Gradient Descent

Nipun Batra

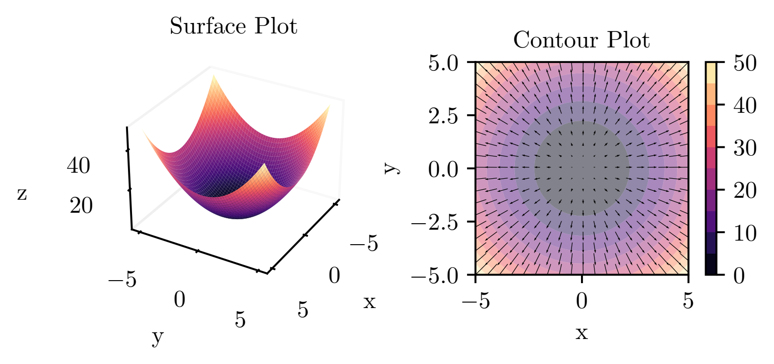
February 4, 2024

IIT Gandhinagar

Revision

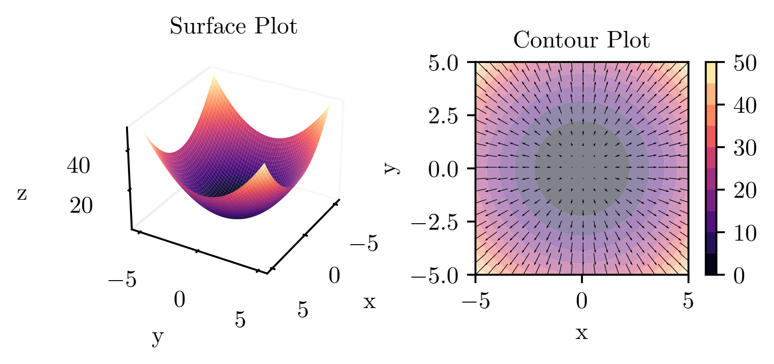
Contour Plot And Gradients

$z = f(x, y) = x^2 + y^2$



Contour Plot And Gradients

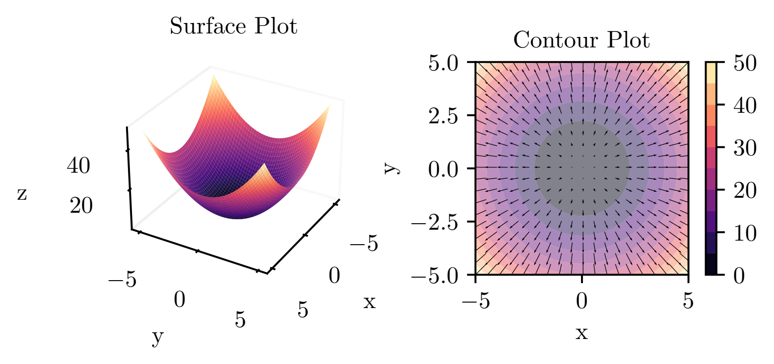
$z = f(x, y) = x^2 + y^2$



Gradient denotes the direction of steepest ascent or the direction in which there is a maximum increase in f(x,y)

Contour Plot And Gradients

$z = f(x, y) = x^2 + y^2$



Gradient denotes the direction of steepest ascent or the direction in which there is a maximum increase in f(x,y)

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$$

Introduction

Optimization algorithms

- We often want to minimize/maximize a function

Optimization algorithms

- We often want to minimize/maximize a function
- We wanted to minimize the cost function:

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

Optimization algorithms

- We often want to minimize/maximize a function
- We wanted to minimize the cost function:

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

- Note, here θ is the parameter vector

Optimization algorithms

- In general, we have following components:

Optimization algorithms

- In general, we have following components:
- Maximize or Minimize a function subject to some constraints

Optimization algorithms

- In general, we have following components:
- Maximize or Minimize a function subject to some constraints
- Today, we will focus on unconstrained optimization (no constraints)

Optimization algorithms

- In general, we have following components:
- Maximize or Minimize a function subject to some constraints
- Today, we will focus on unconstrained optimization (no constraints)
- We will focus on minimization

Optimization algorithms

- In general, we have following components:
- Maximize or Minimize a function subject to some constraints
- Today, we will focus on unconstrained optimization (no constraints)
- We will focus on minimization
- Goal:

$$\theta^* = \arg\min_{\theta} f(\theta) \tag{2}$$

Introduction

- Gradient descent is an optimization algorithm

Introduction

- Gradient descent is an optimization algorithm
- It is used to find the minimum of a function in unconstrained settings

Introduction

- Gradient descent is an optimization algorithm
- It is used to find the minimum of a function in unconstrained settings
- It is an iterative algorithm

Introduction

- Gradient descent is an optimization algorithm
- It is used to find the minimum of a function in unconstrained settings
- It is an iterative algorithm
- It is a first order optimization algorithm

Introduction

- Gradient descent is an optimization algorithm
- It is used to find the minimum of a function in unconstrained settings
- It is an iterative algorithm
- It is a first order optimization algorithm
- It is a local search algorithm/greedy

Gradient Descent Algorithm

1. Initialize θ to some random value

Gradient Descent Algorithm

1. Initialize θ to some random value
2. Compute the gradient of the cost function at θ, $\nabla f(\theta)$

Gradient Descent Algorithm

1. Initialize θ to some random value
2. Compute the gradient of the cost function at θ, $\nabla f(\theta)$
3. For Iteration i ($i = 1, 2, \ldots$) or until convergence:

Gradient Descent Algorithm

1. Initialize θ to some random value
2. Compute the gradient of the cost function at θ, $\nabla f(\theta)$
3. For Iteration i $(i = 1, 2, \ldots)$ or until convergence:
    - $\theta_i \leftarrow \theta_{i-1} - \alpha \nabla f(\theta_{i-1})$

Taylor's Series

Taylor's Series

- Taylor's series is a way to approximate a function $f(x)$ around a point x_0 using a polynomial

Taylor's Series

- Taylor's series is a way to approximate a function $f(x)$ around a point x_0 using a polynomial
- The polynomial is given by

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots \quad (3)$$

Taylor's Series

- Taylor's series is a way to approximate a function $f(x)$ around a point x_0 using a polynomial
- The polynomial is given by

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots \quad (3)$$

- The vector form of the above equation is given by:

$$f(\vec{x}) = f(\vec{x_0}) + \nabla f(\vec{x_0})^T(\vec{x} - \vec{x_0}) + \frac{1}{2}(\vec{x} - \vec{x_0})^T \nabla^2 f(\vec{x_0})(\vec{x} - \vec{x_0}) + \dots \quad (4)$$

Taylor's Series

- Taylor's series is a way to approximate a function $f(x)$ around a point x_0 using a polynomial
- The polynomial is given by

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots \quad (3)$$

- The vector form of the above equation is given by:

$$f(\vec{x}) = f(\vec{x_0}) + \nabla f(\vec{x_0})^T(\vec{x} - \vec{x_0}) + \frac{1}{2}(\vec{x} - \vec{x_0})^T \nabla^2 f(\vec{x_0})(\vec{x} - \vec{x_0}) + \dots \quad (4)$$

- where $\nabla^2 f(\vec{x_0})$ is the Hessian matrix and $\nabla f(\vec{x_0})$ is the gradient vector

Taylor's Series

- Let us consider $f(x) = \cos(x)$ and $x_0 = 0$

Taylor's Series

- Let us consider $f(x) = \cos(x)$ and $x_0 = 0$
- Then, we have:

Taylor's Series

- Let us consider $f(x) = \cos(x)$ and $x_0 = 0$
- Then, we have:
- $f(x_0) = \cos(0) = 1$

Taylor's Series

- Let us consider $f(x) = \cos(x)$ and $x_0 = 0$
- Then, we have:
- $f(x_0) = \cos(0) = 1$
- $f'(x_0) = -\sin(0) = 0$

Taylor's Series

- Let us consider $f(x) = \cos(x)$ and $x_0 = 0$
- Then, we have:
- $f(x_0) = \cos(0) = 1$
- $f'(x_0) = -\sin(0) = 0$
- $f''(x_0) = -\cos(0) = -1$

Taylor's Series

- Let us consider $f(x) = \cos(x)$ and $x_0 = 0$
- Then, we have:
- $f(x_0) = \cos(0) = 1$
- $f'(x_0) = -\sin(0) = 0$
- $f''(x_0) = -\cos(0) = -1$
- We can write the second order Taylor's series as:

Taylor's Series

- Let us consider $f(x) = \cos(x)$ and $x_0 = 0$
- Then, we have:
- $f(x_0) = \cos(0) = 1$
- $f'(x_0) = -\sin(0) = 0$
- $f''(x_0) = -\cos(0) = -1$
- We can write the second order Taylor's series as:
- $f(x) = 1 + 0(x - 0) + \frac{-1}{2!}(x - 0)^2 = 1 - \frac{x^2}{2}$

Taylor's series

- Let us consider another example: $f(x) = x^2 + 2$ and $x_0 = 2$

Taylor's series

- Let us consider another example: $f(x) = x^2 + 2$ and $x_0 = 2$
- Question: How does the first order Taylor's series approximation look like?

Taylor's series

- Let us consider another example: $f(x) = x^2 + 2$ and $x_0 = 2$
- Question: How does the first order Taylor's series approximation look like?
- First order Taylor's series approximation is given by:

Taylor's series

- Let us consider another example: $f(x) = x^2 + 2$ and $x_0 = 2$
- Question: How does the first order Taylor's series approximation look like?
- First order Taylor's series approximation is given by:
- $f(x) = f(x_0) + f'(x_0)(x - x_0) = 6 + 4(x - 2) = 4x - 2$

Taylor's Series (Alternative form)

- We have:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots \quad (5)$$

Taylor's Series (Alternative form)

- We have:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots \quad (5)$$

- Let us consider $x = x_0 + \Delta x$ where Δx is a small quantity

Taylor's Series (Alternative form)

- We have:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots \quad (5)$$

- Let us consider $x = x_0 + \Delta x$ where Δx is a small quantity
- Then, we have:

$$f(x_0 + \Delta x) = f(x_0) + \frac{f'(x_0)}{1!}\Delta x + \frac{f''(x_0)}{2!}\Delta x^2 + \dots \quad (6)$$

Taylor's Series (Alternative form)

- We have:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots \quad (5)$$

- Let us consider $x = x_0 + \Delta x$ where Δx is a small quantity
- Then, we have:

$$f(x_0 + \Delta x) = f(x_0) + \frac{f'(x_0)}{1!}\Delta x + \frac{f''(x_0)}{2!}\Delta x^2 + \dots \quad (6)$$

- Let us assume Δx is small enough such that Δx^2 and higher order terms can be ignored

Taylor's Series (Alternative form)

- We have:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots \quad (5)$$

- Let us consider $x = x_0 + \Delta x$ where Δx is a small quantity
- Then, we have:

$$f(x_0 + \Delta x) = f(x_0) + \frac{f'(x_0)}{1!}\Delta x + \frac{f''(x_0)}{2!}\Delta x^2 + \dots \quad (6)$$

