CS395: Selected CS1 (Introduction to Machine Learning)

Associate Prof. Wessam El-Behaidy

Fall 2021

References:

https://www.coursera.org/learn/machine-learning (Andrew Ng)

Machine learning A to Z: **Kirill Eremenko ©superdatascience**

Linear Algebra review: Matrices and vectors

(Self review)

Vector: Special case of the matrix

n x 1 matrix

$\mathbb{R}^4$

$$y = \begin{bmatrix} 4 \\ 50 \\ 3 \\ 25 \end{bmatrix} \quad 4 - dimentional\ vector\ n = 4$$

$y_i = i^{\text{th}}$ element.

$y_2 = 50$

1-indexed vs 0-indexed:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \qquad y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Matrix Addition

$$\begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 11 \\ 3 & 7 & 12 \end{bmatrix} + \begin{bmatrix} 10 & 0 & 5 \\ 2 & 4 & 3 \\ 5 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 11 & 5 & 14 \\ 4 & 10 & 14 \\ 8 & 9 & 14 \end{bmatrix}$$

3 x 3   3 x 3   3 x 3

We can add only two matrices that are of **the same dimensions.**

So this example is a 3 x 3 matrix, the result in same dimension 3x3.

Addition and subtraction are **element-wise**, so you simply add or subtract each corresponding element

Scalar Multiplication

$$3 \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{x3}$$

2 x 2

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} / 4 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ x } \tfrac{1}{4} = \begin{bmatrix} 1/4 & 1/2 \\ 3/4 & 1 \end{bmatrix}$$

Combination of Operands

Scaler Multiplication

$$3 \times \begin{bmatrix}1\\4\\2\end{bmatrix} + \begin{bmatrix}0\\0\\5\end{bmatrix} - \begin{bmatrix}3\\0\\2\end{bmatrix}/3 =$$

Scaler Multiplication

$$\begin{bmatrix}3\\12\\6\end{bmatrix} + \begin{bmatrix}0\\0\\5\end{bmatrix} - \begin{bmatrix}1\\0\\\frac{2}{3}\end{bmatrix} =$$

$$\begin{bmatrix}2\\12\\10\frac{1}{3}\end{bmatrix}$$

Matrix Vector Multiplication

$$\begin{bmatrix} 5 & 1 \\ 2 & 6 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 15 \\ 34 \\ 26 \end{bmatrix}$$

A x x = y

$$\begin{bmatrix} 5 & 1 \\ 2 & 6 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 15 \\ 34 \\ 26 \end{bmatrix}$$

3 × 2 2 × **1** **3** × **1 matrix → 3 dim vector**

To get y_i, $multiply\ A's\ i^{th}\ row\ with\ elements\ of\ vector\ x, and\ add\ them\ up.$

$5 \times 2 + 1 \times 5 = 15$

$2 \times 2 + 6 \times 5 = 34$

$3 \times 2 + 4 \times 5 = 26$

An **m x n matrix** multiplied by an **n x 1 vector** results in an **m x 1 vector**.

Matrix Multiplication

$$\overset{\substack{\text{A}\\ \text{2x3}}}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}} \times \overset{\substack{\text{B}\\ \text{3x2}}}{\begin{bmatrix} 1 & 2 \\ 0 & 5 \\ 2 & 1 \end{bmatrix}} = \overset{\substack{\text{C}\\ \text{2x2}}}{\begin{bmatrix} 7 & 15 \\ 16 & 39 \end{bmatrix}}$$

C_{11}=1+0+6 = 7

C_{21} =4 + 0+ 12 = 16

C_{12} = 2+10+3= 15

C_{22} =8+25+6= 39

An **m x n matrix** multiplied by an **n x o matrix** results in an **m x o** matrix. In the above example, a 2 x 3 matrix times a 3 x 2 matrix resulted in a 2 x 2 matrix.

