

Week 2

Multiple Linear Regression

Multiple Features

Notation

Vectorisation vs No Vectorisation

Gradient Descent with Multiple Variables

Feature Scaling

Scaling

Mean Normalization

Z-Score Normalization

General Rule of Thumb for Feature Scaling

Feature Engineering

Polynomial Regression

Multiple Linear Regression

Multiple Features

Multiple features (variables)

	Size in feet ² x_1	Number of bedrooms x_2	Number of floors x_3	Age of home in years x_4	Price (\$) in \$1000's	$j=1...4$ $n=4$
$i=2$	2104	5	1	45	460	
	1416	3	2	40	232	
	1534	3	2	30	315	
	852	2	1	36	178	
	

$x_j = j^{th}$ 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

$\vec{x}^{(2)} = [1416 \ 3 \ 2 \ 40]$
 $x_3^{(2)} = 2$

i: row

j: column

→: Optional, used to indicate a vector

Notation

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

$\vec{w} = [w_1 \ w_2 \ w_3 \ \dots \ w_n]$ parameters of the model
 b is a number

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

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n + b$$

dot product multiple linear regression

Vectorisation vs No Vectorisation

Parameters and features

$\vec{w} = [w_1 \ w_2 \ w_3]$ $n=3$

b is a number

$\vec{x} = [x_1 \ x_2 \ x_3]$

linear algebra: count from 1

NumPy

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

code: count from 0

Without vectorization $n=100,000$

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + w_3x_3 + b$$

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



Without vectorization

$$f_{\vec{w},b}(\vec{x}) = \left(\sum_{j=1}^n w_j x_j \right) + b$$

$\sum_{j=1}^n \rightarrow j=1 \dots n$
1, 2, 3

$\text{range}(0, n) \rightarrow j=0 \dots n-1$

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



Vectorization

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

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



NB: Vectorisation uses parallelism

Gradient Descent with Multiple Variables

- One vs multiple
- For multiple, you need to update every w

Gradient descent

One feature	n features (n ≥ 2)
<p>repeat {</p> $\underline{w} = w - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$ <p style="text-align: center; margin-left: 100px;">$\frac{\partial}{\partial w} J(w, b)$</p> $b = b - \alpha \frac{1}{m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})$ <p style="text-align: center;">simultaneously update w, b</p> <p>}</p>	<p>repeat {</p> $\underline{w}_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\underline{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_1^{(i)}$ <p style="text-align: center; margin-left: 100px;">$\frac{\partial}{\partial w_1} J(\underline{w}, b)$</p> <p style="text-align: center;">⋮</p> $\underline{w}_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\underline{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_n^{(i)}$ $b = b - \alpha \frac{1}{m} \sum_{i=1}^m (f_{\underline{w},b}(\vec{x}^{(i)}) - y^{(i)})$ <p style="text-align: center;">simultaneously update w_j (for j = 1, ..., n) and b</p> <p>}</p>

Feature Scaling

- How to choose your w's?
- big x_i, small w_i, vice versa.

Feature and parameter values

$\widehat{price} = w_1 x_1 + w_2 x_2 + b$ <p style="text-align: center; margin-left: 100px;"> size # bedrooms </p>	<table border="0" style="width: 100%;"> <tr> <td style="width: 50%;"> x_1: size (feet²) range: 300 – 2,000 large </td> <td style="width: 50%;"> x_2: # bedrooms range: 0 – 5 small </td> </tr> </table>	x_1 : size (feet ²) range: 300 – 2,000 large	x_2 : # bedrooms range: 0 – 5 small
x_1 : size (feet ²) range: 300 – 2,000 large	x_2 : # bedrooms range: 0 – 5 small		
<p>House: $x_1 = 2000$, $x_2 = 5$, $price = \\$500k$ one training example</p> <p>size of the parameters w_1, w_2?</p>			
<p>$w_1 = 50, w_2 = 0.1, b = 50$</p> $\widehat{price} = \frac{50 * 2000}{100,000K} + \frac{0.1 * 5}{0.5K} + \frac{50}{50K}$ $\widehat{price} = \$100,050.5K = \$100,050,500$	<p>$w_1 = 0.1, w_2 = 50, b = 50$</p> <p style="text-align: center; margin-left: 40px;">small large</p> $\widehat{price} = \frac{0.1 * 2000k}{200K} + \frac{50 * 5}{250K} + \frac{50}{50K}$ $\widehat{price} = \$500k \text{ more reasonable}$		

