CoE202
Fundamentals of Artificial intelligence
<Big Data Analysis and Machine Learning>

Gradient-based Optimization

Prof. Young-Gyu Yoon
School of EE, KAIST

Contents

- Recap
- Gradient
- Gradient descent
  - for linear regression
  - With momentum
  - Stochastic gradient descent

Recap: Supervised learning

- **Supervised learning**: learning a function that maps an input to an output based on example input-output pairs

$$\text{For a data set } \mathcal{D} = \{(\vec{x_1}, \vec{y_1}), (\vec{x_2}, \vec{y_2}), \cdots, (\vec{x_N}, \vec{y_N})\}$$

$$\text{Seeks a function } f : X \to Y$$

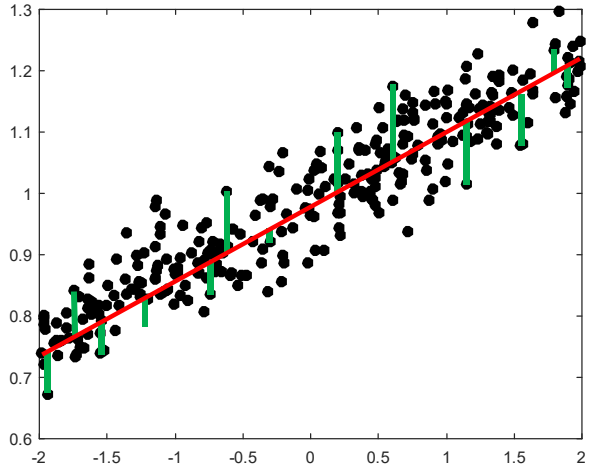
$$\text{Such that a loss function } \mathcal{L} : X \times Y \to \mathcal{R} \text{ is minimized}$$

Recap: simple linear regression

For a data set $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$

Seeks a function $f(x; \theta_0, \theta_1) = \theta_1 x + \theta_0$

Such that the mean squared error,
$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (f(x_i) - y_i)^2$, is minimized



Recap: linear regression (matrix)

$$\mathcal{L}(\theta) = \frac{1}{4}(X\theta - Y)^T(X\theta - Y)$$

$$\nabla_\theta \mathcal{L}(\theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_0}\mathcal{L}(\theta_1, \theta_0) \\ \frac{\partial}{\partial \theta_1}\mathcal{L}(\theta_1, \theta_0) \end{bmatrix} = 0$$

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{2}X^T(X\theta - Y) = 0$$

$$X^TY = X^TX\theta$$

$$\theta = (X^TX)^{-1}X^TY$$

Recap: polynomial regression (matrix)

$$f(x_1) = \theta_2 x_1^2 + \theta_1 x_1 + \theta_0$$
$$f(x_2) = \theta_2 x_2^2 + \theta_1 x_2 + \theta_0$$
$$f(x_3) = \theta_2 x_3^2 + \theta_1 x_3 + \theta_0$$
$$f(x_4) = \theta_2 x_4^2 + \theta_1 x_4 + \theta_0$$

$$\begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \end{bmatrix} = \theta_0 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \theta_1 \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \theta_2 \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \end{bmatrix}$$

$$\begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

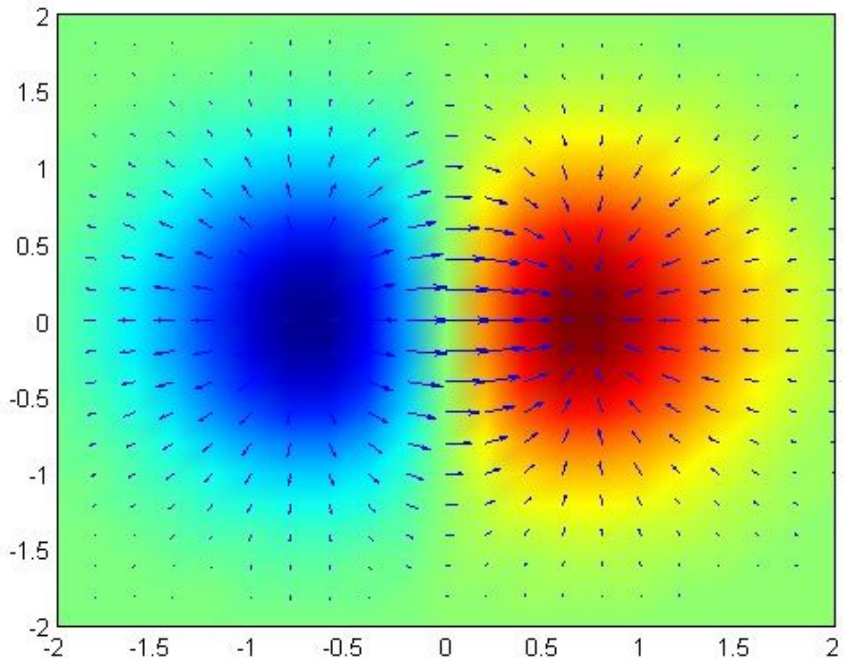
Φ Feature matrix

Gradient: finding optimal parameters

$$\nabla_\theta \mathcal{L}(\theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} \mathcal{L}(\theta_1, \theta_0) \\ \frac{\partial}{\partial \theta_1} \mathcal{L}(\theta_1, \theta_0) \end{bmatrix} = 0$$

- This requires the loss function written as a closed-form function of θ_1 and θ_0
- In many practical cases, we do not have a loss function written as a closed-form function of θ_1 and θ_0
- In many practical cases, we can still calculate the numerical value of the gradients at certain locations

