MACHINE LEARNING COURSE

PRESENTED BY ABDEL RAHMAN ALSABBAGH

LECTURE #3 – SAT - 17.5.2023

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

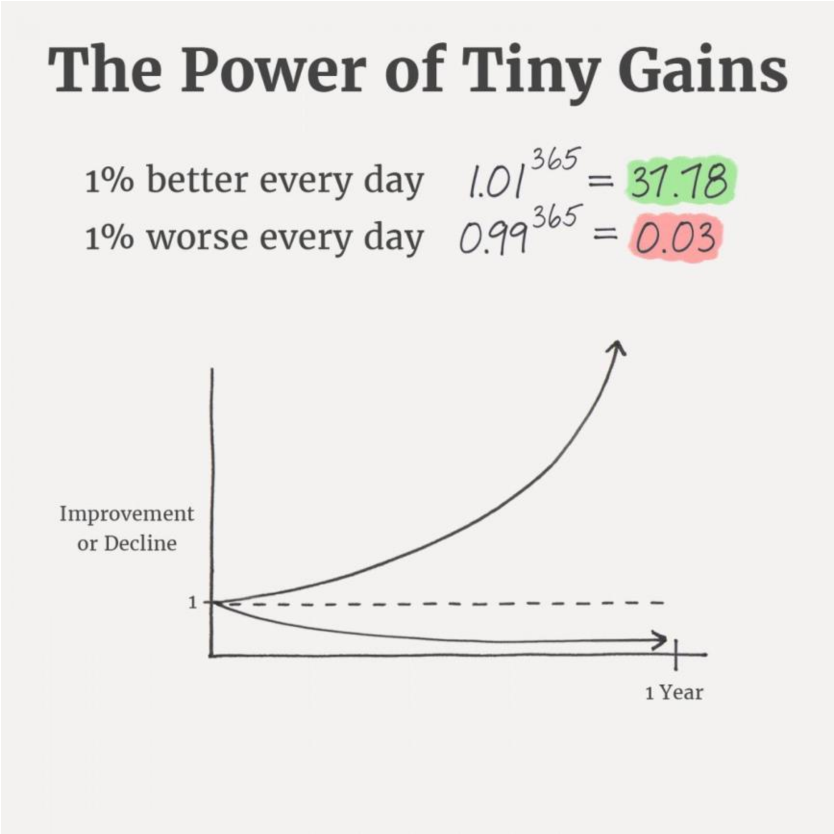
In the name of Allah, the most gracious, the most merciful, we start :)

Today's Quote

"Growth happens without requiring much effort"

- Chris Gardener.

Today's Quote



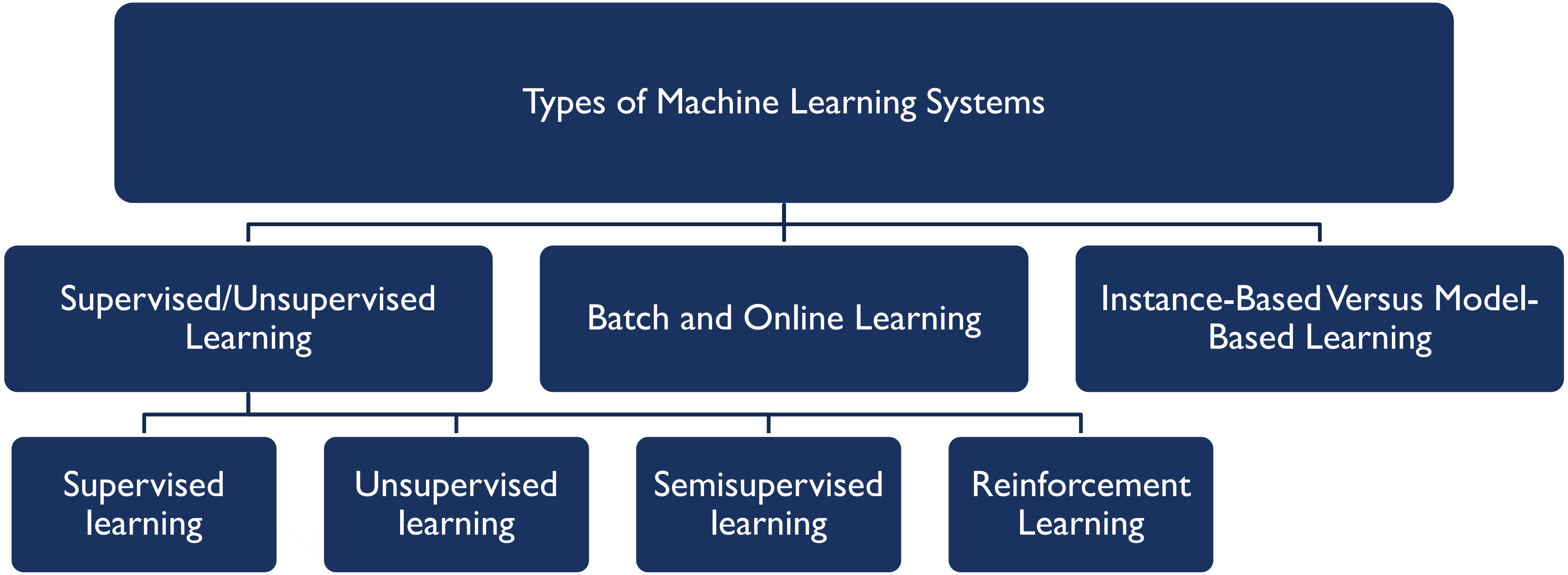
Course Outline

1. Machine Learning Overview
2. The Machine Learning Process
3. Introduction to Classification
4. Machine Learning Model Training
5. Support Vector Machines
6. Decision Trees
7. Ensemble Learning and Random Forests
8. Unsupervised Learning Techniques
9. Artificial Neural Networks

Resources used:

- Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron
- Machine Learning Specialization by Andrew Ng and Stanford Online

Types of Machine Learning Systems



Recap

Main Challenges of Machine Learning

Bad Data

- Insufficient Quantity of Training Data
- Nonrepresentative Training Data
- Poor-Quality Data
- Irrelevant Features

Bad Algorithm

- Overfitting the Training Data
- Underfitting the Training Data

Recap

Testing and Validating

Hyperparameter Tuning and Model Selection

Data Mismatch

The Machine Learning Process

- Linear regression.
- Loss and cost functions.
- Visualizing the cost function.
- Gradient descent.
- Learning rate.
- Gradient descent for linear regression.
- Evaluation metrics.
- Linear regression for multiple features.
- Gradient descent for multiple features.
- Feature scaling.

Source: Machine Learning Specialization by Andrew Ng and Stanford Online.

