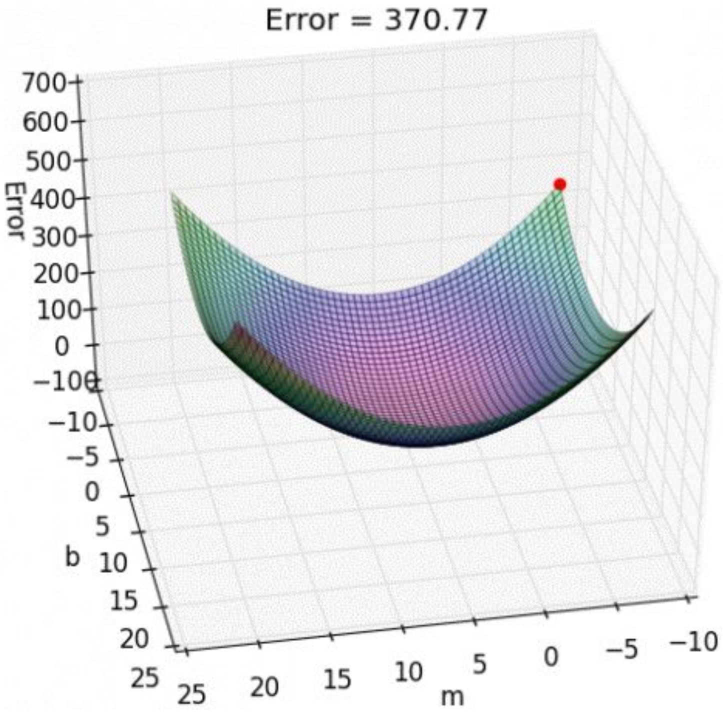
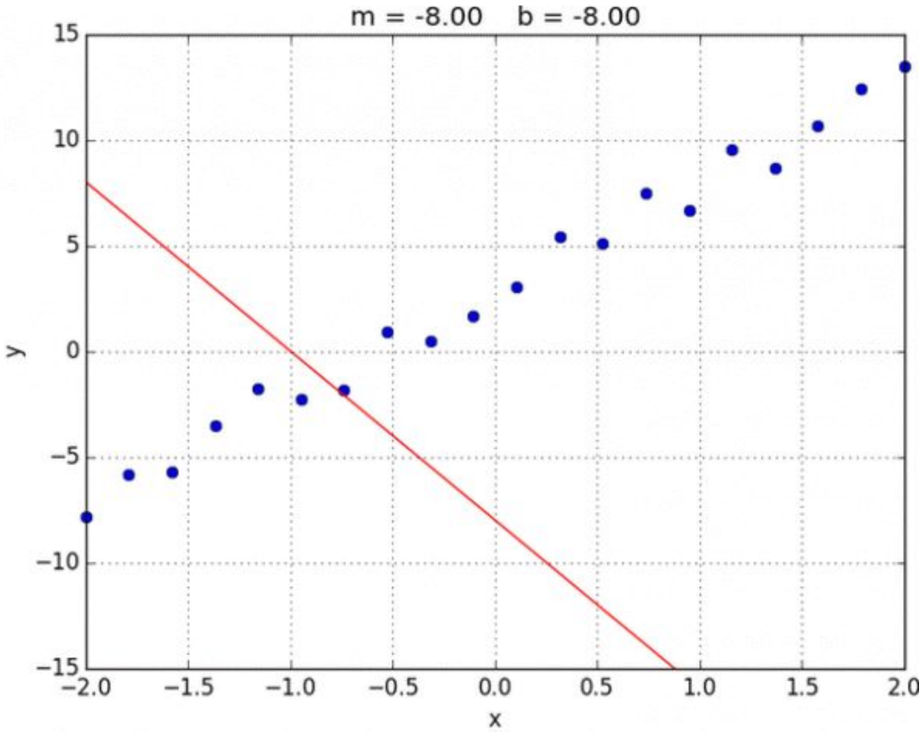
Recall from last time ...

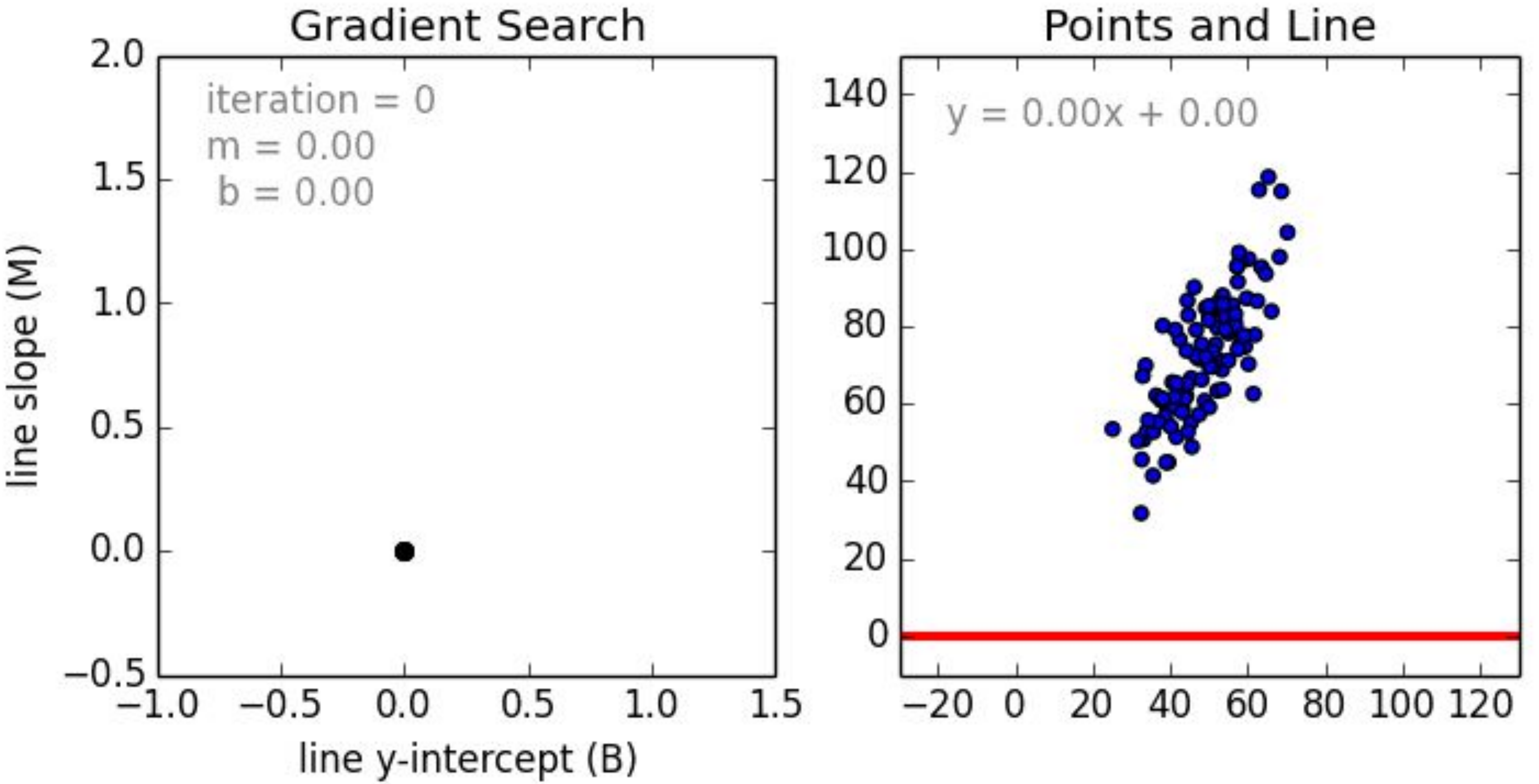
$$y = b + mx$$





Credit: https://alykhantejani.github.io/a-brief-introduction-to-gradient-descent/

$$h_\theta(x) = \theta_0 + \theta_1 x \quad \Rightarrow \quad y = b + mx$$



Credit: https://github.com/mattnedrich/GradientDescentExample/raw/master/gradient_descent_example.gif

“Batch” Gradient Descent

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

“Batch” Gradient Descent

repeat until convergence {

$$\left.\begin{aligned} \theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \\ \theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\, x^{(i)} \end{aligned}\right\} \text{update } \theta_0 \text{ and } \theta_1 \text{ simultaneously}$$

}

“Batch” Gradient Descent

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

```
for i in range(nb_epochs):
  params_grad = evaluate_gradient(loss_function,data,params)
  params = params - learning_rate * params_grad
```

“Batch” Gradient Descent

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

```
for i in range(nb_epochs):
  params_grad = evaluate_gradient(loss_function,data,params)
  params = params - learning_rate * params_grad
```

Epochs: One epoch is usually defined to be **ONE** complete run through **ALL** of the training data.

