

Lecture 9: Iterative Optimization with Gradient Descent

COMP90049

Introduction to Machine Learning

Semester 2, 2021

Lida Rashidi, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Acknowledgement: Lea Frermann



Finding Optimal Points I

Finding the **parameters** that optimize a **target**

Ex1: Estimate the **study time** which leads to the **best grade** in COMP90049.

Ex2: Find the **shoe price** which leads to **maximum profit** of our shoe shop.

Ex3: Predicting **housing prices** from a **weighted** combination of house age and house location

Ex4: Find the **parameters** θ of a spam classifier which lead to the **lowest error**

Ex5: Find the **parameters** θ of a spam classifier which lead to the **highest data log likelihood**



Recipe for finding Minima / Maxima

1. Define your function of interest $f(x)$ (e.g., data log likelihood)
2. Compute its first derivative wrt its input x
3. Set the derivative to zero
4. Solve for x

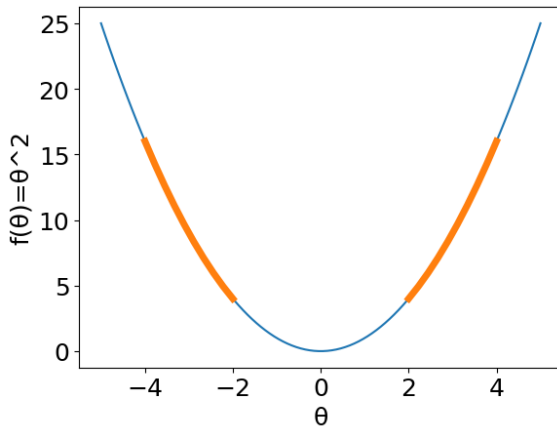
Closed-form solutions

- Previously, we computed the **closed form** solution for the MLE of the binomial distribution
- We follow our recipe, and arrive at a single solution

Unfortunately, life is not always as easy

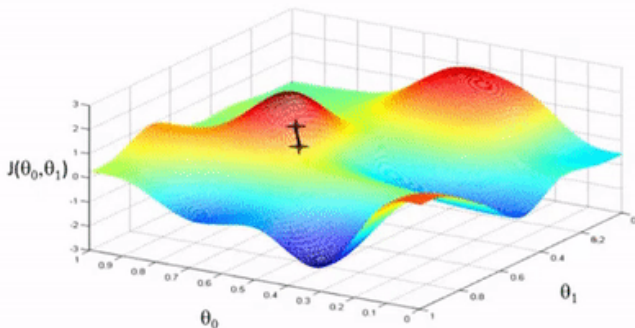
- Often, no closed-form solution exists
- Instead, we have to **iteratively** improve our estimate of $\hat{\theta}$ until we arrive at a satisfactory solution
- Gradient descent is one popular iterative optimization method

'Descending' the function to find the Optimum



- 1-dimensional case: find parameter θ that minimizes the function
- follow the curvature of the line step by step

‘Descending’ the function to find the Optimum



- 2-dimensional case: find parameters $\theta = [\theta_0, \theta_1]$ that minimize the function J
- follow the curvature step by step along the steepest way

Andrew Ng



Source: <https://medium.com/binaryandmore/>

beginners-guide-to-deriving-and-implementing-backpropagation-e3c1a5a1e536

Intuition

- Descending a mountain (aka. our function) as fast as possible: at every position take the next step that takes you most directly into the valley
- We compute a series of solutions $\theta^{(0)}, \theta^{(1)}, \theta^{(2)}, \theta^{(3)}, \dots$ by ‘walking’ along the function and taking steps in the direction with the steepest local slope (or gradient).
- each solution depends on the current location

Learn the model parameters θ

- such that we **minimize the error**
- traverse over the loss function step by step ('descending into a valley')
- we would like an algorithm that tells how to update our parameters

$$\theta \leftarrow \theta + \Delta\theta$$

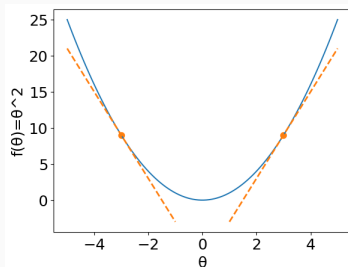
Gradient Descent: Details

Learn the model parameters θ

- such that we **minimize the error**
- traverse over the loss function step by step ('descending into a valley')
- we would like an algorithm that tells how to update our parameters

$$\theta \leftarrow \theta + \Delta\theta$$

- $\Delta\theta$ is the **derivative** $\frac{\partial f}{\partial \theta}$
- tells us how much f changes in response to a change in θ .
- a measure of the **slope** or **gradient** of a function f at point θ
- the **gradient** points to the greatest **increase** of a function