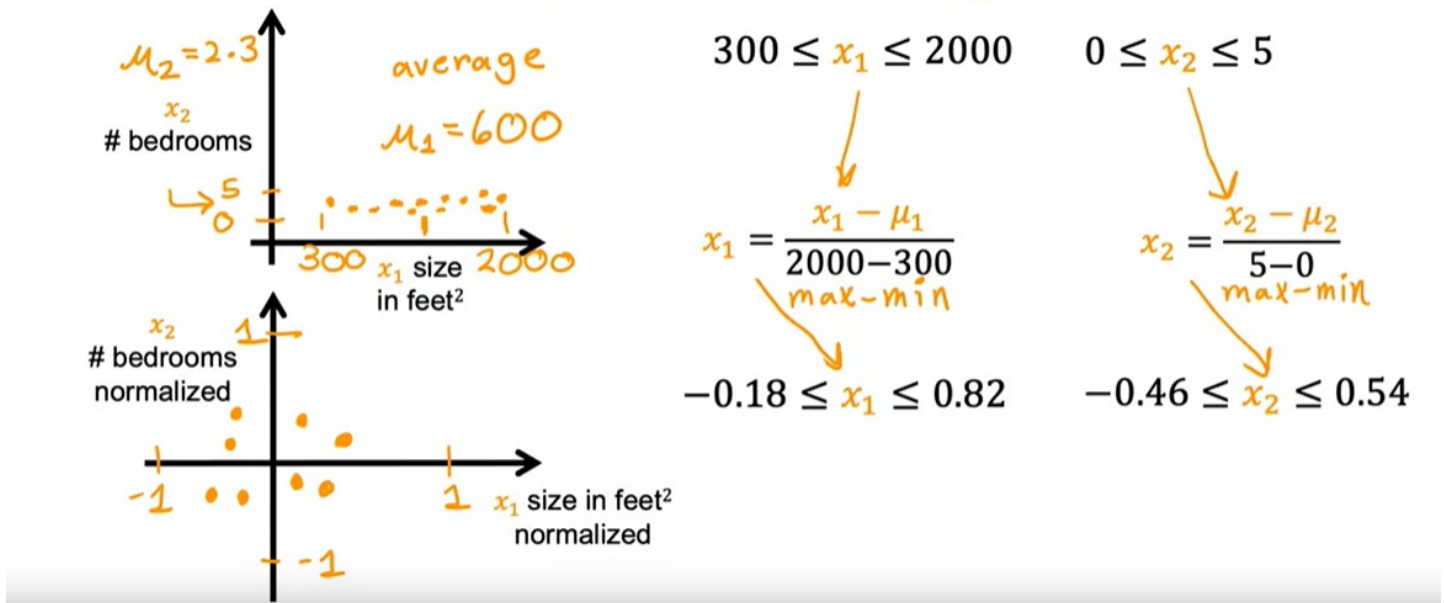
Scaling

- Divide by the max range

Mean Normalization

- Minus the avg, divide by max - min

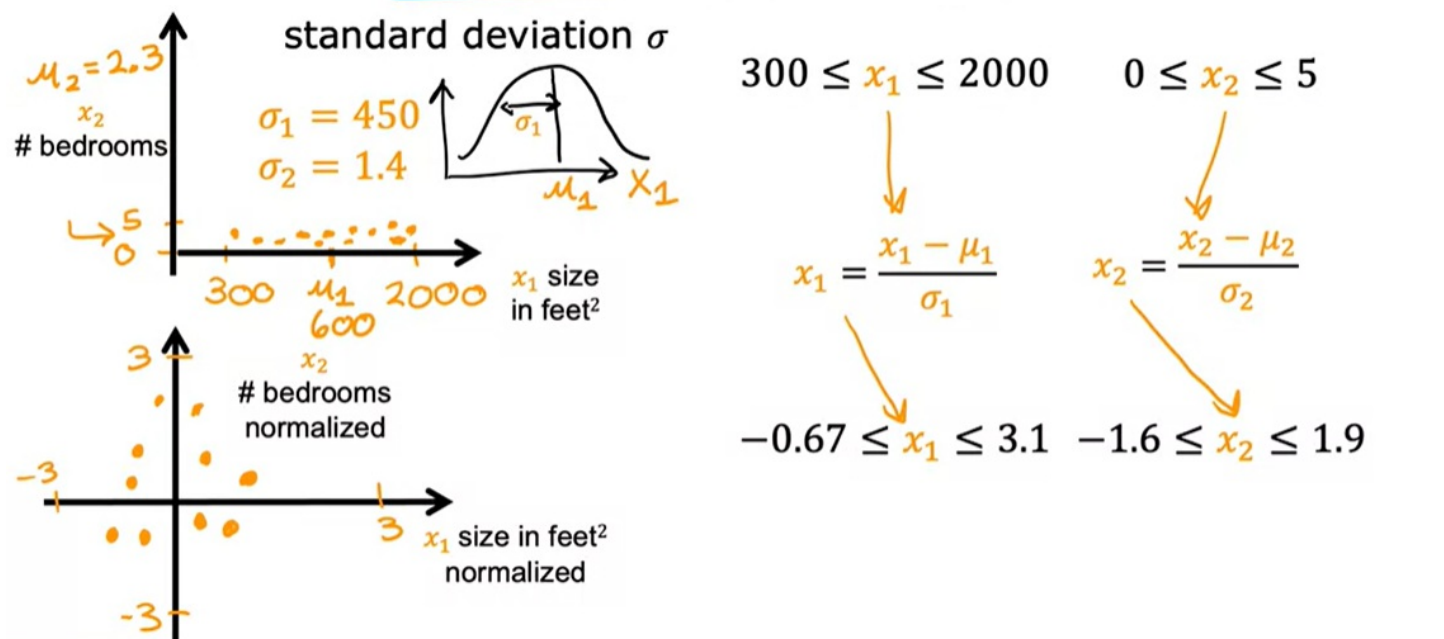
Mean normalization



Z-Score Normalization

- Minus mean divide by standard deviation

Z-score normalization



General Rule of Thumb for Feature Scaling

- Feature scaling helps to improve gradient descent speeds!

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

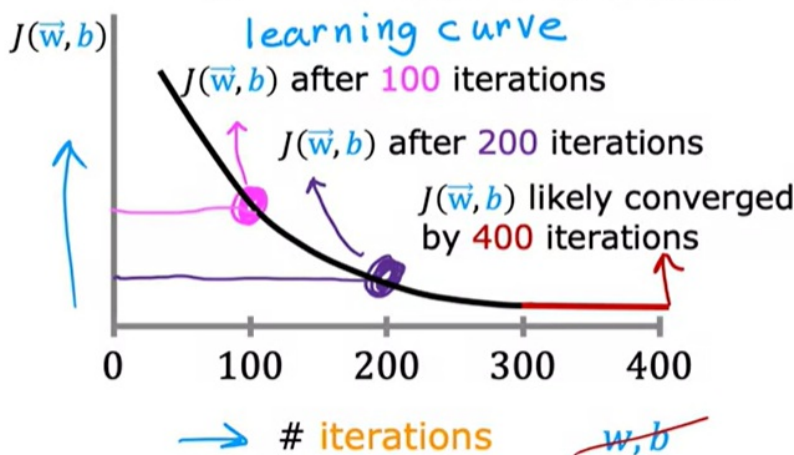
$$98.6 \leq x_5 \leq 105$$

too large → rescale

- You can also plot a graph, and ensure function J does not increase at any iterations
- Helps to ensure model is done correctly
- Always try a range of values for α

Make sure gradient descent is working correctly

objective: $\min_{\bar{w}, b} J(\bar{w}, b)$ $J(\bar{w}, b)$ 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 $J(\bar{w}, b)$ decreases by $\leq \epsilon$ in one iteration, declare convergence.

(found parameters \bar{w}, b to get close to global minimum)

Feature Engineering

- Defining a new feature by combining existing features which can be a better fit for the model.

Feature engineering

$$f_{\vec{w},b}(\vec{x}) = w_1 \underbrace{x_1}_{\text{frontage}} + w_2 \underbrace{x_2}_{\text{depth}} + b$$

$$\text{area} = \text{frontage} \times \text{depth}$$

$$x_3 = x_1 x_2$$

new feature

$$f_{\vec{w},b}(\vec{x}) = \underbrace{w_1}_{\text{frontage}} x_1 + \underbrace{w_2}_{\text{depth}} x_2 + \underbrace{w_3}_{\text{area}} x_3 + b$$



Feature engineering:
Using **intuition** to design
new features, by
transforming or combining
original features.

Polynomial Regression

Polynomial regression

