Introduction to Optimization

WiMLDS Accra and AIMS Mathematics Bootcamp

AIMS Ghana

African Institute for Mathematical Sciences
Ghana

April 11, 2024

AIMS African Institute for Mathematical Sciences GHANA

WiMLDS Women in Machine Learning & Data Science

Overview

Introduction

Optimization is defined as the process of selecting the best possible solution with regard to some criterion/criteria from some set of available alternatives.
The purpose of optimization is to maximize a desired result and minimise an unwanted outcome.

Types of Optimization

- **Discrete Optimization**: Deals with problems where variables take on distinct values. Combinatorial optimization, integer programming and constraint programming are areas under discrete optimization.
- **Continuous Optimization**: Deals with problems where variables take on values in a specific range. It also allows the use of calculus techniques.

Applications of optimization

- **Transportation**: It is used to find the shortest possible route for delivery and traffic congestion
- **Finance**: Used in building investment portfolios and manage risk.
- **Machine Learning**: Used to train algorithms to perform tasks with the highest accuracy or efficiency.

Optimization in Machine Learning

The most common optimization algorithm is **gradient descent** which updates parameters iteratively until it finds an optimal set of values for the model being optimized.

Cost Functions

AIMS Ghana

Definition

The **Cost function** is a mathematical function used to quantify the error produced by a machine learning model. It is expressed as the difference between the actual and predicted values.

Uses of the cost function

- Used for the quantification of errors produced by predictions made using a model.
- Reduction of errors.

Calculus review

Definition: Partial derivative

Consider a function $f : \mathbb{R}^n \to \mathbb{R}$. The partial derivative of f with respect to θ_i is

$$\frac{\partial f(\theta)}{\partial \theta_i} = \lim_{\epsilon \to 0} \frac{f(\theta + \varepsilon \mathbf{e_i}) - f(\theta)}{\epsilon} \tag{1}$$

- We often use the notation $\partial_{\theta_i} f$ for $\dfrac{\partial f(\theta)}{\partial \theta_i}$

For example, let $f : \mathbb{R}^2 \to \mathbb{R}$ where $f(\theta) = \theta_1^2 + 2\theta_1\theta_2$,

$$\frac{\partial f}{\partial \theta_1} = 2\theta_1 + 2\theta_2; \quad \frac{\partial f}{\partial \theta_2} = 2\theta_1$$

Calculus review

Convexity

To put it simply, a real-valued function f is **convex** if the line segment (or chord) between any two points $f(x)$ and $f(y)$ lies above the function graph.

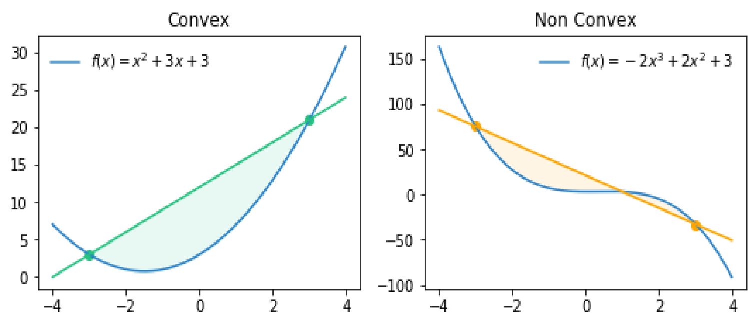


Figure: Convex and Non convex functions (rhome, Medium 2020)

Cost Functions in ML

The type of cost function to be used is largely dependent on the type of machine learning problem.

Types of Machine Learning Problems

- **Regression Problems**: Dealing with continuous values such as price housing makes use of the the **mean error** or of the **mean squared error(MSE)** cost functions
- **Classification Problems**: For tasks where the motive is to predict discrete outputs(e.g; yes or no, cat or dog) appropriate cost functions to use are the **log loss/cross- entropy loss function** for binary classification and **categorical-cross entropy**for multi classification.

