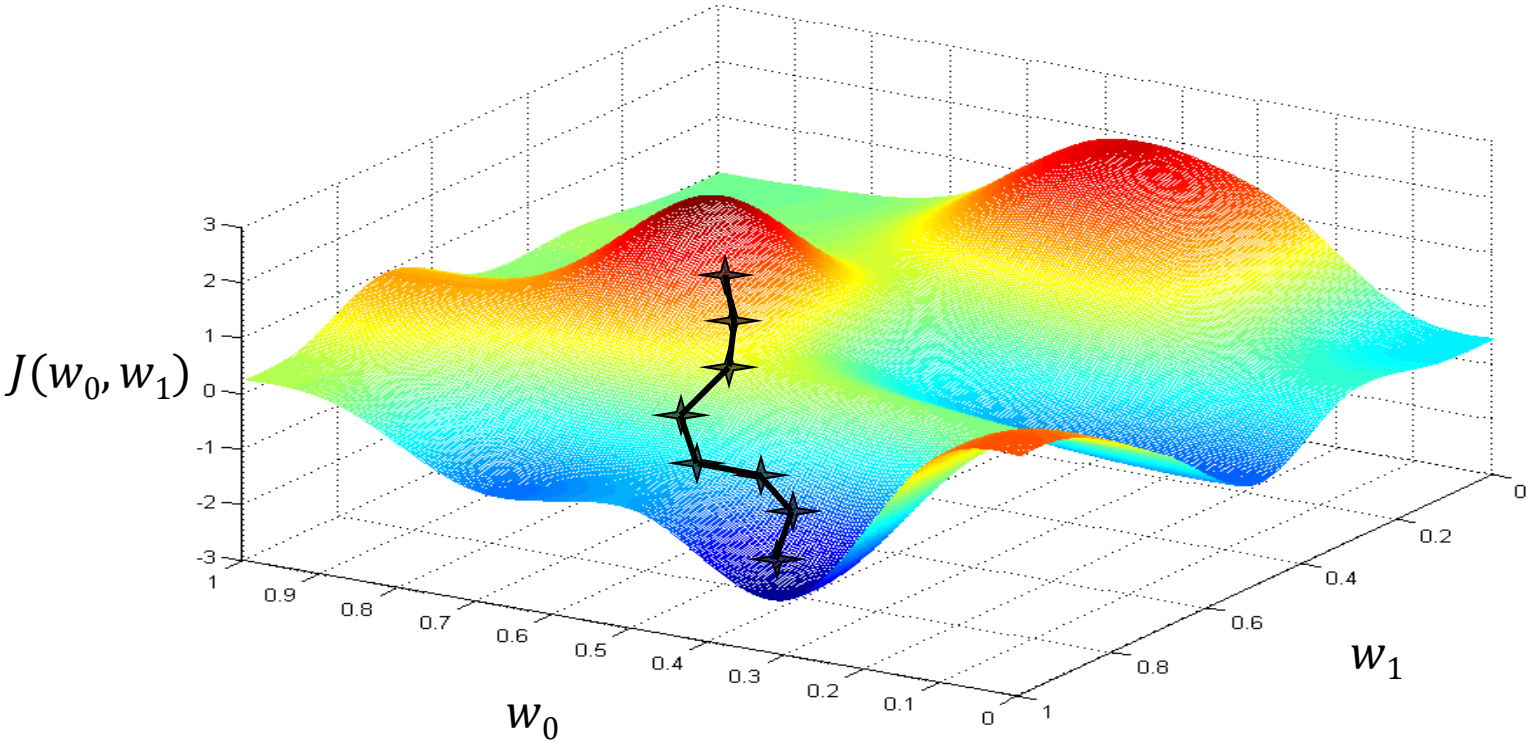
$$w_0 \leftarrow w_0 - \alpha \frac{1}{n}\sum_{i=1}^{n}\left(h_w\left(x^i\right) - y^i\right)$$

$$w_1 \leftarrow w_1 - \alpha \frac{1}{n}\sum_{i=1}^{n}\left(h_w\left(x^i\right) - y^i\right).x^i$$