Gradient

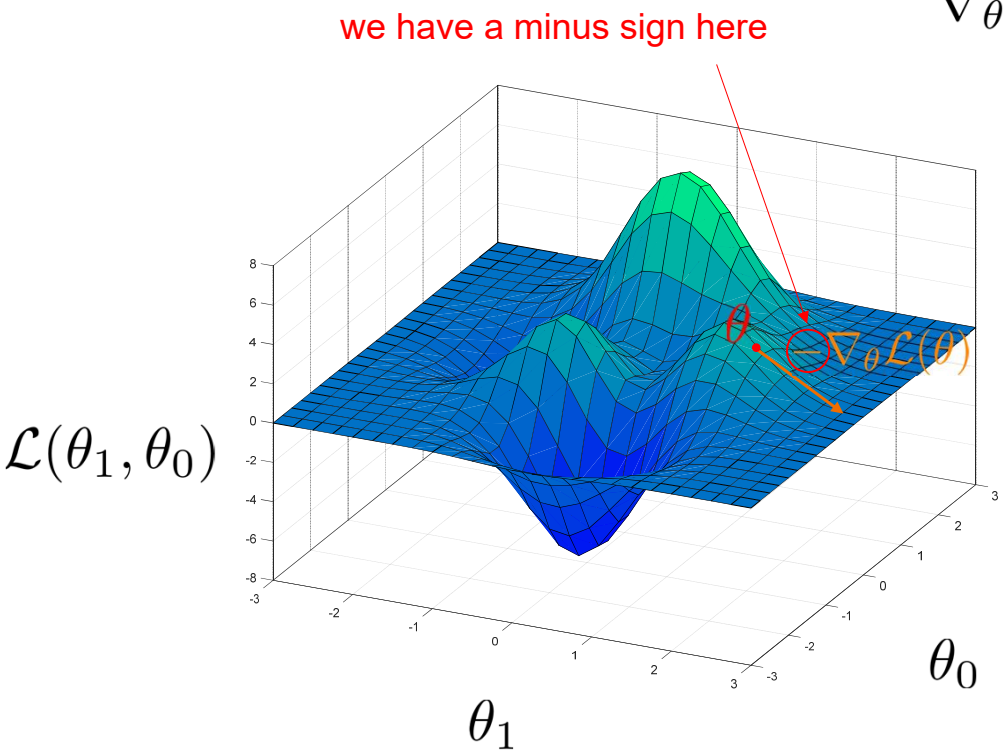


$$\nabla_\theta \mathcal{L}(\theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} \mathcal{L}(\theta_1, \theta_0) \\ \frac{\partial}{\partial \theta_1} \mathcal{L}(\theta_1, \theta_0) \end{bmatrix}$$

The gradient at a point is a vector pointing in the direction of the steepest slope at that point.

Gradient

$$\nabla_\theta \mathcal{L}(\theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} \mathcal{L}(\theta_1, \theta_0) \\ \frac{\partial}{\partial \theta_1} \mathcal{L}(\theta_1, \theta_0) \end{bmatrix}$$

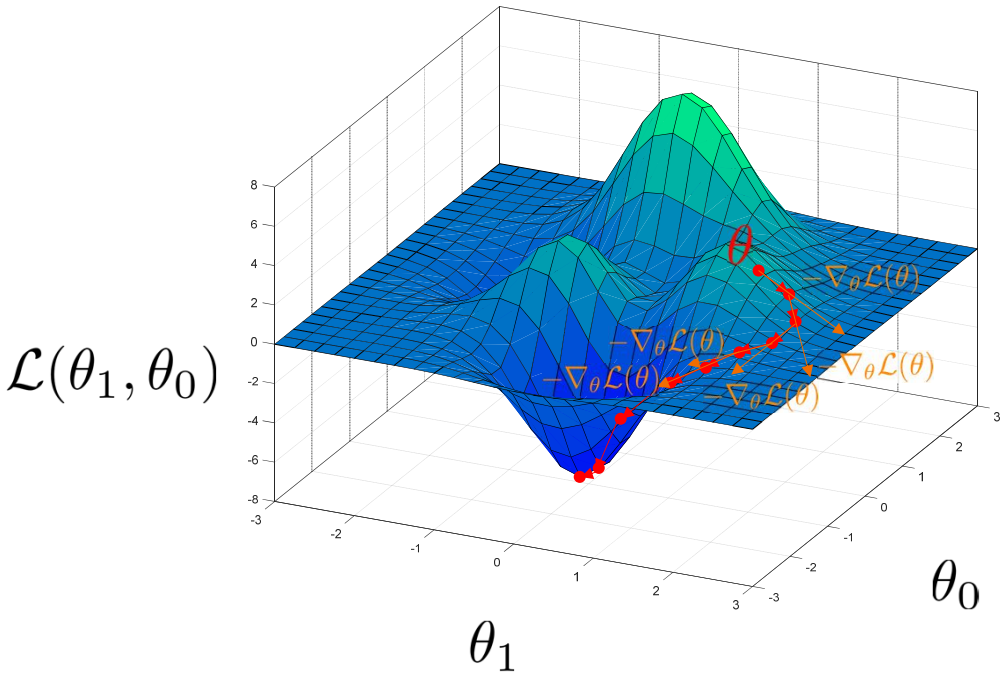


The gradient at a point is a vector pointing in the direction of the steepest slope at that point.

Gradient Descent

$$\theta^{(k+1)} = \theta^{(k)} - \gamma \nabla \mathcal{L}(\theta^{(k)})$$

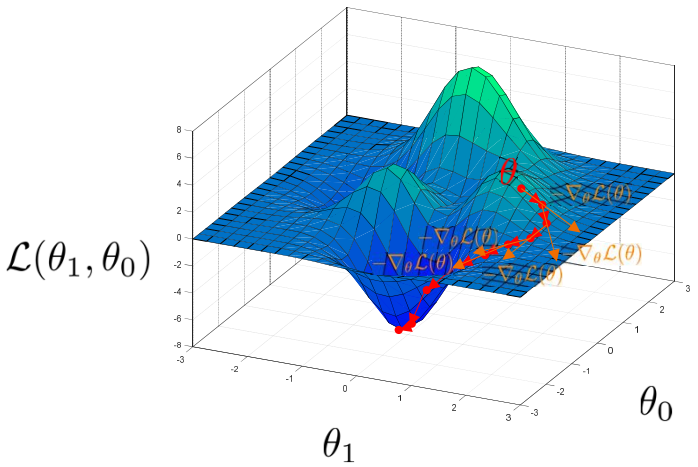
γ : learning rate



Just follow the gradient!

Gradient Descent

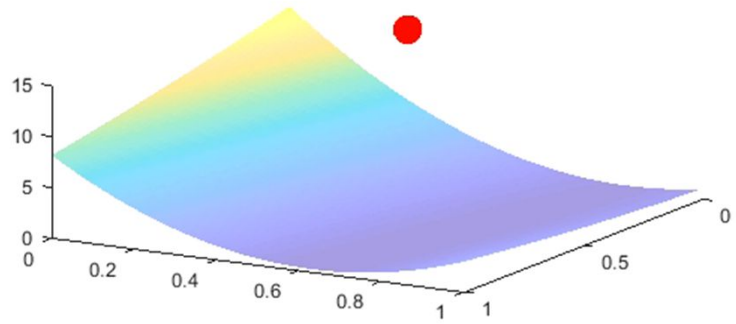
$$\theta^{(k+1)} = \theta^{(k)} - \gamma \nabla \mathcal{L}(\theta^{(k)})$$