Linear Regression

Model: $f_{w,b}(x) = wx + b$

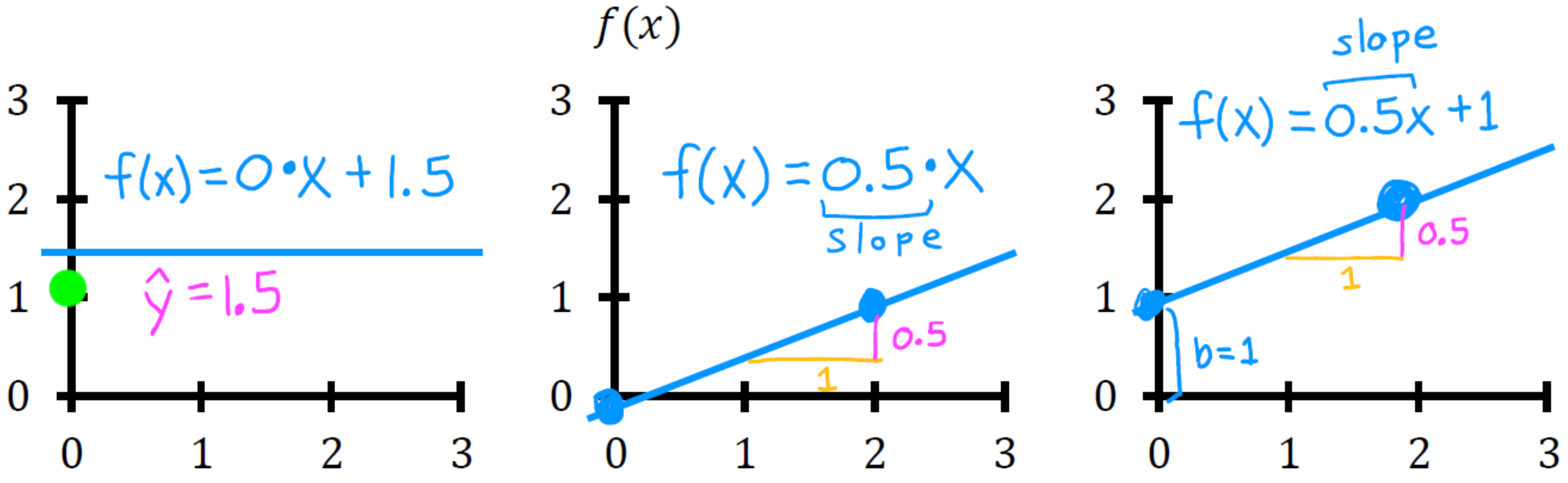
w, b: parameters

coefficients

weights

Linear Regression

$f_{w,b}(x) = wx + b$

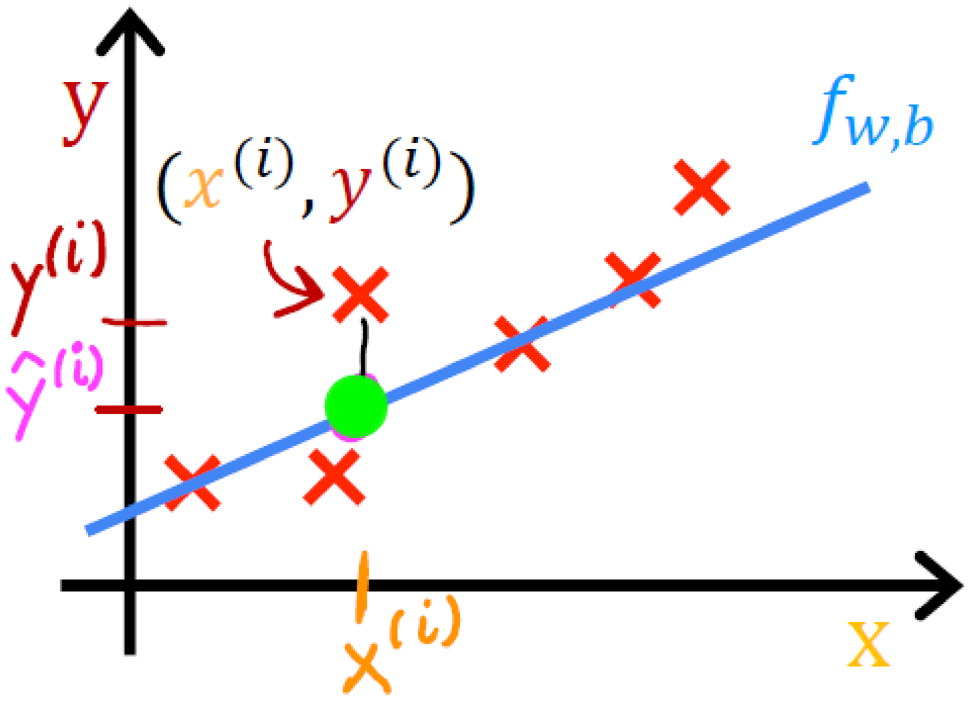


→ $w = 0$
→ $b = 1.5$
y-intercept

→ $w = 0.5$
→ $b = 0$

→ $w = 0.5$
→ $b = 1$

Linear Regression



$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b$$

Loss and Cost Functions

$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b$$

Find w, b:
$\hat{y}^{(i)}$ is close to $y^{(i)}$ for all $(x^{(i)}, y^{(i)})$.

Loss function

$$\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

Cost function

m = number of training examples

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

intuition (next!)

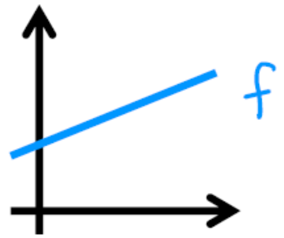
Cost Function

model:

$$f_{w,b}(x) = wx + b$$

parameters:

$$w, b$$



cost function:

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2$$

goal:

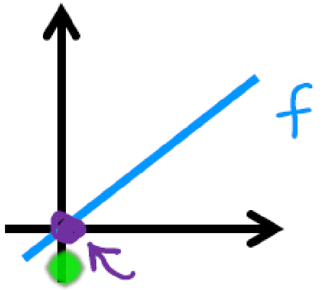
$$\underset{w,b}{\text{minimize}}\, J(w,b)$$

Cost Function

simplified

$f_w(x) = wx$ $\quad b = \emptyset$

w

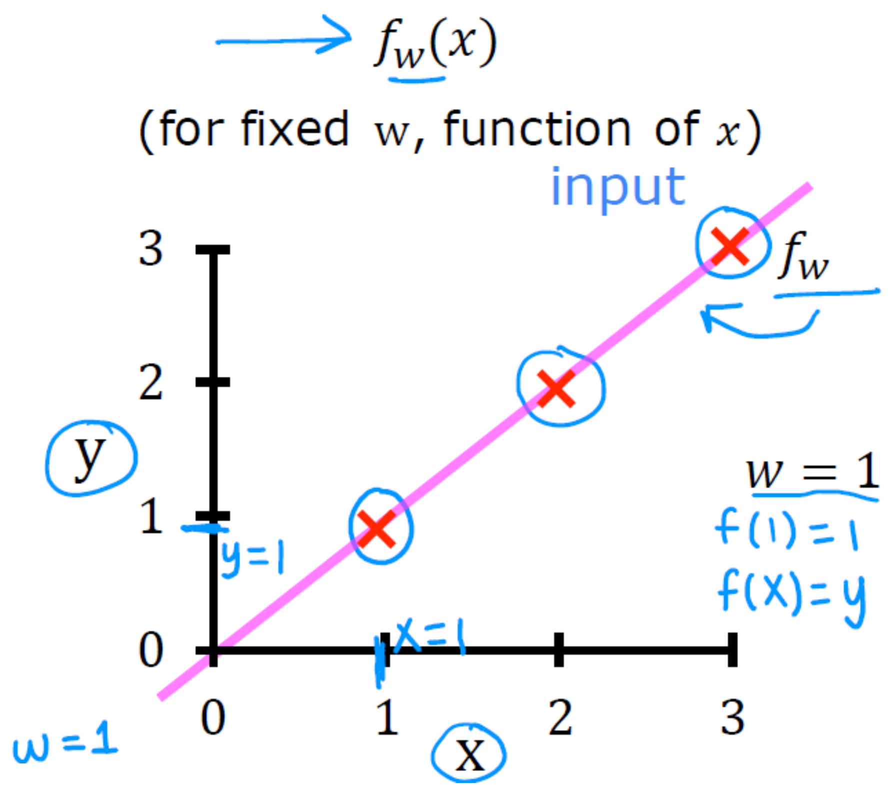


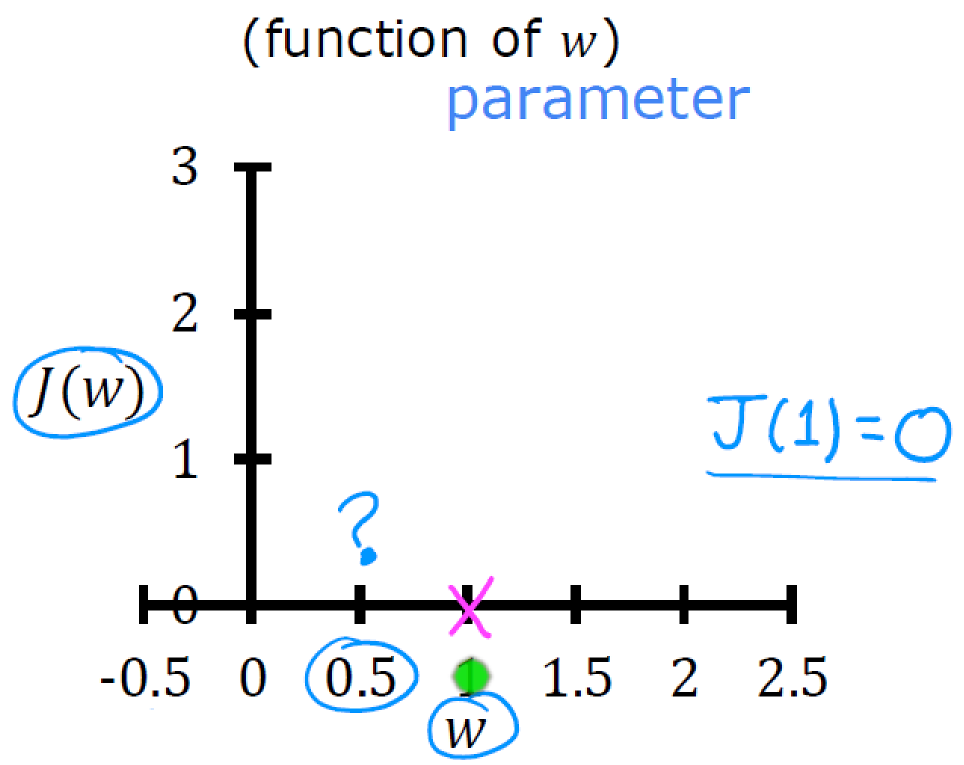
$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_w(x^{(i)}) - y^{(i)}\right)^2$$

$wx^{(i)}$

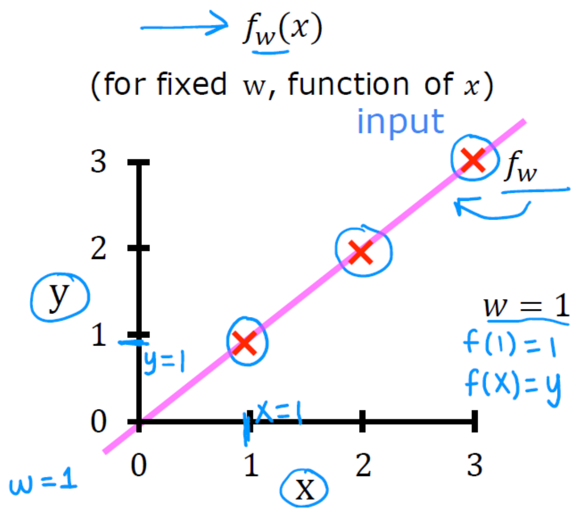
$$\underset{w}{\text{minimize}}\; J(w)$$

