GRADIENT DESCENT AND MULTIPLE LINEAR REGRESSION

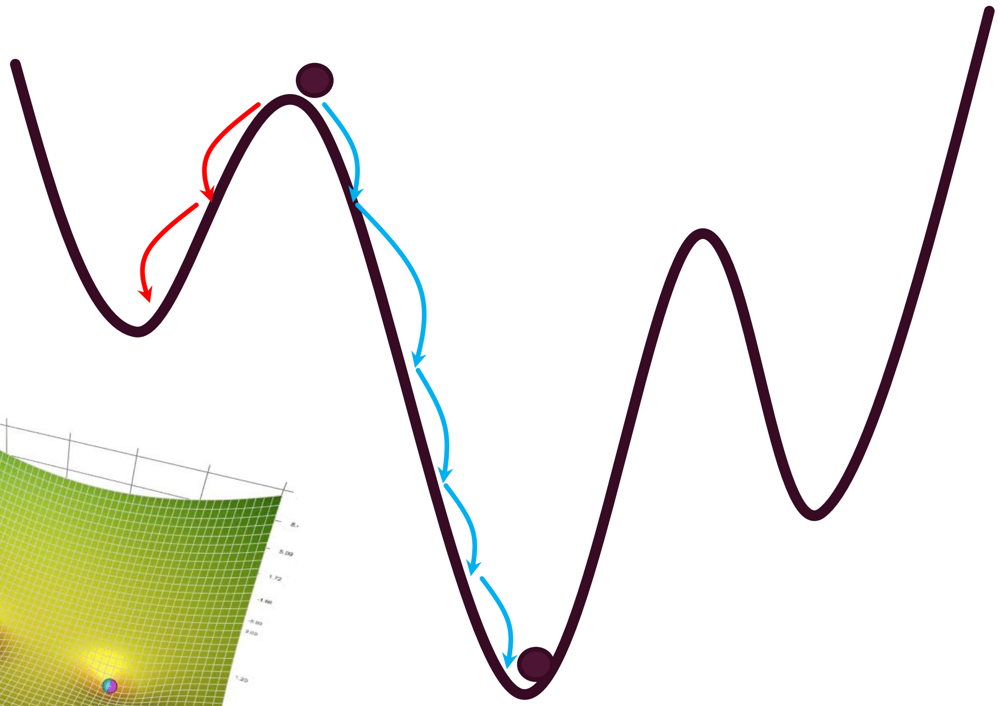
DR. FARHAD RAZAVI

OUTLINE

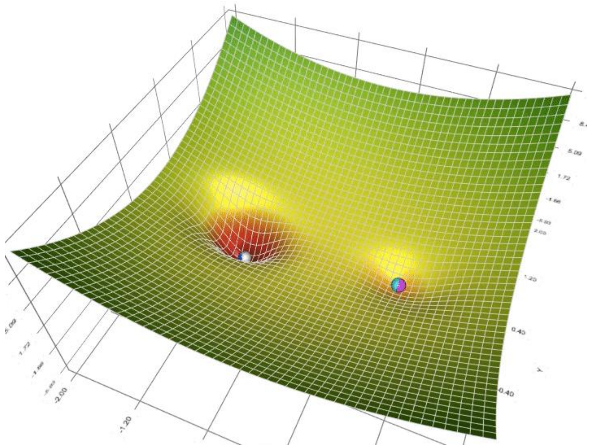
- Gradient Descent
- Linear regression for multiple features
- Gradient Descent for multiple features regression
- Feature engineering
- Polynomial regression

GRADIENT DESCENT INTUITION

- Gradient Descent Algorithm:
  - Start with some w and b.
  - Keep changing w, and b to reduce $J(w, b)$.
  - Continue until we settle at or near a minimum.



- Drawbacks:
  - It could get stuck in a local minima.



GRADIENT DESCENT FOR LINEAR REGRESSION

- Linear Regression:
  - $f_{w,b}(x) = wx + b$
  - $J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2$
- Gradient Descent
  - Repeat until convergence {

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

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

}

- Cost function of Linear Regression is quadratic. This means it has only one minimum value, so we are guaranteed to reach global minimum.