- **Gradient descent** is an iterative algorithm for finding a local minimum of a differentiable function
- It requires only the gradient value at one point at each iteration step (does not require closed-form gradient function)

Gradient Descent

$$\theta^{(k+1)} = \theta^{(k)} - \gamma \nabla \mathcal{L}(\theta^{(k)})$$



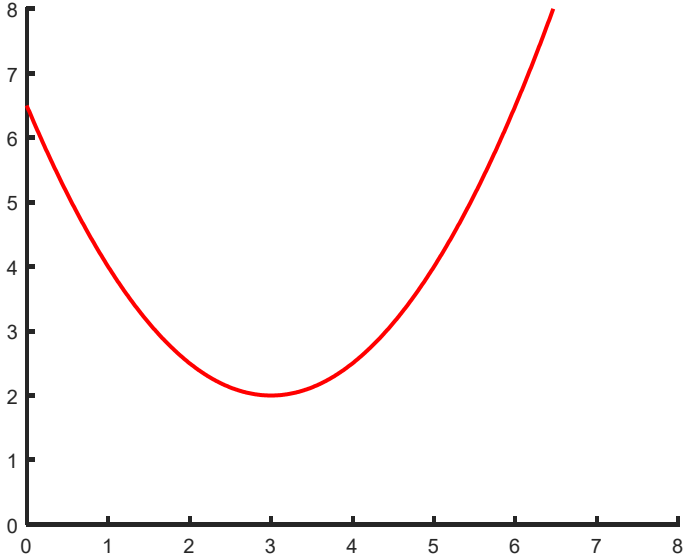
- **Gradient descent** is an iterative algorithm for finding a local minimum of a differentiable function
- It requires only the gradient value at one point at each iteration step (does not require closed-form gradient function)

Gradient Descent: simple example

$$\mathcal{L}(\theta) = \frac{1}{2}(\theta - 3)^2 + 2$$

$$\nabla_\theta \mathcal{L}(\theta) = (\theta - 3)$$

$$\gamma = 0.5$$

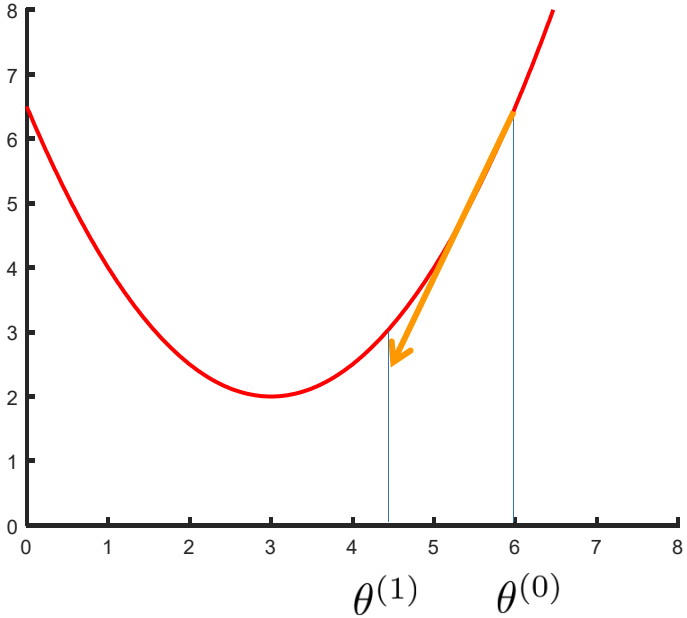


Gradient Descent: simple example

$$\mathcal{L}(\theta) = \frac{1}{2}(\theta - 3)^2 + 2$$

$$\nabla_\theta \mathcal{L}(\theta) = (\theta - 3)$$

$$\gamma = 0.5$$



$$\theta^{(0)} = 6$$

$$\nabla_\theta \mathcal{L}(\theta^{(0)}) = (\theta^{(0)} - 3) = 3$$

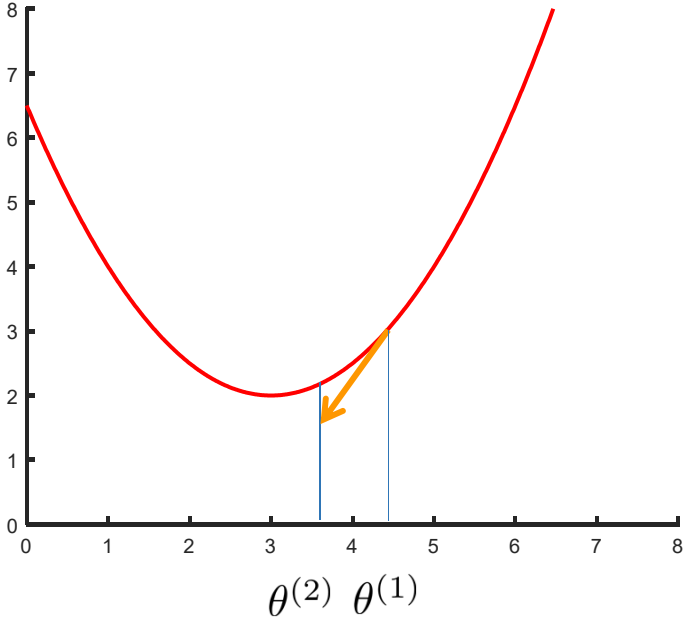
$$\theta^{(1)} = 6 - 0.5 \times 3 = 4.5$$

Gradient Descent: simple example

$$\mathcal{L}(\theta) = \frac{1}{2}(\theta - 3)^2 + 2$$

$$\nabla_\theta \mathcal{L}(\theta) = (\theta - 3)$$

$$\gamma = 0.5$$



$$\theta^{(1)} = 4.5$$

$$\nabla_\theta \mathcal{L}(\theta^{(1)}) = (\theta^{(1)} - 3) = 1.5$$

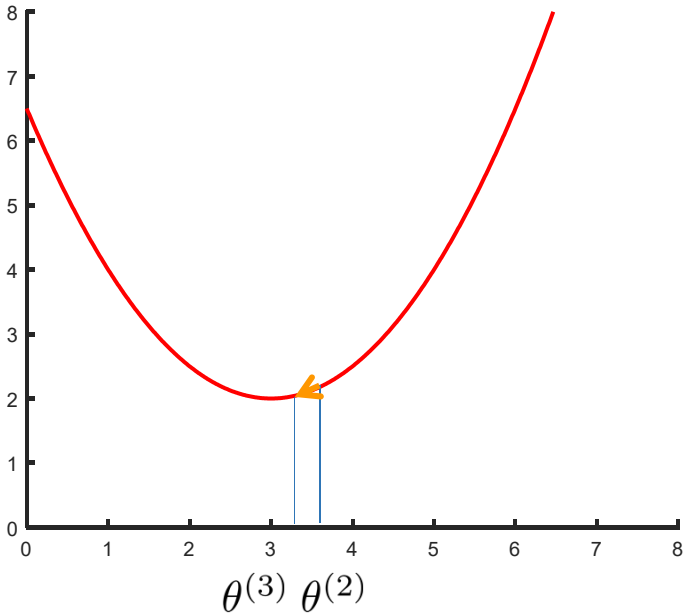
$$\theta^{(2)} = 4.5 - 0.5 \times 1.5 = 3.75$$