Matrix Multiplication Properties

- Matrices are **not commutative**: $A \times B \neq B \times A$
- Matrices are **associative**: $(A \times B) \times C = A \times (B \times C)$
- Identity Matrix $I, or\ I_{n \times n}$ for any A, $A.\ I = I.\ A = A$

- $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Inverse and Transpose

- The **inverse** of a matrix A $(m \times m)$ is denotedA^{-1} . Multiplying by the inverse results in the identity matrix.

$$A \times A^{-1} = 1$$

- A non square matrix does not have an inverse matrix.

$$\begin{bmatrix} 3 & 4 \\ 2 & 16 \end{bmatrix} \times \begin{bmatrix} 0.4 & -0.1 \\ -0.05 & 0.075 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Transpose

- The **transposition** $B_{ij} = A_{ji}$ of a matrix is like rotating the matrix 90° in clockwise direction and then reversing it.
- $A_{ij} = A_{ij}^T$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

m x n matrix

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

nx m matrix

Numerical Example on Linear Regression with one variable

Recap:

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

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

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Gradient Descent

- We make steps down the cost function in the direction with the steepest descent, and the size of each step is determined by the parameter **α**, which is called the **learning rate.**
- The gradient descent algorithm is:

Repeat until **convergence** {

$$\theta_j := \theta_j - \underbrace{\alpha}_{\text{Learning rate (step size)}} \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

}

Gradient Descent for Linear Regression

- When specifically applied to the case of **linear regression**, a new form of the gradient descent equation can be derived. We can substitute our actual cost function and our actual hypothesis function and modify the equation to:

Start by initializing the parameters θ_0, θ_1 **randomly**

$$\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \qquad \frac{\partial}{\partial_{\theta_0}} J(\theta_0, \theta_1) \\
&\quad \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)} \qquad \frac{\partial}{\partial_{\theta_1}} J(\theta_0, \theta_1) \\
&\}
\end{aligned}$$

Should be done simultaneously

Hypothesis using Matrix Product

- $h_\theta(x) = \theta_0 + \theta_1 x$

$x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \in \mathbb{R}^2$

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \in \mathbb{R}^2$

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

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 = [\theta_0 \theta_1] \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \theta^T x$$

Linear Regression with one variable Example:

- Suppose we are given a dataset, and we need to find the hypothesis of linear regression:

Experience (X)	Salary (y) (in lakhs)
2	3
6	10
5	4
7	13

https://www.geeksforgeeks.org/mathematical-explanation-for-linear-regression-working/

Experience (X)	Salary (y) (in lakhs)
2	3
6	10
5	4
7	13

Iteration 1

a) In the start, θ_0 and θ_1 values are randomly chosen.

- Let us suppose, $\theta_0 = 0$ and $\theta_1 = 0$.

b) **Predicted values after iteration 1 with Linear regression hypothesis.**

$$h_\theta = \theta_0 x_0 + \theta_1 x_1 = [\theta_0 \theta_1] \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

$$h_\theta = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix} \begin{bmatrix} \boxed{\begin{matrix} x_0 \\ x_1 \end{matrix}} & \begin{matrix} x_0 \\ x_2 \end{matrix} & \begin{matrix} x_0 \\ x_3 \end{matrix} & \begin{matrix} x_0 \\ x_4 \end{matrix} \end{bmatrix}$$

One training example

$$= \begin{bmatrix} 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 6 & 5 & 7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

Experience (X)	Salary (y) (in lakhs)
2	3
6	10
5	4
7	13

Iteration 1

c) Cost function Error

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}[h_\theta(x_i) - y_i]^2$$

$$= \frac{1}{2*4}[(0-3)^2+(0-10)^2+(0-4)^2+(0-13)^2]$$

$$=\frac{1}{8}[9+100+16+169]$$

= 36.75

Experience (X)	Salary (y) (in lakhs)
2	3
6	10
5	4
7	13

Iteration 1

d) **Gradient Descent – Updating $\boldsymbol{\theta_0}$ value**

Here, j=0

$$\theta_0 = \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)$$

$$= 0 - \frac{0.001}{4}[(0-3) + (0-10) + (0-4) + (0-13)]$$

$$= - \frac{0.001}{4}[(-3) + (-10) + (-4) + (-13)]$$

$$=- \frac{0.001}{4}[-30]$$

= 0.0075

Algorithm GD for linear regression:

repeat until convergence {

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

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

}

Experience (X)	Salary (y) (in lakhs)
2	3
6	10
5	4
7	13

Iteration 1

e) **Gradient Descent – Updating $\boldsymbol{\theta_1}$ value**

Here, j=1

$$\theta_1 = \theta_1 - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i).\,x_i$$

$$= 0 - \frac{0.001}{4}[(0-3)2 + (0-10)6 + (0-4)5 + (0-13)7]$$

$$= -\ \frac{0.001}{4}[(-6) + (-60) + (-20) + (-91)]$$

$$= -\frac{0.001}{4}[-177]$$

= 0.04425

Algorithm GD for linear regression:

repeat until convergence {

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

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

}

Iteration 2

Iteration 2 – $\boldsymbol{\theta_0 = 0.0075 \ \ and \ \ \theta_1 = 0.04425}$
Predicting values after iteration 1 with linear regression hypothesis

$$h_\theta = \begin{bmatrix}\theta_0 & \theta_1\end{bmatrix}\begin{bmatrix}x_0 & x_0 & x_0 & x_0 \\ x_1 & x_2 & x_3 & x_4\end{bmatrix}$$

$$= \begin{bmatrix}0.0075 & 0.04425\end{bmatrix}\begin{bmatrix}1 & 1 & 1 & 1 \\ 2 & 6 & 5 & 7\end{bmatrix}$$

$$= \begin{bmatrix}0.096 & 0.273 & 0.22875 & 0.31725\end{bmatrix}$$

Iteration 2

- Similar to iteration no. 1 performed above
  - we will again calculate Cost function and update θ_j values using Gradient Descent.
  - We will keep on iterating until Cost function doesn't reduce further **(converge)**. At that point, model achieves best θ values.
  - Using these θ values in the model hypothesis will give the best prediction results.

Testing:

If we have the house sizes and the hypotheses (h_θ), and you need to predict the prices of these houses

$house\ sizes$:

2100

3150

4152

1108

$$h_\theta(x) = -40 + 0.25\,x$$

Matrix **4x2**:

Prediction **4x1 matrix**:

$$\begin{bmatrix} 1 & 2100 \\ 1 & 3150 \\ 1 & 4152 \\ 1 & 1108 \end{bmatrix} \times \begin{bmatrix} -40 \\ 0.25 \end{bmatrix} = \begin{array}{c} h_\theta(2100) \\ \begin{bmatrix} -40 \times 1 + 0.25 \times 2100 \\ -40 \times 1 + 0.25 \times 3150 \\ -40 \times 1 + 0.25 \times 4152 \\ -40 \times 1 + 0.25 \times 1108 \end{bmatrix} \end{array}$$

2x1

Testing:

If we have the house sizes and more than one hypotheses (h_θ):

$house\ sizes$:

2100
3150
4152
1108

$$h_\theta(x) = \boxed{-40 + 0.25\,x}$$
$$h_\theta(x) = \boxed{200 + 0.1\,x}$$
$$h_\theta(x) = \boxed{-150 + 0.4\,x}$$

m x n matrix

Matrix **4x2**:

$$\begin{bmatrix} 1 & 2100 \\ 1 & 3150 \\ 1 & 4152 \\ 1 & 1108 \end{bmatrix} \times \begin{bmatrix} -40 & 200 & -150 \\ 0.25 & 0.1 & 0.4 \end{bmatrix} =$$

n x o matrix

2x3

m x o matrix

Prediction **4x3 matrix**:

$$\begin{bmatrix} 485 & 410 & 690 \\ 748 & 515 & 1110 \\ 998 & 615 & 1511 \\ 237 & 311 & 293 \end{bmatrix}$$

Linear regression with multiple variables

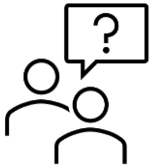
Multiple variables (Features)

- Linear regression with multiple variables is also known as "multivariate linear regression".

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$\boldsymbol{Size\ (feet)^2}$ $\boldsymbol{x}$	$\boldsymbol{Price\ (\$1000)}$ $\boldsymbol{y}$
2104	460
1416	232
1534	315
852	178

Could the prediction be more accurate if we add #of rooms?



Multiple variables (Features)

$\boldsymbol{Size\ (feet)^2}$ $\boldsymbol{x_1}$	$\boldsymbol{Number\ of\ bedrooms}$ $\boldsymbol{x_2}$	$\boldsymbol{Number\ of\ floors}$ $\boldsymbol{x_3}$	$\boldsymbol{Age\ of\ home\ (years)}$ $\boldsymbol{x_4}$	$\boldsymbol{Price\ (\$1000)}$ $\boldsymbol{y}$
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

$\boldsymbol{m}$, *number of training examples*

$\mathbf{n}$, number of features

$x^{(i)}$, the input (features) of the i^{th} training example

$x_j^{(i)}$, value of feature *j* in the i^{th} training example

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$x_4^{(3)} = 30$$

Hypothesis

Size $(feet)^2$ x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

$$h_\theta(x) = 80 \quad 0.1x_1 \quad 0.01x_2 \quad 3x_3 \quad 2x_4$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

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

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ . \\ . \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ . \\ . \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