- Let us assume Δx is small enough such that Δx^2 and higher order terms can be ignored
- Then, we have: $f(x_0 + \Delta x) \approx f(x_0) + \frac{f'(x_0)}{1!}\Delta x$

Taylor's Series to Gradient Descent

- Then, we have: $f(x_0 + \Delta x) \approx f(x_0) + \frac{f'(x_0)}{1!}\Delta x$

Taylor's Series to Gradient Descent

- Then, we have: $f(x_0 + \Delta x) \approx f(x_0) + \frac{f'(x_0)}{1!}\Delta x$
- Or, in vector form: $f(\vec{x_0} + \Delta\vec{x}) \approx f(\vec{x_0}) + \nabla f(\vec{x_0})^T \Delta\vec{x}$

Taylor's Series to Gradient Descent

- Then, we have: $f(x_0 + \Delta x) \approx f(x_0) + \frac{f'(x_0)}{1!}\Delta x$
- Or, in vector form: $f(\vec{x_0} + \Delta\vec{x}) \approx f(\vec{x_0}) + \nabla f(\vec{x_0})^T \Delta\vec{x}$
- Goal: Find $\Delta\vec{x}$ such that $f(\vec{x_0} + \Delta\vec{x})$ is minimized

Taylor's Series to Gradient Descent

- Then, we have: $f(x_0 + \Delta x) \approx f(x_0) + \frac{f'(x_0)}{1!}\Delta x$
- Or, in vector form: $f(\vec{x_0} + \Delta\vec{x}) \approx f(\vec{x_0}) + \nabla f(\vec{x_0})^T \Delta\vec{x}$
- Goal: Find $\Delta\vec{x}$ such that $f(\vec{x_0} + \Delta\vec{x})$ is minimized
- This is equivalent to minimizing $f(\vec{x_0}) + \nabla f(\vec{x_0})^T \Delta\vec{x}$

Taylor's Series to Gradient Descent

- Then, we have: $f(x_0 + \Delta x) \approx f(x_0) + \frac{f'(x_0)}{1!}\Delta x$
- Or, in vector form: $f(\vec{x_0} + \Delta\vec{x}) \approx f(\vec{x_0}) + \nabla f(\vec{x_0})^T \Delta\vec{x}$
- Goal: Find $\Delta\vec{x}$ such that $f(\vec{x_0} + \Delta\vec{x})$ is minimized
- This is equivalent to minimizing $f(\vec{x_0}) + \nabla f(\vec{x_0})^T \Delta\vec{x}$
- This happens when vectors $\nabla f(\vec{x_0})$ and $\Delta\vec{x}$ are at phase angle of 180°

Taylor's Series to Gradient Descent

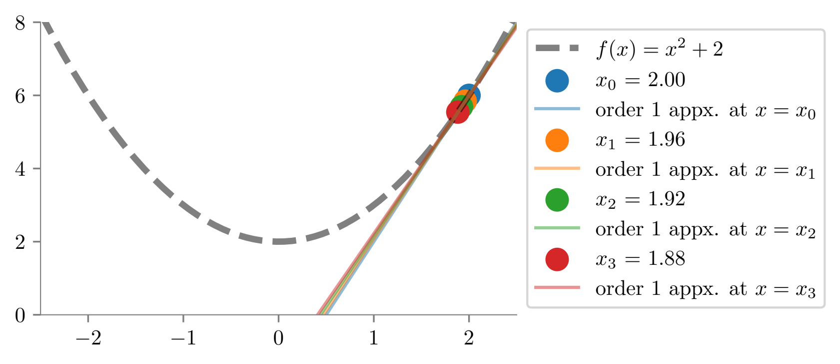
- Then, we have: $f(x_0 + \Delta x) \approx f(x_0) + \frac{f'(x_0)}{1!}\Delta x$
- Or, in vector form: $f(\vec{x_0} + \Delta\vec{x}) \approx f(\vec{x_0}) + \nabla f(\vec{x_0})^T \Delta\vec{x}$
- Goal: Find $\Delta\vec{x}$ such that $f(\vec{x_0} + \Delta\vec{x})$ is minimized
- This is equivalent to minimizing $f(\vec{x_0}) + \nabla f(\vec{x_0})^T \Delta\vec{x}$
- This happens when vectors $\nabla f(\vec{x_0})$ and $\Delta\vec{x}$ are at phase angle of 180°
- This happens when $\Delta\vec{x} = -\alpha\nabla f(\vec{x_0})$ where α is a scalar

Taylor's Series to Gradient Descent

- Then, we have: $f(x_0 + \Delta x) \approx f(x_0) + \frac{f'(x_0)}{1!}\Delta x$
- Or, in vector form: $f(\vec{x_0} + \Delta\vec{x}) \approx f(\vec{x_0}) + \nabla f(\vec{x_0})^T \Delta\vec{x}$
- Goal: Find $\Delta\vec{x}$ such that $f(\vec{x_0} + \Delta\vec{x})$ is minimized
- This is equivalent to minimizing $f(\vec{x_0}) + \nabla f(\vec{x_0})^T \Delta\vec{x}$
- This happens when vectors $\nabla f(\vec{x_0})$ and $\Delta\vec{x}$ are at phase angle of 180°
- This happens when $\Delta\vec{x} = -\alpha\nabla f(\vec{x_0})$ where α is a scalar
- This is the gradient descent algorithm: $\vec{x_1} = \vec{x_0} - \alpha\nabla f(\vec{x_0})$

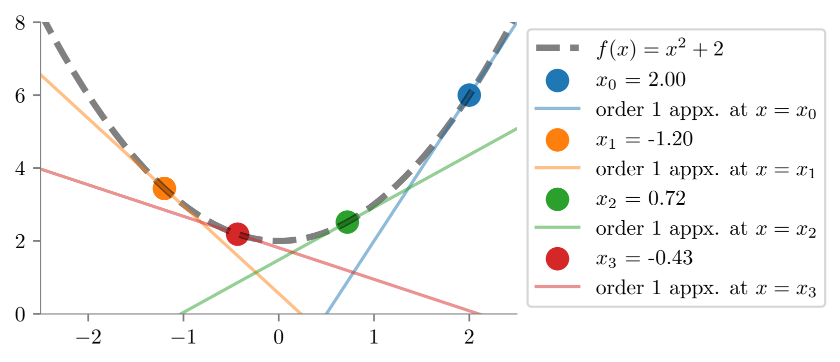
Effect of learning rate

Low learning rate $\alpha = 0.01$: Converges slowly



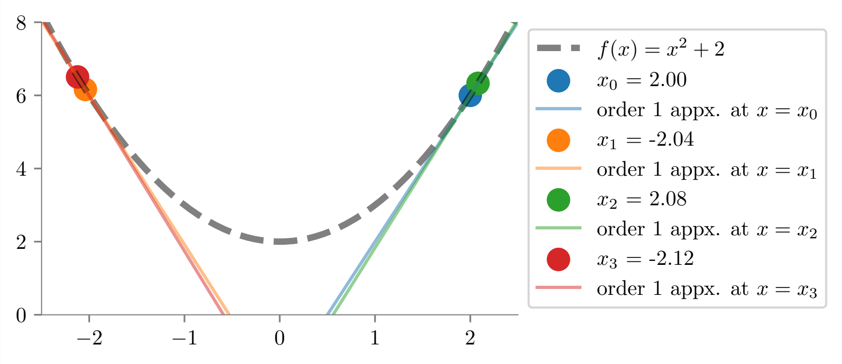
Effect of learning rate

High learning rate $\alpha = 0.8$: Converges quickly, but might overshoot



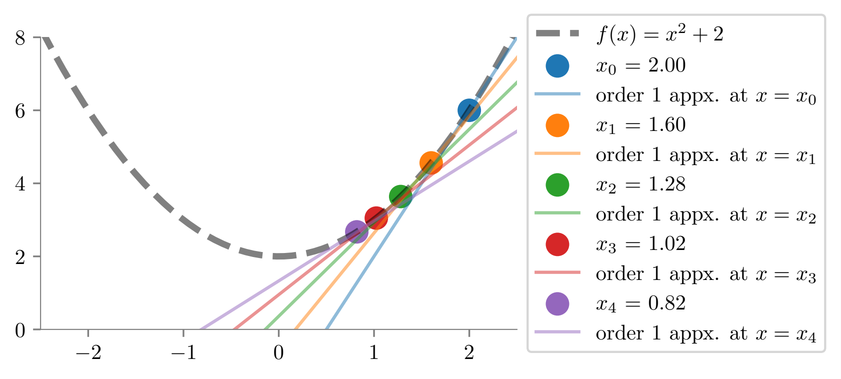
Effect of learning rate

Very high learning rate $\alpha = 1.01$: Diverges



Effect of learning rate

Appropriate learning rate $\alpha = 0.1$



Gradient Descent for linear regression

Some commonly confused terms

- **Loss function** is usually a function defined on a data point, prediction and label, and measures the penalty.

Some commonly confused terms

- **Loss function** is usually a function defined on a data point, prediction and label, and measures the penalty.
- square loss $l\left(f\left(x_i|\theta\right), y_i\right) = \left(f\left(x_i|\theta\right) - y_i\right)^2$, used in linear regression

Some commonly confused terms

- **Loss function** is usually a function defined on a data point, prediction and label, and measures the penalty.
- square loss $l\left(f\left(x_i|\theta\right), y_i\right) = \left(f\left(x_i|\theta\right) - y_i\right)^2$, used in linear regression
- **Cost function** is usually more general. It might be a sum of loss functions over your training set plus some model complexity penalty (regularization). For example:

Some commonly confused terms

- **Loss function** is usually a function defined on a data point, prediction and label, and measures the penalty.
- square loss $l\left(f\left(x_i|\theta\right), y_i\right) = \left(f\left(x_i|\theta\right) - y_i\right)^2$, used in linear regression
- **Cost function** is usually more general. It might be a sum of loss functions over your training set plus some model complexity penalty (regularization). For example:
- Mean Squared Error $MSE(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(f\left(x_i|\theta\right) - y_i\right)^2$

Some commonly confused terms

- **Loss function** is usually a function defined on a data point, prediction and label, and measures the penalty.
- square loss $l\left(f\left(x_i|\theta\right), y_i\right) = \left(f\left(x_i|\theta\right) - y_i\right)^2$, used in linear regression
- **Cost function** is usually more general. It might be a sum of loss functions over your training set plus some model complexity penalty (regularization). For example:
- Mean Squared Error $MSE(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(f\left(x_i|\theta\right) - y_i\right)^2$
- **Objective function** is the most general term for any function that you optimize during training.