MULTIPLE FEATURES

Index i	Size in feet² x_1	Number of bedrooms x_2	Number of floors x_3	Age of home in years x_4	Price ($) in $1000's y
1	2104	5	1	45	460
2	1416	3	2	40	232
3	1534	3	2	30	315
…	…	…	…	…	…

- $x_j = j^{th}$ feature
- $n =$ number of features
- $\vec{x}^{(i)} =$ features of i^{th} training example
- $x_j^{(i)}$ value of feature j in i^{th} training example

MODEL

- $f_{w,b}(x) = wx + b$
- $f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$
- $f_{w,b}(x) = \underset{\substack{\uparrow \\ \text{size}}}{0.1x_1} + \underset{\substack{\uparrow \\ \text{\#bedrooms}}}{4x_2} + \underset{\substack{\uparrow \\ \text{\#floors}}}{10x_3} - \underset{\substack{\uparrow \\ \text{years}}}{2x_4} + \underset{\substack{\uparrow \\ \text{base price}}}{80}$

VECTOR REPRESENTATION

- $f_{w,b}(x) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$

  $\vec{w} = [w_1 \quad w_2 \quad w_3 \quad \ldots \quad w_n]$ parameters of the model

  b is a number

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

- $f_{\vec{w},b}(\vec{x}) = \vec{w}.\vec{x} + b = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$

dot product

Multiple Linear Regression (MLR)

VECTORIZATION

- $f_{\vec{w},b}(\vec{x}) = \vec{w}.\vec{x} + b = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$

```
w = np.array([0.1, 4.0, 10, -2])
b = 4
w = np.array([1513, 3, 2, 30])
```

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

```
n = 4
f_wb = 0
for j in range(0, n):
    f_wb = f_wb + w[j] * x [j]
f_wb = f_wb + b
```

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

NumPy

GRADIENT DESCENT FOR MLR

- Parameters: $w_1, w_2, w_3, \dots, w_n$

  b
- Models: $f_{\vec{w},b}(\vec{x}) = \vec{w}.\vec{x} + b = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$
- Cost function: $J(w_1, w_2, w_3, \dots, w_n, b) = J(\vec{w}, b)$
- Gradient descent
  - Repeat {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, w_2, w_3, \dots, w_n, b) = w_j - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)} \qquad (j=1,\dots,n)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w_1, w_2, w_3, \dots, w_n, b) = b - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)$$

}

CLOSED FORM SOLUTION

- There is a closed form solution for the "linear regression" parameters! It is called normal equation.

$$w = (XX^T)^{-1}Xy^T$$

- Disadvantages:
  - This is only available for Linear regression and does not generalize to other learning algorithms.
  - Slow when number of features are large ($> 10,000$)

FEATURE SCALING

$\widehat{price} = w_1 x_1 + w_2 x_2$

size #bedrooms

x_1: size (feet^2)

range: 300 – 2,000

large

x_2: #bedrooms

range: 0 – 5

small

House: $x_1 = 2000, x_2 = 5, price = \$500K$ one training example

size of the parameters w_1, w_2 ?