Epochs: One epoch is usually defined to be **ONE** complete run through **ALL** of the training data.

Batch Size: Total number of training examples present in a **SINGLE** batch.

Epochs: One epoch is usually defined to be **ONE** complete run through **ALL** of the training data.

Batch Size: Total number of training examples present in a **SINGLE** batch.

Note: Batch size and number of batches are two different things.

Epochs: One epoch is usually defined to be **ONE** complete run through **ALL** of the training data.

Batch Size: Total number of training examples present in a **SINGLE** batch.

Iterations: The number of batches needed to complete **ONE** epoch.

Epochs: One epoch is usually defined to be **ONE** complete run through **ALL** of the training data.

Batch Size: Total number of training examples present in a **SINGLE** batch.

Iterations: The number of batches needed to complete **ONE** epoch.

Note: The number of batches is equal to number of iterations for one epoch.

Epochs & Batch size & Iterations

Let's say we have 10,000 training examples that we are going to use.

We can divide the dataset of 10,000 examples into **batches of 16** then it will take **625 iterations** to complete **1 epoch**.

Stochastic Gradient Descent

Each step of gradient descent uses **one training example**.

repeat until convergence {

for $i \coloneqq 1, \ldots, m$ {

$$\theta_0 := \theta_0 - \alpha(h_\theta(x^{(i)}) - y^{(i)})$$

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

}

}

Stochastic Gradient Descent

Each step of gradient descent uses **one training example**.

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function,example,params)
    params = params - learning_rate * params_grad
```

Mini-batch Gradient Descent

Each step of gradient descent uses **b training examples**.

Say $b = 10$, $m = 1000$.
repeat until convergence {

for $i \coloneqq 1, 11, 21\ldots, 991$ {

$$\theta_0 := \theta_0 - \alpha \frac{1}{10} \sum_{i=k}^{i+9} (h_\theta(x^{(k)}) - y^{(k)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{10} \sum_{i=k}^{i+9} (h_\theta(x^{(k)}) - y^{(k)})\, x^{(k)}$$

} }

Mini-batch Gradient Descent

Each step of gradient descent uses b **training examples**.

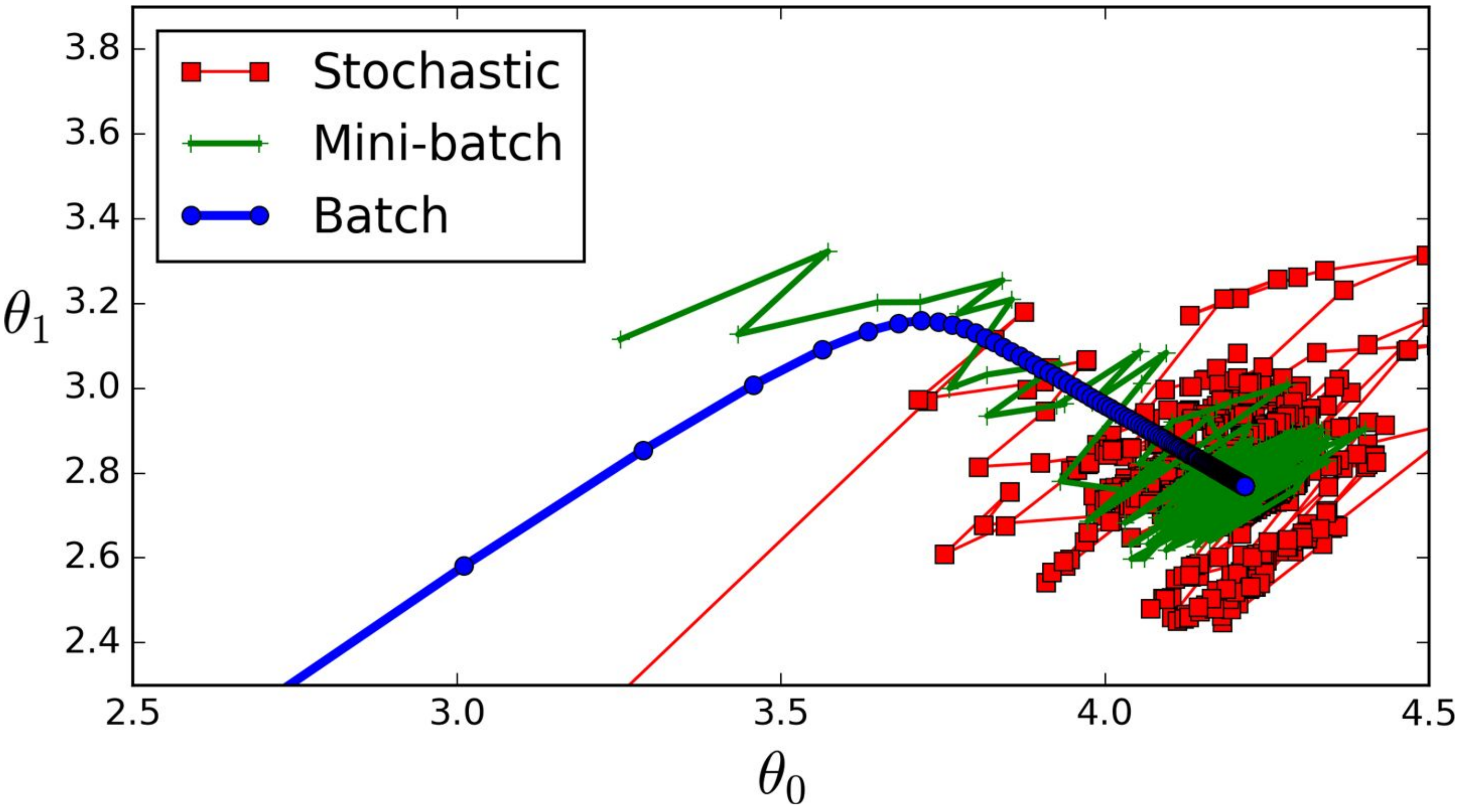
```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for batch in get_batches(data,batch_size=16):
    params_grad = evaluate_gradient(loss_function,batch,params)
    params = params - learning_rate * params_grad
```

```python
for i in range(nb_epochs):
  params_grad = evaluate_gradient(loss_function,data,params)
  params = params - learning_rate * params_grad
```

```python
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function,example,params)
    params = params - learning_rate * params_grad
```