Cost Functions in ML

Computing the predictions of a Machine learning model on a data set of **n samples** $(\mathbf{x_i}, \mathbf{y_i})$ can be considered as computing a function $f(x_i, \theta)$, where $\theta = (\theta_1, \theta_2, \theta_3, \cdots)$ are the parameters of the model.

Mean error and MSE cost functions

- Mean error

$$L(x;\theta) = \frac{1}{n}\sum_{i=1}^{n} |y_i - f(x_i, \theta)|$$

- Mean squared error

$$L(x;\theta) = \frac{1}{n}\sum_{i=1}^{n} (y_i - f(x_i, \theta))^2$$

Cost Functions in ML

Log loss/cross entropy cost function

$$L(x,\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i log(f(x_i,\theta)) + (1-y_i)log(1-f(x_i,\theta))\right]$$

Main properties of cost functions:

- **Differentiable**
- **Convex**

Essentially, cost functions act as guideposts for machine learning models, helping them navigate the learning process and achieve optimal performance.

Gradient descent

AIMS Ghana

What is Gradient Descent?

- Gradient descent is an optimization algorithm used to find the values of parameters(coefficients) of a function,*f* that minimizes the cost function(cost).
- Gradient descent is best used when the parameters cannot be calculated analytically for example by using linear algebra and thus must be searched for by an optimization algorithm.

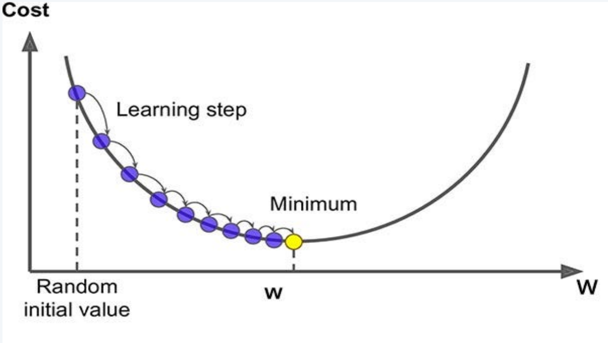


Figure: Understanding of Gradient Descent (Richi, GoPenAI)

Calculus review

Definition: Gradient vector

Again, consider a function $f : \mathbb{R}^n \to \mathbb{R}$. The gradient vector of f is

$$\nabla f(\theta) = \begin{pmatrix} \frac{\partial f}{\partial \theta_1} \\ \vdots \\ \frac{\partial f}{\partial \theta_n} \end{pmatrix}$$

- **Note**: The gradient points in the direction where the function increases the most rapidly.

Calculus review

Properties

Consider a family of functions E_i $(E_1, E_2, \cdots, E_m) : \mathbb{R}^n \to \mathbb{R}$

- **Additivity:**
$$\frac{\partial}{\partial \theta_j}(E_1 + E_2) = \frac{\partial E_1}{\partial \theta_j} + \frac{\partial E_2}{\partial \theta_j}$$
  More generally,
$$\frac{\partial}{\partial \theta_j} \sum_{i=1}^{m} E_i(\theta_j) = \sum_{i=1}^{m} \frac{\partial E_i}{\partial \theta_j}$$
- **Chain rule:** Given a function $Z : \mathbb{R}^n \to \mathbb{R}$
$$\frac{\partial}{\partial \theta_j} E(Z(\theta)) = \frac{\partial E(Z)}{\partial Z} \times \frac{\partial Z}{\partial \theta_j}$$

Gradient descent

Application in ML: Minimize Error (cost)

Gradient descent determines a weight vector θ that minimizes the error, $L(\theta)$ by:

- Starting with an arbitrary initial weight vector.
- Repeatedly modify the weight vector in small steps.
- At each step, the wieght vector is modified in the direction that produces the steepest descent along the error surface.
- The gradient points directly uphill and the negative gradient points directly downhill.