Hypothesis for Multiple Features

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ . \\ . \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ . \\ . \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$$h_\theta = \theta^T x = [\, \theta_0 \quad \theta_1 \ldots\ldots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ x_n \end{bmatrix}$$

GD. for Multiple Variables

$x_0 = 1$

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

$Parameters: \theta_0, \quad \theta_1, \ldots\ldots, \quad \theta_n$ $\qquad \theta\ is\ n+1\ dimension\ vector$

$$Cost\ function: J(\theta_0, \theta_1, \ldots, \theta_n) = J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \left(\left(\sum_{j=0}^{n} \theta_j x_j^{(i)}\right) - y^{(i)}\right)^2$$

GD for Multiple Variables

The gradient descent equation itself is generally the same form; we just have to repeat it for our 'n' features

$repeat\ until\ convergence : \{$

$$\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_j} J(\theta),$$ simultaneously updates for every $j = 0,1,2,\dots,n$

$\}$

Previously (n=1):

Repeat $\{$

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

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

(simultaneously update θ_0, θ_1)

$\}$

For (n>1):

$\boldsymbol{repeat\ until\ convergence} : \{$

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

, simultaneously updates θ_j for every $j = 0,1,2,\dots,n$

$\}$

Gradient Descent in Practice I - Feature Scaling

- We can speed up gradient descent by having each of our input values in roughly the **same range**.
- Because θ will:
  - descend quickly on small ranges
  - descend slowly on large ranges, and
  - oscillate inefficiently down to the optimum when the variables are very uneven.
- The way to prevent this is to modify the ranges of our input variables so that they are all roughly the same.

Make sure features are on same scale.

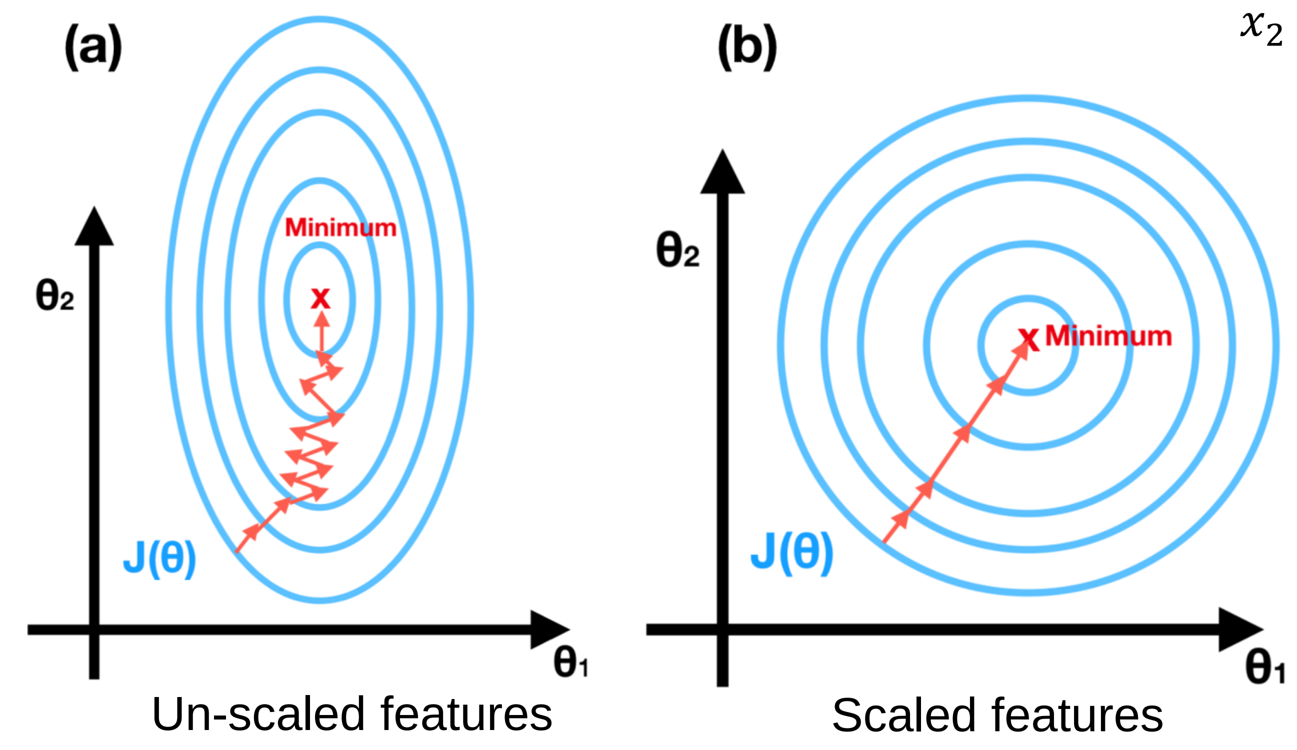
Example:

$x_1 = size(0 - 2000\ feet^2)$

$x_2 = number\ of\ bedrooms\ (1 - 5)$

$$x_1 = \frac{size\ (feet)^2}{2000} \qquad 0 \le x_1 \le 1$$

$$x_2 = \frac{No\ of\ bedrooms}{5} \qquad 0 \le x_2 \le 1$$



https://www.blog.nipunarora.net/ml_multi_variate_linear_regression/

Get every feature into approximately $-1 \leq x_j \leq 1$ range

(**Do not apply to $x_0 = 1$**)

Rule a thumb regarding acceptable ranges

- -3 to +3 is generally fine - any bigger bad
- -1/3 to +1/3 is ok - any smaller bad

Mean Normalization

- Replace x_i with μ_i to make features have approximately zero mean (**<u>Do not apply to $x_0 = 1$</u>**)

$$x_i = \frac{x_i - \mu_i}{s_i},$$

$$x_1 = \frac{size - 1000}{2000}$$

$\mu_i,$ $\quad$ *is the average value of x in the training dataset*

s_i , $\quad$ is the range of values (max - min) or the standard deviation.

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

σ = population standard deviation

N = the size of the population

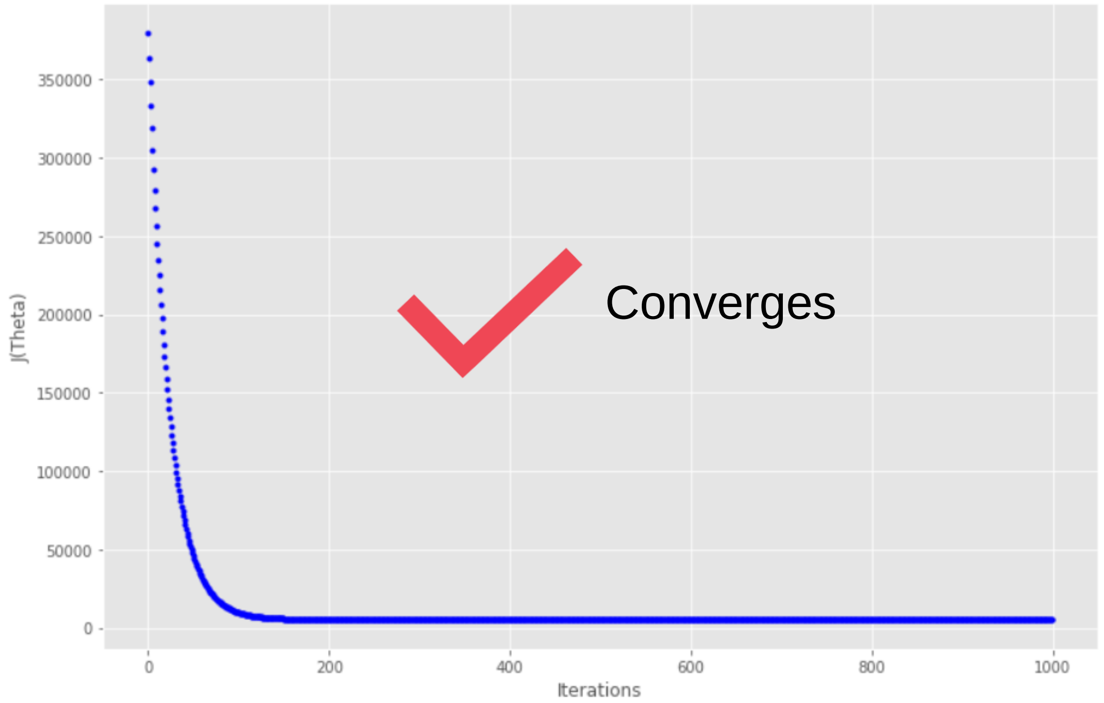
x_i = each value from the population

μ = the population mean

Gradient Descent in Practice II - Learning Rate

- **Debugging gradient descent.**
- Make a plot with *number of iterations* of gradient descent on the $x - axis$ and the cost function $J(\theta)$ on the $y - axis$.
- $J(\theta)$ should decrease after each iteration else decrease α

Automatic convergence test. Declare convergence if $J(\theta)$ decreases by less than ε in one iteration, where ε is some small value such as 10^{-3}. However in practice it's difficult to choose this threshold value.