```python
for i in range(nb_epochs):
  np.random.shuffle(data)
  for batch in get_batches(data,batch_size=16):
    params_grad = evaluate_gradient(loss_function,batch,params)
    params = params - learning_rate * params_grad
```

Batch vs. Stochastic vs. Mini-batch



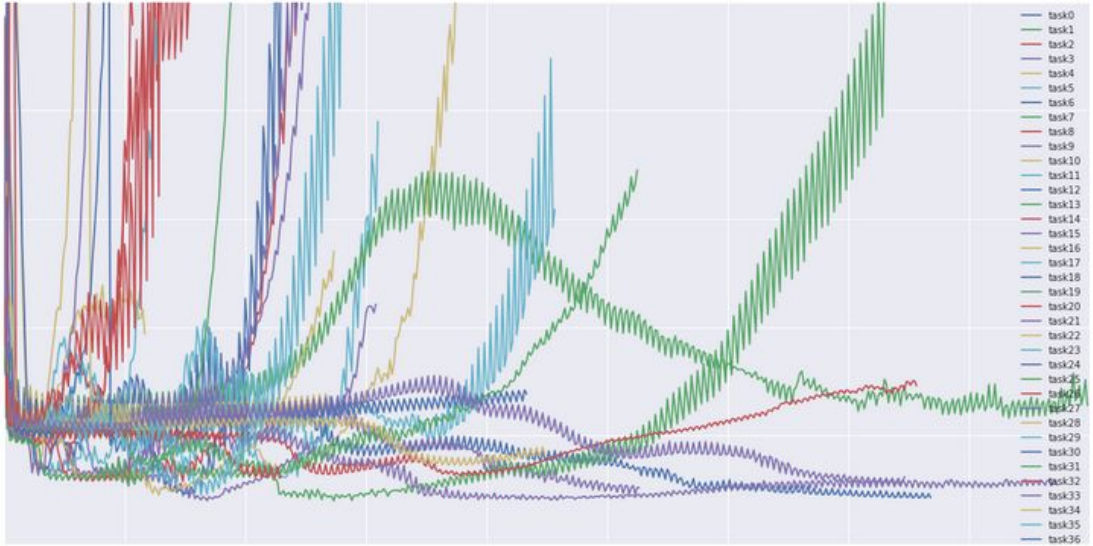
http://ruder.io/optimizing-gradient-descent

Sebastian Ruder

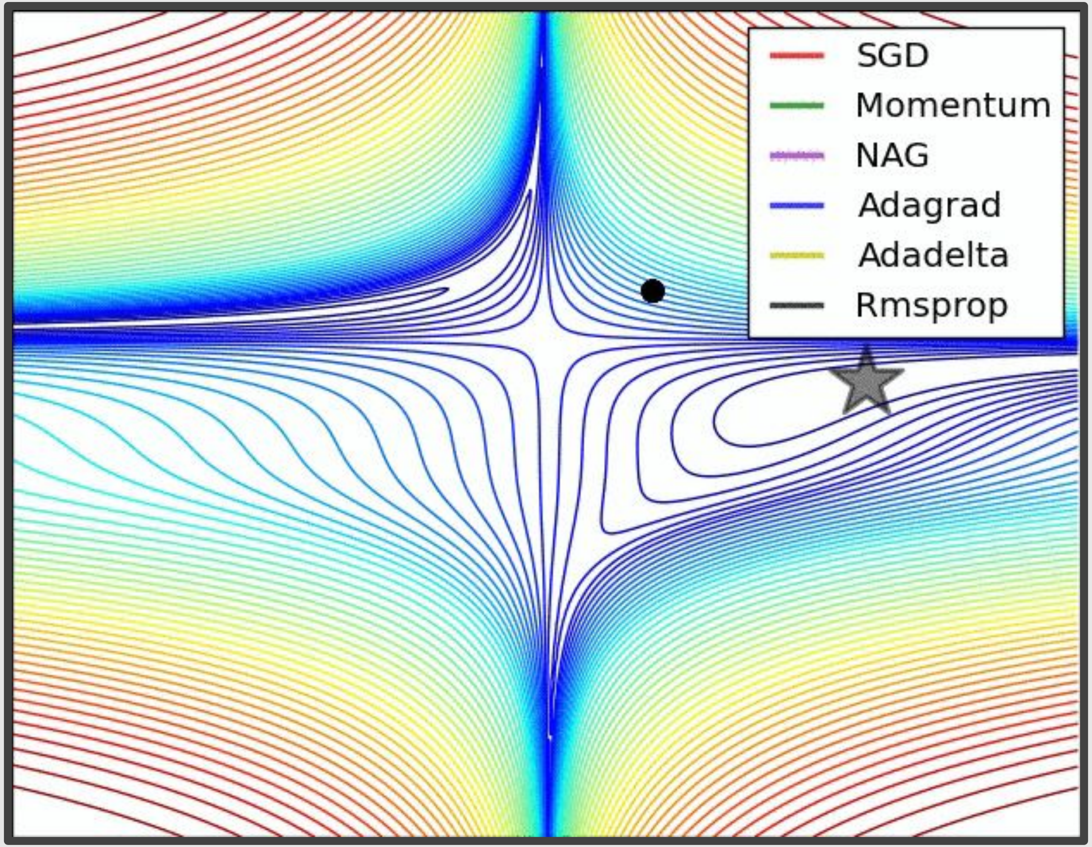
I'm a PhD student in Natural Language Processing and a research scientist at AYLIEN. I blog about Machine Learning, Deep Learning, NLP, and startups.

Blog
About
Papers
News
Newsletter
FAQ
Progress

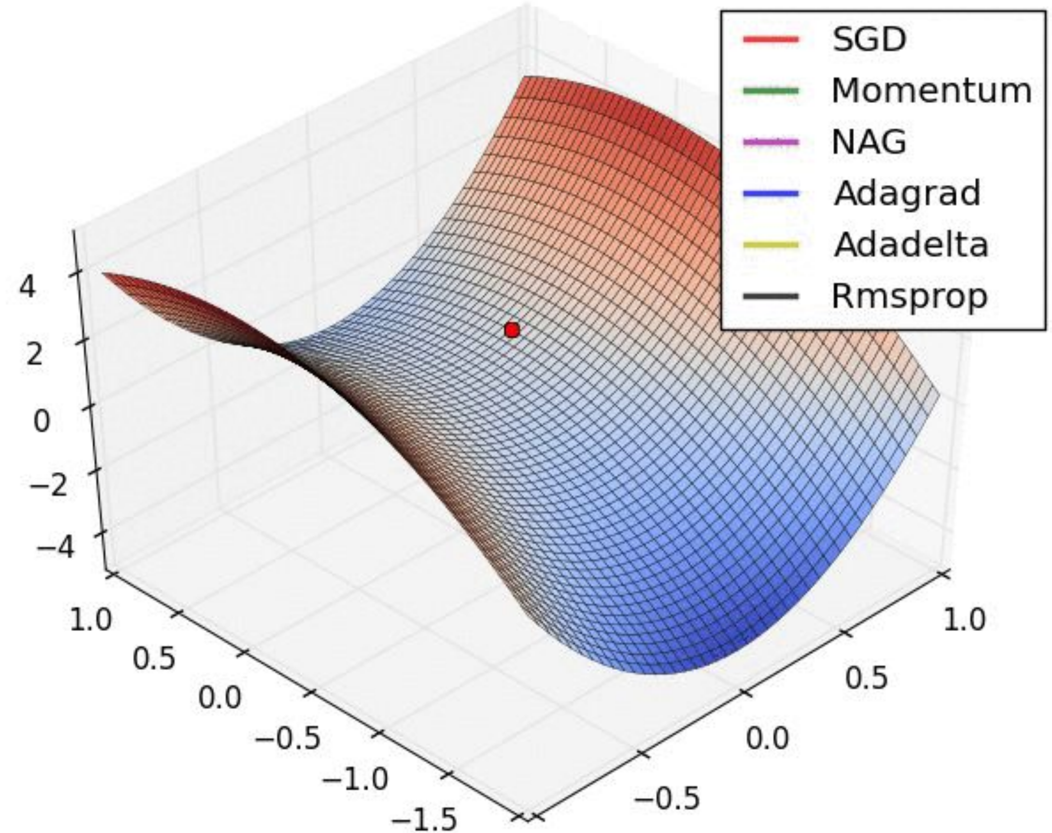
19 Jan 2016 in OPTIMIZATION DEEP LEARNING SGD ~ 25 min read.



An overview of gradient descent optimization algorithms



Credit: Alec Radford: https://i.imgur.com/pD0hWu5.gif



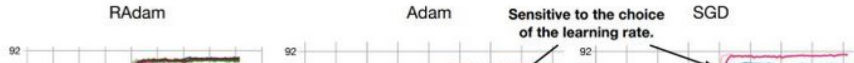
Credit: Alec Radford: https://i.imgur.com/2dKCQHh.gif https://medium.com/@lessw/new-state-of-the-art-ai-optimizer-rectified-adam-radam-5d854730807b

Medium | Machine Learning

New State of the Art AI Optimizer: Rectified Adam (RAdam). Improve your AI accuracy instantly versus Adam, and why it works.

Less Wright Follow
Aug 15 · 5 min read ★

A new paper by Liu, Jian, He et al introduces **RAdam**, or “Rectified Adam”. It’s a new variation of the classic Adam optimizer that provides an automated, dynamic adjustment to the adaptive learning rate based on their detailed study into the effects of variance and momentum during training. RAdam holds the promise of immediately improving every AI architecture compared to vanilla Adam as a result:



Linear Regression

Machine Learning

(Largely based on slides from Andrew Ng)

Prof. Sandra Avila

Institute of Computing (IC/Unicamp)

MC886, August 19, 2019

Today's Agenda

- Linear Regression with One Variable
  - Model Representation
  - Cost Function
  - Gradient Descent
- Linear Regression with Multiple Variables
  - Gradient Descent for Multiple Variables
  - **Feature Scaling**
  - **Learning Rate**
  - **Features and Polynomial Regression**
  - **Normal Equation**

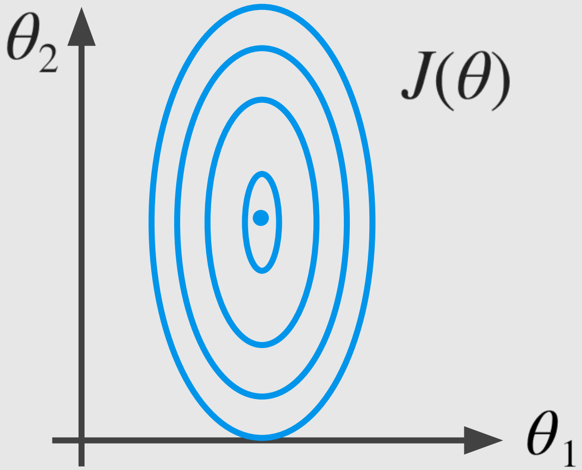
Feature Scaling

Feature Scaling

Idea: Make sure features are on similar scale.

E.g. x_1= size (0–2000 feet^2)

x_2= number of bedrooms (1–5)

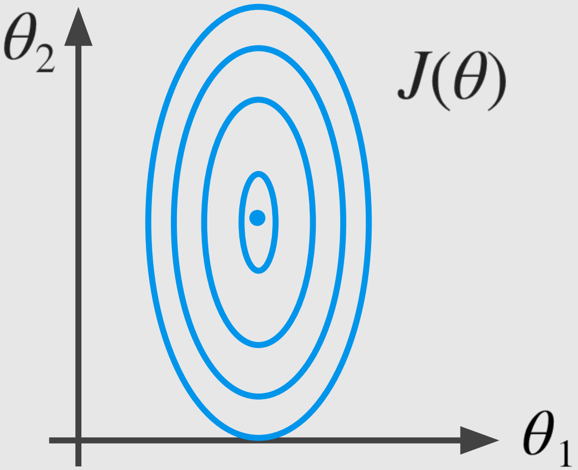