Gradient Descent : Example

Learn $y = \theta_0 + \theta_1 x$ on following dataset, using gradient descent where initially $(\theta_0, \theta_1) = (4, 0)$ and step-size, $\alpha = 0.1$, for 2 iterations.

x	**y**
1	1
2	2
3	3

Gradient Descent : Example

Our predictor, $\hat{y} = \theta_0 + \theta_1 x$

Error for i^{th} datapoint, $\epsilon_i = y_i - \hat{y}_i$

$\epsilon_1 = 1 - \theta_0 - \theta_1$

$\epsilon_2 = 2 - \theta_0 - 2\theta_1$

$\epsilon_3 = 3 - \theta_0 - 3\theta_1$

$$\text{MSE} = \frac{\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2}{3} = \frac{14 + 3\theta_0^2 + 14\theta_1^2 - 12\theta_0 - 28\theta_1 + 12\theta_0\theta_1}{3}$$

Difference between SSE and MSE

$$\sum \epsilon_i^2 \textit{ increases as the number of examples increase}$$

So, we use MSE

$$MSE = \frac{1}{n} \sum \epsilon_i^2$$

Here n denotes the number of samples

Gradient Descent : Example

$$\frac{\partial MSE}{\partial \theta_0} = \frac{2\sum_i (y_i - \theta_0 - \theta_1 x_i)(-1)}{N} = \frac{2\sum_i \epsilon_i (-1)}{N}$$

$$\frac{\partial MSE}{\partial \theta_1} = \frac{2\sum_i (y_i - \theta_0 - \theta_1 x_i)(-x_i)}{N} = \frac{2\sum_i \epsilon_i (-x_i)}{N}$$

Gradient Descent : Example

Iteration 1

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

Gradient Descent : Example

Iteration 1

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_0 = 4 - 0.2 \frac{((1-(4+0))(-1)+(2-(4+0))(-1)+(3-(4+0))(-1))}{3}$$

$$\theta_0 = 3.6$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

Gradient Descent : Example

Iteration 1

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_0 = 4 - 0.2 \frac{((1-(4+0))(-1)+(2-(4+0))(-1)+(3-(4+0))(-1))}{3}$$

$$\theta_0 = 3.6$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

$$\theta_1 = 0 - 0.2 \frac{((1-(4+0))(-1)+(2-(4+0))(-2)+(3-(4+0))(-3))}{3}$$

$$\theta_1 = -0.67$$

Gradient Descent : Example

Iteration 2

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

Gradient Descent : Example

Iteration 2

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_0 = 3.6 - 0.2\frac{((1-(3.6-0.67))(-1)+(2-(3.6-0.67\times2))(-1)+(3-(3.6-0.67\times3))(-1))}{3}$$

$$\theta_0 = 3.54$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

Gradient Descent : Example

Iteration 2

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_0 = 3.6 - 0.2 \frac{((1-(3.6-0.67))(-1)+(2-(3.6-0.67\times 2))(-1)+(3-(3.6-0.67\times 3))(-1))}{3}$$

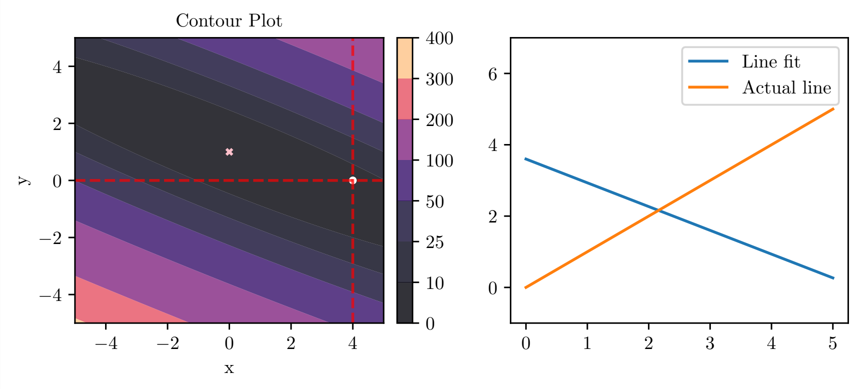
$$\theta_0 = 3.54$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

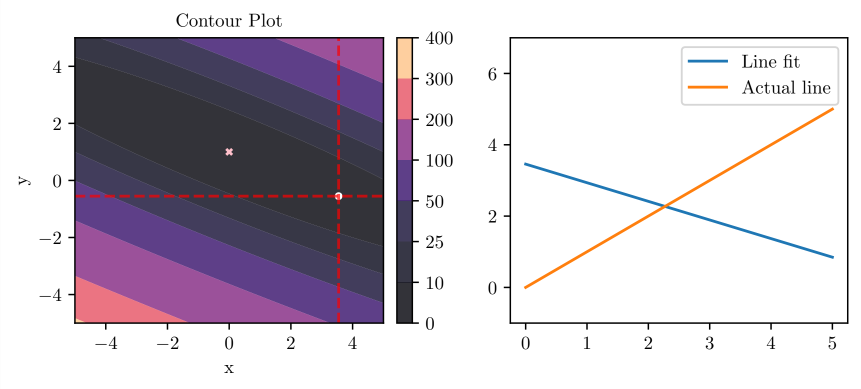
$$\theta_0 = 3.6 - 0.2 \frac{((1-(3.6-0.67))(-1)+(2-(3.6-0.67\times 2))(-2)+(3-(3.6-0.67\times 3))(-3))}{3}$$

$$\theta_0 = -0.55$$

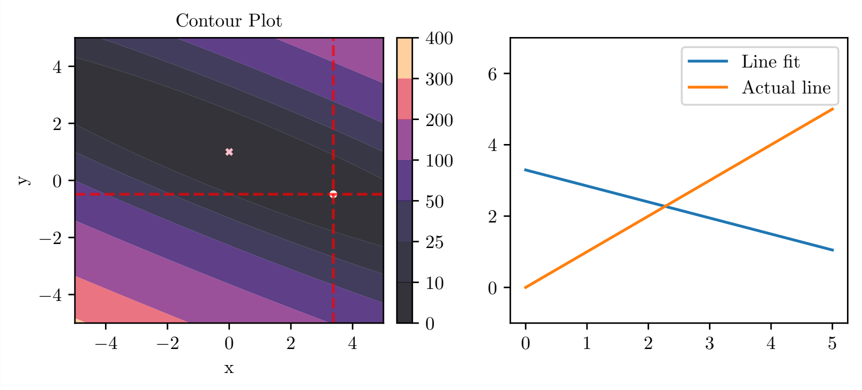
Gradient Descent : Example (Iteraion 0)



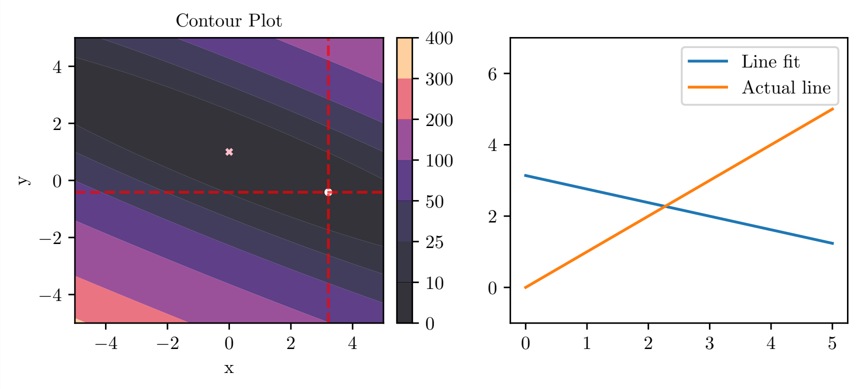
Gradient Descent : Example (Iteraion 2)



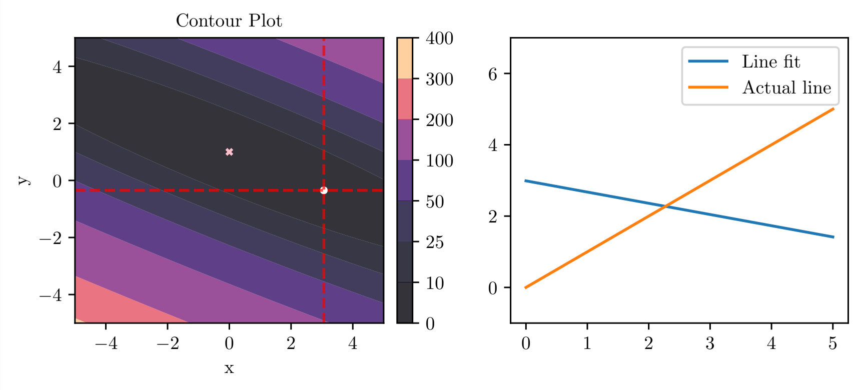
Gradient Descent : Example (Iteraion 4)



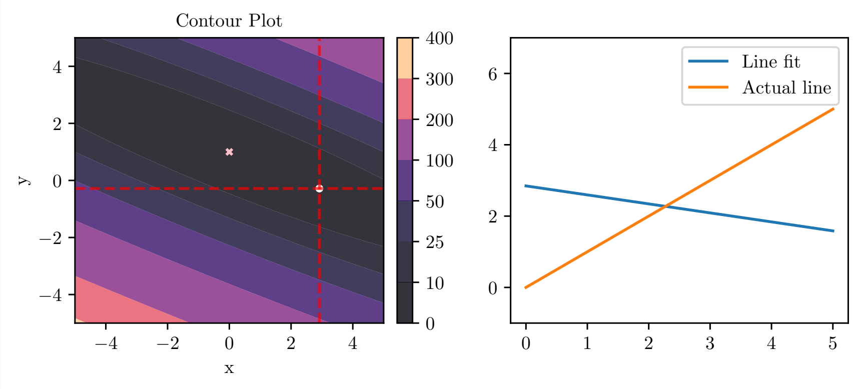
Gradient Descent : Example (Iteraion 6)



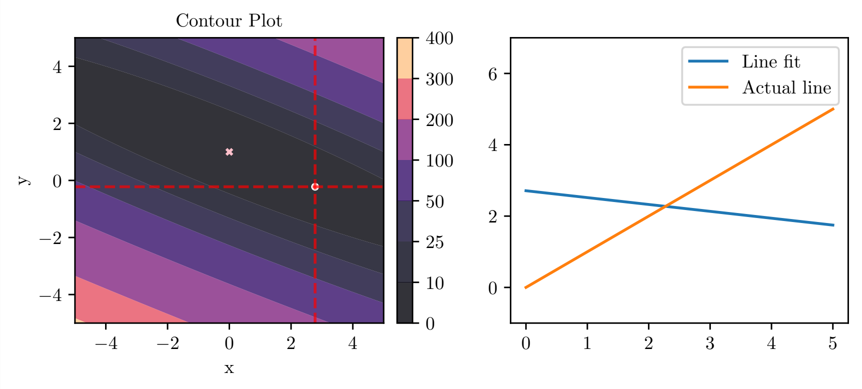
Gradient Descent : Example (Iteraion 8)



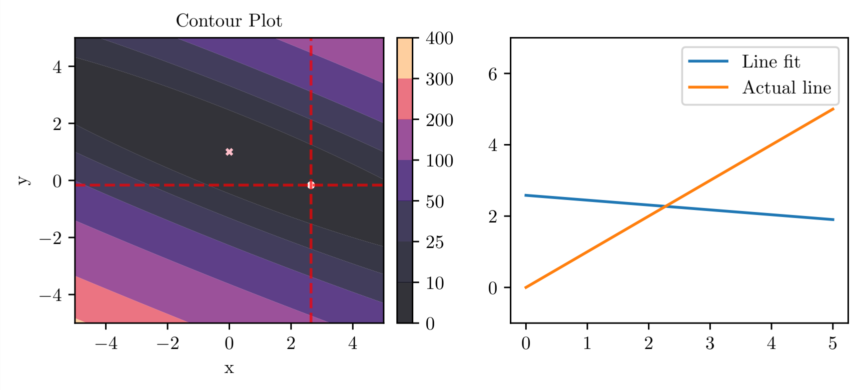
Gradient Descent : Example (Iteraion 10)