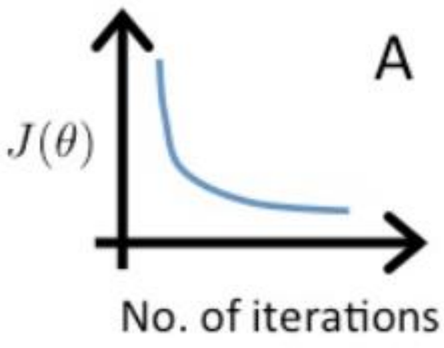
Gradient Descent in Practice II - Learning Rate

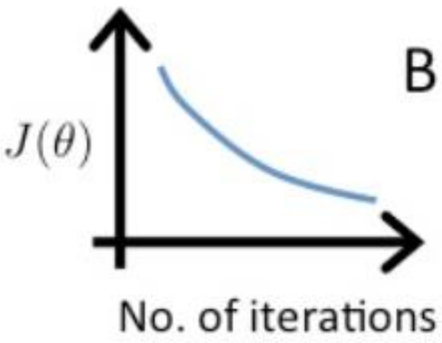
It has been proven that if learning rate α is sufficiently small, then $J(\theta)$ will decrease on every iteration. When Gradient Descent can't decrease the cost-function anymore and remains more or less on the same level, we say it has **converged**.

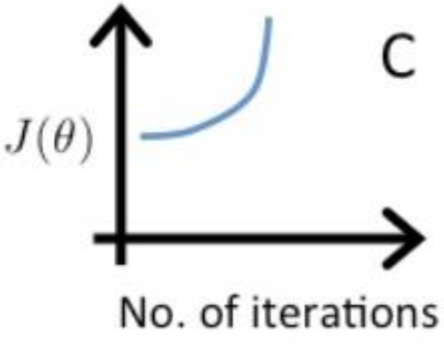
Note:
If you see in the plot that your learning curve is just going up and down, without really reaching a lower point, you also should try to decrease the learning rate.

If α is too small:
slow convergence.

If α is too high:
the cost function is increasing







Features and Polynomial Regression / choice of features

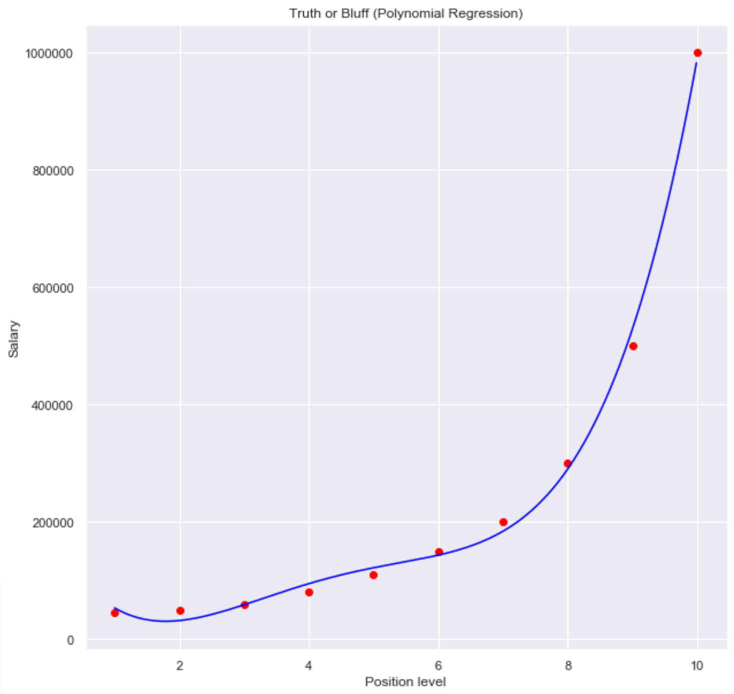
- We can improve our features and the form of our hypothesis function.
- We can **combine** multiple features into one. For example, we can combine x_1 and x_2 into a new feature x_3 by taking $x_1 . x_2$

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x_1$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 (x_1)^2 + \theta_3 (x_1)^3$$

$$h_\theta(x) = \theta_0 + \theta_1 (x_1) + \theta_2 \sqrt{x_1}$$

square root function



feature scaling becomes very important.

Thanks