$w_1 = 50, w_2 = 0.1, b = 50$

$\widehat{price} = 50 * 2000 + 0.1 * 5 + 50$

$\widehat{price} = 100{,}050.5K$

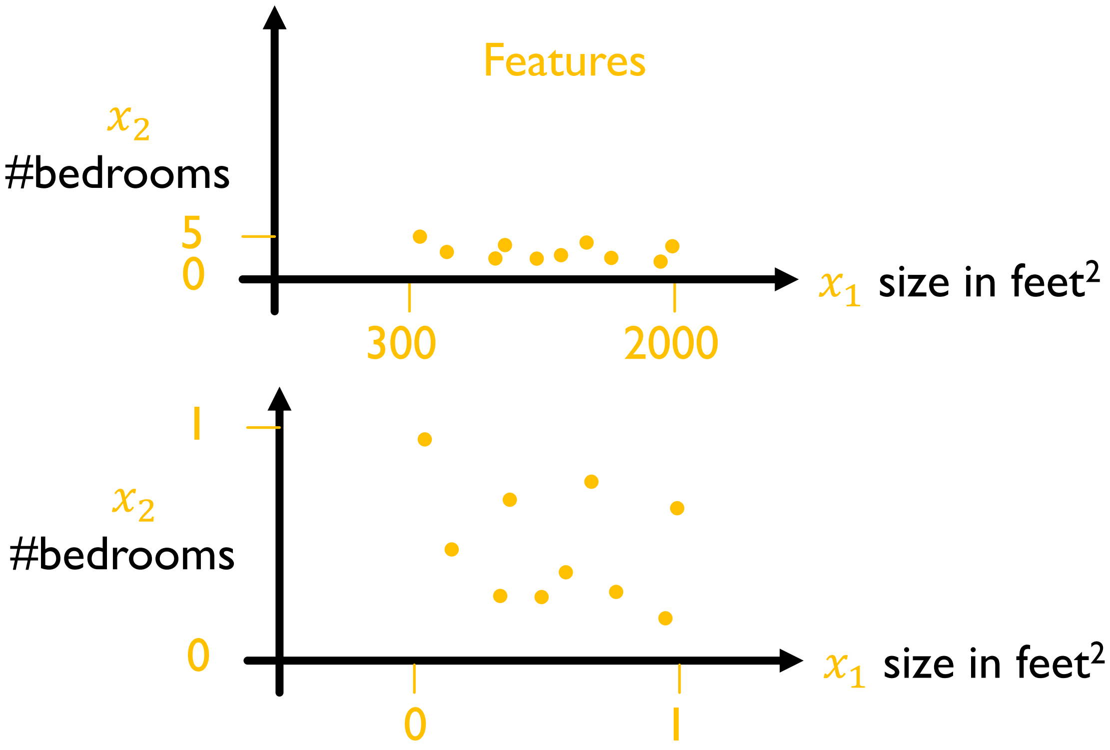
$w_1 = 0.1, w_2 = 50, b = 50$

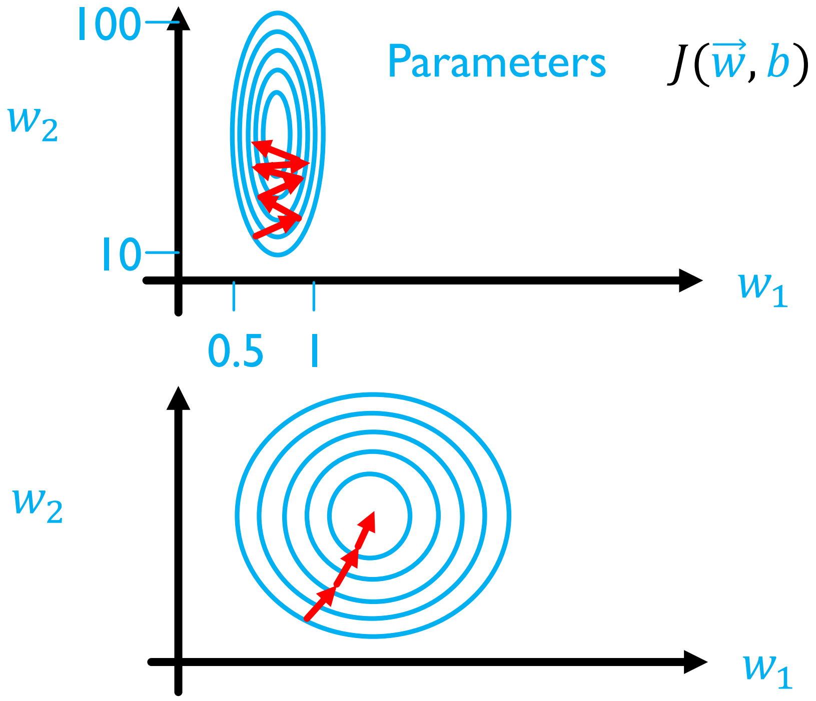
$\widehat{price} = 0.1 * 2000 + 50 * 5 + 50$