Feature Scaling

Idea: Make sure features are on similar scale.

E.g. x_1 = size (0–2000 feet^2)

x_2 = number of bedrooms (1–5)



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

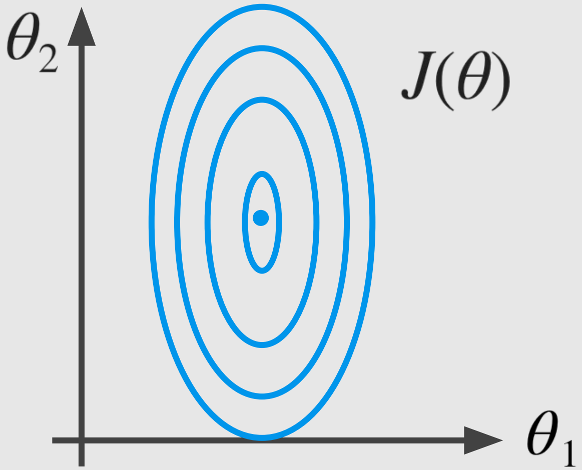
$$x_2 = \frac{\text{number of bedrooms}}{5}$$

Feature Scaling

Idea: Make sure features are on similar scale.

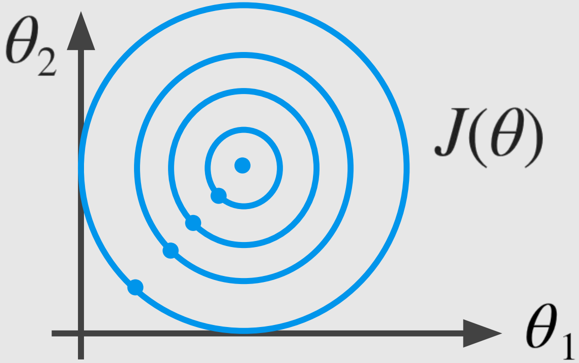
E.g. $x_1 =$ size (0–2000 feet^2)

$x_2 =$ 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.

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. $x_1 = \dfrac{\text{size} - 1000}{2000}$ $\Rightarrow$ $-0.5 \leq x_1 \leq 0.5$

$x_2 = \dfrac{\text{\#bedrooms} - 2.5}{5}$ $\Rightarrow$ $-0.5 \leq x_2 \leq 0.5$

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. $x_1 = \frac{\text{size} - 1000}{2000}$ ➡ $-0.5 \leq x_1 \leq 0.5$

$x_2 = \frac{\#\text{bedrooms} - 2.5}{5}$ ➡ $-0.5 \leq x_2 \leq 0.5$

$$x_1 = \frac{x_1 - \mu_1}{s_1} \qquad x_2 = \frac{x_2 - \mu_2}{s_2}$$

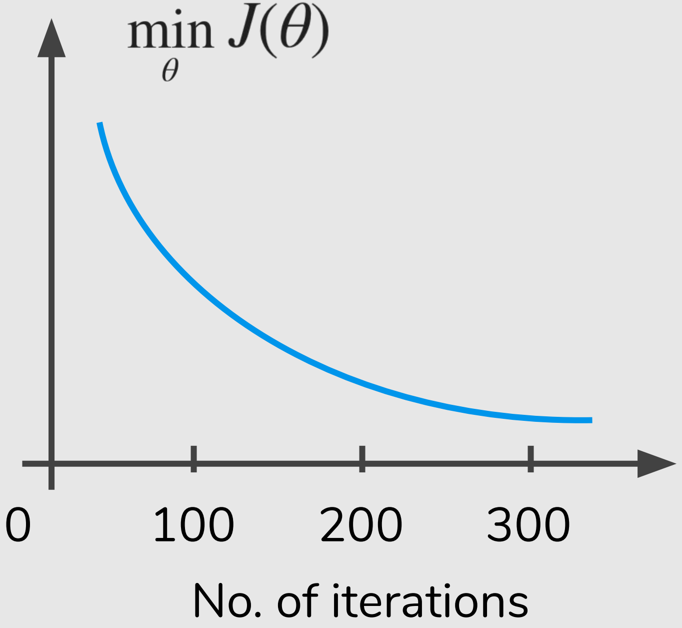
Learning Rate

Gradient Descent

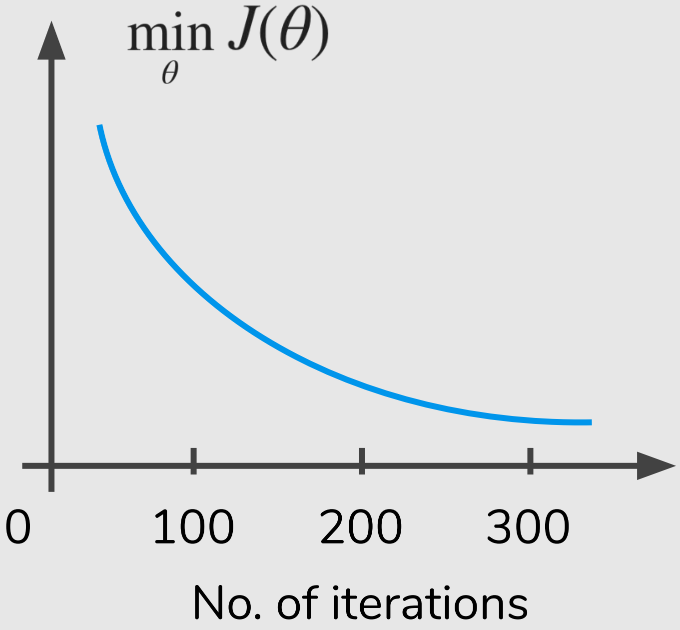
$$\theta_j := \theta_{\mathrm{j}} - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

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

Making sure gradient descent is working correctly.



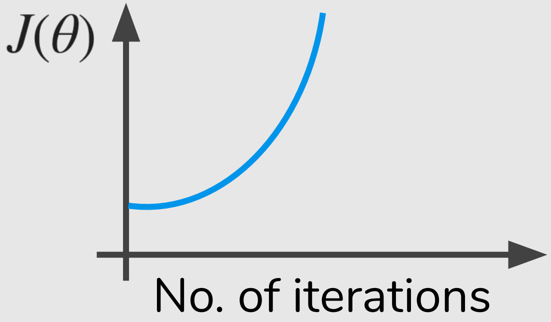
Making sure gradient descent is working correctly.



Example automatic convergence test:

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

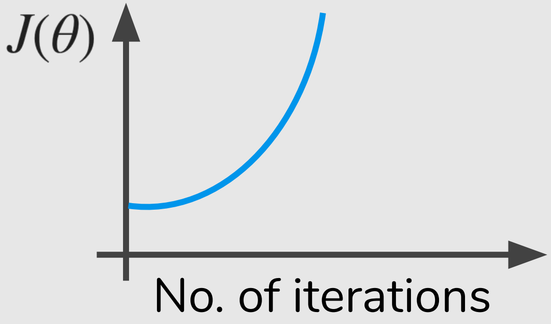
Making sure gradient descent is working correctly.



Gradient descent not working.

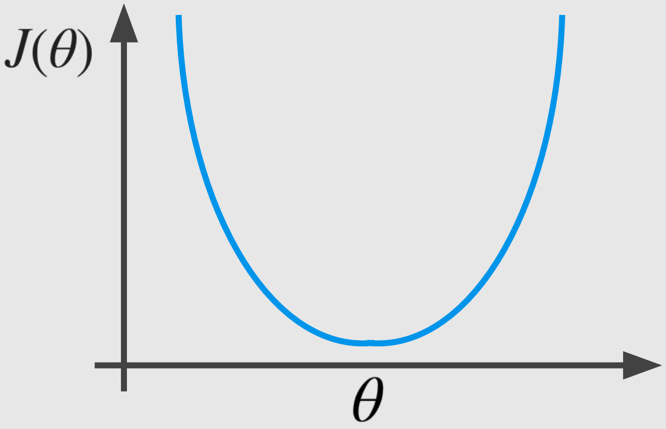
Use smaller α.

Making sure gradient descent is working correctly.

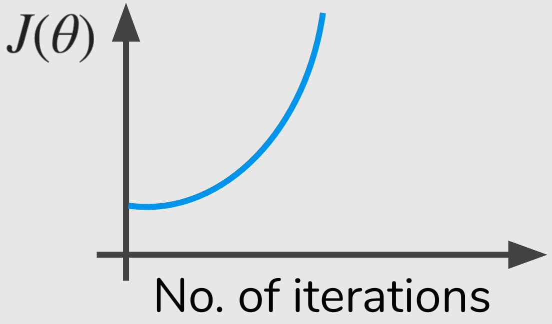


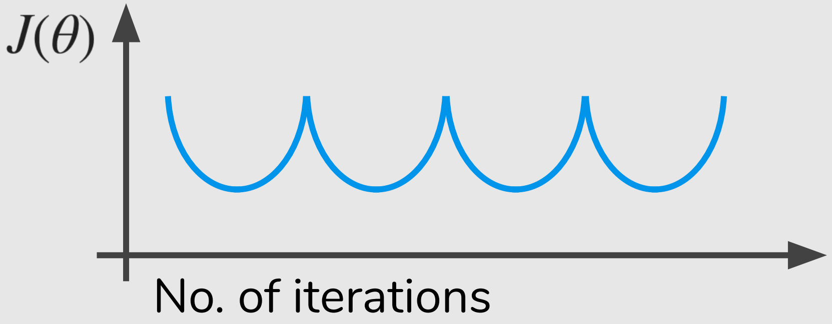
Gradient descent not working.

Use smaller α.



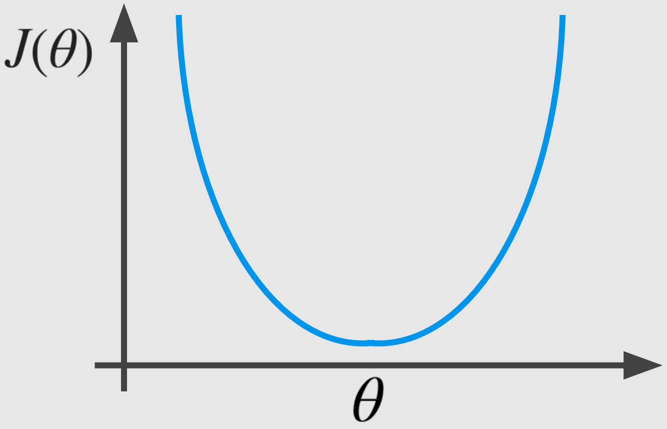
Making sure gradient descent is working correctly.



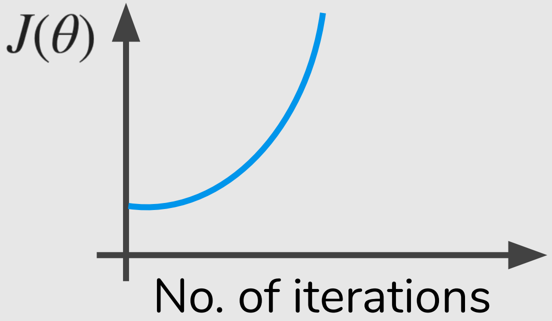


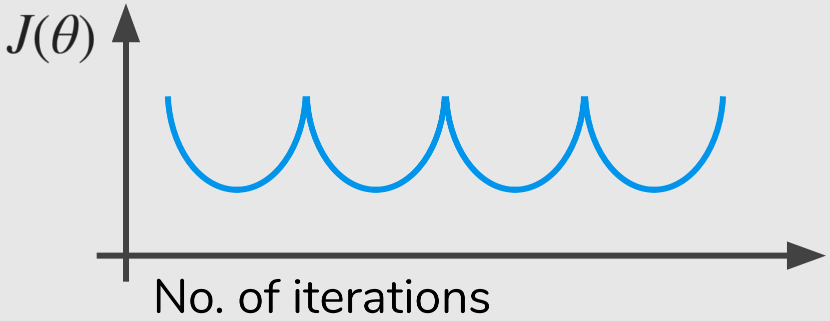
Gradient descent not working.

Use smaller α.



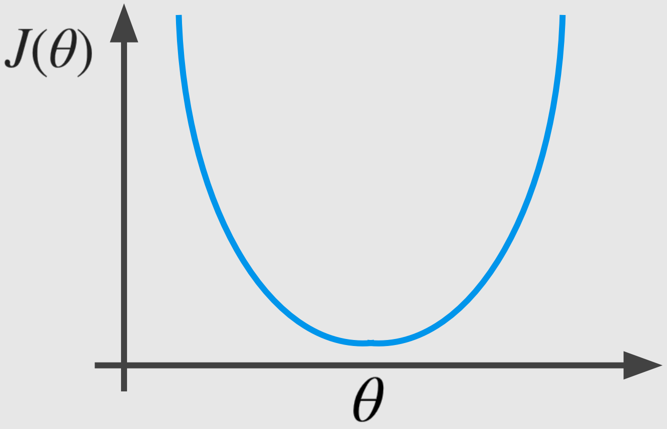
Making sure gradient descent is working correctly.





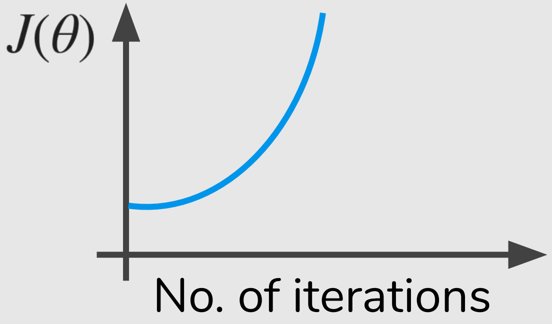
Gradient descent not working.

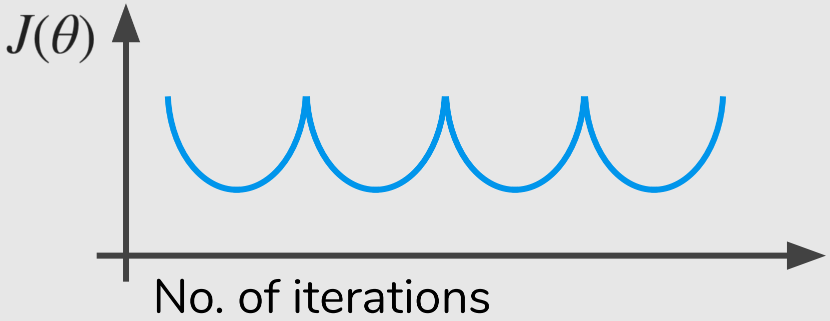
Use smaller α.



