Gradient Descent

Nipun Batra

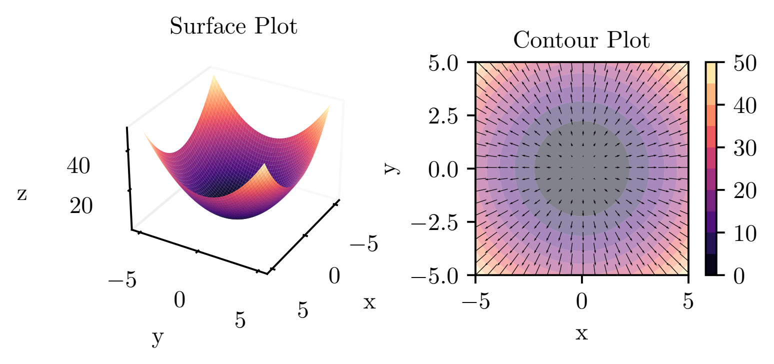
January 28, 2024

IIT Gandhinagar

Revision

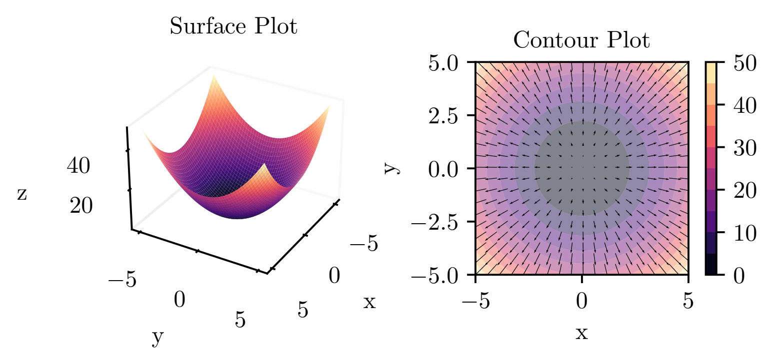
Contour Plot And Gradients

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



Contour Plot And Gradients

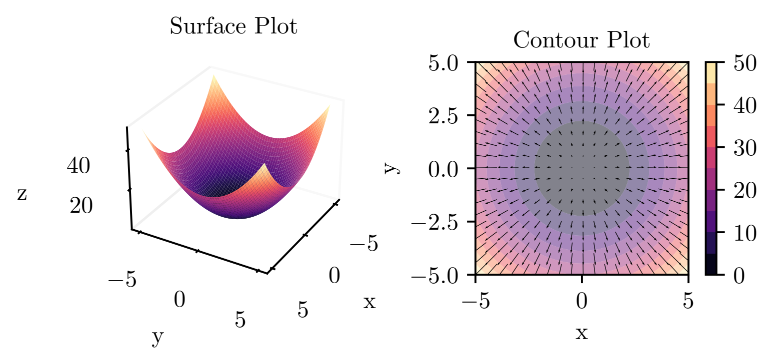
$z = f(x, y) = x^2 + y^2$



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
- 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:
- 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
- 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
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
- 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 $x_0 = 0$
- 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 $x_0 = 2$
- 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:

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

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

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