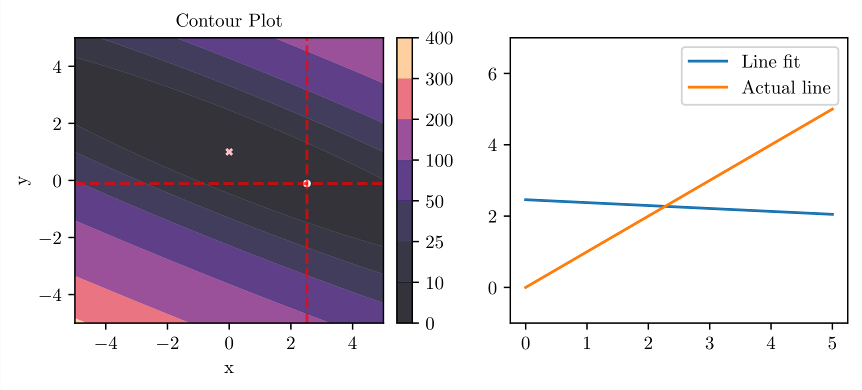
Gradient Descent : Example (Iteraion 12)



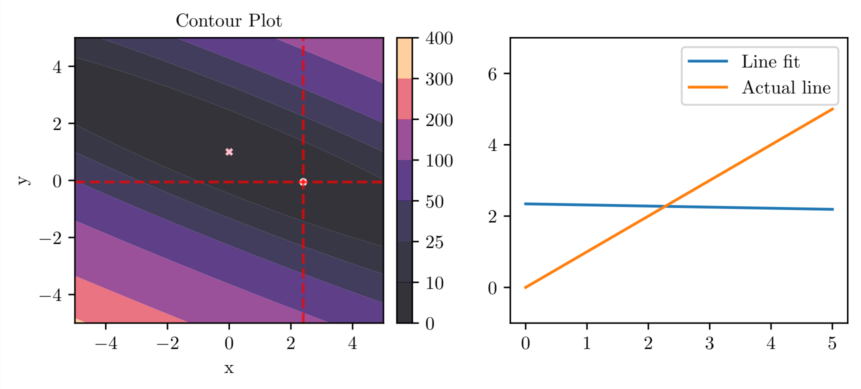
Gradient Descent : Example (Iteraion 14)



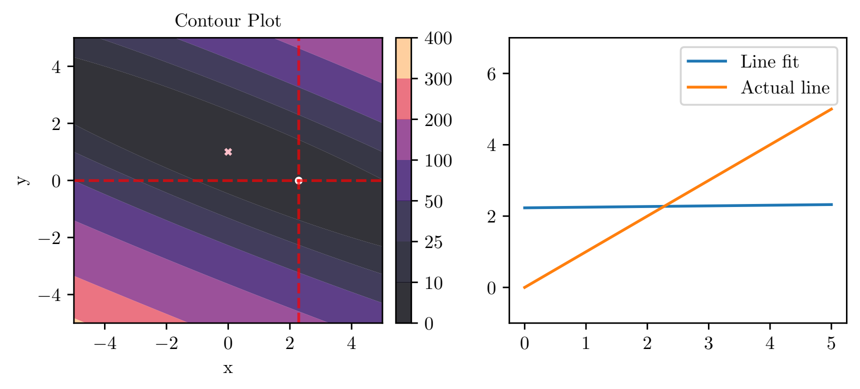
Gradient Descent : Example (Iteraion 16)



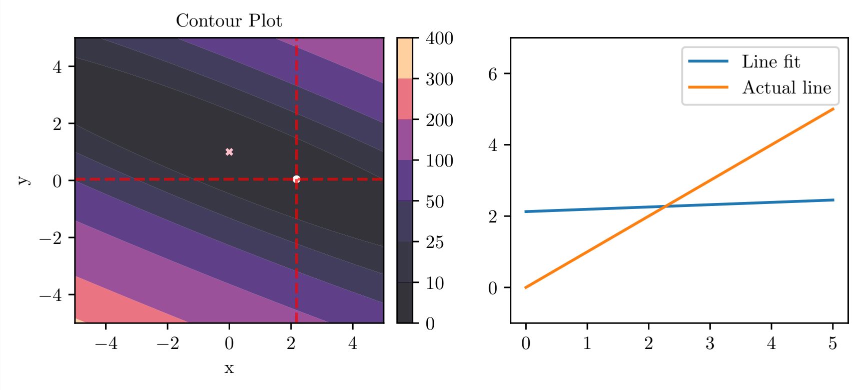
Gradient Descent : Example (Iteraion 18)



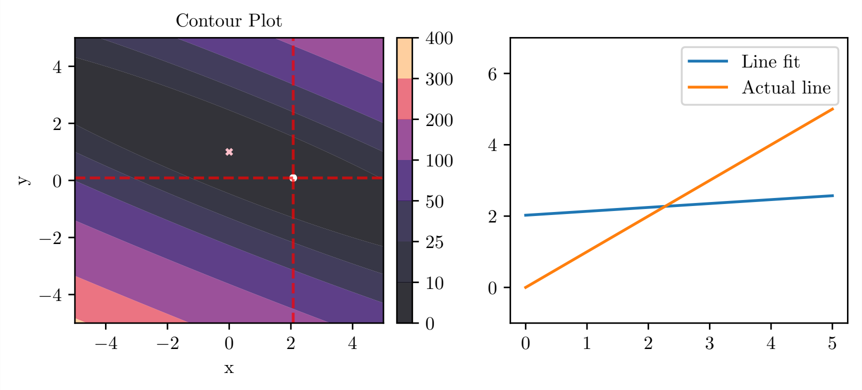
Gradient Descent : Example (Iteraion 20)



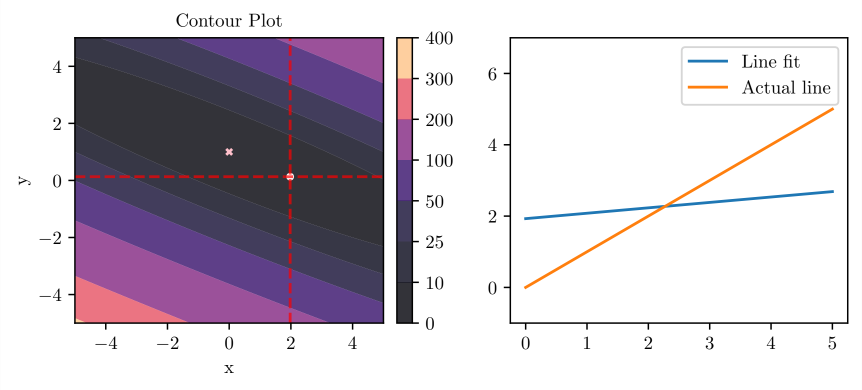
Gradient Descent : Example (Iteraion 22)



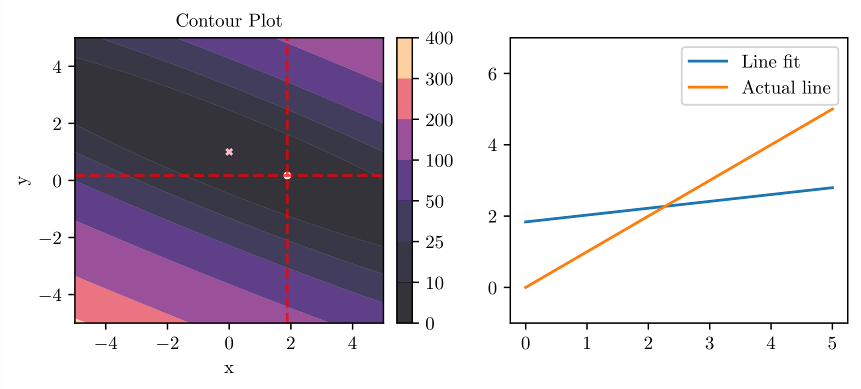
Gradient Descent : Example (Iteraion 24)



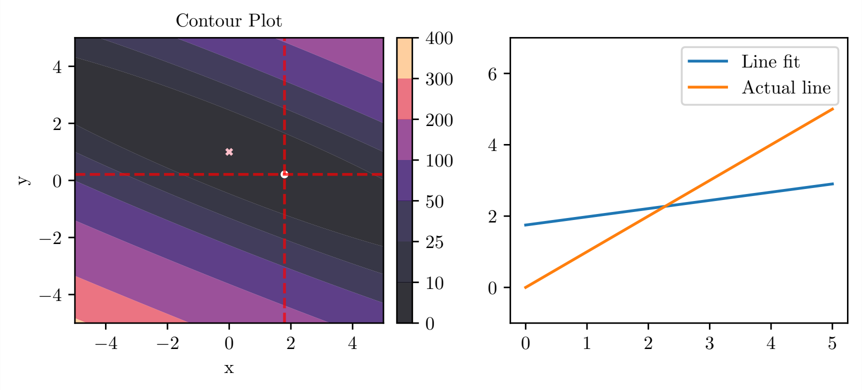
Gradient Descent : Example (Iteraion 26)



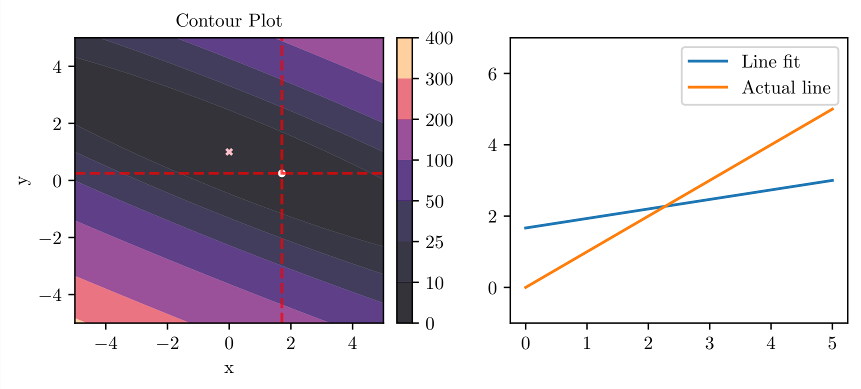
Gradient Descent : Example (Iteraion 28)