- For sufficiently small α, $J(\theta)$ should decrease on every iteration.

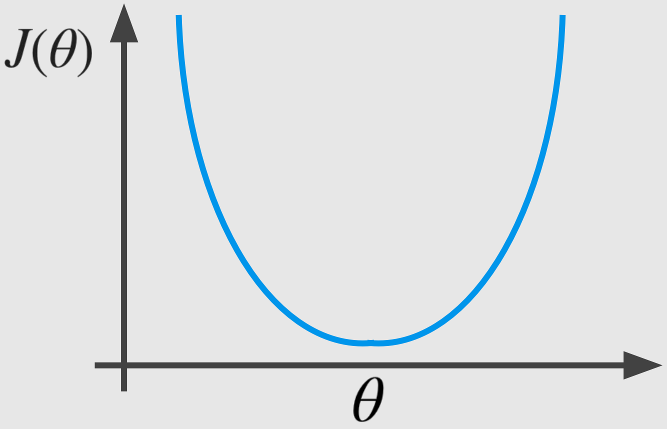
Making sure gradient descent is working correctly.





Gradient descent not working.

Use smaller α.



- But if α is too small, gradient descent can be slow to converge.

Summary

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

To choose α, try

$$\ldots, 0.001, \ldots, 0.01, \ldots, 0.1, \ldots, 1, \ldots$$

Today's Agenda

- Linear Regression with One Variable
  - Model Representation
  - Cost Function
  - Gradient Descent
- Linear Regression with Multiple Variables
  - Gradient Descent for Multiple Variables
  - **Feature Scaling**
  - **Learning Rate**
  - **Features and Polynomial Regression**
  - **Normal Equation**

Features and Polynomial Regression

Housing prices prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times \text{frontage} + \theta_2 \times \text{depth}$$

Housing prices prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{\text{frontage}}_{x_1} + \theta_2 \times \underbrace{\text{depth}}_{x_2}$$

Housing prices prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{\text{frontage}}_{x_1} + \theta_2 \times \underbrace{\text{depth}}_{x_2}$$

Area x = frontage $\times$ depth

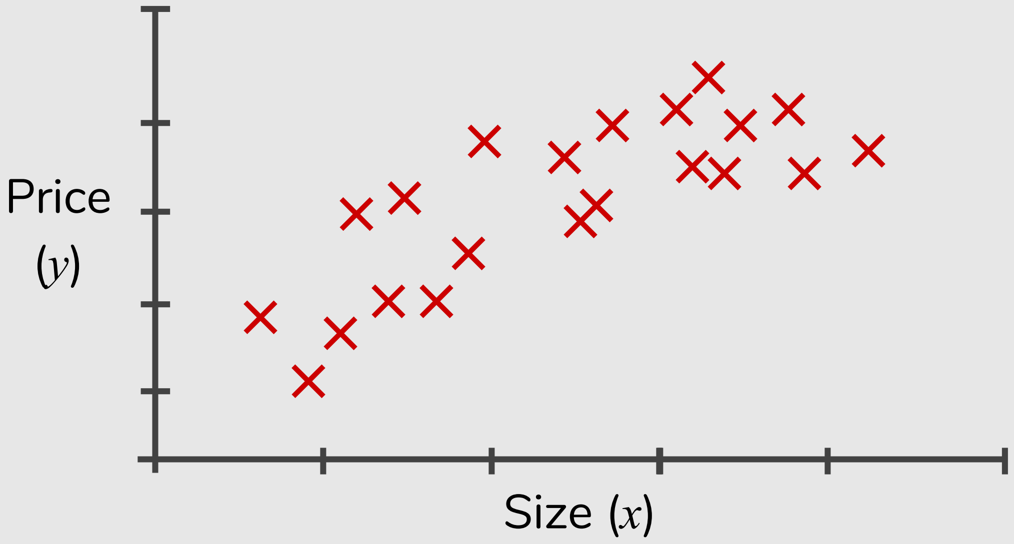
Housing prices prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times \underset{\substack{\downarrow \\ x_1}}{\text{frontage}} + \theta_2 \times \underset{\substack{\downarrow \\ x_2}}{\text{depth}}$$

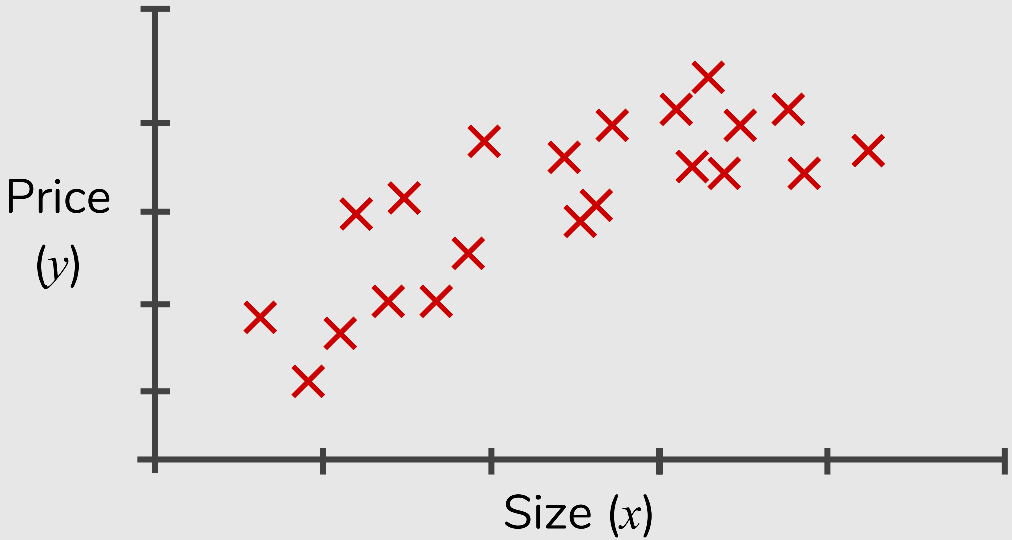
Area $x = \text{frontage} \times \text{depth}$

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

Polynomial Regression

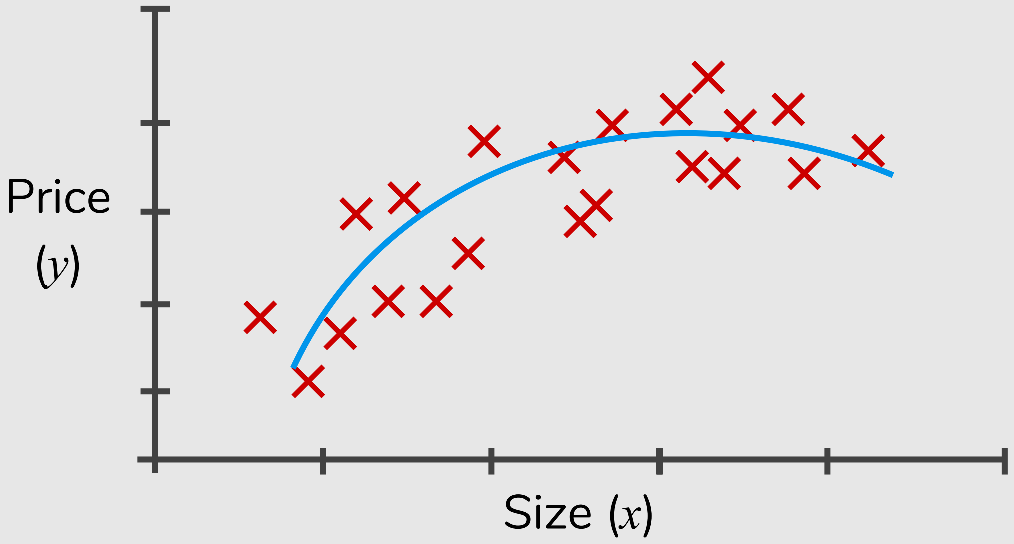


Polynomial Regression



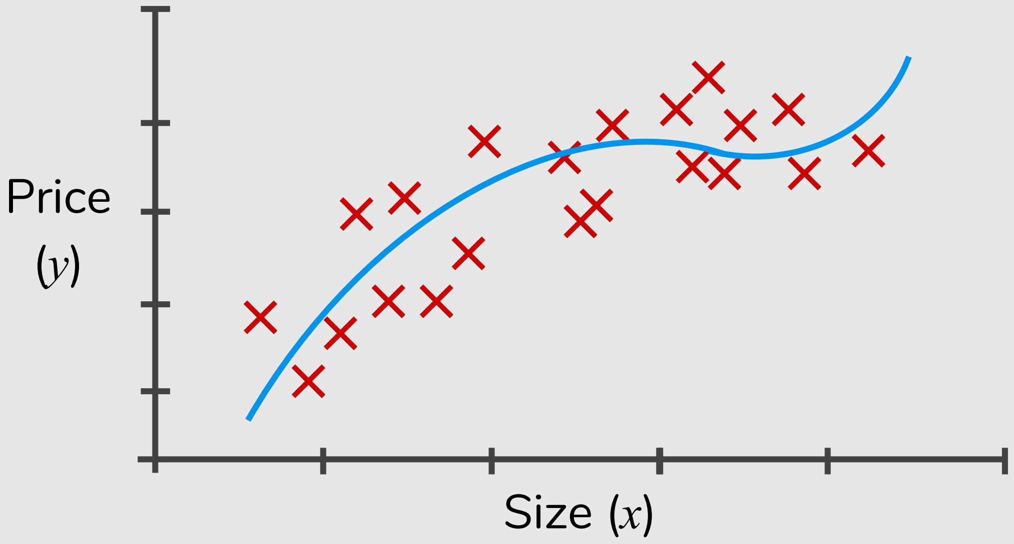
$$\theta_0 + \theta_1 x + \theta_2 x^2$$

Polynomial Regression



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

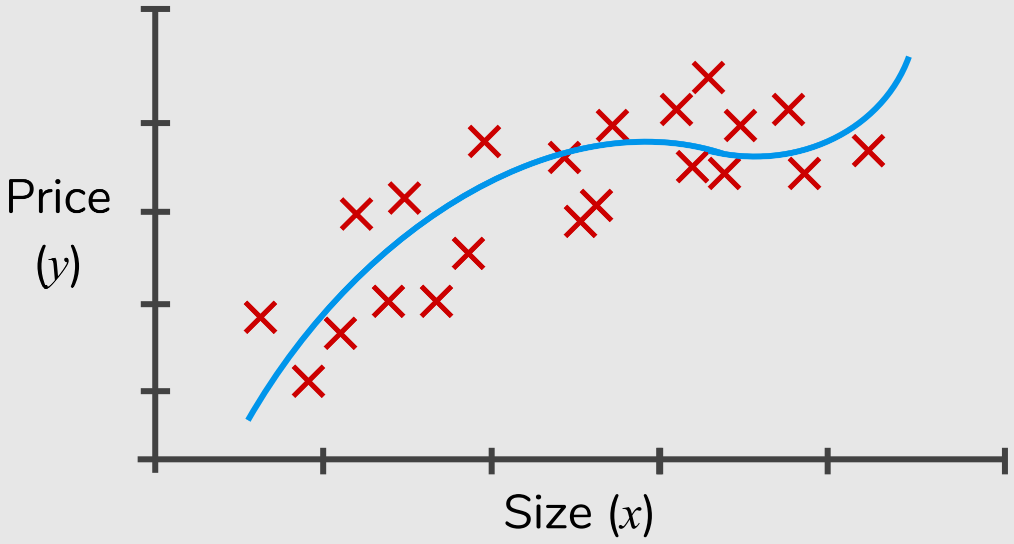
Polynomial Regression



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

Polynomial Regression



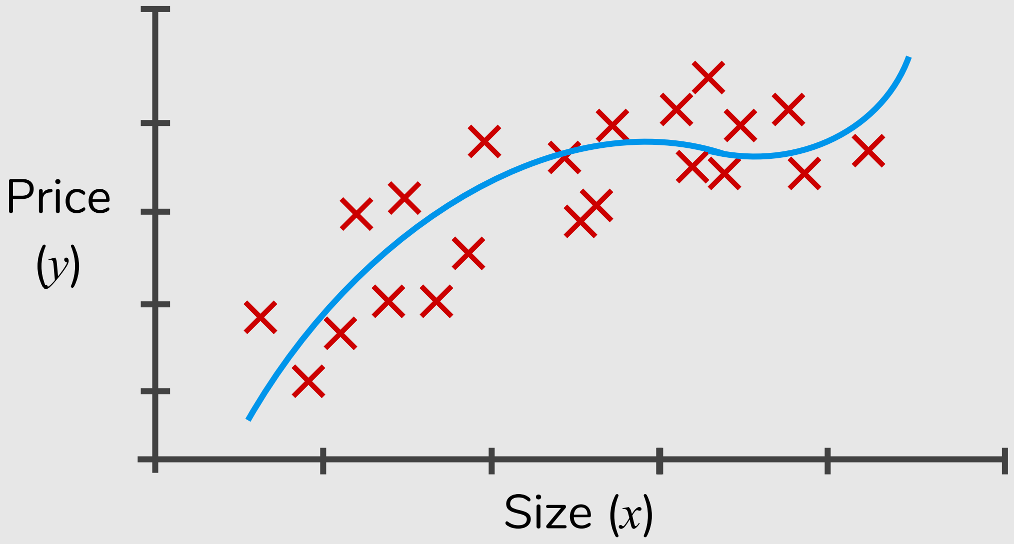
$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3 \end{aligned}$$

$$x_1 = (\text{size})$$
$$x_2 = (\text{size})^2$$
$$x_3 = (\text{size})^3$$

Polynomial Regression



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

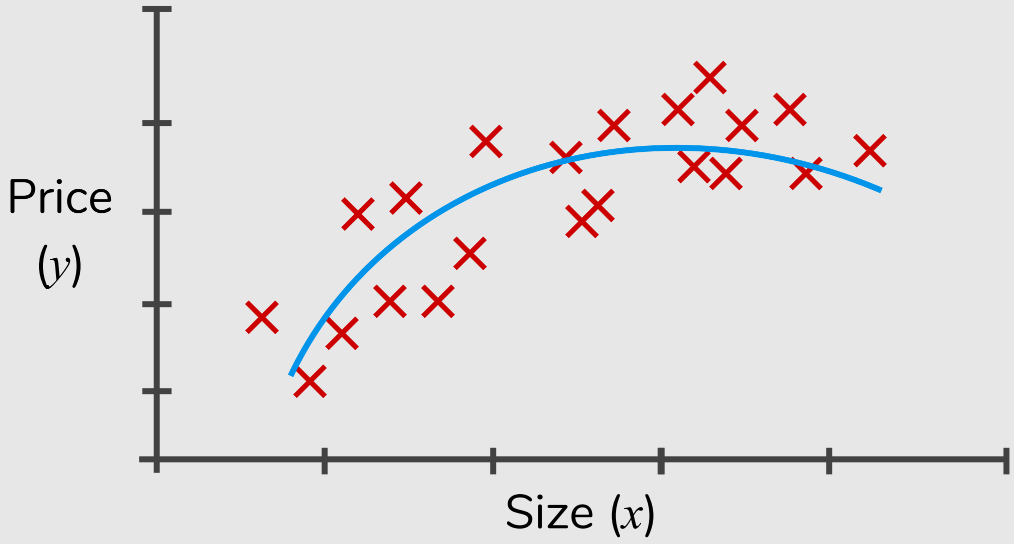