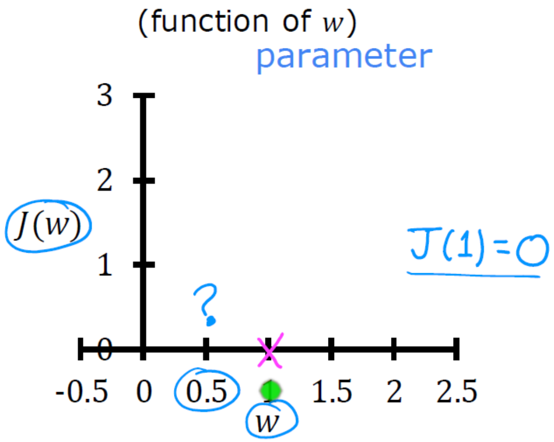
Cost Function





Cost Function





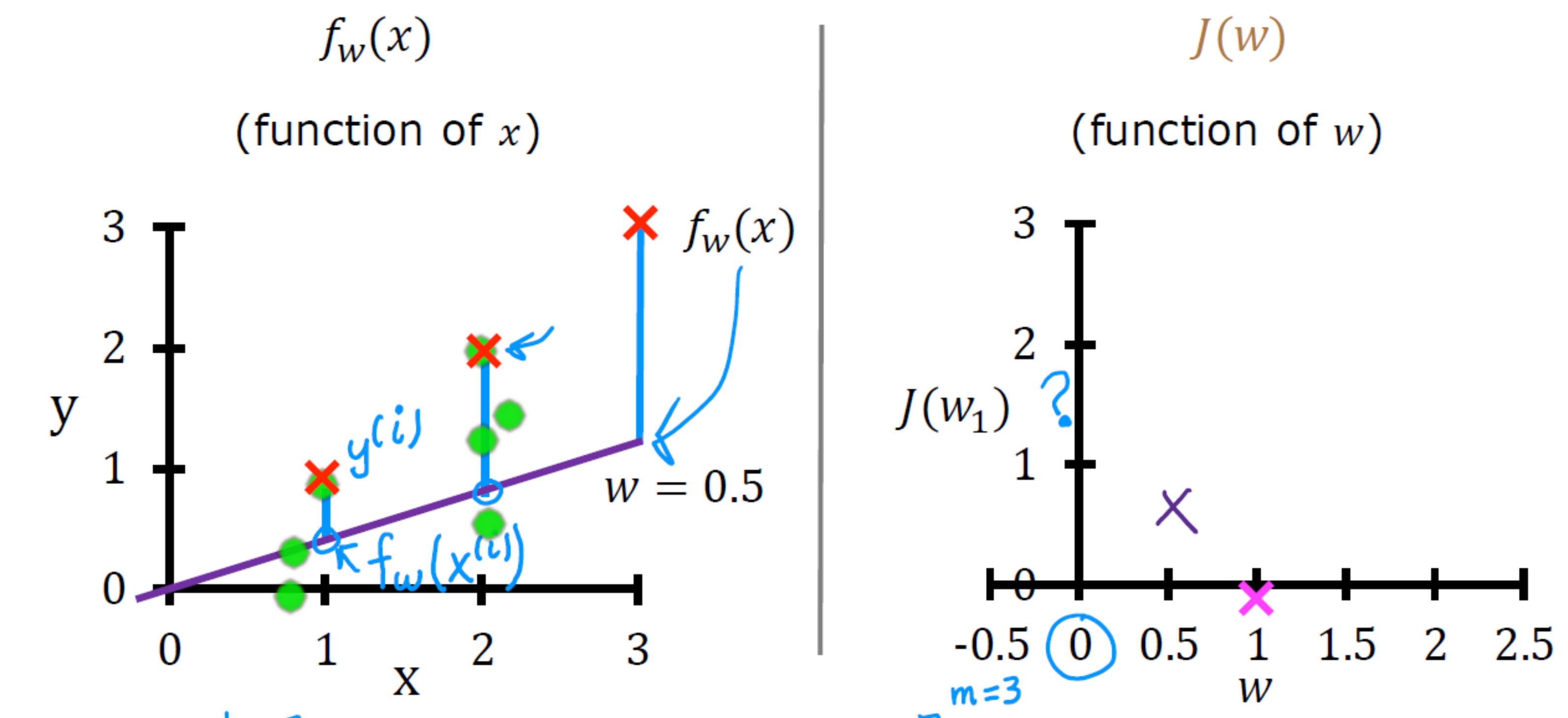
w=1

$$\underline{J(w)} = \frac{1}{2m}\sum_{i=1}^{m}(\underbrace{f_w(x^{(i)})}_{w\,x^{(i)}} - y^{(i)})^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}(\overbrace{w x^{(i)} - y^{(i)}}^{\emptyset})^2$$

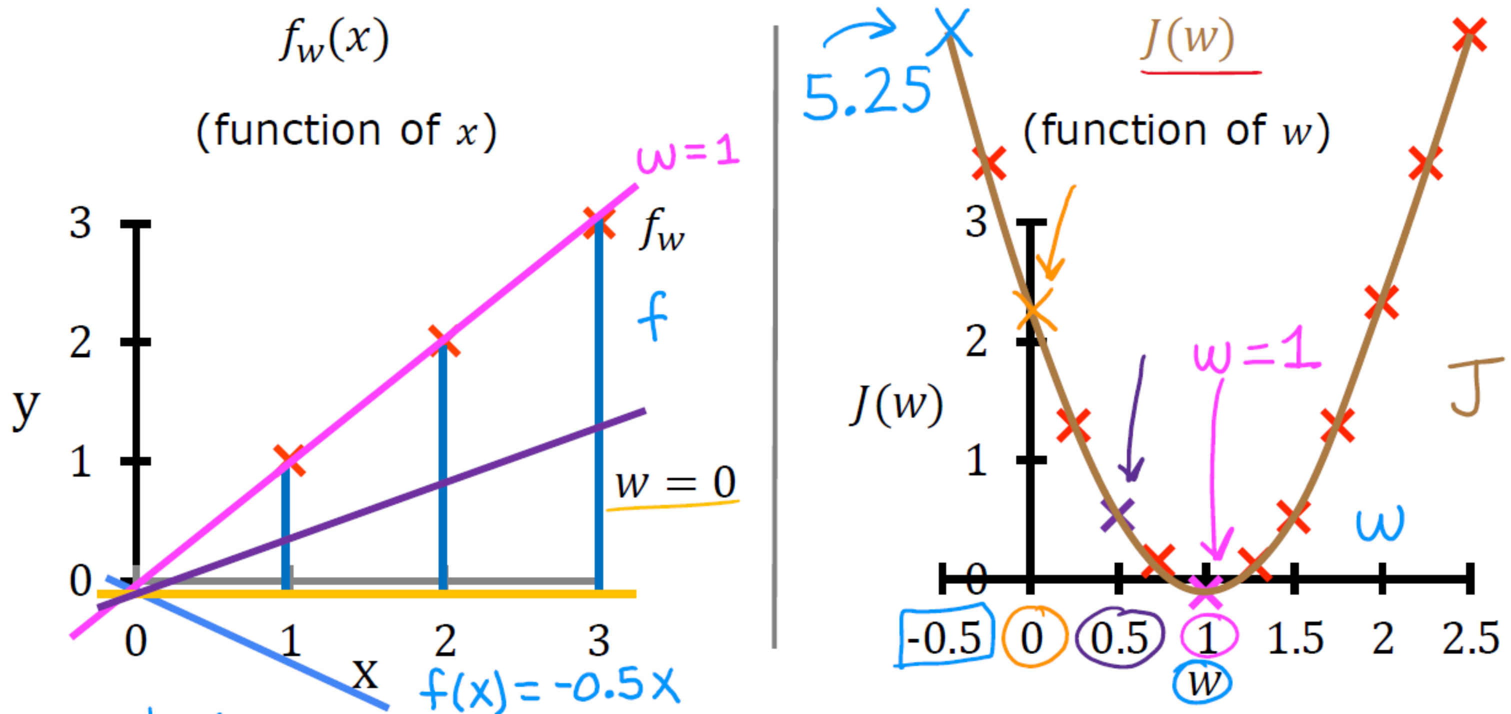
$$= \frac{1}{2m}(\,0^2 + 0^2 + 0^2\,) = 0$$

Cost Function



$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2+(1-2)^2+(1.5-3)^2\right] = \frac{1}{2\times 3}[3.5] = \frac{3.5}{6} \approx 0.58$

Cost Function



$J(0) = \frac{1}{2m}\left(1^2 + 2^2 + 3^2\right) = \frac{1}{6}[14] \approx 2.3$

how to choose w?

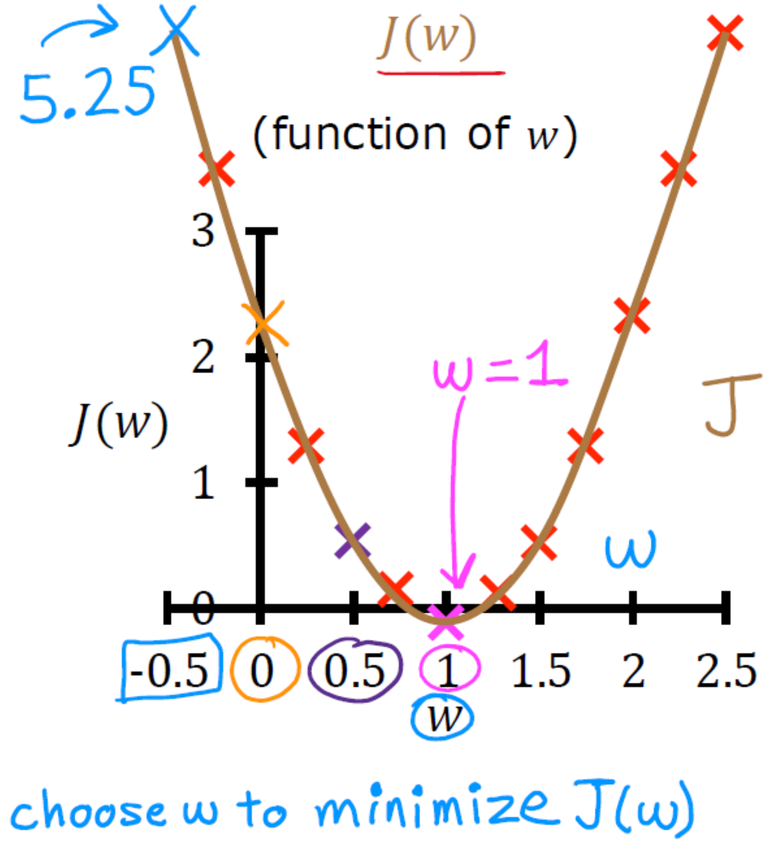
Cost Function

goal of linear regression:

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

general case:

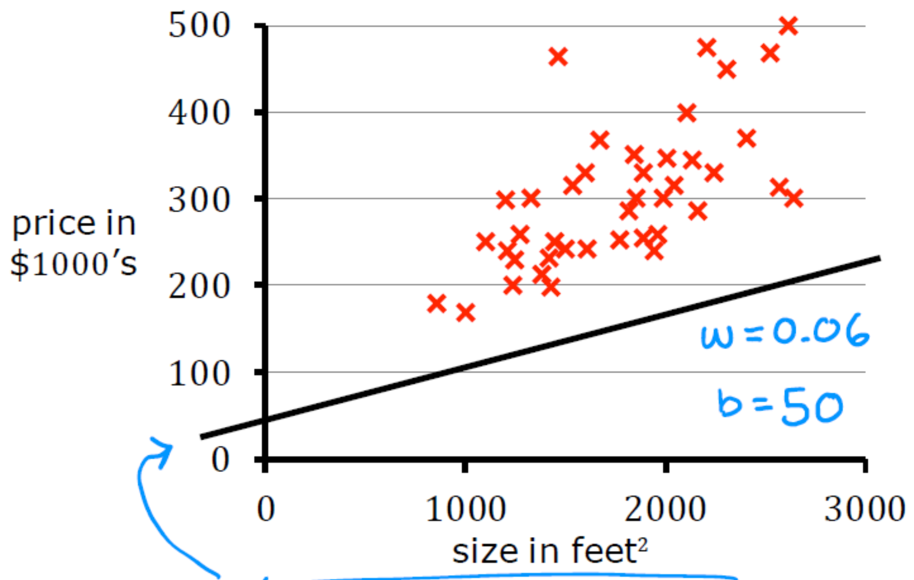
$$\underset{w,b}{\text{minimize}}\, J(w,b)$$



Visualizing the Cost Function

$f_{w,b}$

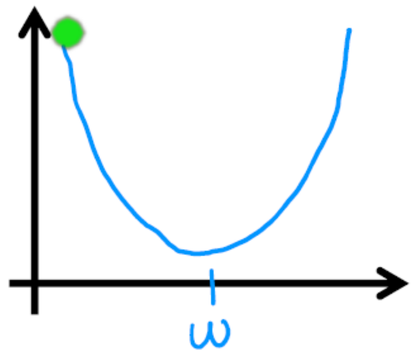
(function of x)



$$f_{w,b}(x) = 0.06x + 50$$

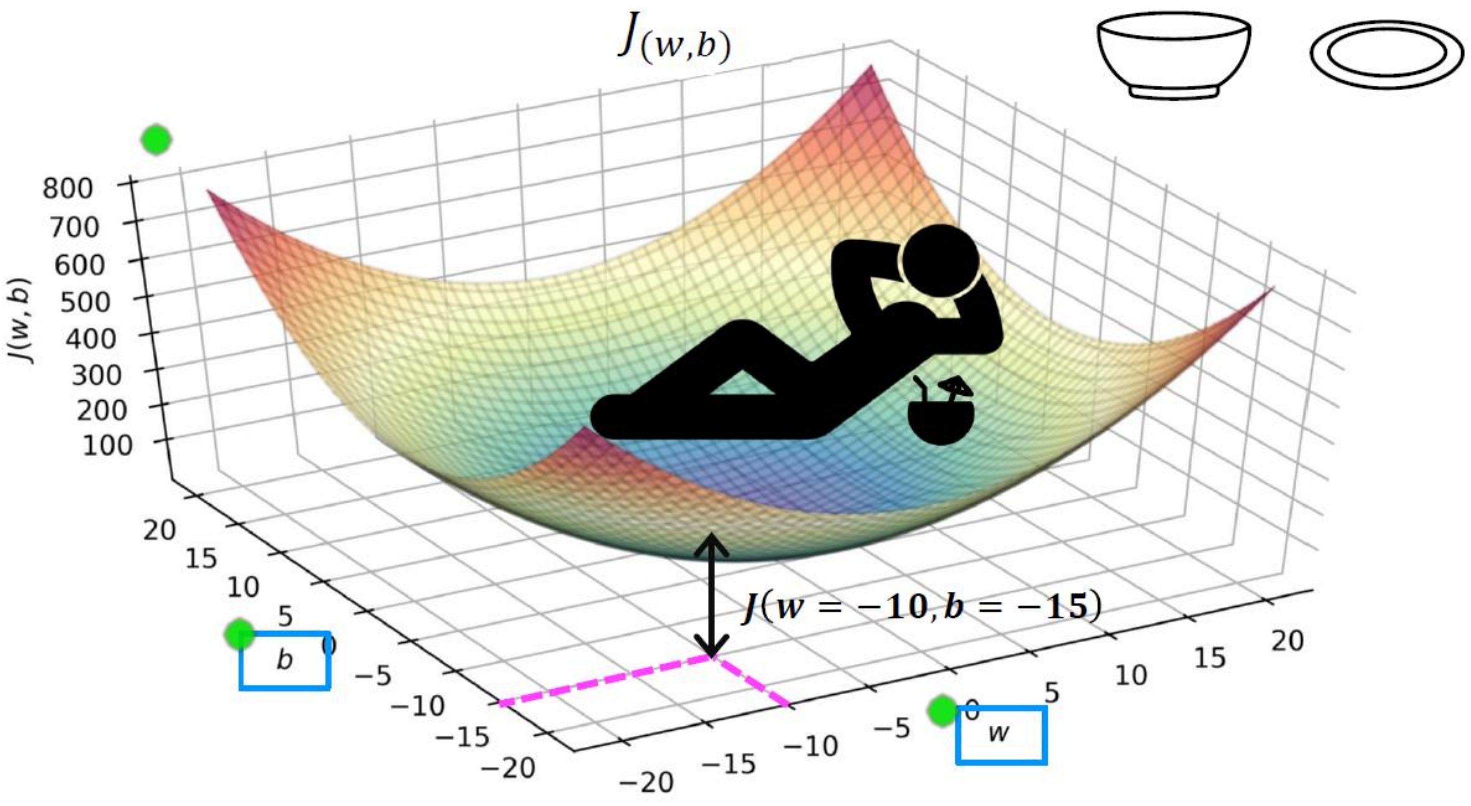
J

(function of w, b)