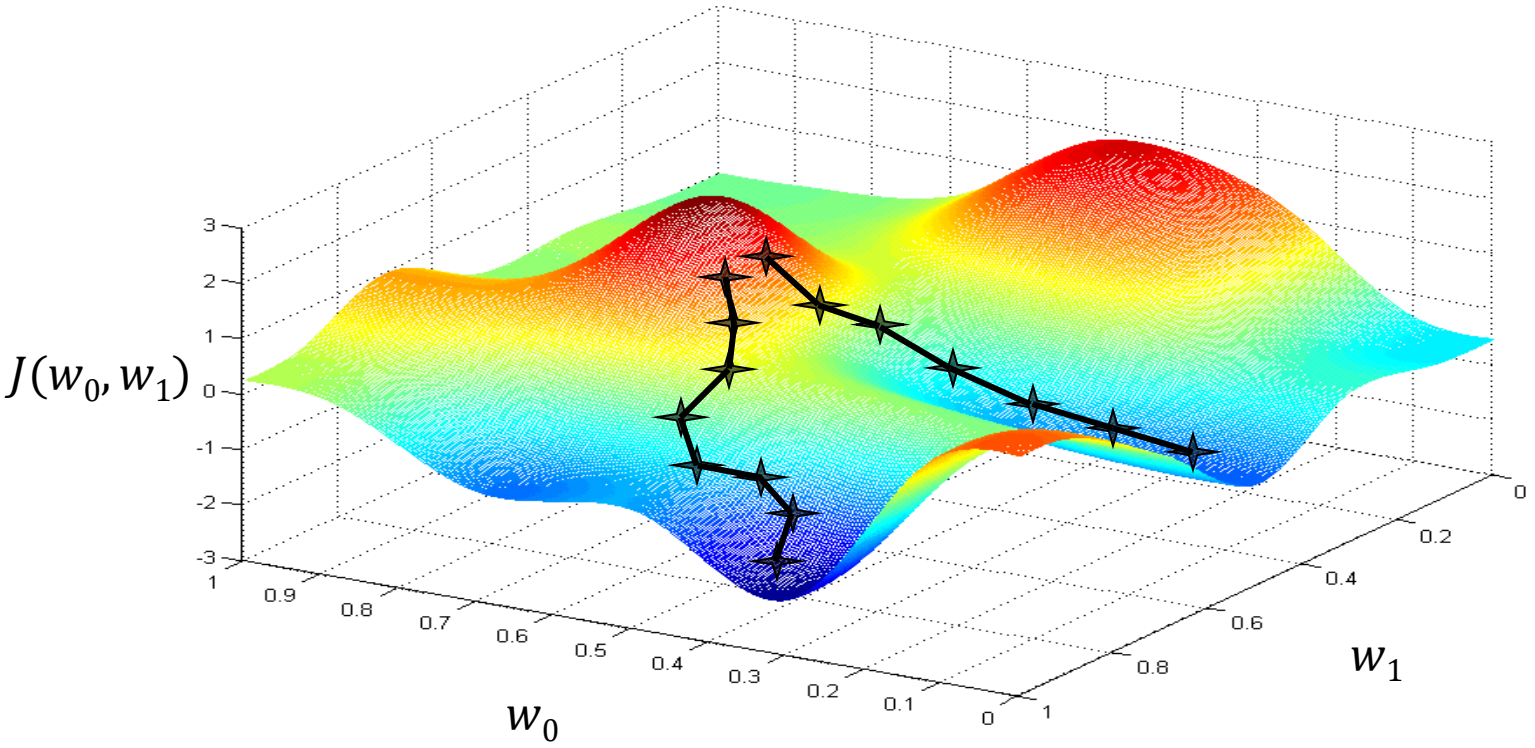
}

Update w_0 and w_1 simultaneously

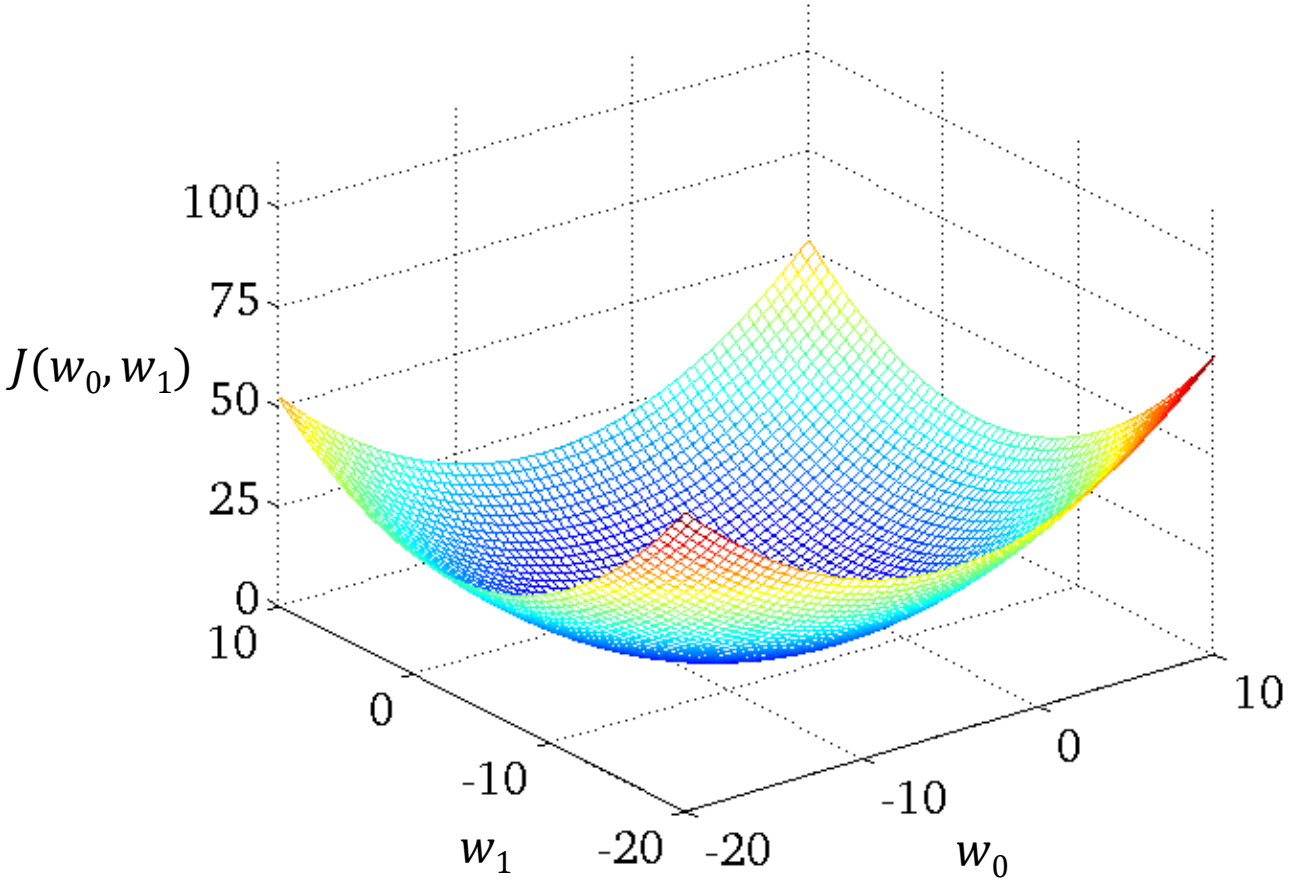
Gradient Descent for Linear Regression



Gradient Descent for Linear Regression



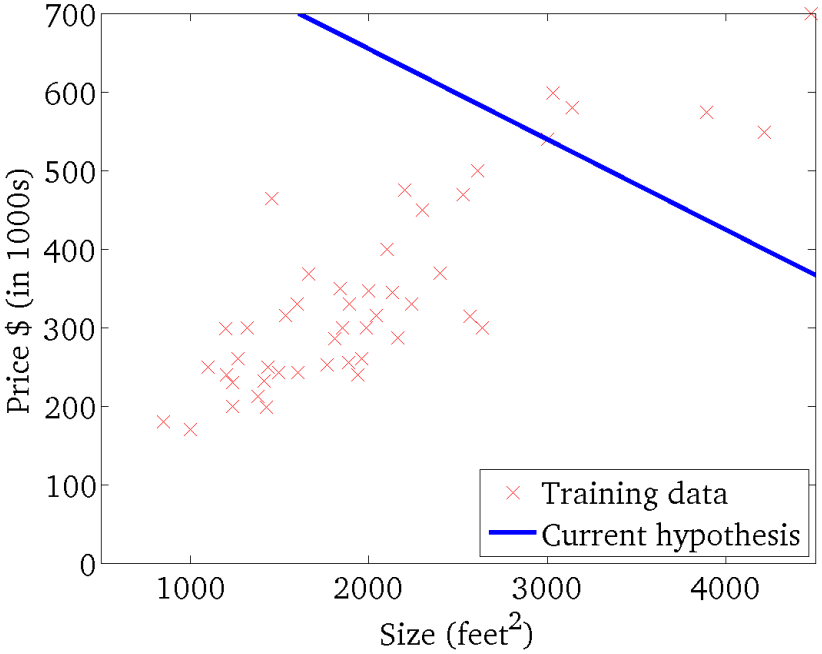
Gradient Descent for Linear Regression

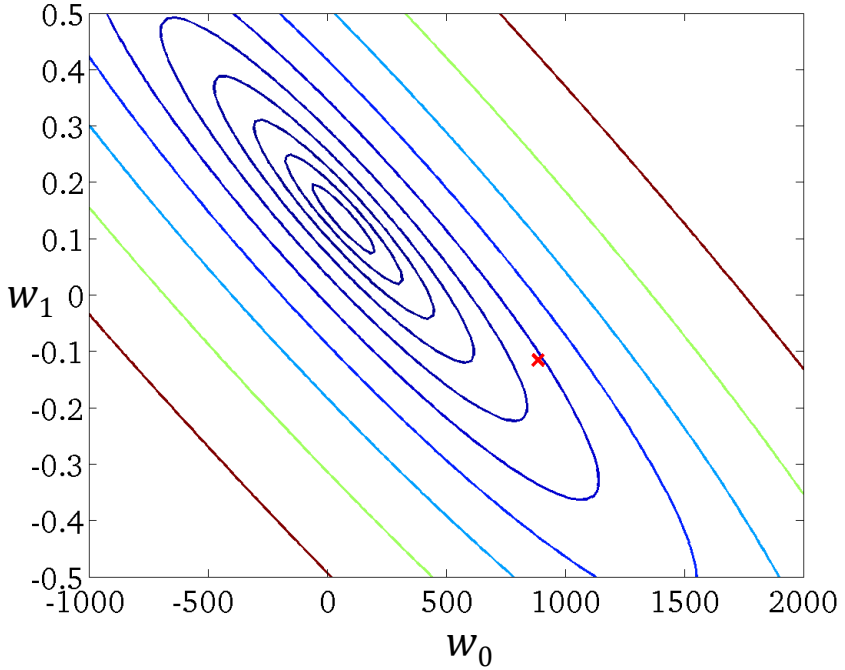


Gradient Descent for Linear Regression

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)