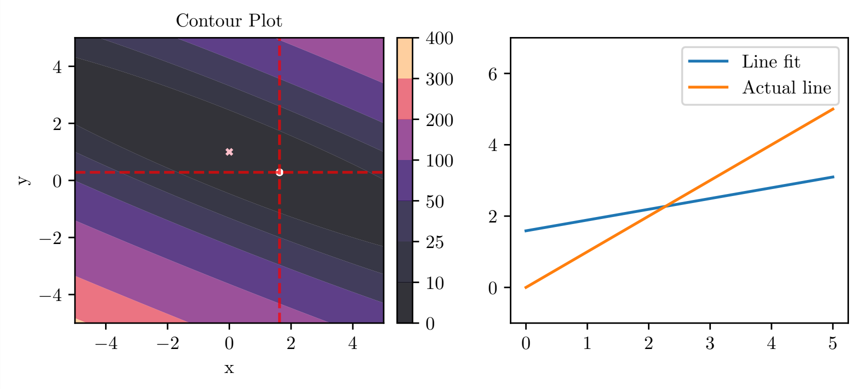
Gradient Descent : Example (Iteraion 30)



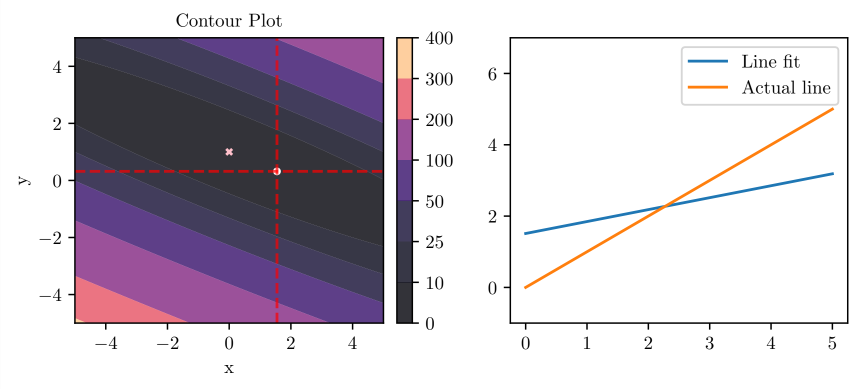
Gradient Descent : Example (Iteraion 32)



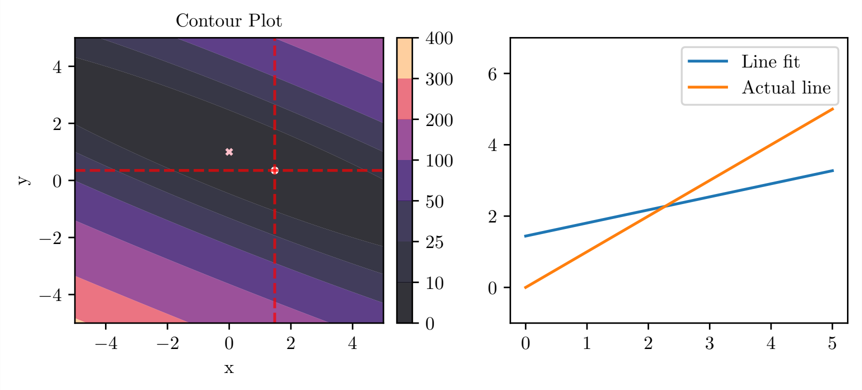
Gradient Descent : Example (Iteraion 34)



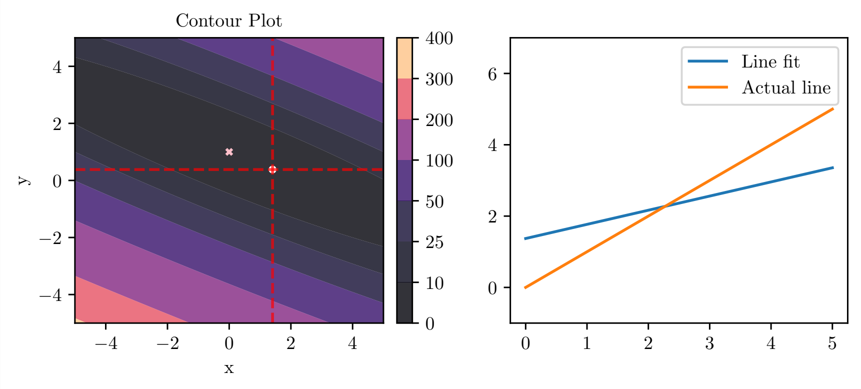
Gradient Descent : Example (Iteraion 36)



Gradient Descent : Example (Iteraion 38)



Gradient Descent : Example (Iteraion 40)



Iteration v/s Epcohs for gradient descent

- Iteration: Each time you update the parameters of the model

Iteration v/s Epcohs for gradient descent

- Iteration: Each time you update the parameters of the model
- Epoch: Each time you have seen all the set of examples

Gradient Descent (GD)

- Dataset: $D = \{(X, y)\}$ of size N
- Initialize θ
- For epoch e in $[1, E]$
  - Predict $\hat{y} = pred(X, \theta)$
  - Compute loss: $J(\theta) = loss(y, \hat{y})$
  - Compute gradient: $\nabla J(\theta) = grad(J)(\theta)$
  - Update: $\theta = \theta - \alpha \nabla J(\theta)$

Stochastic Gradient Descent (SGD)

- Dataset: $D = \{(X, y)\}$ of size N
- Initialize θ
- For epoch e in $[1, E]$
  - Shuffle D
  - For i in $[1, N]$
    - Predict $\hat{y}_i = pred(X_i, \theta)$
    - Compute loss: $J(\theta) = loss(y_i, \hat{y}_i)$
    - Compute gradient: $\nabla J(\theta) = grad(J)(\theta)$
    - Update: $\theta = \theta - \alpha \nabla J(\theta)$

Mini-Batch Gradient Descent (MBGD)

- Dataset: $D = \{(X, y)\}$ of size N
- Initialize θ
- For epoch e in $[1, E]$
    - Shuffle D
    - $Batches = make\_batches(D, B)$
    - For b in $Batches$
        - $X\_b, y\_b = b$
        - Predict $\hat{y\_b} = pred(X\_b, \theta)$
        - Compute loss: $J(\theta) = loss(y\_b, \hat{y\_b})$
        - Compute gradient: $\nabla J(\theta) = grad(J)(\theta)$
        - Update: $\theta = \theta - \alpha \nabla J(\theta)$

Gradient Descent vs SGD

Vanilla Gradient Descent

- in Vanilla (Batch) gradient descent: We update params after going through all the data

Gradient Descent vs SGD

Vanilla Gradient Descent

- in Vanilla (Batch) gradient descent: We update params after going through all the data
- Smooth curve for Iteration vs Cost

Gradient Descent vs SGD

Vanilla Gradient Descent

- in Vanilla (Batch) gradient descent: We update params after going through all the data
- Smooth curve for Iteration vs Cost
- For a single update, it needs to compute the gradient over all the samples. Hence takes more time

Gradient Descent vs SGD

Vanilla Gradient Descent

- in Vanilla (Batch) gradient descent: We update params after going through all the data
- Smooth curve for Iteration vs Cost
- For a single update, it needs to compute the gradient over all the samples. Hence takes more time

Gradient Descent vs SGD

Vanilla Gradient Descent

- in Vanilla (Batch) gradient descent: We update params after going through all the data
- Smooth curve for Iteration vs Cost
- For a single update, it needs to compute the gradient over all the samples. Hence takes more time

Stochastic Gradient Descent

- In SGD, we update parameters after seeing each each point

Gradient Descent vs SGD

Vanilla Gradient Descent

- in Vanilla (Batch) gradient descent: We update params after going through all the data
- Smooth curve for Iteration vs Cost
- For a single update, it needs to compute the gradient over all the samples. Hence takes more time

Stochastic Gradient Descent

- In SGD, we update parameters after seeing each each point
- Noisier curve for iteration vs cost

Gradient Descent vs SGD

Vanilla Gradient Descent

- in Vanilla (Batch) gradient descent: We update params after going through all the data
- Smooth curve for Iteration vs Cost
- For a single update, it needs to compute the gradient over all the samples. Hence takes more time

Stochastic Gradient Descent

- In SGD, we update parameters after seeing each each point
- Noisier curve for iteration vs cost
- For a single update, it computes the gradient over one example. Hence lesser time

Stochastic Gradient Descent : Example

Learn $y = \theta_0 + \theta_1 x$ on following dataset, using SGD where initially $(\theta_0, \theta_1) = (4, 0)$ and step-size, $\alpha = 0.1$, for 1 epoch (3 iterations).

x	**y**
2	2
3	3
1	1

Stochastic Gradient Descent : Example

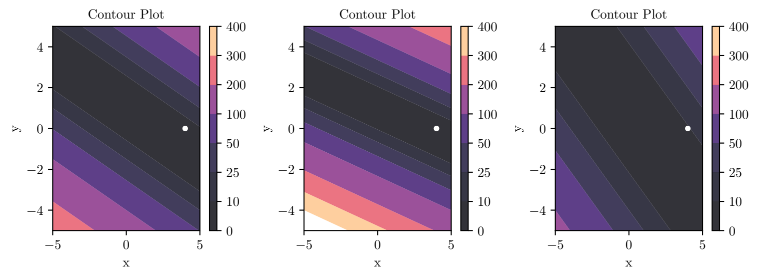
Our predictor, $\hat{y} = \theta_0 + \theta_1 x$

Error for i^{th} datapoint, $e_i = y_i - \hat{y}_i$
$\epsilon_1 = 2 - \theta_0 - 2\theta_1$
$\epsilon_2 = 3 - \theta_0 - 3\theta_1$
$\epsilon_3 = 1 - \theta_0 - \theta_1$

While using SGD, we compute the MSE using only 1 datapoint per iteration.
So MSE is ϵ_1^2 for iteration 1 and ϵ_2^2 for iteration 2.