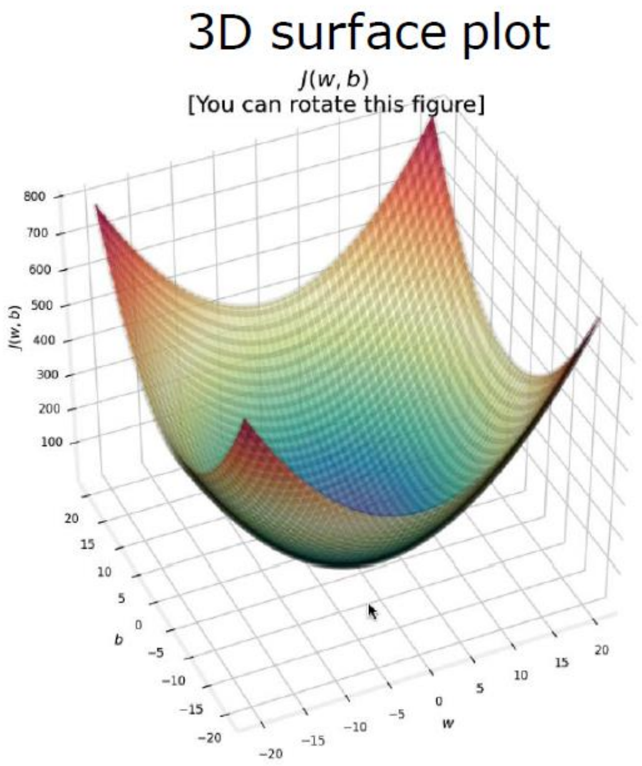


$$\boxed{w, b}$$

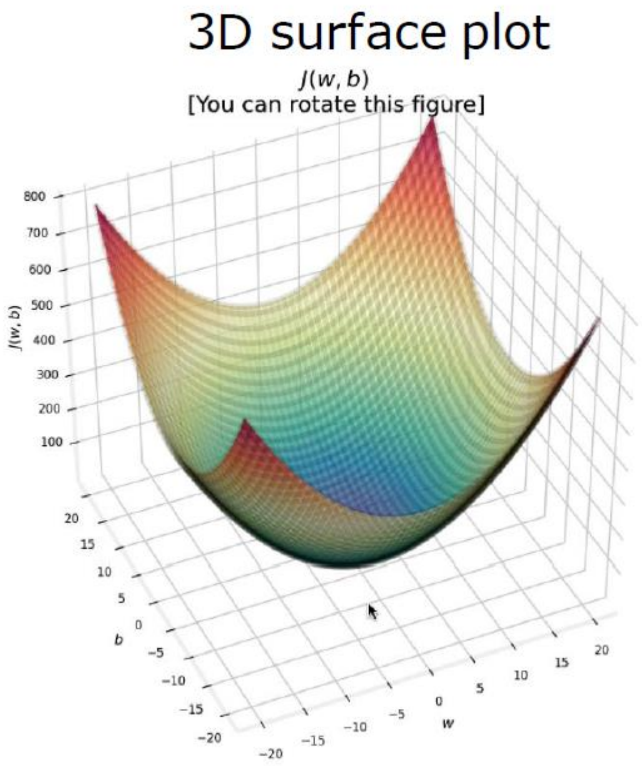
Visualizing the Cost Function



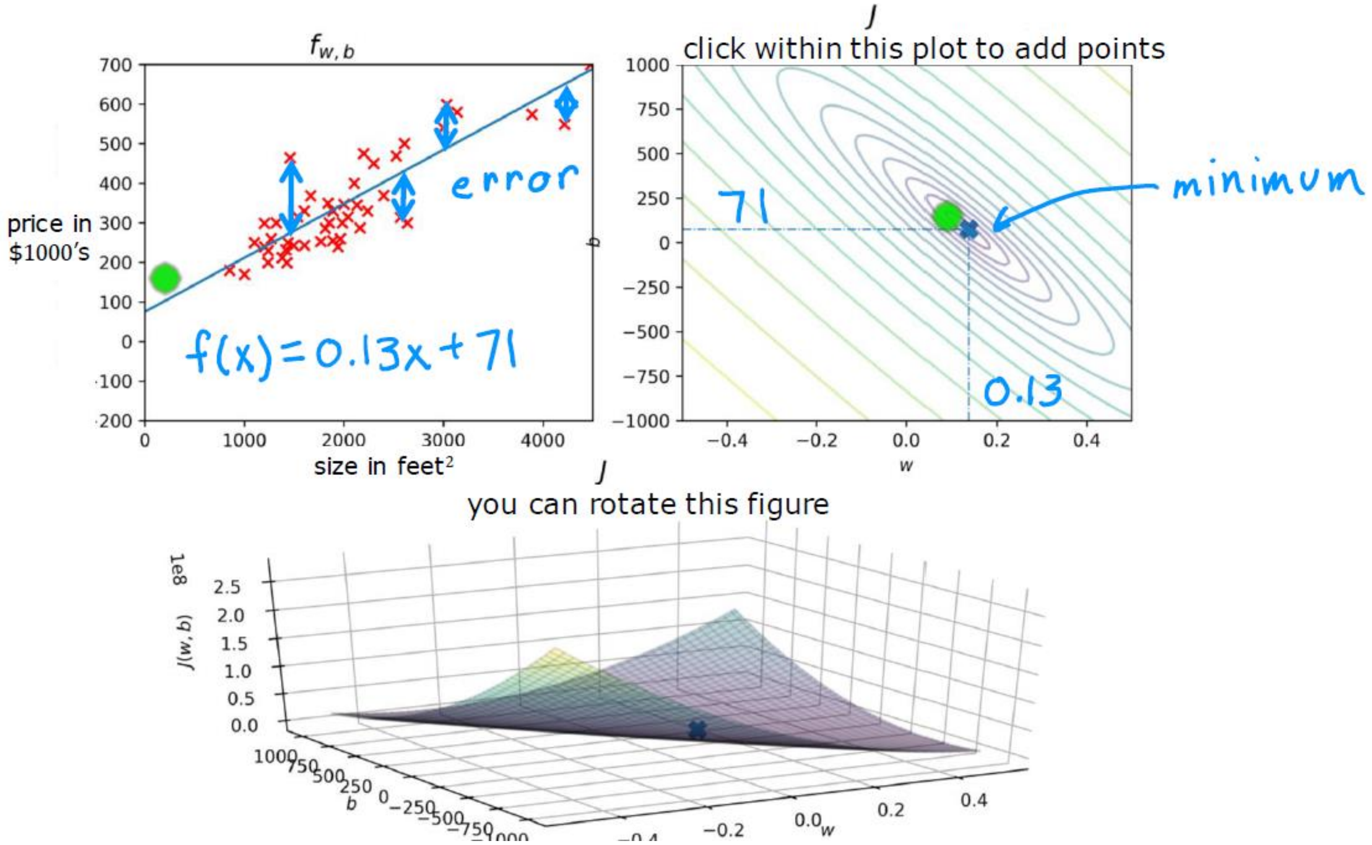
Visualizing the Cost Function

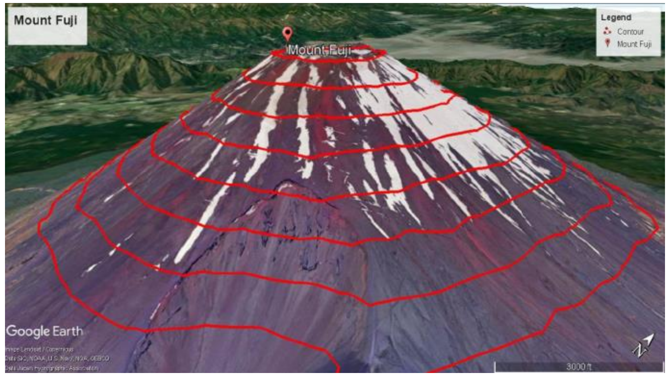


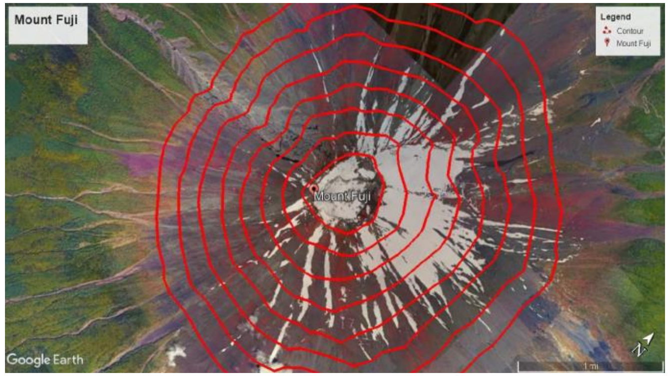
Visualizing the Cost Function



Visualizing the Cost Function







Gradient Descent

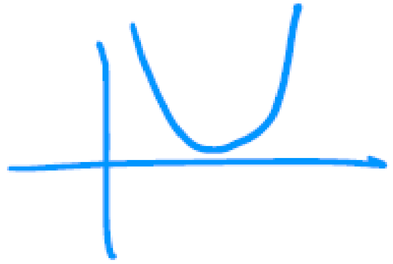
Have some function $J(w, b)$ for linear regression or any function

Want $\min_{w,b} J(w, b)$ $\qquad \min_{w_1, \ldots, w_n, b} J(w_1, w_2, \ldots, w_n, b)$

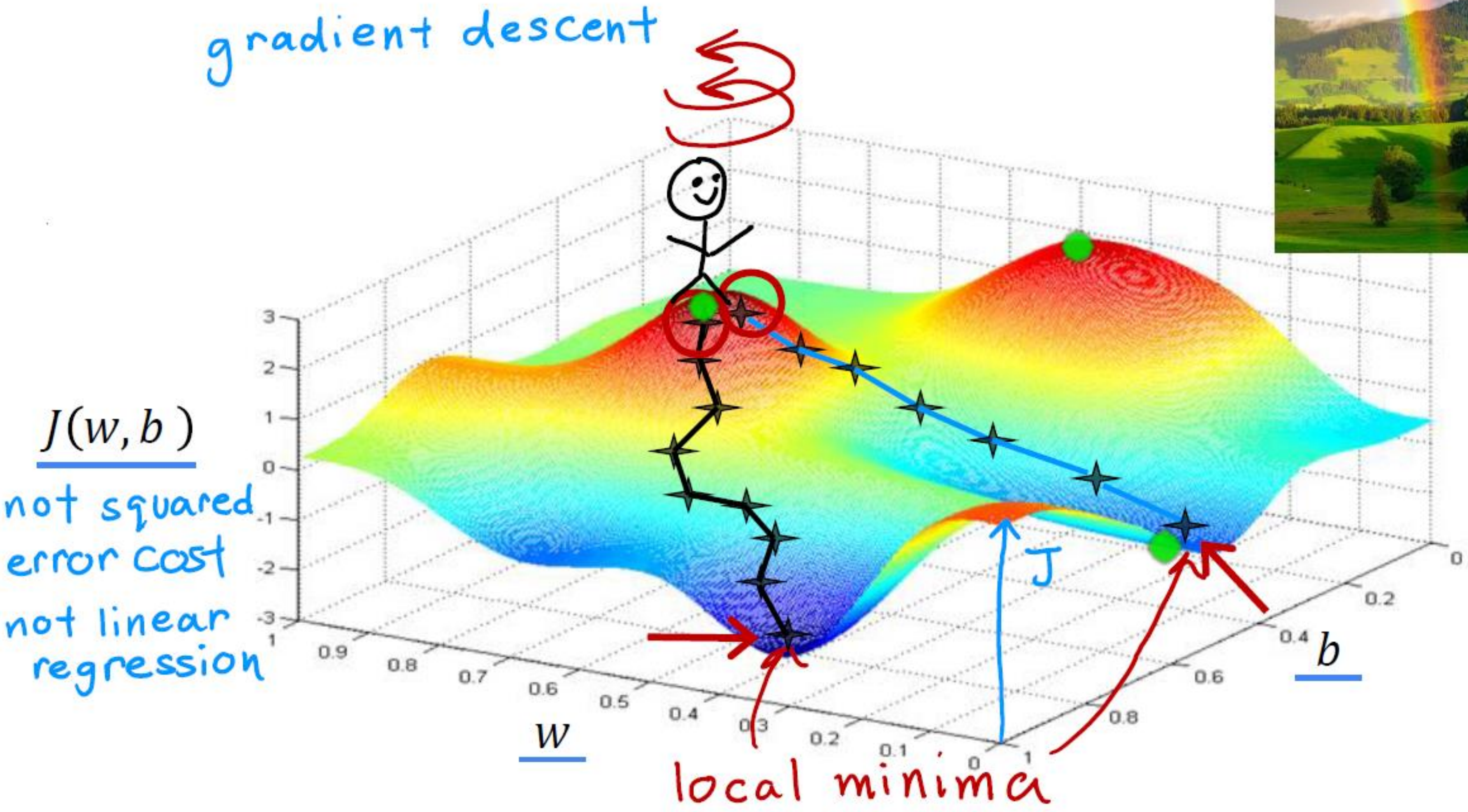
Outline:

- Start with some w, b (set $w=0$, $b=0$)
- Keep changing w, b to reduce $J(w, b)$
- Until we settle at or near a minimum

J not always



Gradient Descent



Gradient Descent Algorithm

Repeat until convergence

$$\begin{cases} w = w - \alpha \boxed{\frac{\partial}{\partial w} J(w,b)} \\ b = b - \alpha \frac{\partial}{\partial b} J(w,b) \end{cases}$$