$\widehat{price} = 500K$

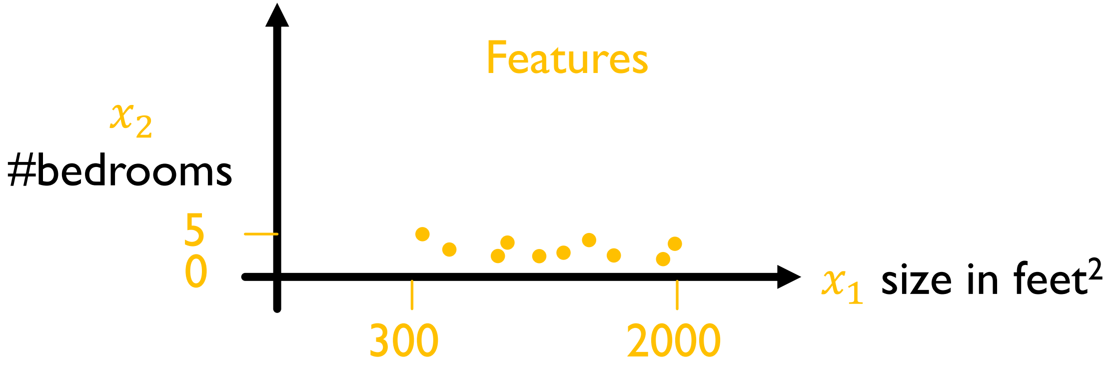
FEATURE SIZE AND PARAMETER SIZE

	Size of feature x_j	Size of parameter w_j
Size in feet^2		
#bedrooms		

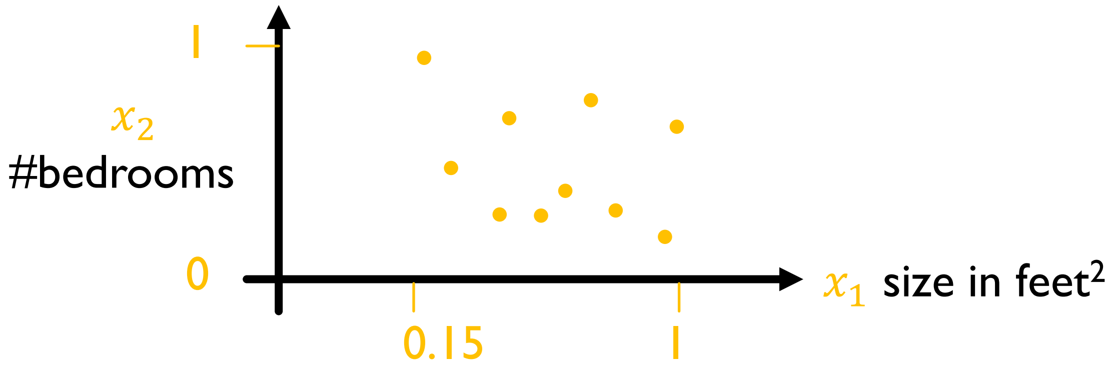




FEATURE SCALING (MAX NORMALIZATION)



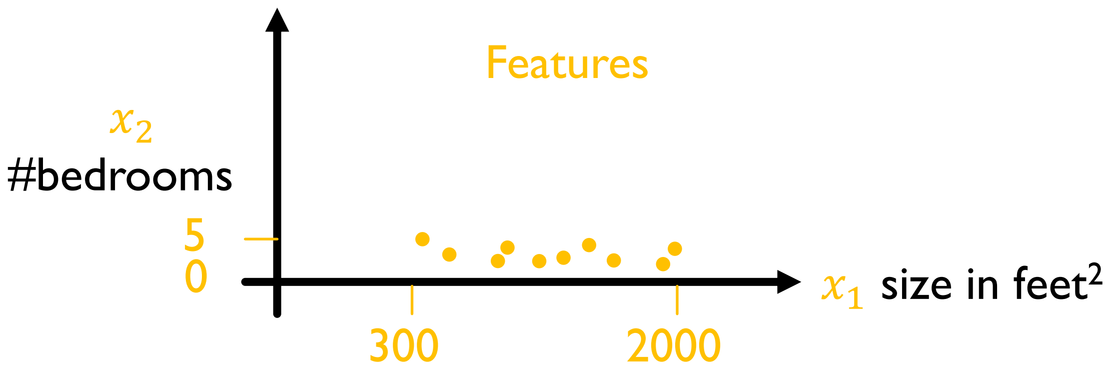
$300 \le x_1 \le 2000$ $\qquad$ $0 \le x_1 \le 5$