Stochastic Gradient Descent : Example

Contour plot of the cost functions for the three datapoints



Stochastic Gradient Descent : Example

For Iteration i

$$\frac{\partial MSE}{\partial \theta_0} = 2\left(y_i - \theta_0 - \theta_1 x_i\right)(-1) = 2\epsilon_i\,(-1)$$

$$\frac{\partial MSE}{\partial \theta_1} = 2\left(y_i - \theta_0 - \theta_1 x_i\right)(-x_i) = 2\epsilon_i\,(-x_i)$$

Stochastic Gradient Descent : Example

Iteration 1

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

Stochastic Gradient Descent : Example

Iteration 1

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_0 = 4 - 0.1 \times 2 \times (2 - (4 + 0))(-1)$$

$$\theta_0 = 3.6$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

Stochastic Gradient Descent : Example

Iteration 1

$$\theta_0 = \theta_0 - \alpha\frac{\partial MSE}{\partial \theta_0}$$

$$\theta_0 = 4 - 0.1 \times 2 \times (2 - (4 + 0))\,(-1)$$

$$\theta_0 = 3.6$$

$$\theta_1 = \theta_1 - \alpha\frac{\partial MSE}{\partial \theta_1}$$

$$\theta_1 = 0 - 0.1 \times 2 \times (2 - (4 + 0))\,(-2)$$

$$\theta_1 = -0.8$$

Stochastic Gradient Descent : Example

Iteration 2

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

Stochastic Gradient Descent : Example

Iteration 2

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_0 = 3.6 - 0.1 \times 2 \times (3 - (3.6 - 0.8 \times 3))\,(-1)$$

$$\theta_0 = 3.96$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

Stochastic Gradient Descent : Example

Iteration 2

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_0 = 3.6 - 0.1 \times 2 \times (3 - (3.6 - 0.8 \times 3))\,(-1)$$

$$\theta_0 = 3.96$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

$$\theta_0 = -0.8 - 0.1 \times 2 \times (3 - (3.6 - 0.8 \times 3))\,(-3)$$

$$\theta_1 = 0.28$$

Stochastic Gradient Descent : Example

Iteration 3

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

Stochastic Gradient Descent : Example

Iteration 3

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_0 = 3.96 - 0.1 \times 2 \times (1 - (3.96 + 0.28 \times 1))(-1)$$

$$\theta_0 = 3.312$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

Stochastic Gradient Descent : Example

Iteration 3

$$\theta_0 = \theta_0 - \alpha \frac{\partial MSE}{\partial \theta_0}$$

$$\theta_0 = 3.96 - 0.1 \times 2 \times (1 - (3.96 + 0.28 \times 1))(-1)$$

$$\theta_0 = 3.312$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial MSE}{\partial \theta_1}$$

$$\theta_0 = 0.28 - 0.1 \times 2 \times (1 - (3.96 + 0.28 \times 1))(-1)$$

$$\theta_1 = -0.368$$

Stochastic gradient is an unbiased estimator of the true gradient

True Gradient

Based on Estimation Theory and Machine Learning by Florian Hartmann

- Let us say we have a dataset $\mathcal{D}$ containing input output pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$

True Gradient

Based on Estimation Theory and Machine Learning by Florian Hartmann

- Let us say we have a dataset $\mathcal{D}$ containing input output pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$
- We can define overall loss as:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} loss(f(x_i, \theta), y_i)$$

True Gradient

Based on Estimation Theory and Machine Learning by Florian Hartmann

- Let us say we have a dataset $\mathcal{D}$ containing input output pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$
- We can define overall loss as:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} loss(f(x_i, \theta), y_i)$$

- loss can be any loss function such as squared loss, cross-entropy loss etc.

$$loss(f(x_i, \theta), y_i) = (f(x_i, \theta) - y_i)^2$$

True Gradient

- The true gradient of the loss function is given by:

$$
\begin{aligned}
\nabla L &= \nabla \frac{1}{n} \sum_{i=1}^{n} \operatorname{loss}\left(f\left(x_i\right), y_i\right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \nabla \operatorname{loss}\left(f\left(x_i\right), y_i\right)
\end{aligned}
$$

True Gradient

- The true gradient of the loss function is given by:

$$\nabla L = \nabla \frac{1}{n} \sum_{i=1}^{n} \operatorname{loss}\left(f\left(x_i\right), y_i\right)$$
$$= \frac{1}{n} \sum_{i=1}^{n} \nabla \operatorname{loss}\left(f\left(x_i\right), y_i\right)$$

- The above is a consequence of linearity of the gradient operator.

Estimator for the true gradient

- In practice, we do not have access to the true gradient

Estimator for the true gradient

- In practice, we do not have access to the true gradient
- We can only estimate the true gradient using a subset of the data

Estimator for the true gradient

- In practice, we do not have access to the true gradient
- We can only estimate the true gradient using a subset of the data
- For SGD, we use a single example to estimate the true gradient, for mini-batch gradient descent, we use a mini-batch of examples to estimate the true gradient

Estimator for the true gradient

- In practice, we do not have access to the true gradient
- We can only estimate the true gradient using a subset of the data
- For SGD, we use a single example to estimate the true gradient, for mini-batch gradient descent, we use a mini-batch of examples to estimate the true gradient
- Let us say we have a sample: (x, y)

Estimator for the true gradient

- In practice, we do not have access to the true gradient
- We can only estimate the true gradient using a subset of the data
- For SGD, we use a single example to estimate the true gradient, for mini-batch gradient descent, we use a mini-batch of examples to estimate the true gradient
- Let us say we have a sample: (x, y)
- The estimated gradient is given by:

$$\nabla \tilde{L} = \nabla \operatorname{loss}(f(x), y)$$

Bias of the estimator

- One measure for the quality of an estimator $\tilde{X}$ is its bias or how far off its estimate is on average from the true value X :

$$\text{bias}(X) = \mathbb{E}[\tilde{X}] - X$$

Bias of the estimator

- One measure for the quality of an estimator $\tilde{X}$ is its bias or how far off its estimate is on average from the true value X :

$$\text{bias}(X) = \mathbb{E}[\tilde{X}] - X$$

- Using the rules of expectation, we can show that the expected value of the estimated gradient is the true gradient:

$$\begin{aligned}
\mathbb{E}[\nabla \tilde{L}] &= \sum_{i=1}^{n} \frac{1}{n} \nabla \operatorname{loss}\left(f\left(x_i\right), y_i\right) \\
&= \frac{1}{n} \nabla \sum_{i=1}^{n} \operatorname{loss}\left(f\left(x_i\right), y_i\right) \\
&= \nabla L
\end{aligned}$$

Bias of the estimator

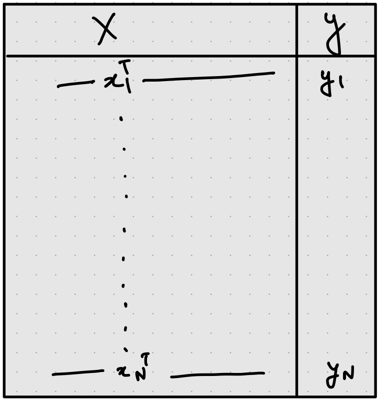
- One measure for the quality of an estimator $\tilde{X}$ is its bias or how far off its estimate is on average from the true value X :

$$\text{bias}(X) = \mathbb{E}[\tilde{X}] - X$$

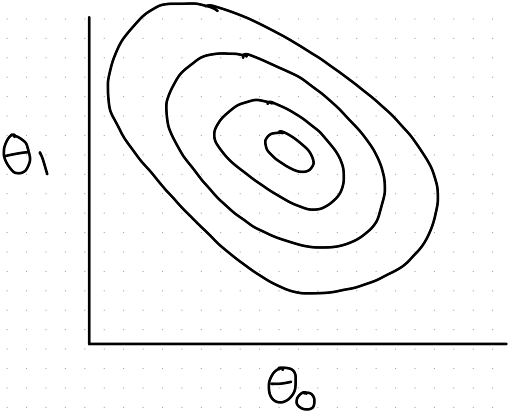
- Using the rules of expectation, we can show that the expected value of the estimated gradient is the true gradient:

$$\begin{aligned}\mathbb{E}[\nabla \tilde{L}] &= \sum_{i=1}^{n} \frac{1}{n} \nabla \operatorname{loss}\left(f\left(x_i\right), y_i\right) \\ &= \frac{1}{n} \nabla \sum_{i=1}^{n} \operatorname{loss}\left(f\left(x_i\right), y_i\right) \\ &= \nabla L\end{aligned}$$

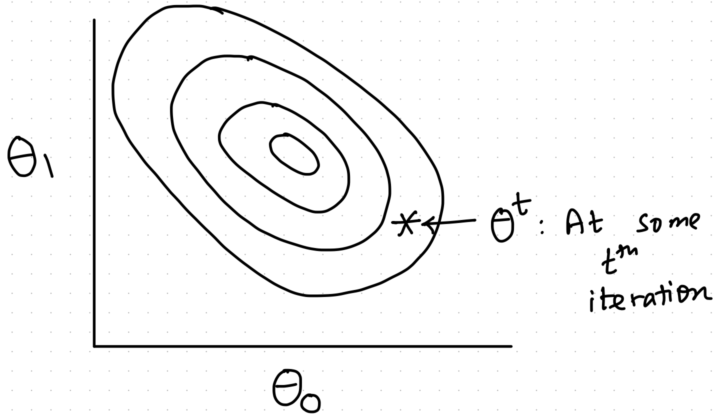
- Thus, the estimated gradient is an unbiased estimator of the true gradient


X
y
x_1^T
y_1
x_N^T
y_N

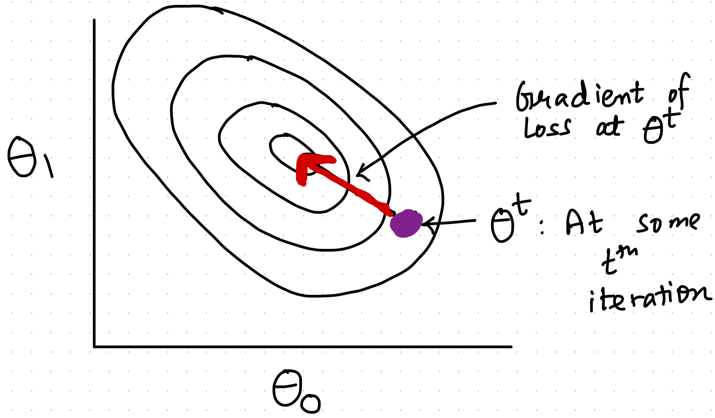
X	y	$\hat{y} = f(x, \theta)$
— x_1^T —	y_1	$\hat{y}_1$
$\vdots$		$\vdots$
— x_N^T —	y_N	$\hat{y}_N$