Learning rate

Derivative

Simultaneously update w and b

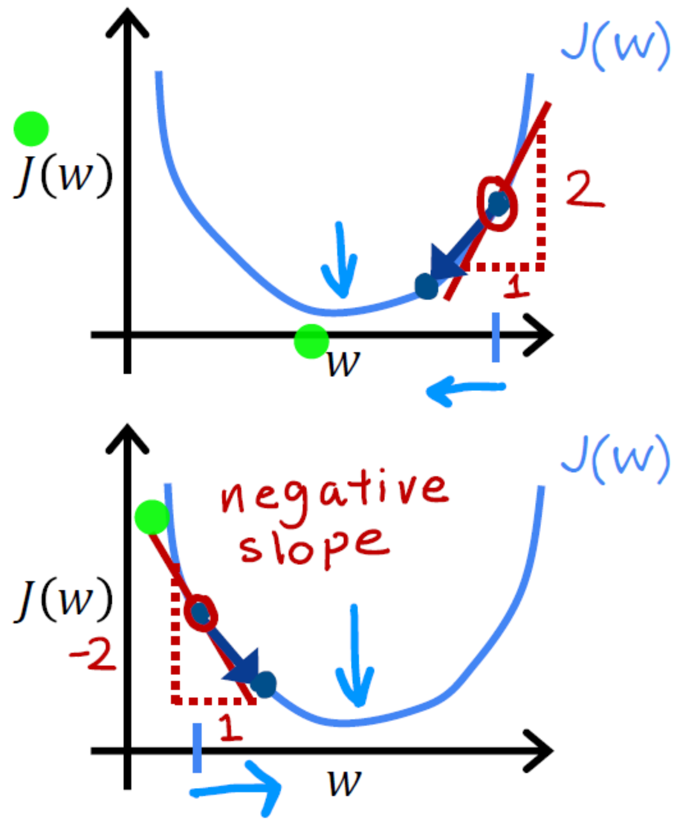
$tmp\_w = w - \alpha \frac{\partial}{\partial w} J(w,b)$

$tmp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$

$w = tmp\_w$

$b = tmp\_b$

Gradient Descent Algorithm



$$w = w - \alpha \underbrace{\boxed{\frac{d}{dw}J(w)}}_{>0}$$

$$w = w - \alpha \cdot (\text{positive number})$$

$$\underbrace{\frac{d}{dw}J(w)}_{<0}$$

$$w = w - \alpha \cdot (\text{negative number})$$

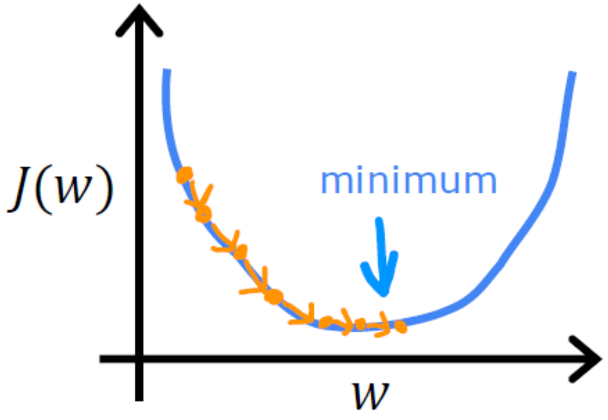
Gradient Descent Algorithm

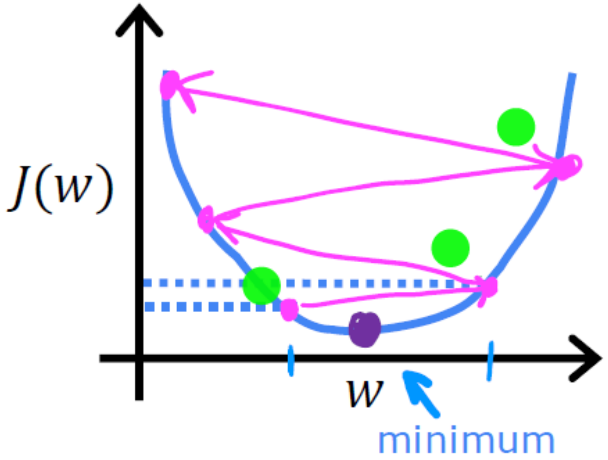
$$w = w - \alpha \frac{d}{dw} J(w)$$

If α is too small…
Gradient descent may be slow.

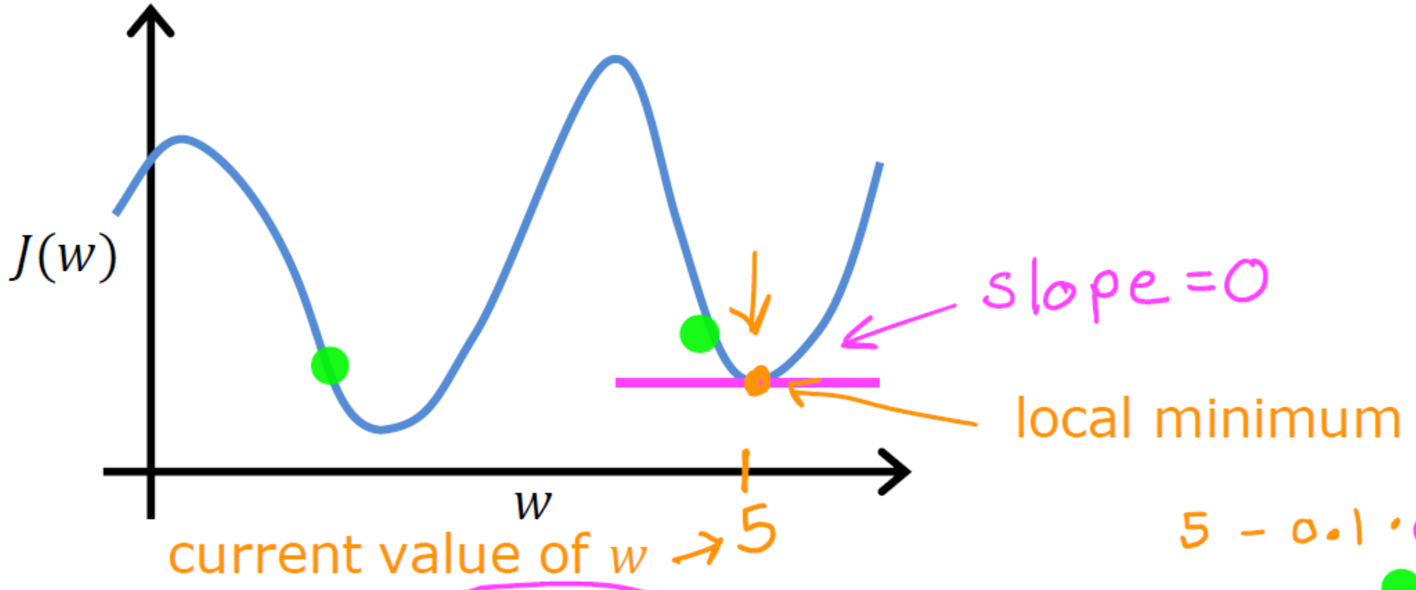
If α is too large…

Gradient descent may:
- Overshoot, never reach minimum
- Fail to converge, diverge





Gradient Descent Algorithm



$$w = w - \underbrace{\alpha}_{0.1}\underbrace{\boxed{\frac{d}{dw}J(w)}}_{=0}$$

$$5 - 0.1 \cdot 0$$

$$w = w - \alpha \cdot 0$$

$$w = w$$

Gradient Descent for Linear Regression

Linear regression model

$$f_{w,b}(x) = wx + b$$

Cost function

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Gradient descent algorithm

repeat until convergence {

$$w = w - \alpha \frac{\partial}{\partial w} J(w,b) \rightarrow \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)x^{(i)}$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b) \rightarrow \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)$$

}

Evaluation Metrics for Linear Regression

Mean Absolute Error

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

Mean Squared Error

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}}$$

R Squared

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Linear Regression for Multiple Features

Model:

Previously: $f_{w,b}(x) = wx + b$

$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b$$

example

$$f_{w,b}(x) = 0.1\, x_1 + 4\, x_2 + 10 x_3 + -2\, x_4 + 80$$

size #bedrooms #floors years base price

$$f_{w,b}(\mathrm{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

Linear Regression for Multiple Features

vector $\vec{x} = [x_1 \quad x_2 \quad x_3 \ldots x_n]$

$\vec{w} = [w_1 \quad w_2 \quad w_3 \ldots w_n]$

b

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n + b$$

dot product

multiple linear regression

Vectorization

```
w = np.array([1.0,2.5,-3.3])
b = 4                x[0]  x[1]  x[2]
x = np.array([10,20,30])
```

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

```
f = w[0] * x[0] +
    w[1] * x[1] +
    w[2] * x[2] + b
```

```
f = 0        range(n)
for j in range(0,n):
    f = f + w[j] * x[j]
f = f + b
```

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

```
f = np.dot(w,x) + b
```

Gradient Descent for Multiple Features

repeat {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$$

}

Gradient Descent for Multiple Features

n features ($n \geq 2$)

repeat {

j=1

$$\underbrace{w_1}_{} = w_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right) \underbrace{x_1^{(i)}}_{}$$