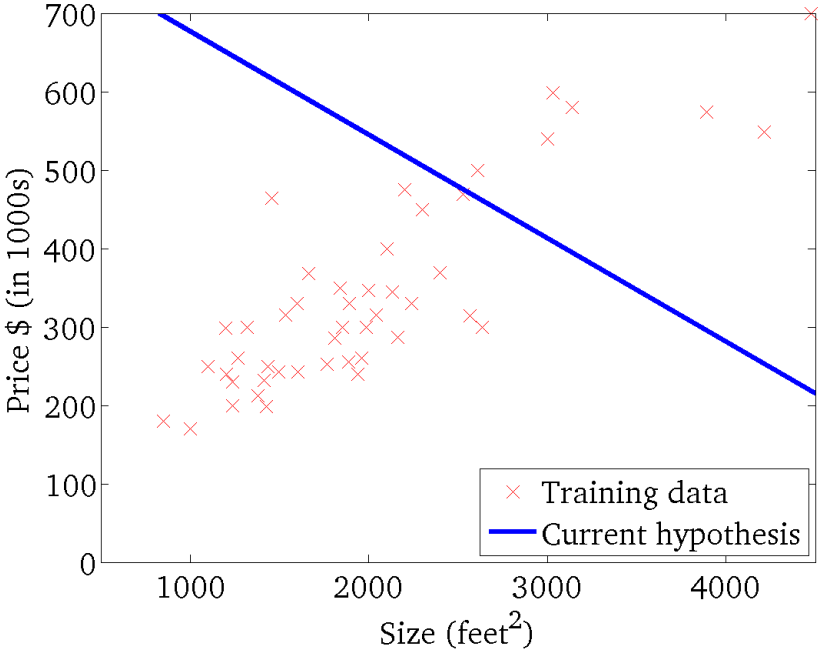


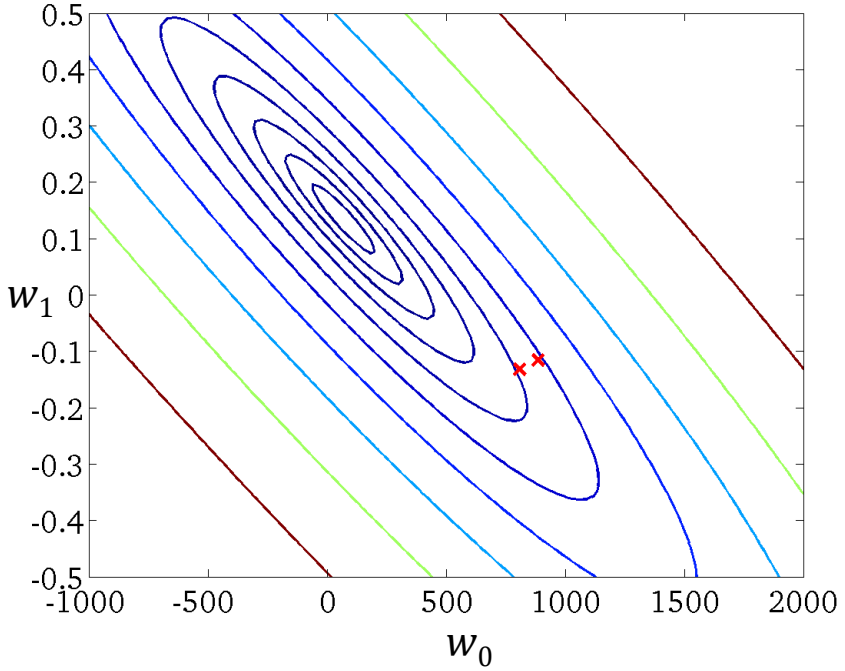


Gradient Descent for Linear Regression

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)

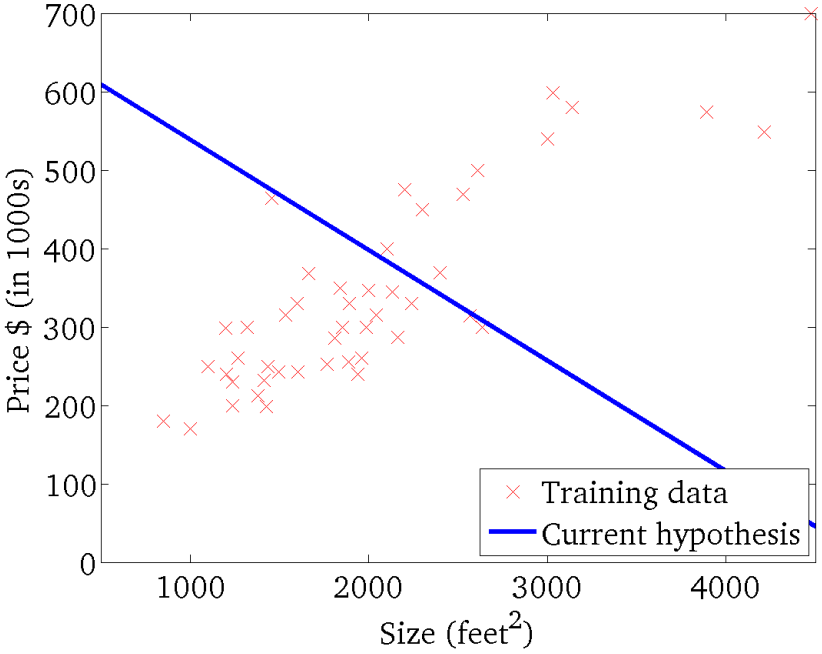


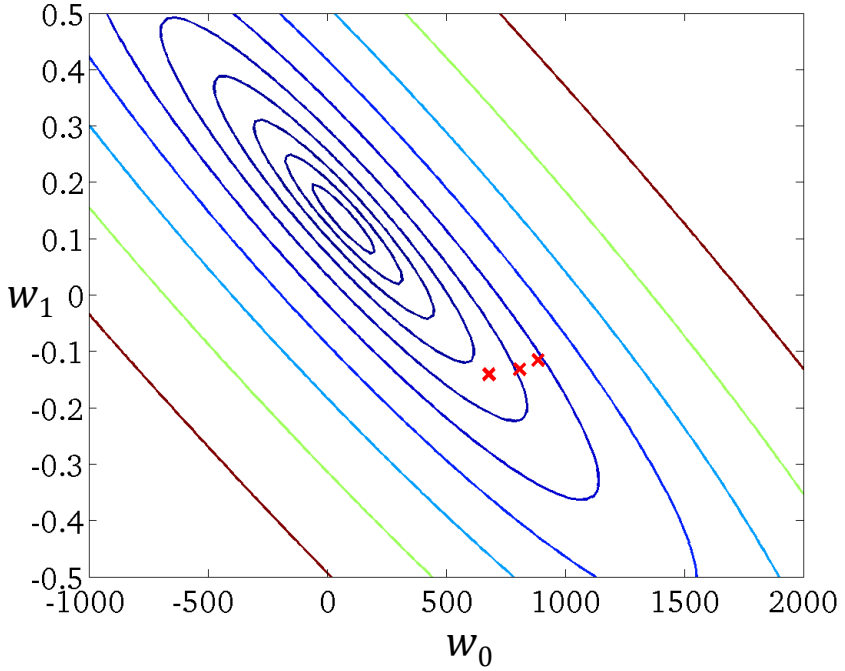


Gradient Descent for Linear Regression

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)