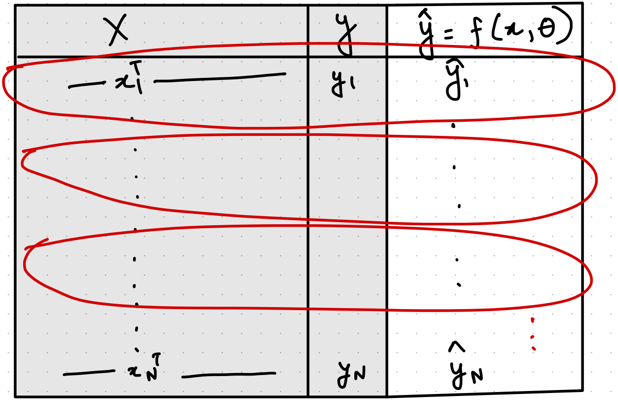
LOSS SURFACE OVER
6N' EXAMPLES



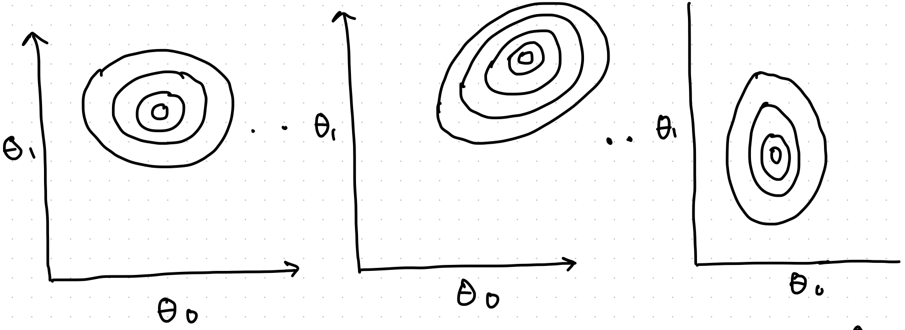
LOSS SURFACE OVER 6N EXAMPLES



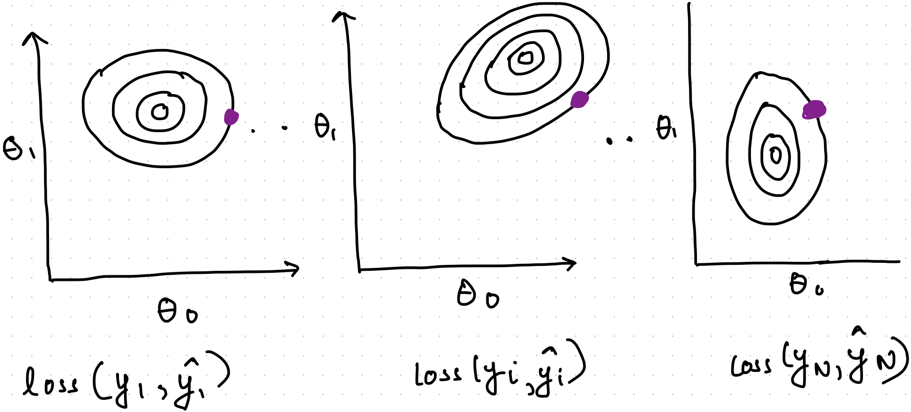
LOSS SURFACE OVER 6N° EXAMPLES



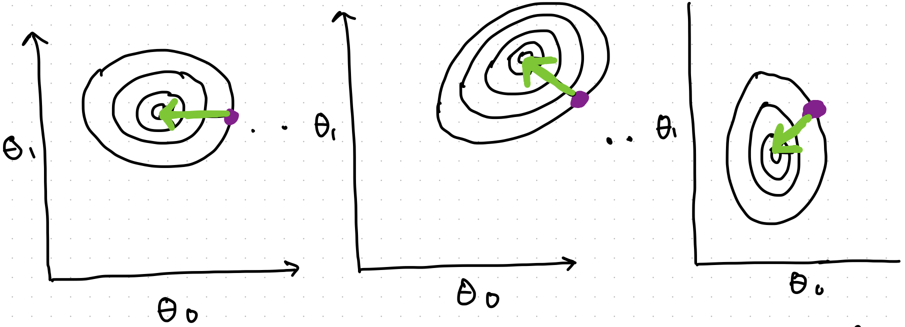
CONSIDER Individual data points to compute loss



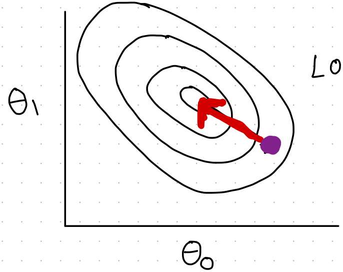
$loss(y_1, \hat{y}_1)$ $loss(y_i, \hat{y}_i)$ $loss(y_N, \hat{y}_N)$



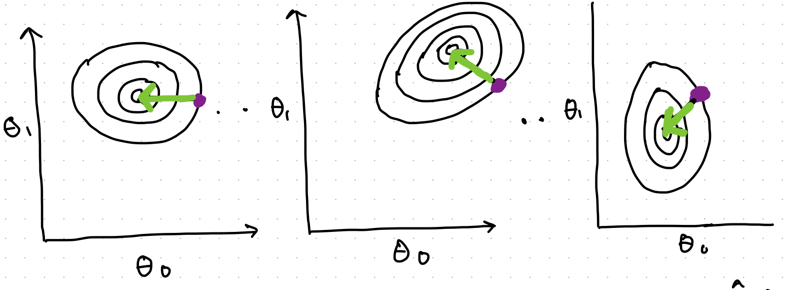
θ_1
θ_0
$loss(y_1, \hat{y}_1)$
θ_1
θ_0
$loss(y_i, \hat{y}_i)$
θ_1
θ_0
$loss(y_N, \hat{y}_N)$



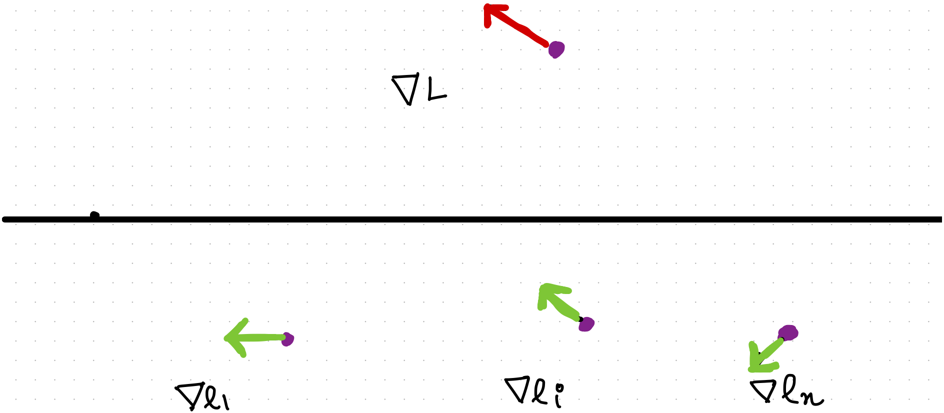
$loss(y_1, \hat{y}_1)$ . . . $loss(y_i, \hat{y}_i)$ . . . $loss(y_N, \hat{y}_N)$

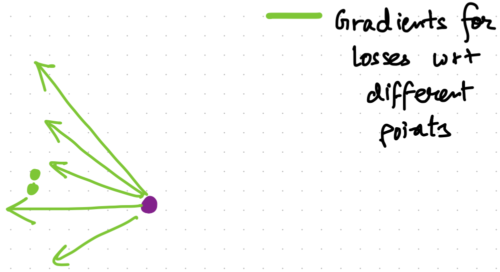


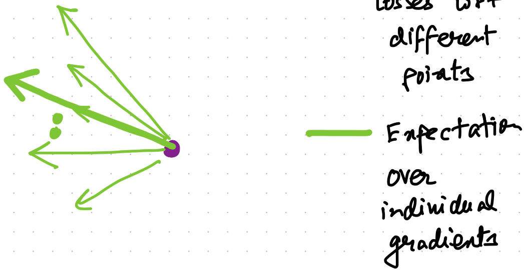
LOSS SURFACE OVER 'N' EXAMPLES

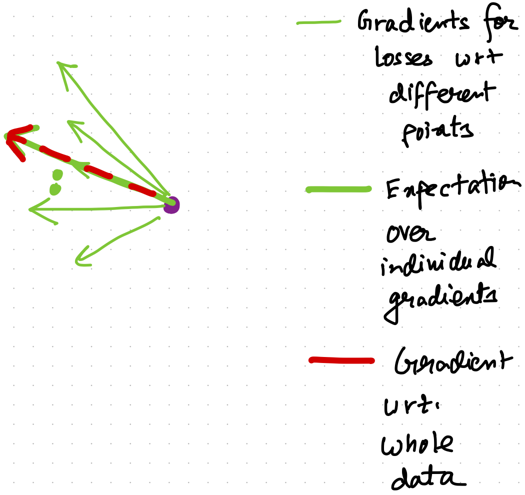


$loss(y_1, \hat{y}_1)$ . . $loss(y_i, \hat{y}_i)$ . . $loss(y_N, \hat{y}_N)$


∇L
$\nabla \ell_1$
$\nabla \ell_i$
$\nabla \ell_n$


Gradients for losses wrt different points


Gradients for losses wrt different points
Expectation over individual gradients


Gradients for losses wrt different points
Expectation over individual gradients
Gradient wrt. whole data

Time Complexity: Gradient Descent v/s Normal Equation for Linear Regression

Normal Equation

- Consider $X \in \mathcal{R}^{N \times D}$

Normal Equation

- Consider $X \in \mathcal{R}^{N \times D}$
- N examples and D dimensions

Normal Equation

- Consider $X \in \mathcal{R}^{N \times D}$
- N examples and D dimensions
- What is the time complexity of solving the normal equation $\hat{\theta} = (X^T X)^{-1} X^T y$?

Normal Equation

- X has dimensions $N \times D$, X^T has dimensions $D \times N$

Normal Equation

- X has dimensions $N \times D$, X^T has dimensions $D \times N$
- $X^T X$ is a matrix product of matrices of size: $D \times N$ and $N \times D$, which is $\mathcal{O}(D^2 N)$

Normal Equation

- X has dimensions $N \times D$, X^T has dimensions $D \times N$
- $X^T X$ is a matrix product of matrices of size: $D \times N$ and $N \times D$, which is $\mathcal{O}(D^2 N)$
- Inversion of $X^T X$ is an inversion of a $D \times D$ matrix, which is $\mathcal{O}(D^3)$

Normal Equation

- X has dimensions $N \times D$, X^T has dimensions $D \times N$
- $X^T X$ is a matrix product of matrices of size: $D \times N$ and $N \times D$, which is $\mathcal{O}(D^2 N)$
- Inversion of $X^T X$ is an inversion of a $D \times D$ matrix, which is $\mathcal{O}(D^3)$
- $X^T y$ is a matrix vector product of size $D \times N$ and $N \times 1$, which is $\mathcal{O}(DN)$

Normal Equation

- X has dimensions $N \times D$, X^T has dimensions $D \times N$
- $X^T X$ is a matrix product of matrices of size: $D \times N$ and $N \times D$, which is $\mathcal{O}(D^2 N)$
- Inversion of $X^T X$ is an inversion of a $D \times D$ matrix, which is $\mathcal{O}(D^3)$
- $X^T y$ is a matrix vector product of size $D \times N$ and $N \times 1$, which is $\mathcal{O}(DN)$
- $(X^T X)^{-1} X^T y$ is a matrix product of a $D \times D$ matrix and $D \times 1$ matrix, which is $\mathcal{O}(D^2)$

Normal Equation

- X has dimensions $N \times D$, X^T has dimensions $D \times N$
- $X^T X$ is a matrix product of matrices of size: $D \times N$ and $N \times D$, which is $\mathcal{O}(D^2 N)$
- Inversion of $X^T X$ is an inversion of a $D \times D$ matrix, which is $\mathcal{O}(D^3)$
- $X^T y$ is a matrix vector product of size $D \times N$ and $N \times 1$, which is $\mathcal{O}(DN)$
- $(X^T X)^{-1} X^T y$ is a matrix product of a $D \times D$ matrix and $D \times 1$ matrix, which is $\mathcal{O}(D^2)$
- Overall complexity: $\mathcal{O}(D^2 N) + \mathcal{O}(D^3) + \mathcal{O}(DN) + \mathcal{O}(D^2)$ $= \mathcal{O}(D^2 N) + \mathcal{O}(D^3)$

Normal Equation

- X has dimensions $N \times D$, X^T has dimensions $D \times N$
- $X^T X$ is a matrix product of matrices of size: $D \times N$ and $N \times D$, which is $\mathcal{O}(D^2 N)$
- Inversion of $X^T X$ is an inversion of a $D \times D$ matrix, which is $\mathcal{O}(D^3)$
- $X^T y$ is a matrix vector product of size $D \times N$ and $N \times 1$, which is $\mathcal{O}(DN)$
- $(X^T X)^{-1} X^T y$ is a matrix product of a $D \times D$ matrix and $D \times 1$ matrix, which is $\mathcal{O}(D^2)$
- Overall complexity: $\mathcal{O}(D^2 N) + \mathcal{O}(D^3) + \mathcal{O}(DN) + \mathcal{O}(D^2)$ $= \mathcal{O}(D^2 N) + \mathcal{O}(D^3)$
- Scales cubic in the number of columns/features of X

Gradient Descent

Start with random values of θ_0 and θ_1

Till convergence

- $\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0}(\sum \epsilon_i^2)$

Gradient Descent

Start with random values of θ_0 and θ_1

Till convergence

- $\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0}(\sum \epsilon_i^2)$
- $\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1}(\sum \epsilon_i^2)$

Gradient Descent

Start with random values of θ_0 and θ_1

Till convergence

- $\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0}(\sum \epsilon_i^2)$
- $\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1}(\sum \epsilon_i^2)$
- Question: Can you write the above for D dimensional data in vectorised form?