$\longrightarrow \frac{\partial}{\partial w_1} J(\vec{w}, b)$

$\vdots$

j=n

$$w_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right) x_n^{(i)}$$

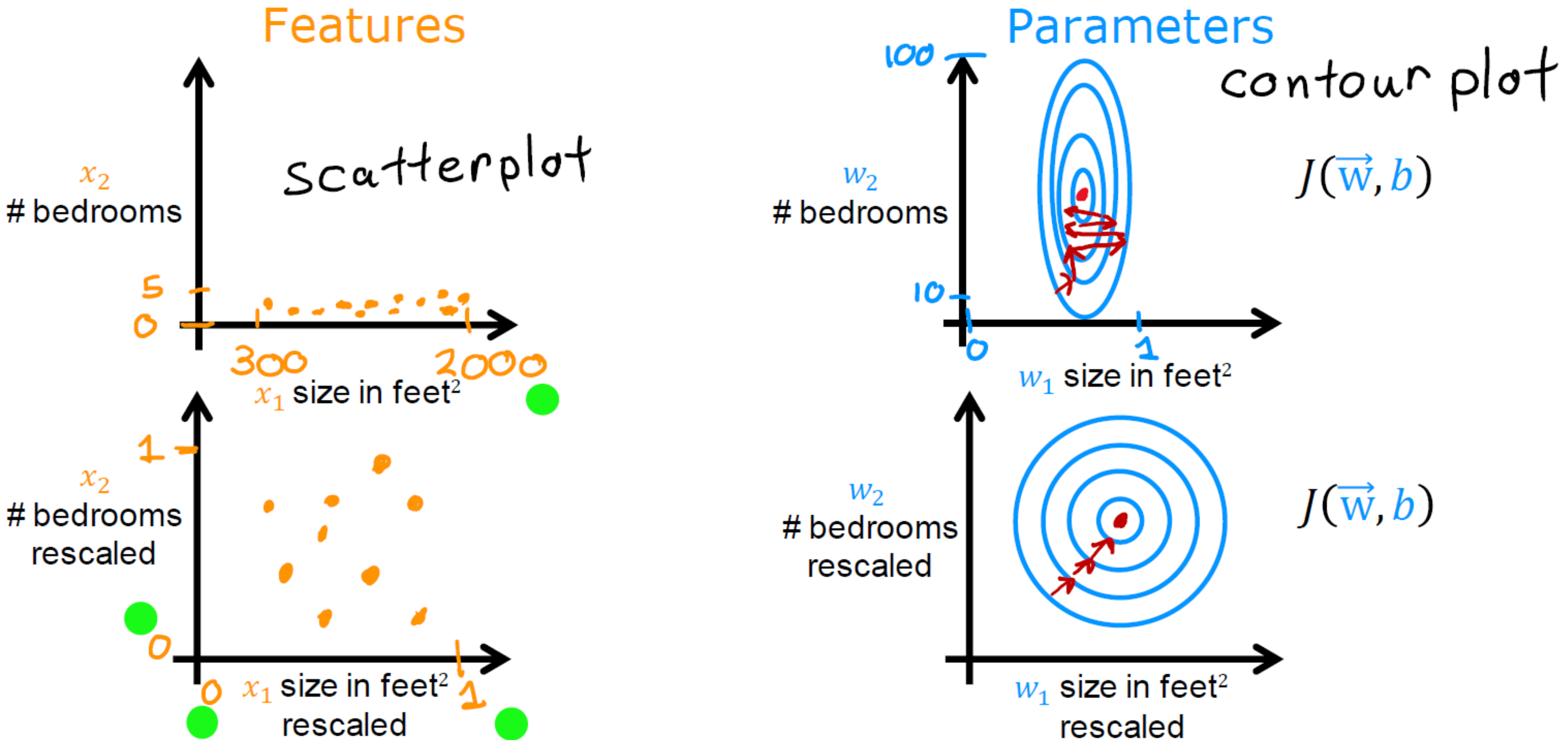
$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)$$

}

simultaneously update

w_j (for $j = 1, \cdots, n$) and b

The Effect of Feature Scaling



The Effect of Feature Scaling

aim for about $-1 \leq x_j \leq 1$ for each feature x_j

$-3 \leq x_j \leq 3$

$-0.3 \leq x_j \leq 0.3$

acceptable ranges

$0 \leq x_1 \leq 3$ okay, no rescaling

$-2 \leq x_2 \leq 0.5$ okay, no rescaling

$-100 \leq x_3 \leq 100$ too large → rescale

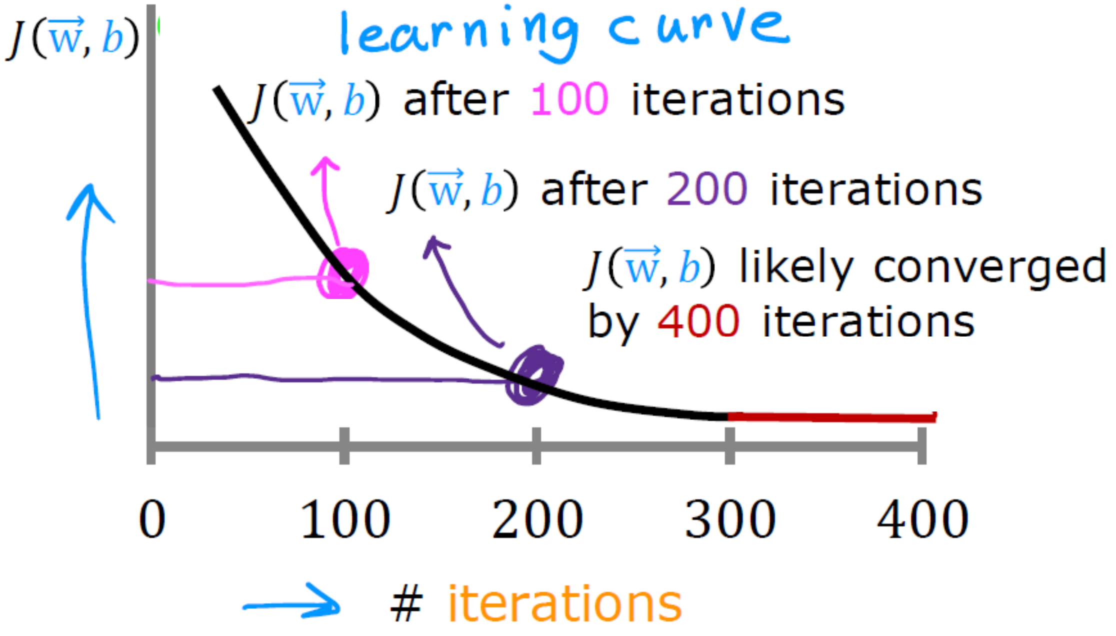
$-0.001 \leq x_4 \leq 0.001$ too small → rescale