$$x_1 = (\text{size}) \quad : 1 - 1{,}000$$

$$x_2 = (\text{size})^2 \quad : 1 - 1{,}000{,}000$$

$$x_3 = (\text{size})^3 \quad : 1 - 10^9$$

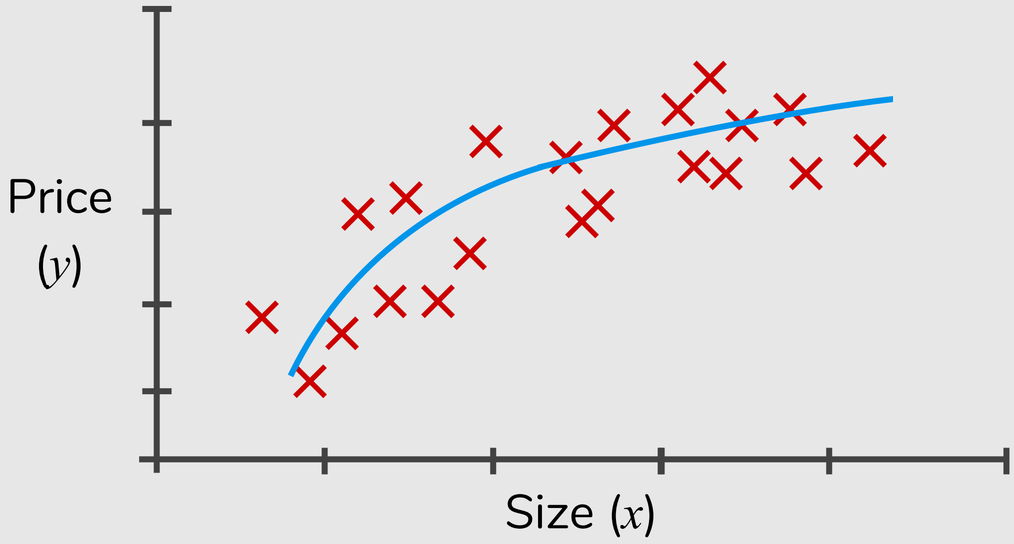
$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3 \end{aligned}$$

Choice of Features



$$h_\theta(x) = \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2$$

Choice of Features

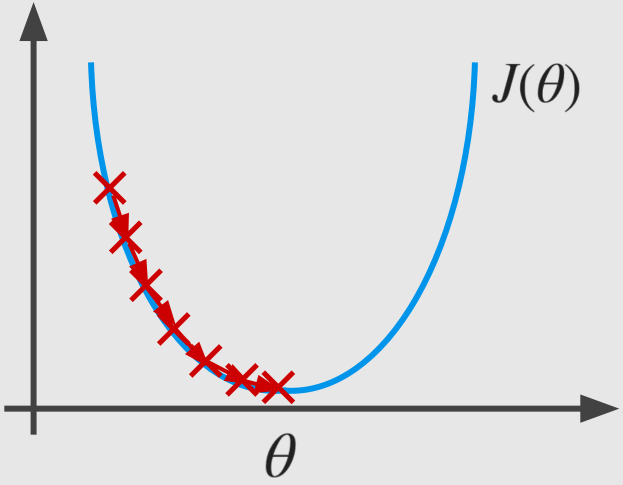


$$h_\theta(x) = \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2$$

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

Normal Equation

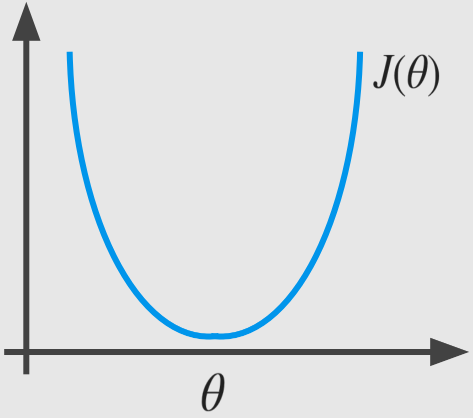
Gradient Descent



Normal equation: Method to solve θ **analytically**.

Intuition: If 1D ($\theta \in \mathbb{R}$)

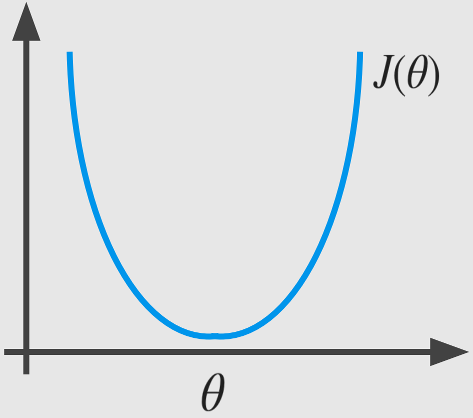
$$J(\theta) = a\theta^2 + b\theta + c$$



Intuition: If 1D ($\theta \in \mathbb{R}$)

$$J(\theta) = a\theta^2 + b\theta + c$$

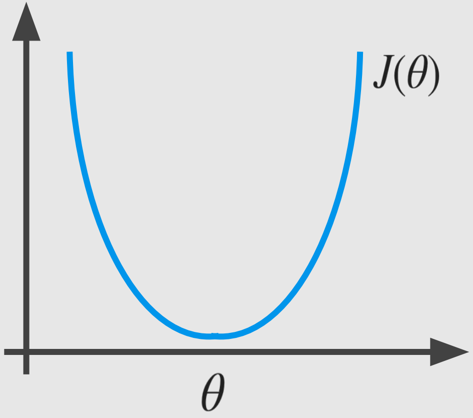
$$\frac{d}{d\theta} J(\theta) = \ldots = 0$$ Solve for θ



Intuition: If 1D ($\theta \in \mathbb{R}$)

$$J(\theta) = a\theta^2 + b\theta + c$$

$$\frac{d}{d\theta} J(\theta) = \ldots = 0$$ Solve for θ



$\theta \in \mathbb{R}^{n+1}$

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

$$\frac{\partial}{\partial \theta_j} J(\theta) = \ldots = 0$$ Solve for $\theta_0, \theta_1, \ldots, \theta_n$

Examples: $m = 4$.

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

Examples: $m = 4$.

x_0	Size (feet^2) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($) in 1000's y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

Examples: $m = 4$.

x_0	Size (feet^2) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price (\$) in 1000's y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

Examples: $m = 4$.

x_0	Size (feet^2) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($) in 1000's y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

Examples: $m = 4$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price (\$) in 1000's y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

Examples: $m = 4$.

x_0	Size (feet^2) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price (\$) in 1000's y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

Examples: $m = 4$.