Gradient Descent: simple example

$$\mathcal{L}(\theta) = \frac{1}{2}(\theta - 3)^2 + 2$$

$$\nabla_\theta \mathcal{L}(\theta) = (\theta - 3)$$

$$\gamma = 0.5$$



$$\theta^{(2)} = 3.75$$

$$\nabla_\theta \mathcal{L}(\theta^{(2)}) = (\theta^{(2)} - 3) = 0.75$$

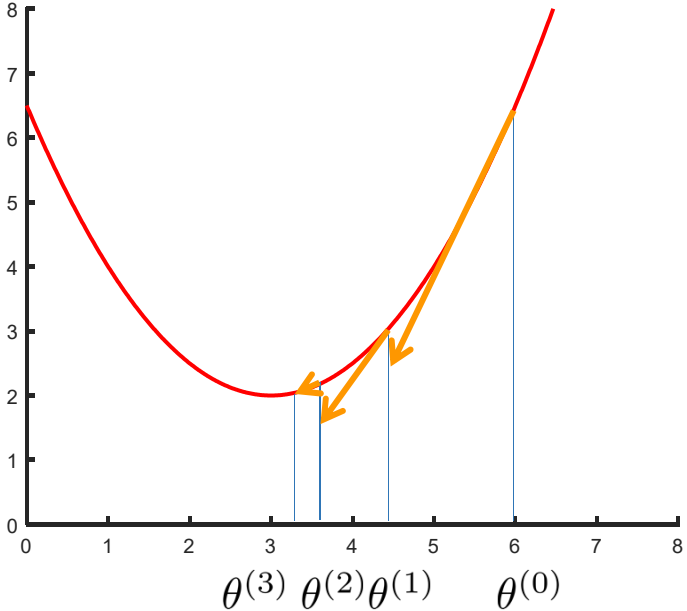
$$\theta^{(3)} = 3.75 - 0.5 \times 0.75 = 3.375$$

Gradient Descent: simple example

$$\mathcal{L}(\theta) = \frac{1}{2}(\theta - 3)^2 + 2$$

$$\nabla_\theta \mathcal{L}(\theta) = (\theta - 3)$$

$$\gamma = 0.5$$



$$\theta^{(k+1)} = \theta^{(k)} - 0.5 \times \nabla_\theta \mathcal{L}(\theta^{(k)})$$

$$= \theta^{(k)} - 0.5 \times (\theta^{(k)} - 3)$$

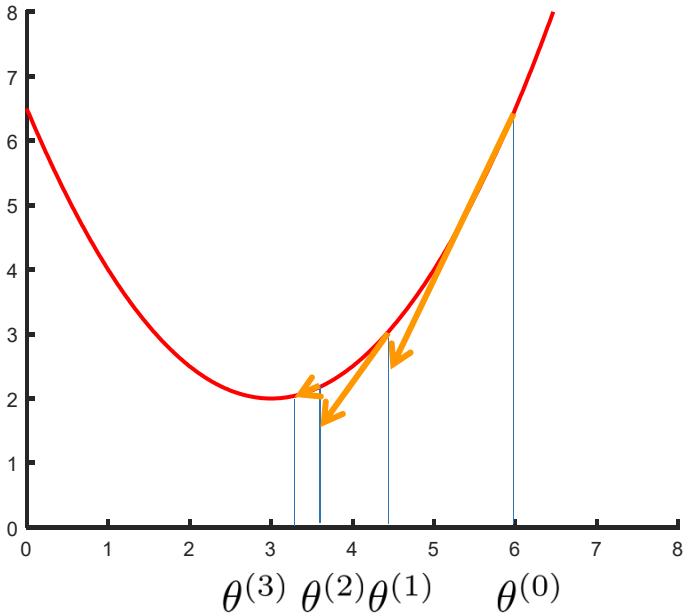
$$= 0.5\theta^{(k)} + 1.5$$

Gradient Descent: simple example

$$\mathcal{L}(\theta) = \frac{1}{2}(\theta - 3)^2 + 2$$

$$\nabla_\theta \mathcal{L}(\theta) = (\theta - 3)$$

$$\gamma = 0.5$$



$$\theta^{(k+1)} = 0.5\theta^{(k)} + 1.5$$

$$= 0.5 \cdots (0.5(0.5\theta^{(0)} + 1.5) + 1.5) \cdots + 1.5$$

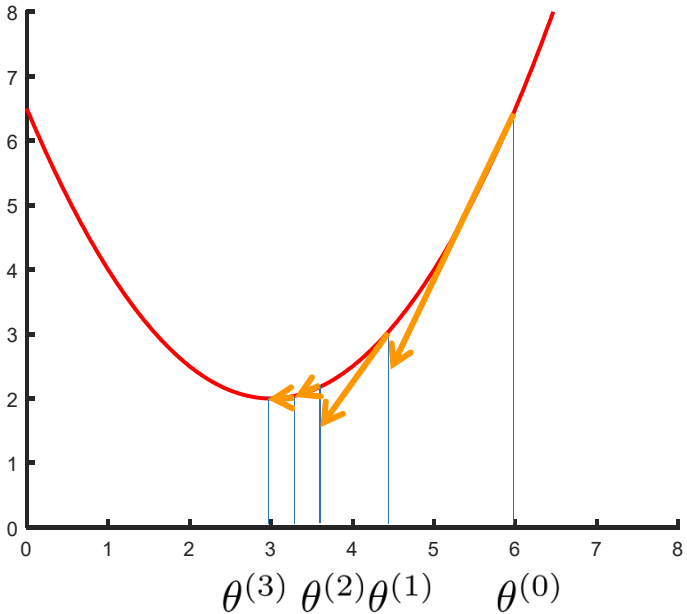
$$= 0.5^k \theta^{(0)} + 1.5 \frac{1 - 0.5^{k+1}}{1 - 0.5}$$