Gradient Descent

Start with random values of θ_0 and θ_1

Till convergence

- $\theta_0 = \theta_0 - \alpha \dfrac{\partial}{\partial \theta_0}(\sum \epsilon_i^2)$
- $\theta_1 = \theta_1 - \alpha \dfrac{\partial}{\partial \theta_1}(\sum \epsilon_i^2)$
- Question: Can you write the above for D dimensional data in vectorised form?
- $\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} (y - X\theta)^\top (y - X\theta)$
  $\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} (y - X\theta)^\top (y - X\theta)$
  $\vdots$
  $\theta_D = \theta_D - \alpha \frac{\partial}{\partial \theta_D} (y - X\theta)^\top (y - X\theta)$

Gradient Descent

Start with random values of θ_0 and θ_1

Till convergence

- $\theta_0 = \theta_0 - \alpha \dfrac{\partial}{\partial \theta_0} (\sum \epsilon_i^2)$
- $\theta_1 = \theta_1 - \alpha \dfrac{\partial}{\partial \theta_1} (\sum \epsilon_i^2)$
- Question: Can you write the above for D dimensional data in vectorised form?
- $\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} (y - X\theta)^\top (y - X\theta)$
  $\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} (y - X\theta)^\top (y - X\theta)$
  $\vdots$
  $\theta_D = \theta_D - \alpha \frac{\partial}{\partial \theta_D} (y - X\theta)^\top (y - X\theta)$
- $\theta = \theta - \alpha \frac{\partial}{\partial \theta} (y - X\theta)^\top (y - X\theta)$

Gradient Descent

$$
\begin{aligned}
&\frac{\partial}{\partial\theta}(y - X\theta)^\top(y - X\theta) \\
&= \frac{\partial}{\partial\theta}\left(y^\top - \theta^\top X^\top\right)(y - X\theta) \\
&= \frac{\partial}{\partial\theta}\left(y^\top y - \theta^\top X^\top y - y^\top x\theta + \theta^\top X^\top X\theta\right) \\
&= -2X^\top y + 2X^\top x\theta \\
&= 2X^\top(X\theta - y)
\end{aligned}
$$

Gradient Descent

We can write the vectorised update equation as follows, for each iteration

$\theta = \theta - \alpha X^\top (X\theta - y)$

Gradient Descent

We can write the vectorised update equation as follows, for each iteration

$\theta = \theta - \alpha X^\top (X\theta - y)$

For t iterations, what is the computational complexity of our gradient descent solution?

Gradient Descent

We can write the vectorised update equation as follows, for each iteration

$\theta = \theta - \alpha X^\top (X\theta - y)$

For t iterations, what is the computational complexity of our gradient descent solution?

Hint, rewrite the above as: $\theta = \theta - \alpha X^\top X\theta + \alpha X^\top y$

Gradient Descent

We can write the vectorised update equation as follows, for each iteration

$\theta = \theta - \alpha X^\top (X\theta - y)$

For t iterations, what is the computational complexity of our gradient descent solution?

Hint, rewrite the above as: $\theta = \theta - \alpha X^\top X\theta + \alpha X^\top y$

Complexity of computing $X^\top y$ is $\mathcal{O}(DN)$

Gradient Descent

We can write the vectorised update equation as follows, for each iteration

$\theta = \theta - \alpha X^\top (X\theta - y)$

For t iterations, what is the computational complexity of our gradient descent solution?

Hint, rewrite the above as: $\theta = \theta - \alpha X^\top X\theta + \alpha X^\top y$

Complexity of computing $X^\top y$ is $\mathcal{O}(DN)$

Complexity of computing $\alpha X^\top y$ once we have $X^\top y$ is $\mathcal{O}(D)$ since $X^\top y$ has D entries

Gradient Descent

We can write the vectorised update equation as follows, for each iteration

$\theta = \theta - \alpha X^\top (X\theta - y)$

For t iterations, what is the computational complexity of our gradient descent solution?

Hint, rewrite the above as: $\theta = \theta - \alpha X^\top X\theta + \alpha X^\top y$

Complexity of computing $X^\top y$ is $\mathcal{O}(DN)$

Complexity of computing $\alpha X^\top y$ once we have $X^\top y$ is $\mathcal{O}(D)$ since $X^\top y$ has D entries

Complexity of computing $X^\top X$ is $\mathcal{O}(D^2 N)$ and then multiplying with α is $\mathcal{O}(D^2)$

Gradient Descent

We can write the vectorised update equation as follows, for each iteration

$\theta = \theta - \alpha X^\top (X\theta - y)$

For t iterations, what is the computational complexity of our gradient descent solution?

Hint, rewrite the above as: $\theta = \theta - \alpha X^\top X\theta + \alpha X^\top y$

Complexity of computing $X^\top y$ is $\mathcal{O}(DN)$

Complexity of computing $\alpha X^\top y$ once we have $X^\top y$ is $\mathcal{O}(D)$ since $X^\top y$ has D entries

Complexity of computing $X^\top X$ is $\mathcal{O}(D^2 N)$ and then multiplying with α is $\mathcal{O}(D^2)$

All of the above need only be calculated once!

Gradient Descent

Gradient Descent

For each of the t iterations, we now need to first multiply $\alpha X^\top X$ with θ which is matrix multiplication of a $D \times D$ matrix with a $D \times 1$, which is $\mathcal{O}(D^2)$

Gradient Descent

For each of the t iterations, we now need to first multiply $\alpha X^\top X$ with θ which is matrix multiplication of a $D \times D$ matrix with a $D \times 1$, which is $\mathcal{O}(D^2)$

The remaining subtraction/addition can be done in $\mathcal{O}(D)$ for each iteration.

Gradient Descent

For each of the t iterations, we now need to first multiply $\alpha X^\top X$ with θ which is matrix multiplication of a $D \times D$ matrix with a $D \times 1$, which is $\mathcal{O}(D^2)$

The remaining subtraction/addition can be done in $\mathcal{O}(D)$ for each iteration.

What is overall computational complexity?

Gradient Descent

For each of the t iterations, we now need to first multiply $\alpha X^\top X$ with θ which is matrix multiplication of a $D \times D$ matrix with a $D \times 1$, which is $\mathcal{O}(D^2)$

The remaining subtraction/addition can be done in $\mathcal{O}(D)$ for each iteration.

What is overall computational complexity?

$\mathcal{O}(tD^2) + \mathcal{O}(D^2N) = \mathcal{O}((t+N)D^2)$

Gradient Descent (Alternative)

Gradient Descent (Alternative)

If we do not rewrite the expression $\theta = \theta - \alpha X^\top (X\theta - y)$

For each iteration, we have:

- Computing $X\theta$ is $\mathcal{O}(ND)$

Gradient Descent (Alternative)

If we do not rewrite the expression $\theta = \theta - \alpha X^\top (X\theta - y)$

For each iteration, we have:

- Computing $X\theta$ is $\mathcal{O}(ND)$
- Computing $X\theta - y$ is $\mathcal{O}(N)$

Gradient Descent (Alternative)

If we do not rewrite the expression $\theta = \theta - \alpha X^\top (X\theta - y)$

For each iteration, we have:

- Computing $X\theta$ is $\mathcal{O}(ND)$
- Computing $X\theta - y$ is $\mathcal{O}(N)$
- Computing $\alpha X^\top$ is $\mathcal{O}(ND)$

Gradient Descent (Alternative)

If we do not rewrite the expression $\theta = \theta - \alpha X^\top (X\theta - y)$

For each iteration, we have:

- Computing $X\theta$ is $\mathcal{O}(ND)$
- Computing $X\theta - y$ is $\mathcal{O}(N)$
- Computing $\alpha X^\top$ is $\mathcal{O}(ND)$
- Computing $\alpha X^\top (X\theta - y)$ is $\mathcal{O}(ND)$

Gradient Descent (Alternative)

If we do not rewrite the expression $\theta = \theta - \alpha X^\top (X\theta - y)$

For each iteration, we have:

- Computing $X\theta$ is $\mathcal{O}(ND)$
- Computing $X\theta - y$ is $\mathcal{O}(N)$
- Computing $\alpha X^\top$ is $\mathcal{O}(ND)$
- Computing $\alpha X^\top (X\theta - y)$ is $\mathcal{O}(ND)$
- Computing $\theta = \theta - \alpha X^\top (X\theta - y)$ is $\mathcal{O}(N)$

Gradient Descent (Alternative)

If we do not rewrite the expression $\theta = \theta - \alpha X^\top (X\theta - y)$

For each iteration, we have:

- Computing $X\theta$ is $\mathcal{O}(ND)$
- Computing $X\theta - y$ is $\mathcal{O}(N)$
- Computing $\alpha X^\top$ is $\mathcal{O}(ND)$
- Computing $\alpha X^\top (X\theta - y)$ is $\mathcal{O}(ND)$
- Computing $\theta = \theta - \alpha X^\top (X\theta - y)$ is $\mathcal{O}(N)$

Gradient Descent (Alternative)

If we do not rewrite the expression $\theta = \theta - \alpha X^\top (X\theta - y)$

For each iteration, we have:

- Computing $X\theta$ is $\mathcal{O}(ND)$
- Computing $X\theta - y$ is $\mathcal{O}(N)$
- Computing $\alpha X^\top$ is $\mathcal{O}(ND)$
- Computing $\alpha X^\top (X\theta - y)$ is $\mathcal{O}(ND)$
- Computing $\theta = \theta - \alpha X^\top (X\theta - y)$ is $\mathcal{O}(N)$

What is overall computational complexity?

Gradient Descent (Alternative)

If we do not rewrite the expression $\theta = \theta - \alpha X^\top(X\theta - y)$

For each iteration, we have:

- Computing $X\theta$ is $\mathcal{O}(ND)$
- Computing $X\theta - y$ is $\mathcal{O}(N)$
- Computing $\alpha X^\top$ is $\mathcal{O}(ND)$
- Computing $\alpha X^\top(X\theta - y)$ is $\mathcal{O}(ND)$
- Computing $\theta = \theta - \alpha X^\top(X\theta - y)$ is $\mathcal{O}(N)$

What is overall computational complexity?

$\mathcal{O}(NDt)$