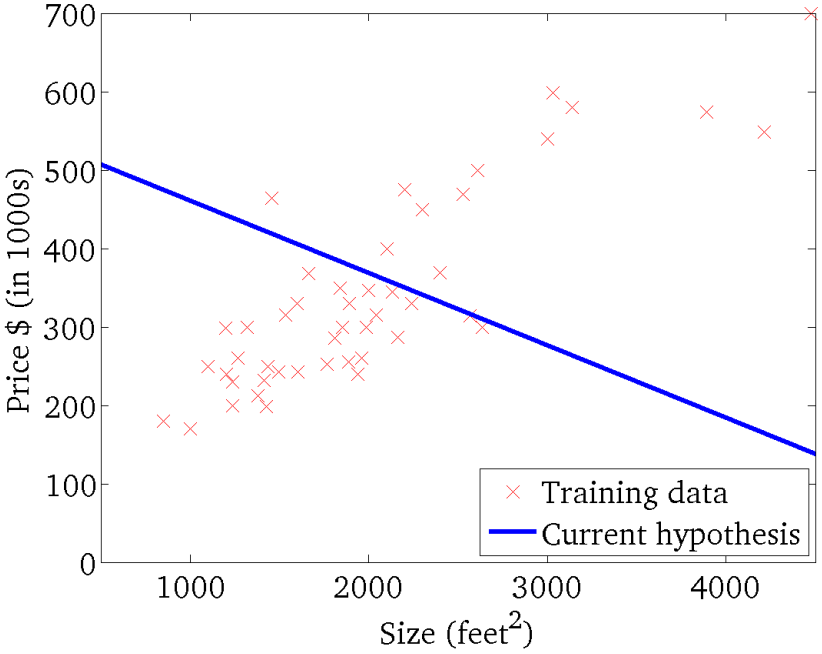


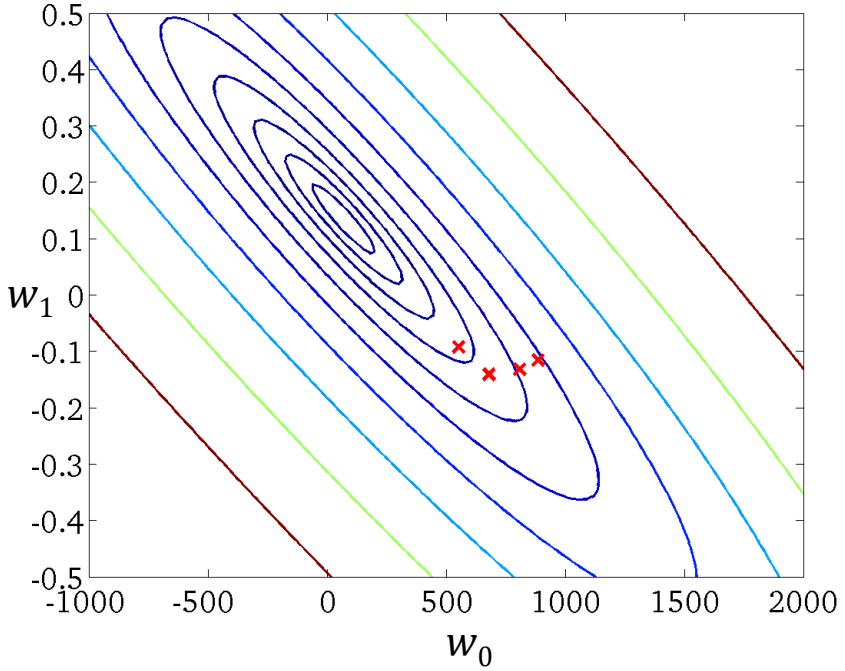


Gradient Descent for Linear Regression

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)

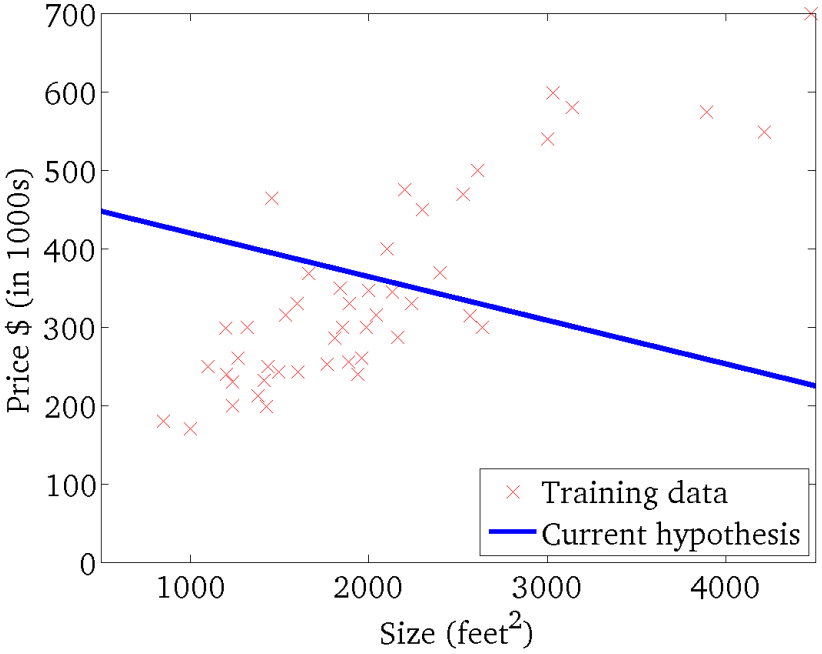


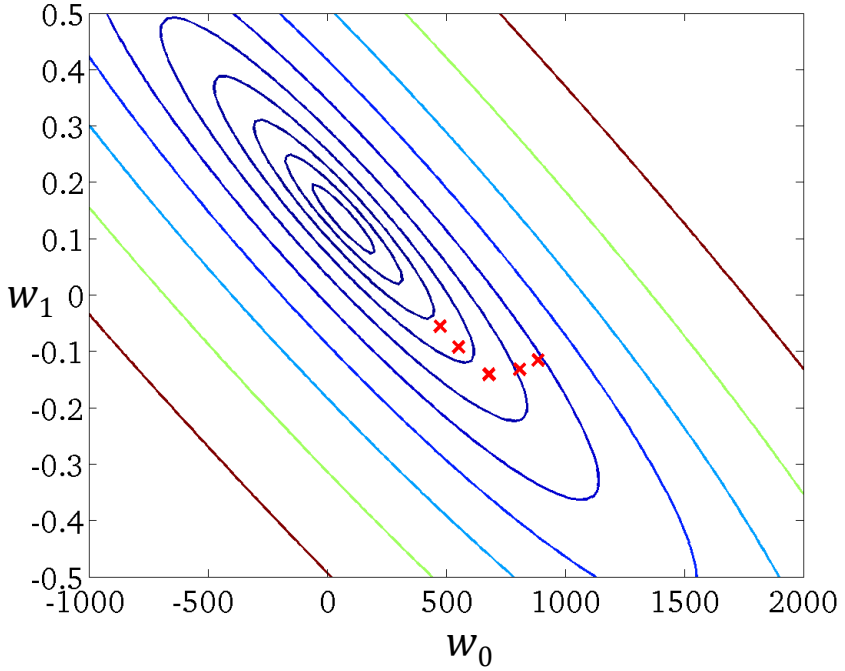


Gradient Descent for Linear Regression

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)