$$x_{1,rescaled} = \frac{x_1}{2000} \qquad x_{2,rescaled} = \frac{x_2}{5}$$

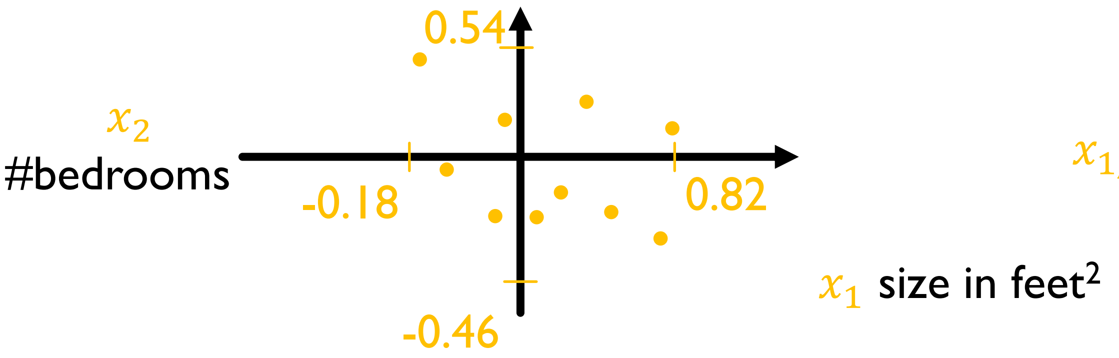
$0.15 \le x_1 \le 1$ $\qquad$ $0 \le x_1 \le 1$

FEATURE SCALING (MEAN NORMALIZATION)



$300 \leq x_1 \leq 2000$ $\quad$ $0 \leq x_1 \leq 5$

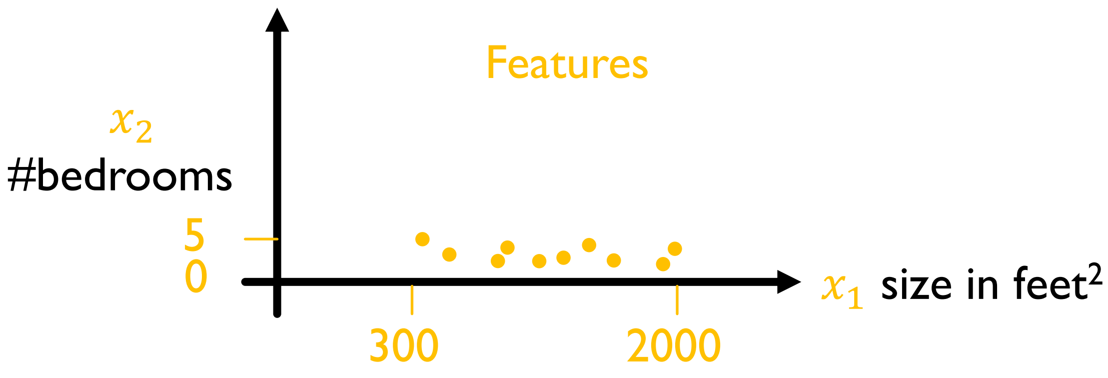
$\mu_1 = 600$ $\quad$ $\mu_2 = 2.3$



$$x_{1,rescaled} = \frac{x_1 - \mu_1}{2000 - 300} \qquad x_{2,rescaled} = \frac{x_2 - \mu_2}{5 - 0}$$

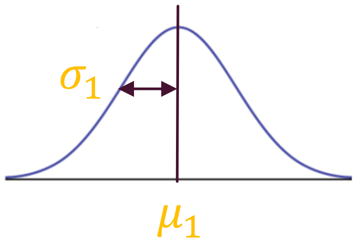
$-0.18 \leq x_1 \leq 0.82$ $\quad$ $-0.46 \leq x_1 \leq 0.54$

FEATURE SCALING (Z-SCORE NORMALIZATION)