$98.6 \leq x_5 \leq 105$ too large → rescale

Making Sure Gradient Descent is Working

objective: $\min_{\vec{w},b} J(\vec{w}, b)$

$J(\vec{w}, b)$ should decrease after every iteration

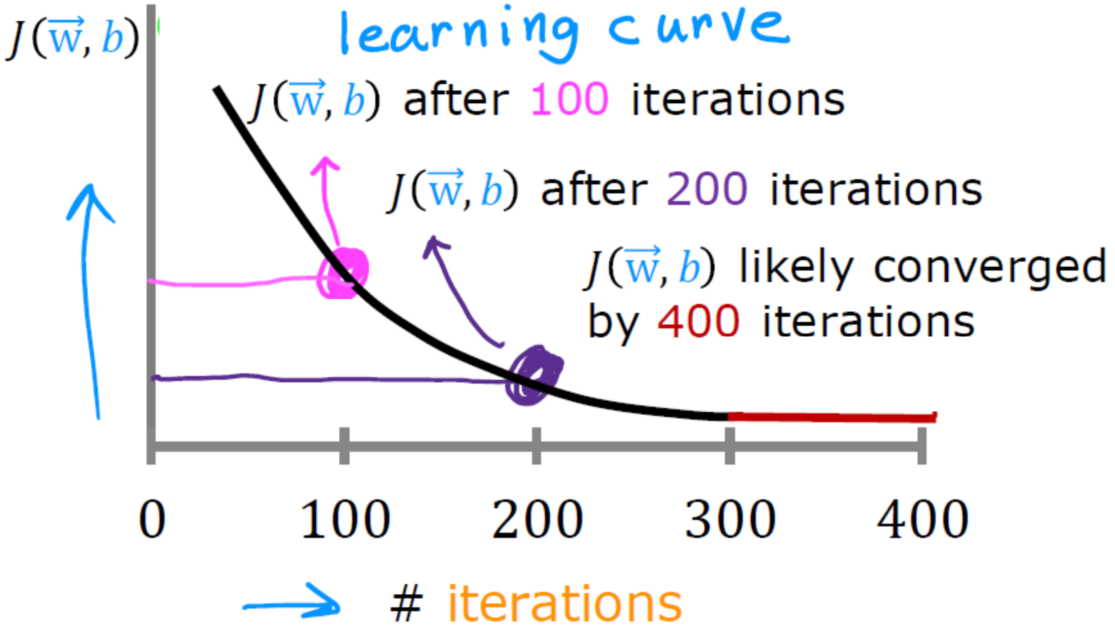


iterations needed varies 30 1,000 100,000

Checking Gradient Descent for Convergence

objective: $\min_{\vec{w},b} J(\vec{w},b)$

$J(\vec{w},b)$ should decrease after every iteration



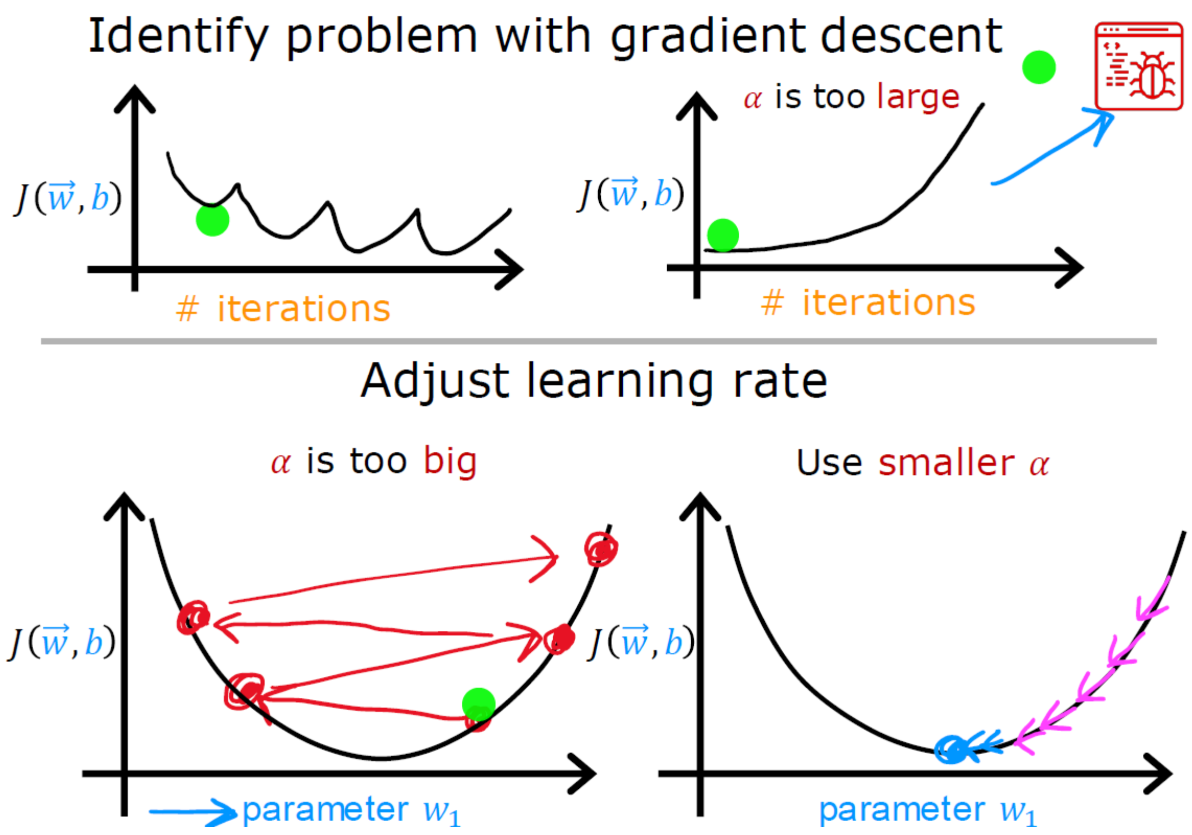
iterations needed varies 30 1,000 100,000

Automatic convergence test

Let ε "epsilon" be 10^{-3}.
0.001

If $J(\vec{w},b)$ decreases by $\leq \varepsilon$ in one iteration, declare convergence.

Choosing the Learning Rate

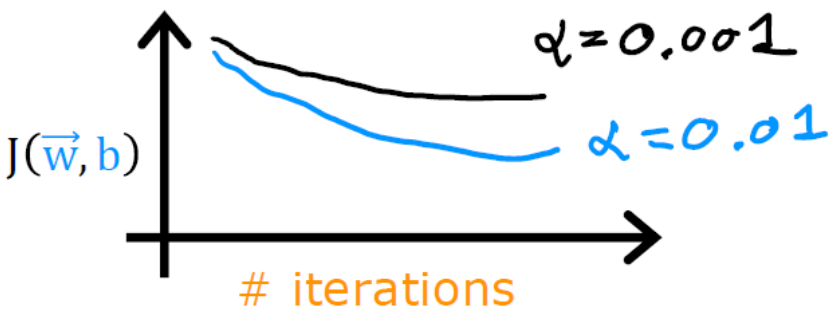


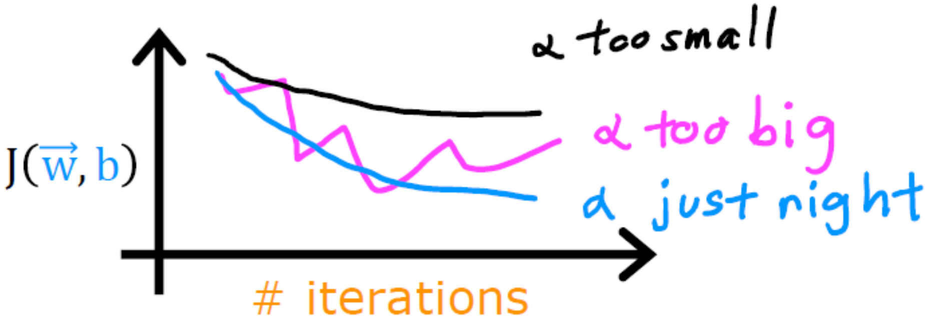
Choosing the Learning Rate

Values of α to try:

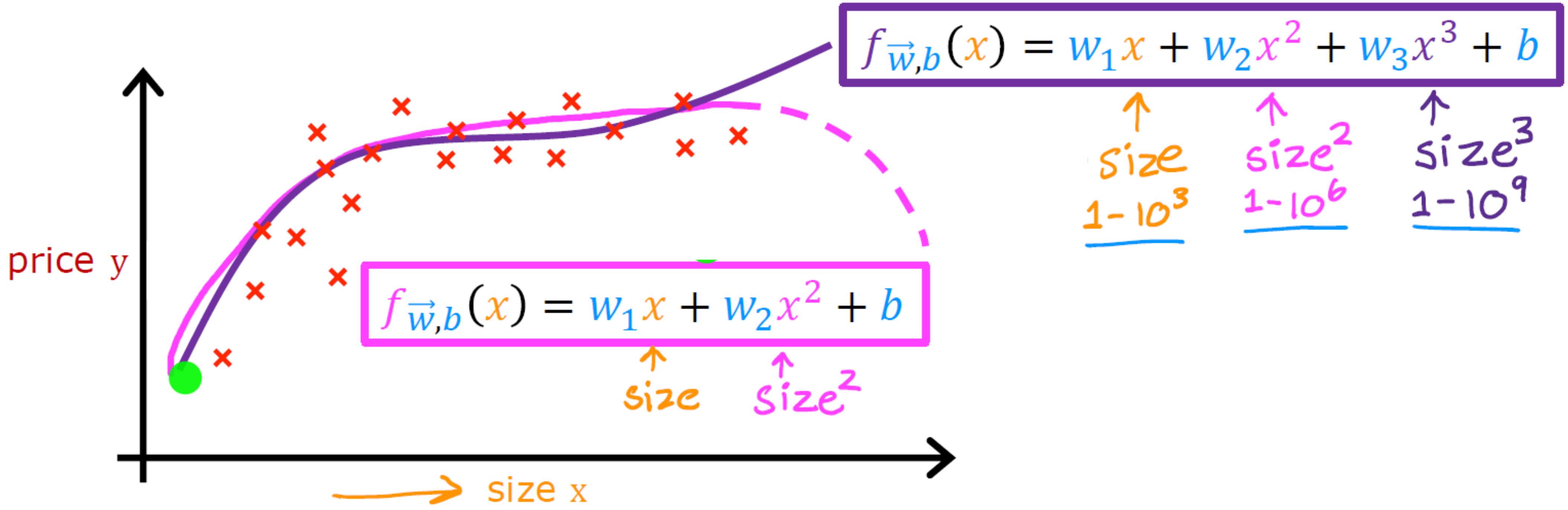
... 0.001 0.003 0.01 0.03 0.1 0.3 1 ...

3X ≈3X 3X ≈3X 3X ≈3X

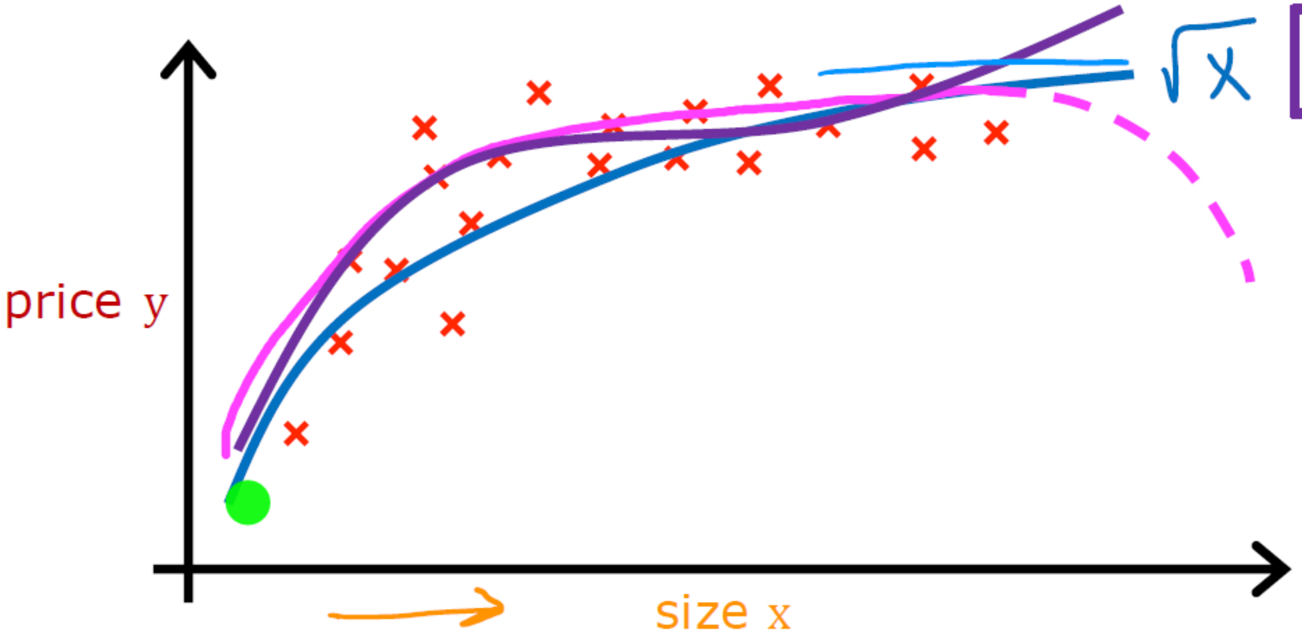




Polynomial Regression



Polynomial Regression



$$\boxed{f_{\vec{w},b}(x) = w_1 x + w_2 \sqrt{x} + b}$$

size

$\sqrt{size}$

THANK YOU

NEXT LECTURE WILL BE ONLINE ON MON, 15.5.2023, IN SHAA ALLAH!

SABBAGH@IEEE.ORG

CONNECT ON LINKEDIN