Gradient Descent: simple example

$$\mathcal{L}(\theta) = \frac{1}{2}(\theta - 3)^2 + 2$$

$$\nabla_\theta \mathcal{L}(\theta) = (\theta - 3)$$

$$\gamma = 0.5$$

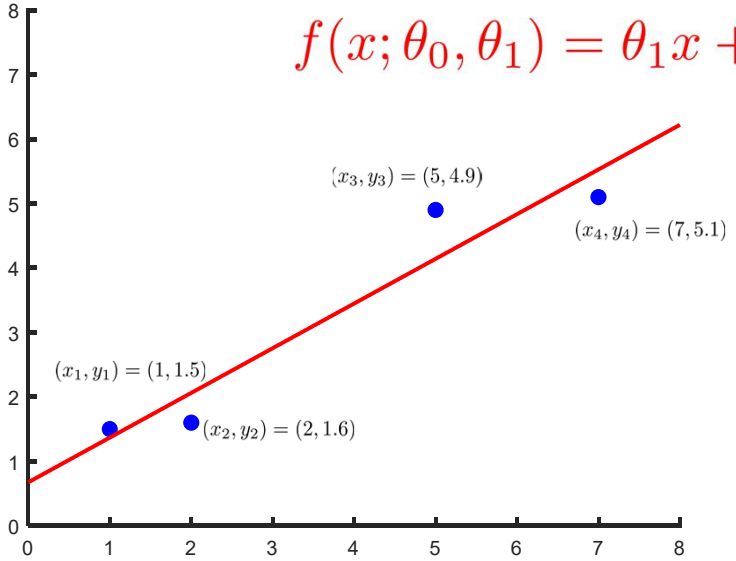


$$\theta^{(k+1)} = 0.5^k \theta^{(0)} + 1.5 \frac{1 - 0.5^{k+1}}{1 - 0.5}$$

$$\lim_{k \to \infty} \theta^{(k+1)} = 1.5 \frac{1}{1 - 0.5} = 3$$

Gradient Descent: linear regression

$$\mathcal{L}(\theta_1, \theta_0) = \theta_0^2 + 7.5\theta_0\theta_1 - 6.55\theta_0 + 19.75\theta_1^2 - 32.45\theta_1 + 13.7075$$



$$\nabla_\theta \mathcal{L}(\theta) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} \mathcal{L}(\theta_1, \theta_0) \\ \frac{\partial}{\partial \theta_1} \mathcal{L}(\theta_1, \theta_0) \end{bmatrix}$$

$$= \begin{bmatrix} 7.5\theta_1 + 2\theta_0 - 6.55 \\ 39.5\theta_1 + 7.5\theta_0 - 32.45 \end{bmatrix}$$

Gradient Descent: linear regression

$$\mathcal{L}(\theta_1, \theta_0) = \theta_0^2 + 7.5\theta_0\theta_1 - 6.55\theta_0 + 19.75\theta_1^2 - 32.45\theta_1 + 13.7075$$

$$\begin{bmatrix} \theta_0^{(k+1)} \\ \theta_1^{(k+1)} \end{bmatrix} = \begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \end{bmatrix} - \gamma \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_0} \\ \frac{\partial \mathcal{L}}{\partial \theta_1} \end{bmatrix}$$

$$= \begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \end{bmatrix} - \gamma \begin{bmatrix} 7.5\theta_1 + 2\theta_0 - 6.55 \\ 39.5\theta_1 + 7.5\theta_0 - 32.45 \end{bmatrix}$$

$$\theta_0^{(0)} = 0 \qquad \theta_0^{(1)} = 0 \qquad \gamma = 0.01$$

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = -6.55 \qquad \frac{\partial \mathcal{L}}{\partial \theta_0} = -32.45$$

$$\theta_0^{(1)} = \theta_0^{(0)} - 0.01 \times (-6.55) = 0.0655$$

$$\theta_1^{(1)} = \theta_1^{(0)} - 0.01 \times (-32.45) = 0.3245$$

Gradient Descent: linear regression

$$\mathcal{L}(\theta_1, \theta_0) = \theta_0^2 + 7.5\theta_0\theta_1 - 6.55\theta_0 + 19.75\theta_1^2 - 32.45\theta_1 + 13.7075$$

$$\begin{bmatrix} \theta_0^{(k+1)} \\ \theta_1^{(k+1)} \end{bmatrix} = \begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \end{bmatrix} - \gamma \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_0} \\ \frac{\partial \mathcal{L}}{\partial \theta_1} \end{bmatrix}$$

$$= \begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \end{bmatrix} - \gamma \begin{bmatrix} 7.5\theta_1 + 2\theta_0 - 6.55 \\ 39.5\theta_1 + 7.5\theta_0 - 32.45 \end{bmatrix}$$

Compare this with

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

$$\theta_0 = 0.6747 \quad \theta_1 = 0.6934$$

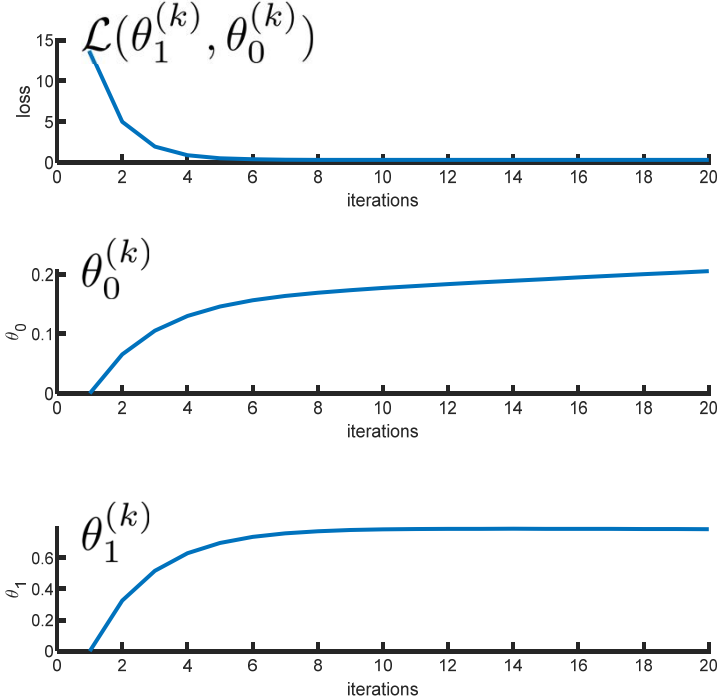
Iter	θ_0	θ_1	$\partial L/\partial\theta_0$	$\partial L/\partial\theta_1$
0	*0*	*0*	*-6.55*	*-32.45*
1	*0.0655*	*0.3245*	*-3.99*	*-19.14*
2	*0.1054*	*0.5159*	*-2.47*	*-11.28*
3	*0.1301*	*0.6287*	*-1.57*	*-6.64*
4	*0.1458*	*0.6951*	*-1.05*	*-3.90*
5	*0.1562*	*0.7341*	*-0.73*	*-2.28*
6	*0.1636*	*0.7569*	*-0.55*	*-1.32*
…	…	…	…	…
100	*0.3741*	*0.7510*	*-0.17*	*0.03*
…	…	…	…	…
1000	*0.6727*	*0.6938*	*-0.001*	*0.0002*