- We can decrease the function L by moving in the direction of negative gradient. This is the method of Steepest descent.
- Given an initial θ_0, then:

$$\theta^{k+1} = \theta^k - \eta \nabla L(\theta^{(k)}),$$

where η is the step size (learning rate).

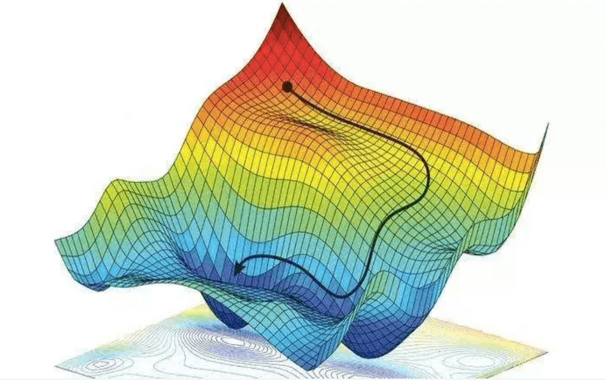


Figure: Understanding of Gradient Descent (Alexander Amini)

Choosing the Learning Rate

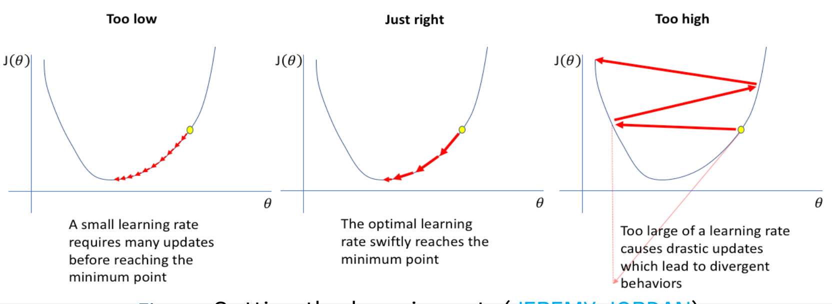


Figure: Setting the learning rate(JEREMY JORDAN)

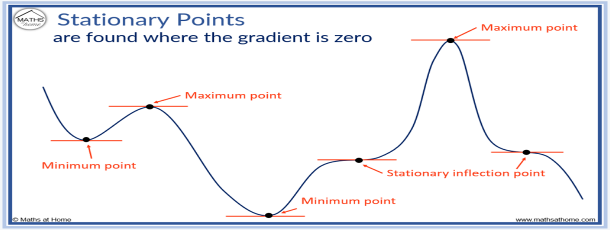
When does the algorithm stop

- When the maximum number of epochs(iterations) is reached.
- When $\partial_\theta L(\theta^{(k)})$ is sufficiently small.

Gradient Descent

Stationary points, Local Optima

$\star$ When $f'(x) = 0$ derivative provides no information about direction of move.
$\star$ Points where $f'(x) = 0$ are known as stationary or critical points
- Local minimum/maximum: a point where $f(x)$ lower/ higher than all its neighbors.
- Saddle/Inflexion Points: neither maxima nor minima.



Presence of Multiple Minima

⋆ Optimization algorithms may fail to find global minimum.
⋆ Generally accept such solutions;

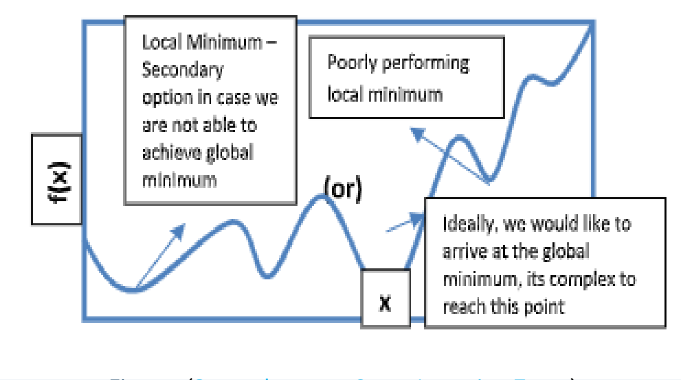


Figure: (Sarveshwaran, Great Learning Team)

Types of Gradient Descent Algorithms

- **Batch Gradient Descent Algorithm:**

Uses the whole dataset to make an update for of the coefficients.