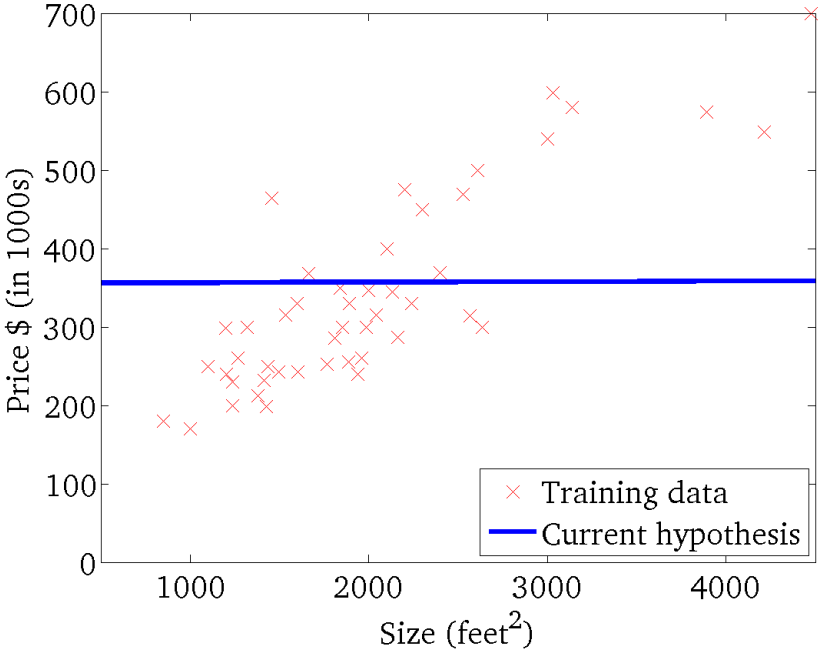


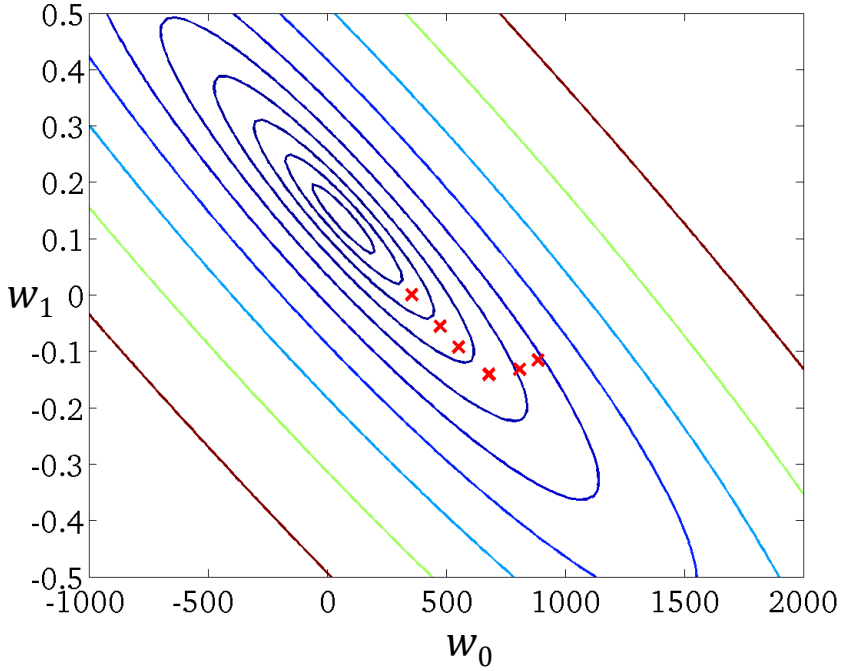


Gradient Descent for Linear Regression

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)

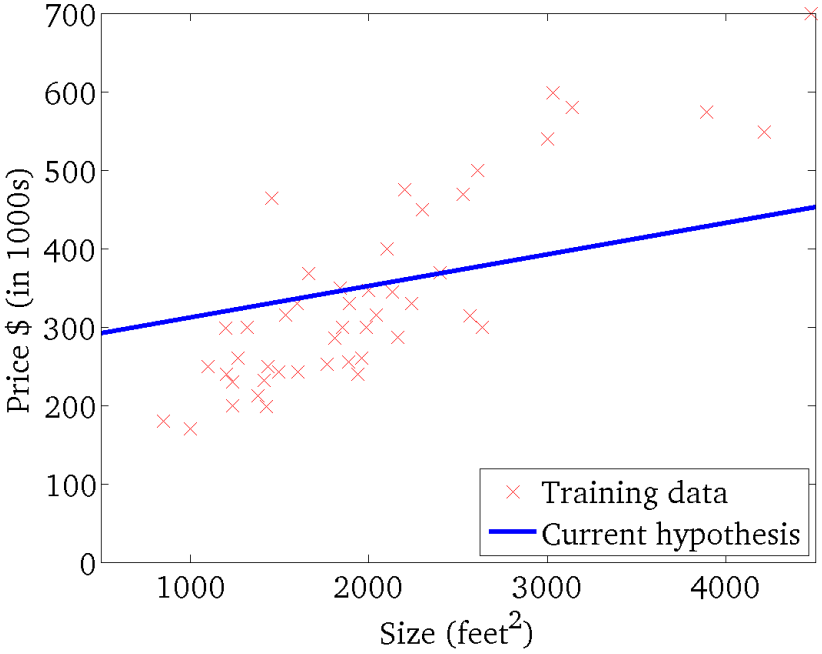


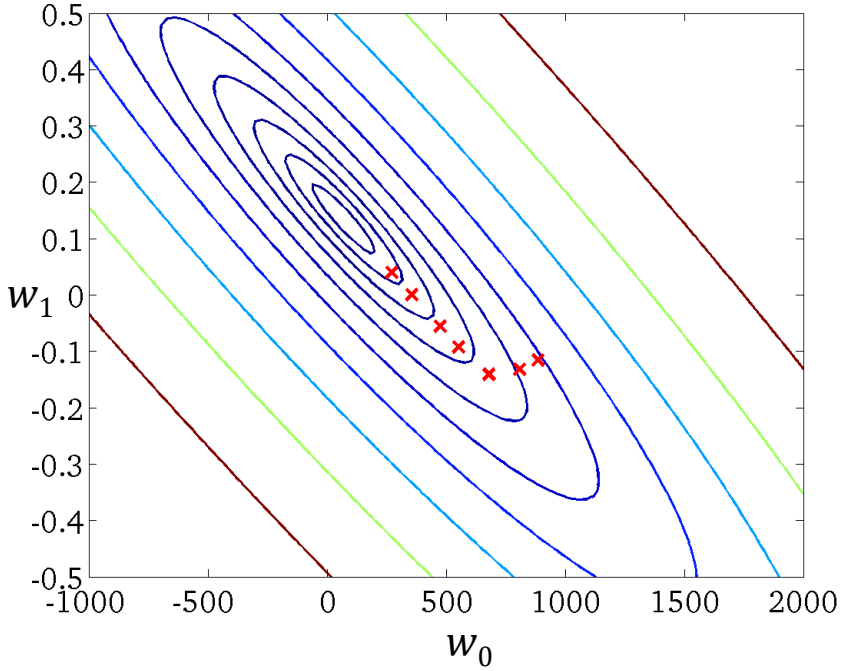


Gradient Descent for Linear Regression

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)