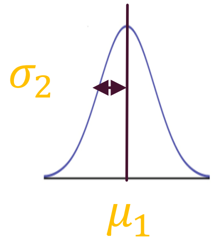
$300 \leq x_1 \leq 2000$

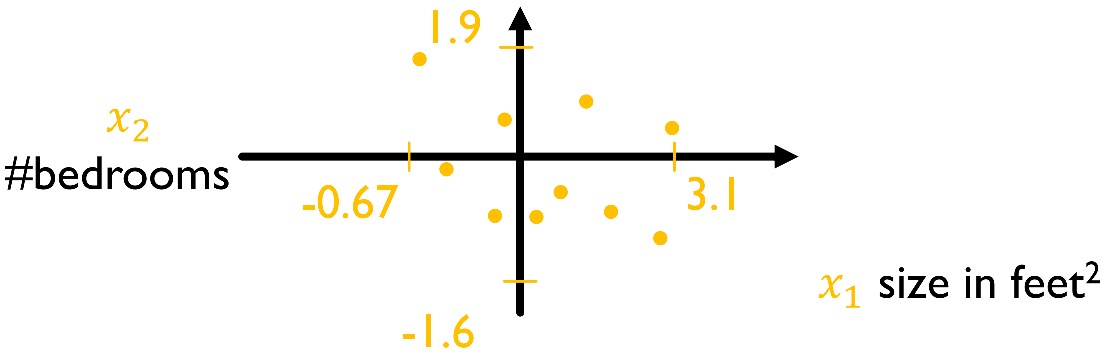
$\mu_1, \sigma_1 = 600, 450$



$0 \leq x_1 \leq 5$

$\mu_2, \sigma_1 = 2.3, 1.4$





$$x_{1,rescaled} = \frac{x_1 - \mu_1}{\sigma_1}$$

$-0.67 \leq x_1 \leq 3.1$

$$x_{2,rescaled} = \frac{x_2 - \mu_2}{\sigma_2}$$

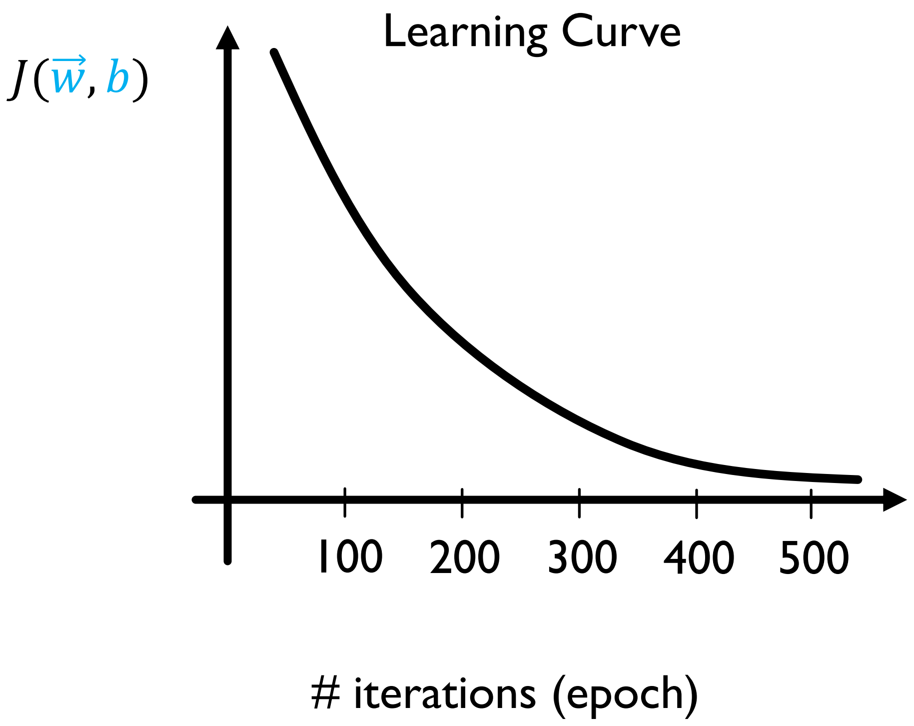
$-1.6 \leq x_1 \leq 1.9$

FEATURE SCALING RULE OF THUMB

- Aim for about for $-1 \leq x_j \leq 1$ for each feature x_j
- Acceptable ranges $\begin{cases} -3 \leq x_j \leq 3 \\ -0.3 \leq x_j \leq 0.3 \end{cases}$
  - $0 \leq x_j \leq 3$ Okay, no rescaling
  - $-2 \leq x_j \leq 0.5$ Okay, no rescaling
  - $-100 \leq x_j \leq 100$ Too large, rescale
  - $-0.001 \leq x_j \leq 0.001$ Too small, rescale
  - $98.6 \leq x_j \leq 105$ Too large, rescale