Gradient Descent: linear regression

$$\mathcal{L}(\theta_1, \theta_0) = \theta_0^2 + 7.5\theta_0\theta_1 - 6.55\theta_0 + 19.75\theta_1^2 - 32.45\theta_1 + 13.7075$$

$$\begin{bmatrix} \theta_0^{(k+1)} \\ \theta_1^{(k+1)} \end{bmatrix} = \begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \end{bmatrix} - \gamma \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_0} \\ \frac{\partial \mathcal{L}}{\partial \theta_1} \end{bmatrix}$$

$$= \begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \end{bmatrix} - \gamma \begin{bmatrix} 7.5\theta_1 + 2\theta_0 - 6.55 \\ 39.5\theta_1 + 7.5\theta_0 - 32.45 \end{bmatrix}$$



Compare this with

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

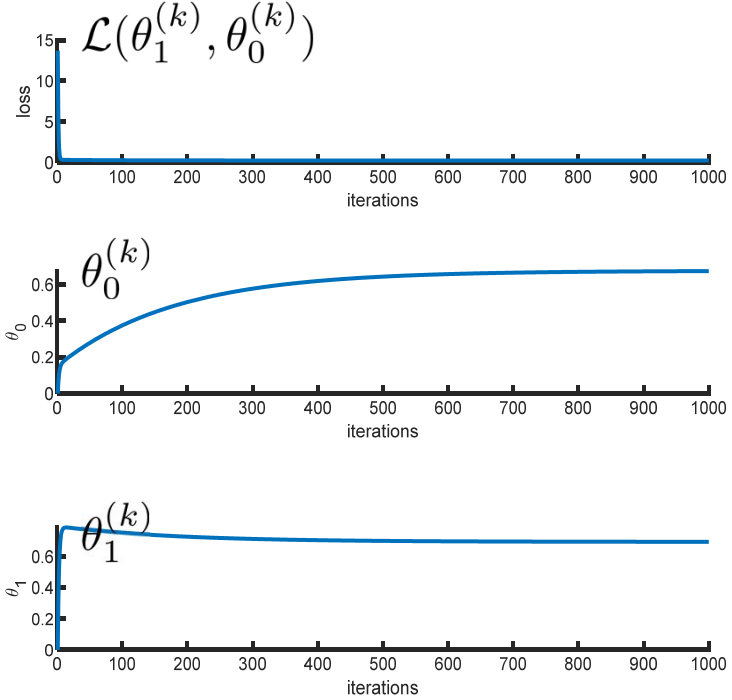
$$\theta_0 = 0.6747 \quad \theta_1 = 0.6934$$

Gradient Descent: linear regression

$$\mathcal{L}(\theta_1, \theta_0) = \theta_0^2 + 7.5\theta_0\theta_1 - 6.55\theta_0 + 19.75\theta_1^2 - 32.45\theta_1 + 13.7075$$

$$\begin{bmatrix} \theta_0^{(k+1)} \\ \theta_1^{(k+1)} \end{bmatrix} = \begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \end{bmatrix} - \gamma \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_0} \\ \frac{\partial \mathcal{L}}{\partial \theta_1} \end{bmatrix}$$

$$= \begin{bmatrix} \theta_0^{(k)} \\ \theta_1^{(k)} \end{bmatrix} - \gamma \begin{bmatrix} 7.5\theta_1 + 2\theta_0 - 6.55 \\ 39.5\theta_1 + 7.5\theta_0 - 32.45 \end{bmatrix}$$