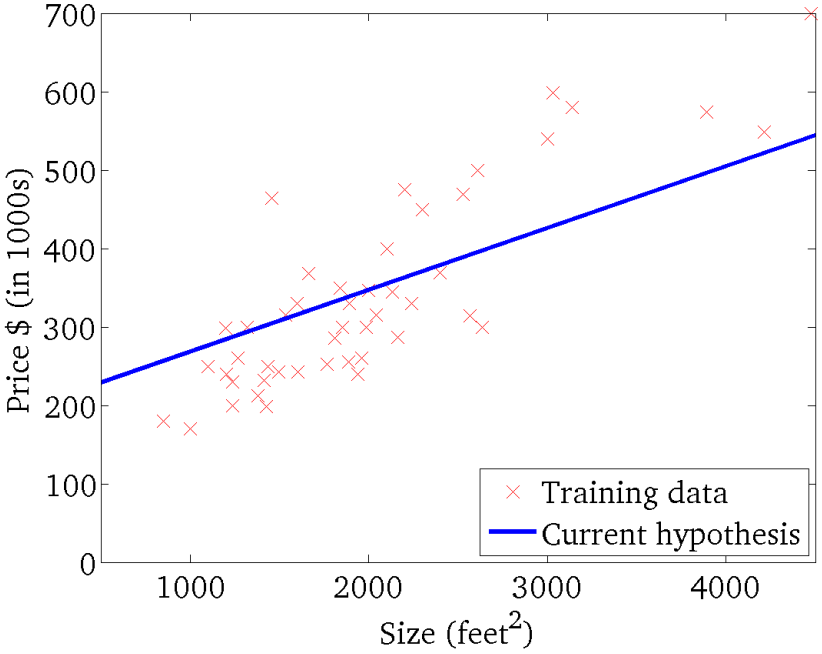


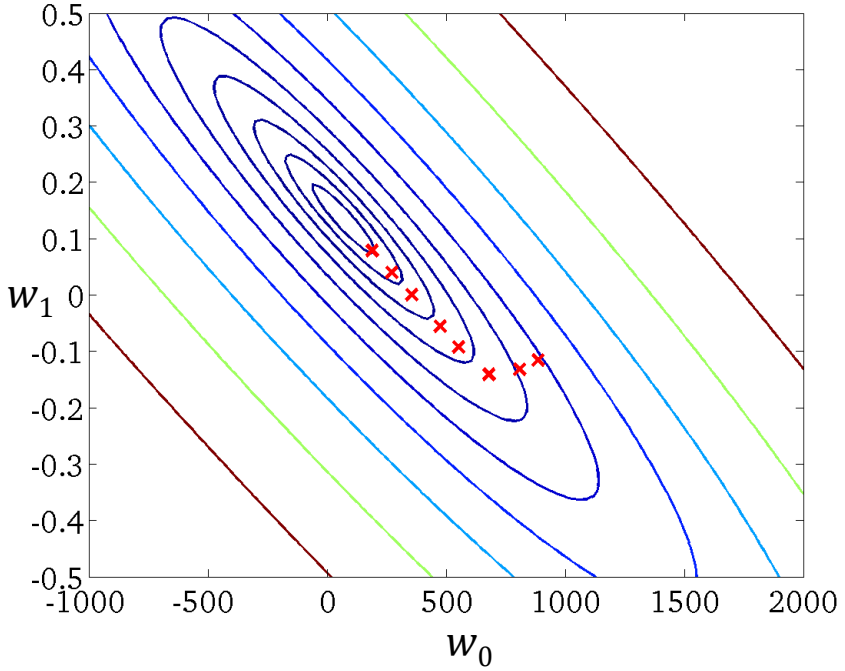


Gradient Descent for Linear Regression

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)

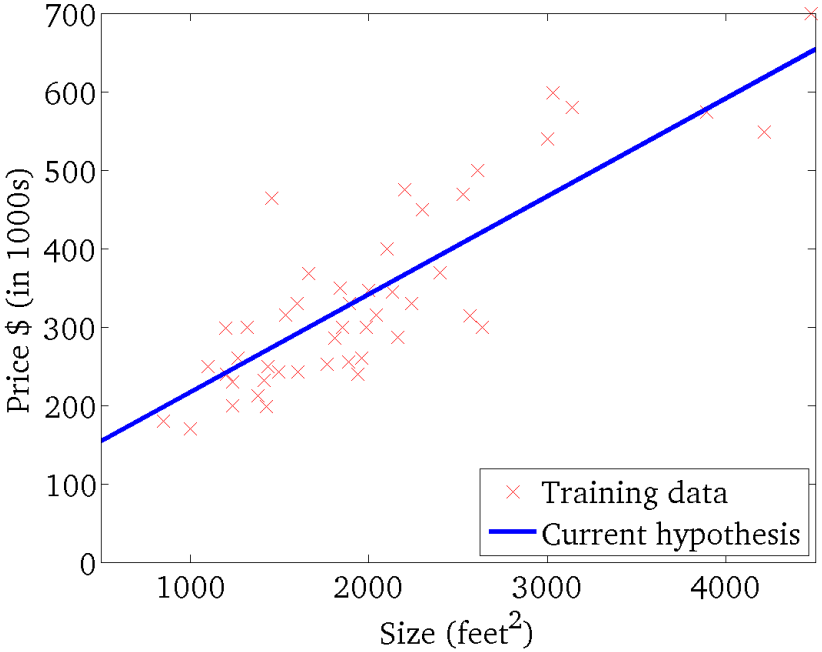


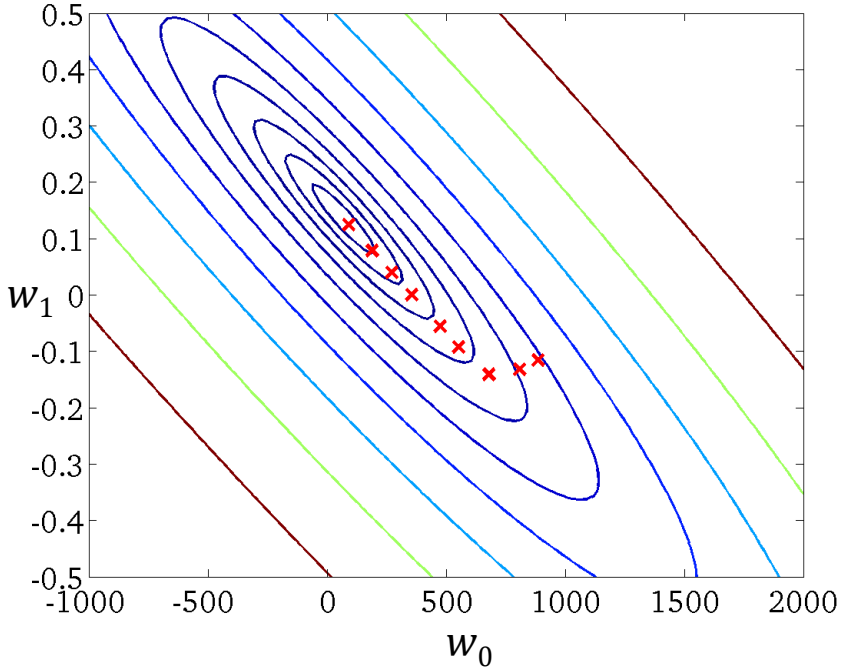


Gradient Descent for Linear Regression

$h_w(x)$ (for fixed w_0, w_1, this is a function of x)

$J(w_0, w_1)$ (function of the parameters w_0, w_1)





“Batch” Gradient Descent

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

“Mini-Batch” Gradient Descent

“Mini-Batch”: Each step of gradient descent uses few of the training examples as per batch size randomly drawn.