CHECKING GRADIENT DESCENT

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

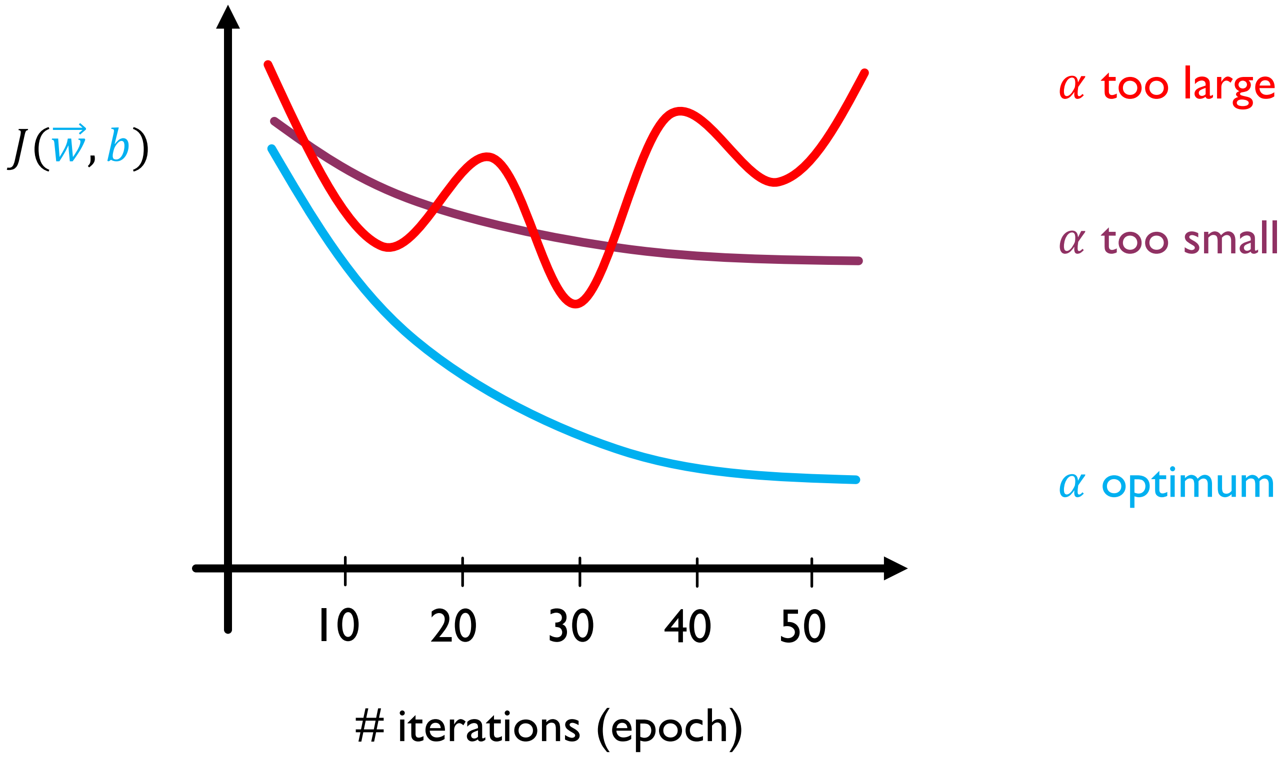


- If $J(\vec{w}, b)$ is getting smaller on each iteration then GD is converging.
  - If you need to speed up the GD, try larger α to see if the GD is still converging or not
- If $J(\vec{w}, b)$ is getting bigger on each iteration then:
  - GD is diverging and your choice of α was too big.
  - Maybe there is a bug in your code!
- The number of iteration is dependent on the application
  - Maybe 50, 1000, 100000 iteration is needed

CHOOSING LEARNING RATE

- Values of α to try:

0.001 0.003 0.01 0.03 0.1 0.3 1



FEATURE ENGINEERING

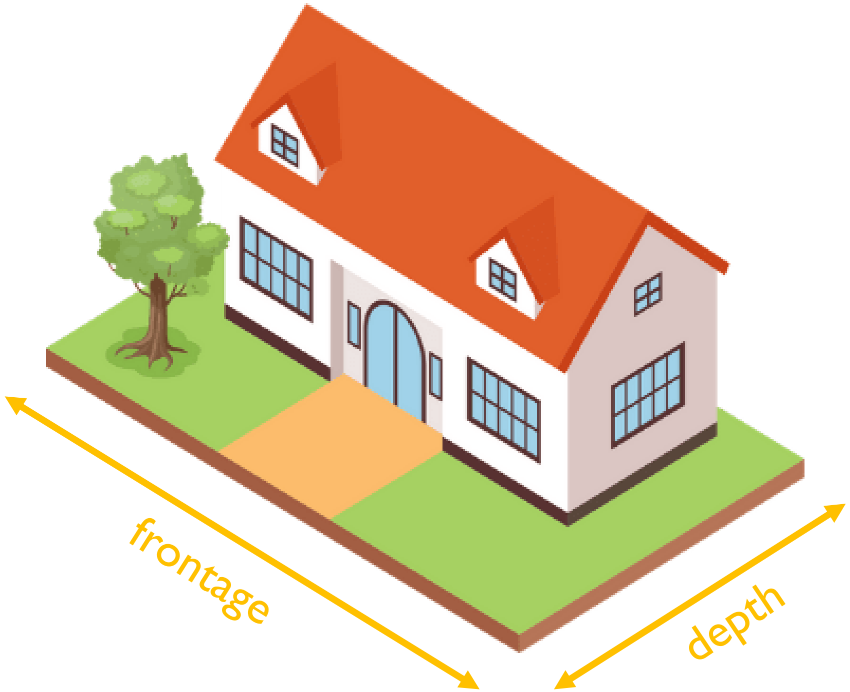
$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + b$$

frontage depth

$$area = frontage \times depth$$

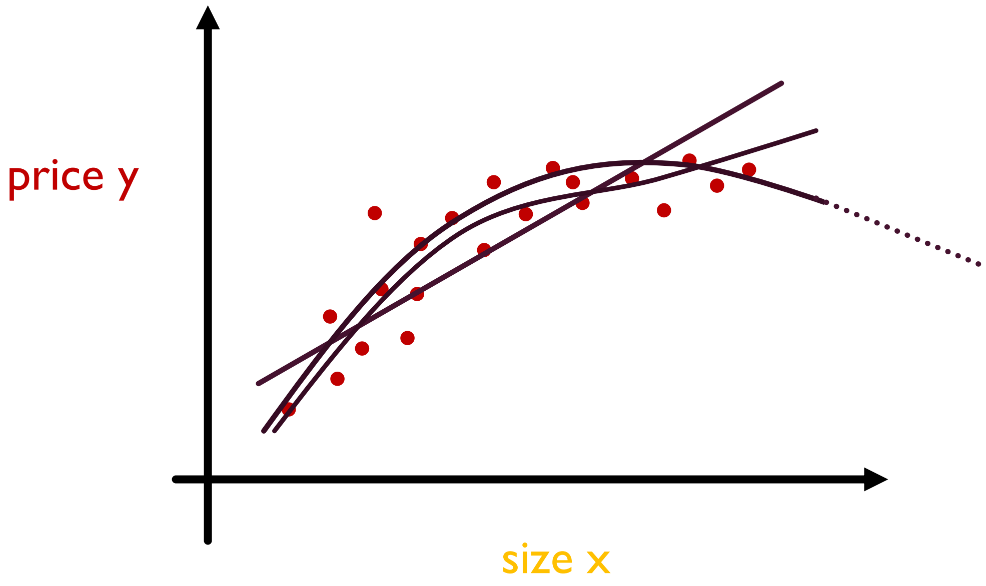
$x_3 = x_1 x_2$ new feature

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



- Feature engineering :
  - Using intuition or knowledge to design new features, by transforming or combining original features.
  - Needs human insight.

POLYNOMIAL REGRESSION



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

$$f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + b$$

$$f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + w_2 x^3 + b$$

size $\quad$ size^2 $\quad$ size^3

Normalization becomes extremely important

ACKNOWLEDGEMENT

- The material are based on Prof. Andrew Ng course on this subject.