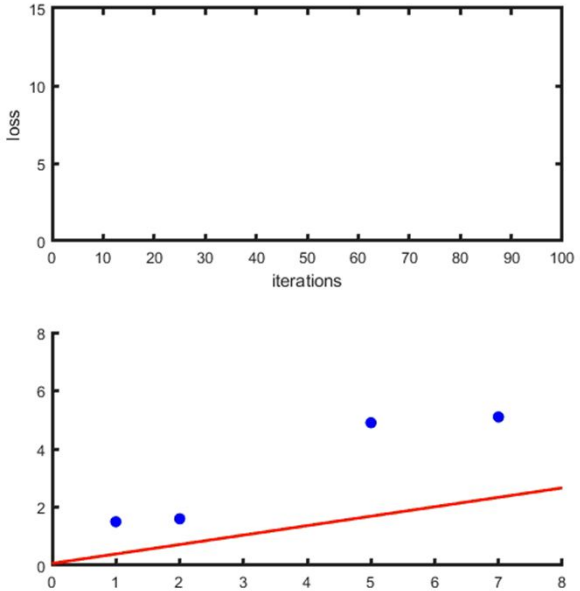


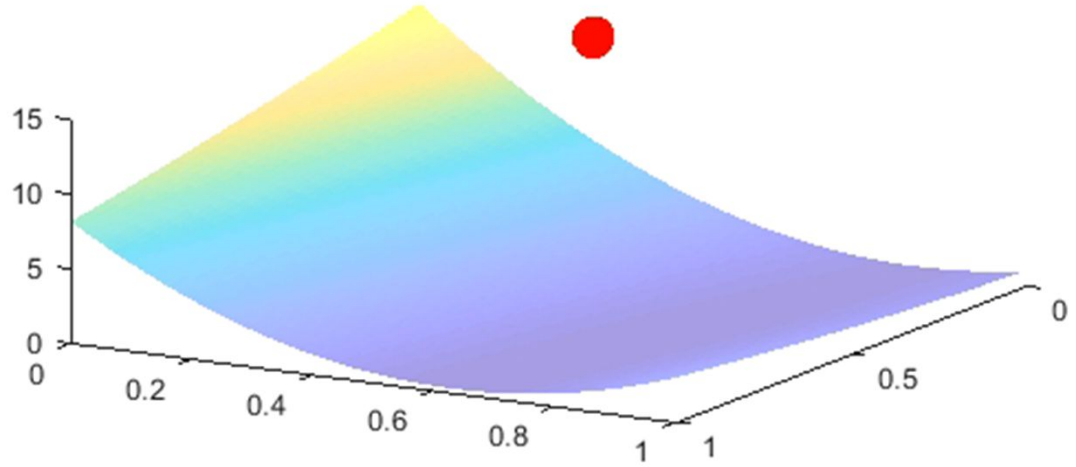
Compare this with

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

$$\theta_0 = 0.6747 \quad \theta_1 = 0.6934$$

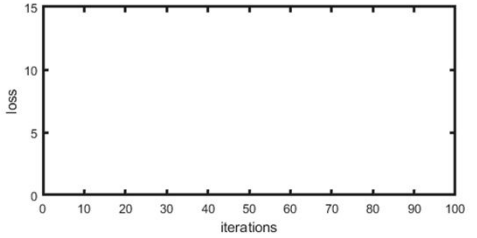
Gradient Descent: linear regression

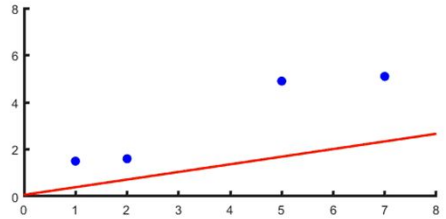


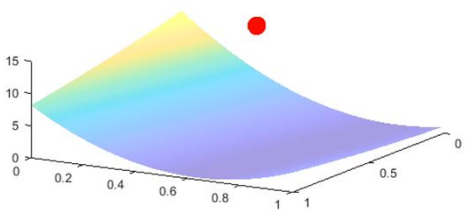


Gradient Descent: importance of learning rate

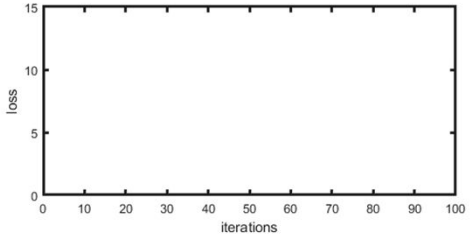
$\gamma = 0.01$

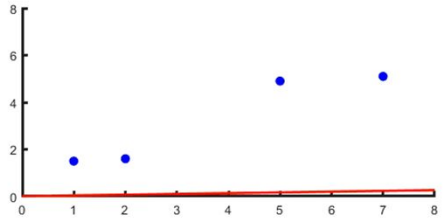


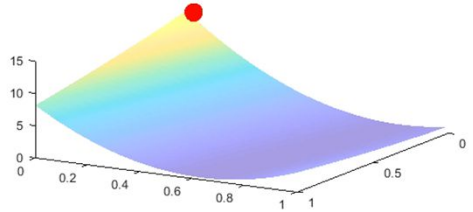




$\gamma = 0.001$

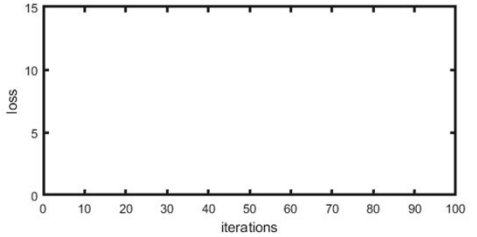


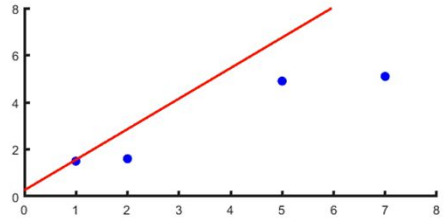


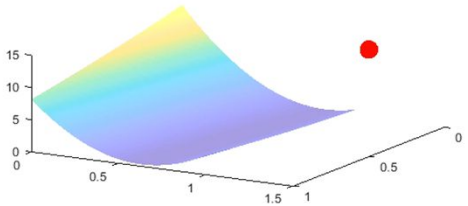


Gradient Descent: importance of learning rate

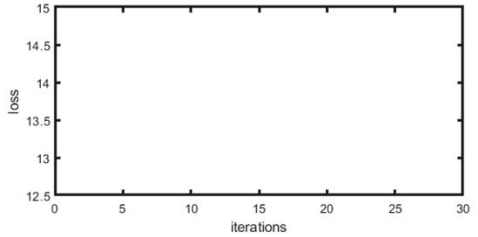
$\gamma = 0.04$

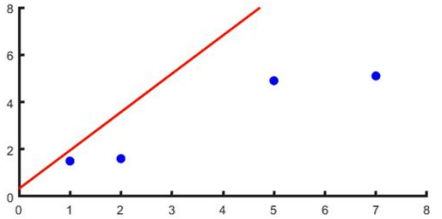


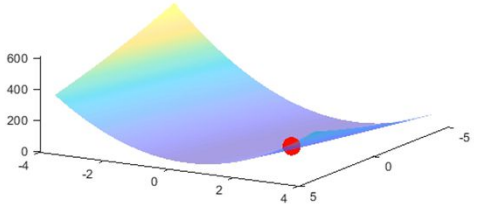




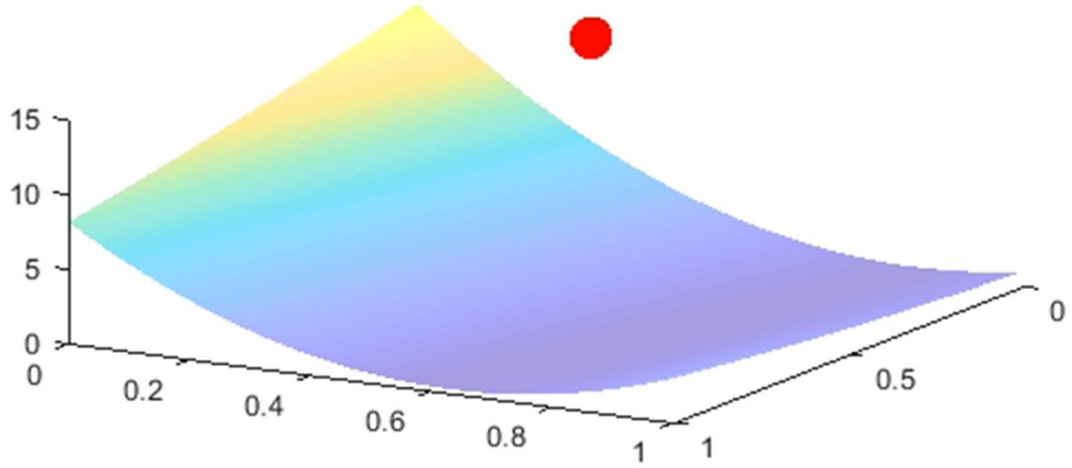
$\gamma = 0.05$





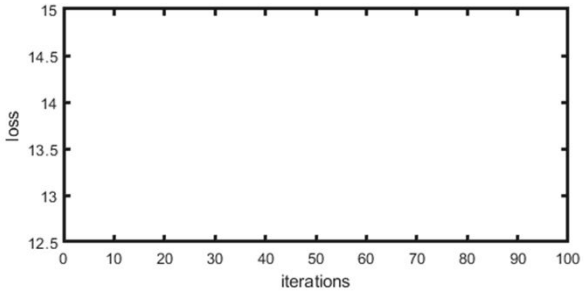


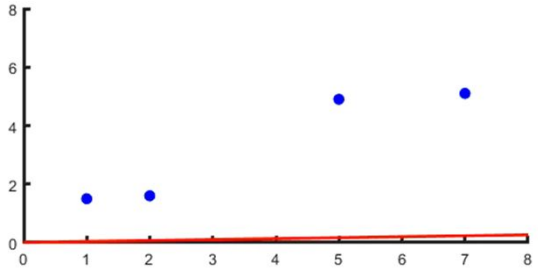
Intuition from the video

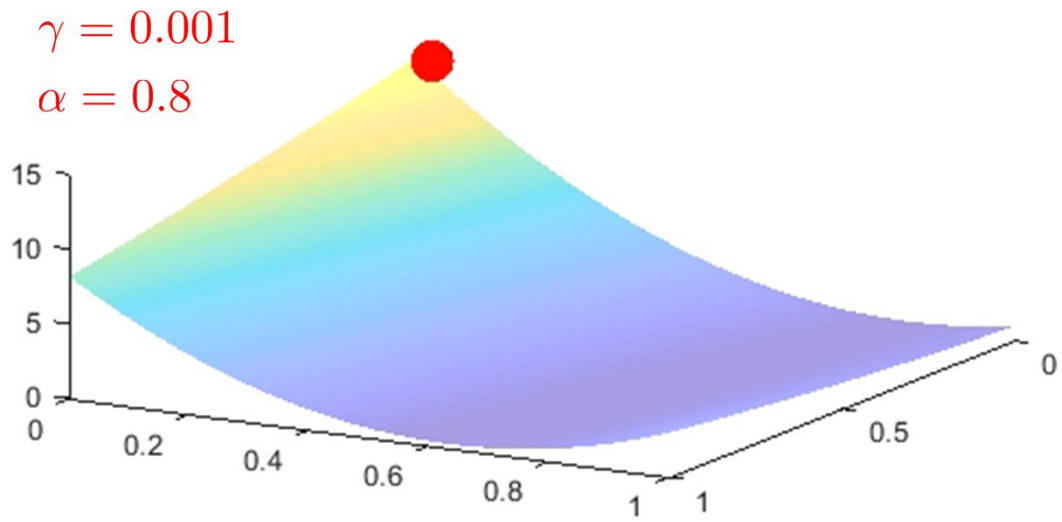


- **Gradient descent** is like placing a ball on a surface and letting it go to the lowest point on the surface
- Speed and direction are determined purely by the slope of the surface (i.e., there is no momentum)

Gradient Descent with momentum







$$\theta^{(k+1)} = \theta^{(k)} + \Delta\theta^{(k+1)}$$

$$\Delta\theta^{(k+1)} = \alpha\Delta\theta^{(k)} - \gamma\nabla\mathcal{L}(\theta^{(k)})$$

Goes back to GD if $\alpha=0$

Stochastic Gradient Descent (SGD)

- Gradient descent requires calculation of $\nabla_\theta \mathcal{L}(\theta)$
- Calculating $\nabla_\theta \mathcal{L}(\theta)$ requires accessing the ENTIRE data set
  - which would require extremely large amount of computation and memory for modern machine learning problems
- Calculating $\nabla_\theta \mathcal{L}(\theta)$ has to be done for every iteration (only one update for one gradient calculation)
- **Stochastic gradient descent (SGD)**: a variant of GD for more efficient update via gradient approximation as follows

For $k = 1, 2, \cdots, M$:

shuffle and split the data

For $i = 1, 2, \cdots, n$:

$$\theta^{(k+1)} = \theta^{(k)} - \gamma \nabla \mathcal{Q}_i(\theta^{(k)})$$