x_0	Size (feet^2) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($) in 1000's y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}_{m \times (n+1)} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}_m$$

Examples: $m = 4$.

x_0	Size (feet^2) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($) in 1000's y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}_{m \times (n+1)} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}_{m}$$

$$\boxed{\theta = (X^T X)^{-1} X^T y}$$

m **examples** $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ **and** n **features**

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

m **examples** $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ **and** n **features**

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$X = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

m **examples** $(x^{(1)}, y^{(1)}), \ \ldots, \ (x^{(m)}, y^{(m)})$ **and** n **features**

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$X = \begin{bmatrix} \text{———} \ (x^{(1)})^{\mathrm{T}} \ \text{———} \\ \\ \\ \\ \end{bmatrix}$$

m examples $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ and n features

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$X = \begin{bmatrix} — (x^{(1)})^{\mathrm{T}} — \\ — (x^{(2)})^{\mathrm{T}} — \\ \\ \\ \end{bmatrix}$$

m examples $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ and n features

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$X = \begin{bmatrix} \text{———} & (x^{(1)})^{\mathrm{T}} & \text{———} \\ \text{———} & (x^{(2)})^{\mathrm{T}} & \text{———} \\ \text{———} & \vdots & \text{———} \\ \text{———} & (x^{(m)})^{\mathrm{T}} & \text{———} \end{bmatrix}$$

m examples $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ and n features

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$X = \begin{bmatrix} \text{———} & (x^{(1)})^{\mathrm{T}} & \text{———} \\ \text{———} & (x^{(2)})^{\mathrm{T}} & \text{———} \\ \text{———} & \vdots & \text{———} \\ \text{———} & (x^{(m)})^{\mathrm{T}} & \text{———} \end{bmatrix}$$

E.g. $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

m **examples** $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ **and** n **features**

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$X = \begin{bmatrix} \text{——} & (x^{(1)})^T & \text{——} \\ \text{——} & (x^{(2)})^T & \text{——} \\ \text{——} & \vdots & \text{——} \\ \text{——} & (x^{(m)})^T & \text{——} \end{bmatrix}$$