Linear Regression with Multiple Features (Variables)

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

Notation:

d = number of features

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

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

Linear Regression with Multiple Features (Variables)

Hypothesis:

$$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$$

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

$$\begin{aligned} h_w(x) &= w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d \\ &= W^T X \end{aligned}$$

Multivariate linear regression.

Gradient Descent for Multiple Variables

Hypothesis: $h_w(x) = W^T X = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$

Parameters: $w_0, w_1, w_2, \ldots, w_d$

Cost function:

$$J(w_0, w_1, \ldots, w_d) = \frac{1}{2n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$

Gradient descent:

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1, \ldots, w_d)$$

}

(simultaneously update for every $j = 0, \ldots, d$)

Gradient Descent for Multiple Variables

Hypothesis: $h_w(x) = W^T X = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$

Parameters: $w_0, w_1, w_2, \ldots, w_d$

Cost function:

$$J(w_0, w_1, \ldots, w_d) = \frac{1}{2n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$

Gradient descent:

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1, \ldots, w_d) \qquad w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} J(W)$$

}

(simultaneously update for every $j = 0, \ldots, d$)

Gradient Descent for Multiple Variables

Repeat $\{$

$$w_j \leftarrow w_j - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_w(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$\}$ (simultaneously update w_j for $j = 0, \ldots, d$)

$$w_0 \leftarrow w_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_w(x^{(i)}) - y^{(i)}) x_0^{(i)} \qquad x_0^{(i)} = 1$$

$$w_1 \leftarrow w_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_w(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$w_2 \leftarrow w_2 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h_w(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

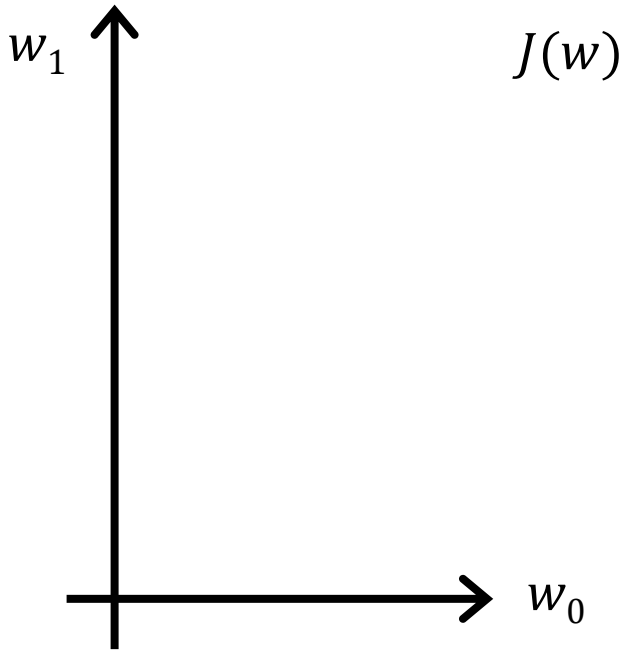
$\ldots$

Feature Scaling

Idea: Make sure features are on a similar scale.

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

x_2 = number of bedrooms (1-5)

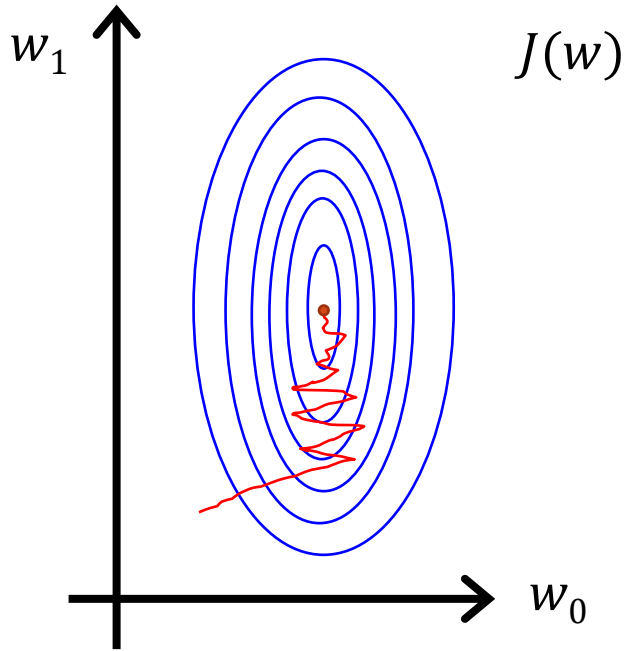


Feature Scaling

Idea: Make sure features are on a similar scale.

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

x_2 = number of bedrooms (1-5)

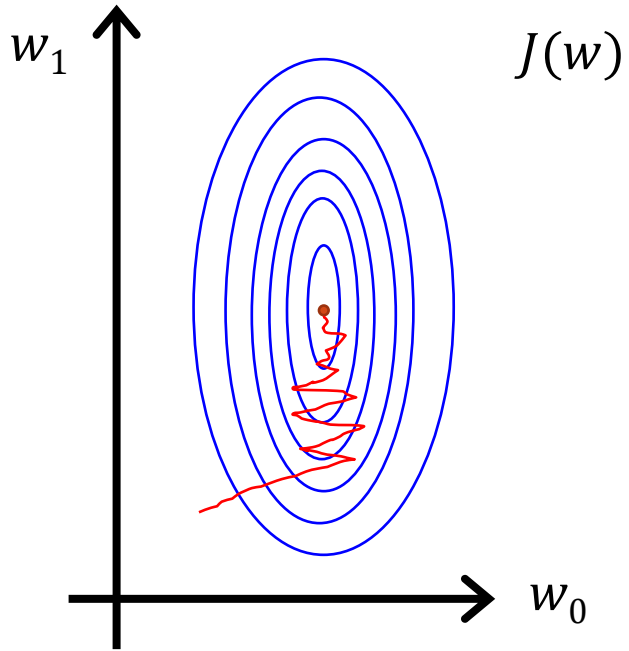


Feature Scaling

Idea: Make sure features are on a similar scale.

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

x_2 = number of bedrooms (1-5)



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

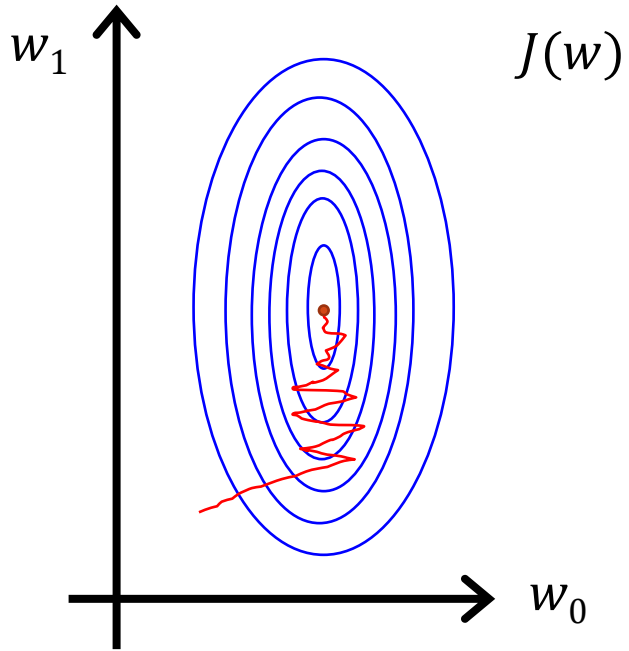
$$x_2 = \frac{\text{number of bedrooms}}{5}$$

Feature Scaling

Idea: Make sure features are on a similar scale.