where $\mathcal{Q}_i(\theta)$ is the loss function for a part of the data set

That is, $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=0}^{n} \mathcal{Q}_i(\theta)$

GD vs. SGD

- **Gradient descent (GD)**

For $k = 1, 2, \cdots, M$:

$$\theta^{(k+1)} = \theta^{(k)} - \gamma \nabla \mathcal{L}(\theta^{(k)})$$

- **Stochastic gradient descent (SGD)**

For $k = 1, 2, \cdots, M$:

shuffle and split the data

For $i = 1, 2, \cdots, n$:

$$\theta^{(k+1)} = \theta^{(k)} - \gamma \nabla \mathcal{Q}_i(\theta^{(k)})$$

where $\mathcal{Q}_i(\theta)$ is the loss function for a part of the data set

That is, $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=0}^{n} \mathcal{Q}_i(\theta)$

GD vs. SGD

- **SGD**
  - The true gradient $\nabla_\theta \mathcal{L}(\theta)$ is approximated by $\nabla_\theta \mathcal{Q}_i(\theta)$ which is computed with the part of the data set (rather than the entire data set)
  - SGD requires access to only part of the data set at each time ☺
    - More memory efficient ☺
    - Less computation per each update ☺
  - Approximated gradient may not be accurate ☹
    - Convergence becomes cumbersome ☹ (however, reducing the learning rate can solve the problem)

Summary

- In most cases, the gradient (of the loss function) is not given in a closed-form (but we can still evaluate the gradient at each point)
- We can use numerical gradient to iteratively update the parameters (to find the optimal parameters) → **gradient descent**
- Gradient descent in a nutshell: follow the (negative) gradient to get to the lowest point
- Gradient descent is like placing a ball on a surface and seeing where it goes to (due to gravity)
- We can introduce momentum to accelerate the convergence → **gradient descent with momentum**
- Gradient descent is not feasible when we have large data sets. We can approximate the true gradient (calculated from the entire data set) by the gradient from a part of the data set → **stochastic gradient descent**

References

- Lecture notes
  - CC229 lecture note
    - http://cs229.stanford.edu/notes2020fall/notes2020fall/cs229-notes1.pdf
  - MIT 6.036 Intro to Machine Learning (Chapter 6)
    - https://www.mit.edu/~lindrew/6.036.pdf

- Website
  - CS231n course website: https://cs231n.github.io/