- **Stochastic Gradient Descent Algorithm(SDG):**

Updates the values of coefficients for each observation in the dataset. These frequent updates of the coefficient provide a good rate of improvement.

- **Mini-Batch Gradient Descent:**

It is a combination of the SGD and BGD. It splits the dataset into smaller batches and the coefficients are updated at the end of each of these batches. • Then at each iteration SGD implements GD on random subset of the training set (minibatch).

Stochastic Gradient descent

Benefits of SGD

$\star$ It can (in principal) escape local minima.

$\star$ Some evidence suggest that SGD finds the parameter for NN that improve generalization performance.

$\star$ SGD Computationally less expensive.

(S)GD performance can be improved by;

$\star$ Normalization and scaling the data

$$x_{new} = \frac{x - \bar{x}}{\sigma}$$

where $\bar{x}$ is the average, and σ the standard deviation $\star$ Change learning rate (adaptively).

Example

Example

Using the data provided 'Data_Week3.out', implement the gradient descent algorithm to find the parameters θ_0 and $\theta 1$ that minimizes the cost function squared loss

$$L(x;\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \theta_0 - \theta_1 x_i)^2 \tag{2}$$

- **Steps :**
  1. Write the cost function and gradients using matrix form
  2. Choose the number of iteration N
  3. Set initial values of θ_0 and θ_1
  4. Run the update

Example

Let consider

$$X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \quad \text{and} \quad \theta = \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix}.$$

These quantities will help to simplify our expressions using vector and matrices operations.

- **Loss function**

$$L(X;\theta) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \theta_0 - \theta_1 x_i)^2 = \frac{1}{m}||y - X\theta||^2$$

Where $||.||$ is the norm (2 norm) operator

Example

- **Gradients**
  We compute first the partial derivatives with respect to θ_0 and θ_1, which give respectively

$$\partial_{\theta_0} L(X;\theta) = \frac{-2}{m}\sum_{i=1}^{m}(y_i - \theta_0 - \theta_1 x_i); \quad \partial_{\theta_1} L(X;\theta) = \frac{-2}{m}\sum_{i=1}^{m} x_i(y_i - \theta_0 - \theta_1 x_i) \qquad (3)$$

  Then, we can observe that

$$\sum_{i=1}^{m}(y_i - \theta_0 - \theta_1 x_i) = (1\,1\,1 \cdots 1\,1) \cdot (y - X\theta)$$

$$\sum_{i=1}^{m} x_i(y_i - \theta_0 - \theta_1 x_i) = (x_1\, x_2\, x_3 \cdots x_m) \cdot (y - X\theta)$$

Example

- **Gradient vector**
  Using the expressions established earlier, we find the gradient vector as follows

$$\nabla_\theta L(X;\theta) = \begin{pmatrix} \partial_{\theta_0} L(X;\theta) \\ \partial_{\theta_1} L(X;\theta) \end{pmatrix}$$
$$= \frac{-2}{m} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_m \end{pmatrix} \cdot (y - X\theta)$$

Therefore,

$$\nabla_\theta L(X;\theta) = \frac{2}{m} X^T (X\theta - y) \tag{4}$$

Let's get coding!!

Conclusion

Some References

- Huy L. Nguyên. (2009). Optimization using gradient descent. Northeastern University Khoury College of Computer Sciences
- Bottou, Léon (1998). “Online Algorithms and Stochastic Approximations”. Online Learning and Neural Networks. Cambridge University Press. ISBN 978-0-521-65263-6
- Christopher M. Bishop, Pattern Recognition and Machine Learning

Thank you for your attention

AIMS Ghana

African Institute for Mathematical Sciences
Ghana

April 11, 2024