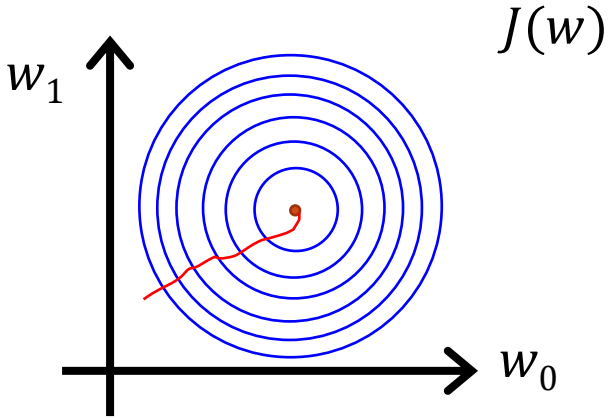
E.g. x_1 = size (0-2000 feet^2)

x_2 = number of bedrooms (1-5)



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

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



Feature Scaling

Get every feature into approximately a $-1 \leq x_i \leq 1$ range.

Good Range	Bad Range
$x_0 = 1$	$-100 \leq x_i \leq 100$
$0 \leq x_1 \leq 3$	$-0.0000001 \leq x_i \leq 0.00000001$
$-2 \leq x_2 \leq 0.5$	
$-3 \leq x_i \leq 3$	
$-1/3 \leq x_i \leq 1/3$	

Mean Normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).

E.g. $x_1 = \frac{size - 1000}{2000}$

$x_2 = \frac{\#bedrooms - 2}{5}$

$-0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5$

$$x_i \leftarrow \frac{x_i - \mu_i}{s_i}$$

Average value of x_i in training set

Range (max - min) or standard deviation

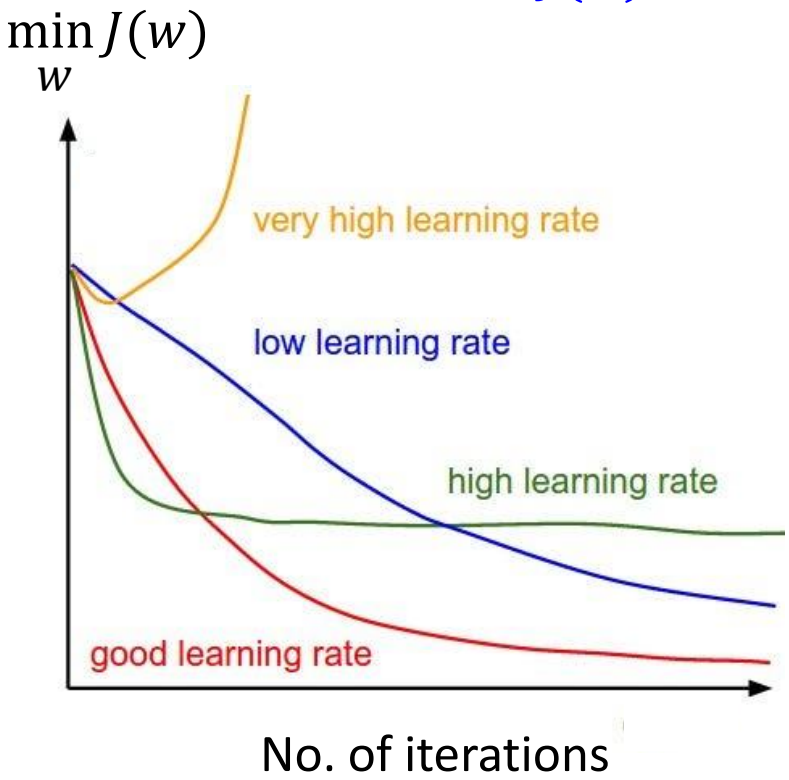
Gradient Descent: Learning Rate

$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} J(W)$$

- "Debugging": How to make sure gradient descent is working correctly.
- How to choose learning rate α.

Gradient Descent: Learning Rate

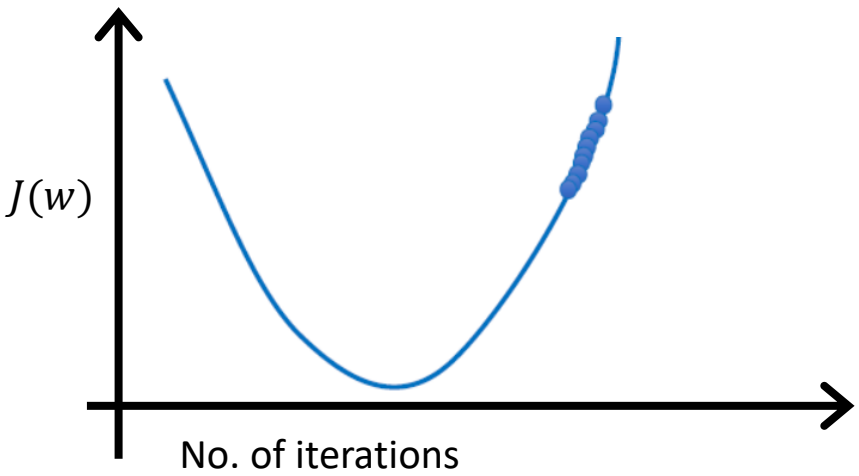
$J(w)$ should decrease after every iteration

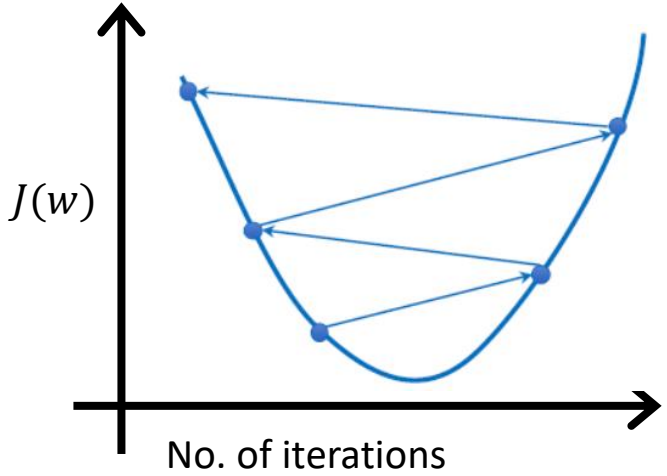


Example automatic convergence test:

Declare convergence if $J(w)$ decreases by less than 10^{-3} in one iteration.

Gradient Descent: Learning Rate





- For sufficiently small α, $J(w)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Gradient Descent: Learning Rate

Summary:

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

To choose α, try

$$\ldots, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \ldots.$$

Gradient Descent vs Normal Equation

n training examples, d features.

Gradient Descent

- Need to choose α.
- Needs many iterations.
- Works well even when d is large.

Normal Equation

- No need to choose α.
- Don't need to iterate.
- Need to compute $(X^T X)^{-1}$
- Slow if d is very large.

Normal Equation

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

- What if $X^T X$ is non-invertible? (singular/ degenerate)

- Redundant features (linearly dependent).

  E.g. size in feet^2

  size in m^2

- Too many features (e.g. $N \leq d$).

  - Delete some features, or use regularization.

Acknowledgement

Most of the slides are based on the ML course of:

- Andrew Ng, Stanford University

THANK YOU