E.g. $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$ $\quad X = \begin{bmatrix} 1 & x_1^{(1)} \\ \vdots & \vdots \\ 1 & x_m^{(1)} \end{bmatrix}_{m \times 2}$

m examples $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ and n features

$$X = \begin{bmatrix} \text{---} & (x^{(1)})^{\mathrm{T}} & \text{---} \\ \text{---} & (x^{(2)})^{\mathrm{T}} & \text{---} \\ \text{---} & \vdots & \text{---} \\ \text{---} & (x^{(m)})^{\mathrm{T}} & \text{---} \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

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

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

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

$(X^T X)^{-1}$ is inverse of matrix $X^T X$.

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

$(X^T X)^{-1}$ is inverse of matrix $X^T X$.

Deriving the Normal Equation using matrix calculus ...

👉 https://ayearofai.com/rohan-3-deriving-the-normal-equation-using-matrix-calculus-1a1b16f65dda

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

$(X^T X)^{-1}$ is inverse of matrix $X^T X$.

Deriving the Normal Equation using matrix calculus ...

👉 https://ayearofai.com/rohan-3-deriving-the-normal-equation-using-matrix-calculus-1a1b16f65dda

What if $X^T X$ is noninvertible?

What if $X^T X$ is noninvertible?

The common causes might be having :

- Redundant features, where two features are very closely related (i.e. they are linearly dependent).
- Too many features (e.g. $m \leq n$). In this case, delete some features or use “regularization”.

Gradient Descent

- 😞 Need to choose α.
- 😞 Needs many iterations.

Normal Equation

- 😃 No need to choose α.
- 😃 Don’t need to iterate.

m examples and n features

Gradient Descent

- 😞 Need to choose α.
- 😞 Needs many iterations.
- 😃 Works well even when n is large.

Normal Equation

- 😃 No need to choose α.
- 😃 Don’t need to iterate.
- 😞 Need to compute $(X^T X)^{-1} \rightarrow O(n^3)$.
- 😞 Slow if n is very large.

m examples and n features

References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 2 & 4
  https://www.oreilly.com/library/view/hands-on-machine-learning/9781491962282/ch04.html
- Pattern Recognition and Machine Learning, Chap. 3

Machine Learning Courses

- https://www.coursera.org/learn/machine-learning, Week 1 & 2