Gradient Descent: Details

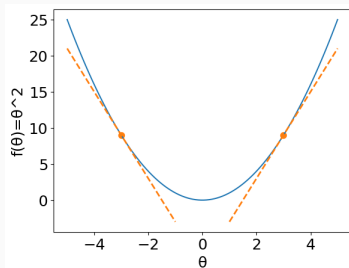
Learn the model parameters θ

- such that we **minimize the error**
- traverse over the loss function step by step ('descending into a valley')
- we would like an algorithm that tells how to update our parameters

$$\theta \leftarrow \theta + \Delta\theta$$

- if $\frac{\partial f}{\partial \theta} > 0$: $f(\theta) : \nearrow$ as $\theta : \nearrow$
- if $\frac{\partial f}{\partial \theta} < 0$: $f(\theta) : \nearrow$ as $\theta : \searrow$
- if $\frac{\partial f}{\partial \theta} = 0$: we are at a minimum
- so, to approach the minimum:

$$\theta \leftarrow \theta - \eta \frac{\partial f}{\partial \theta}$$



Gradient Descent for multiple parameters

- Usually, our models have **several parameters** which need to be optimized to minimize the error
- We compute **partial derivatives** of $f(\theta)$ wrt. individual θ_i
- Partial derivatives measure change in a function of multiple parameters given a change in a single parameter, with all others held constant
- For example for $f(\theta_1, \theta_2)$ we can compute $\frac{\partial f}{\partial \theta_1}$ and $\frac{\partial f}{\partial \theta_2}$
- We then **update each parameter individually**

$$\theta_1 \leftarrow \theta_1 + \Delta \theta_1 \quad \text{with } \Delta \theta_1 = -\eta \frac{\partial f}{\partial \theta_1}$$

$$\theta_2 \leftarrow \theta_2 + \Delta \theta_2 \quad \text{with } \Delta \theta_2 = -\eta \frac{\partial f}{\partial \theta_2}$$



Recipe for Gradient Descent (single parameter)

-
- 1: Define objective function $f(\theta)$
 - 2: Initialize parameter $\theta^{(0)}$
 - 3: **for** iteration $t \in \{0, 1, 2, \dots, T\}$ **do**
 - 4: Compute the first derivative of f at that point $\theta^{(t)} : \frac{\partial f}{\partial \theta^{(t)}}$
 - 5: Update your parameter by subtracting the (scaled) derivative

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \frac{\partial f}{\partial \theta^{(t)}}$$

- η is the **step size** or **learning rate**, a parameter
- When to stop? Fix number of iterations, or define other criteria

Recipe for Gradient Descent (multiple parameters)

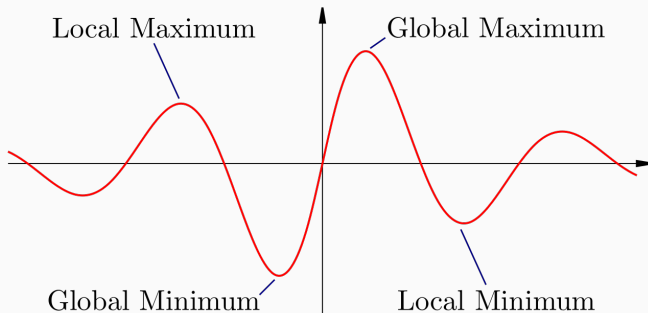
-
- 1: Define objective function $f(\theta)$
 - 2: Initialize parameters $\{\theta_1^{(0)}, \theta_2^{(0)}, \theta_3^{(0)}, \dots\}$
 - 3: **for** iteration $t \in \{0, 1, 2, \dots T\}$ **do**
 - 4: Initialize vector of *gradients* $\leftarrow []$
 - 5: **for** parameter $f \in \{1, 2, 3, \dots F\}$ **do**
 - 6: Compute the first derivative of f at that point $\theta_f^{(t)} : \frac{\partial f}{\partial \theta_f^{(t)}}$
 - 7: append $\frac{\partial f}{\partial \theta_f^{(t)}}$ to *gradients*
 - 8: **Update all** parameters by subtracting the (scaled) gradient

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \frac{\partial f}{\partial \theta^{(t)}}$$

Aside: Global and Local Minima and Maxima

Possible issue: local maxima and minima!

- A function is **convex** if a line between any two points of the function lies above the function
- A global **maximum** is the single highest value of the function
- A global **minimum** is the single lowest value of the function



- with an appropriate learning rate, GD will find the global minimum for differentiable convex functions
- with an appropriate learning rate, GD will find a local minimum for differentiable non-convex functions

Now you know:

- What optimization is, and why it's important
- How to do closed-form optimization (aka. “set the derivative of $f(\theta)$ to zero and solve for θ)
- That closed-form solutions are not always computable
- In that case, iterative optimization can help us
- Gradient descent is one instance of an iterative optimization method
- How gradient descent works!

Next lecture(s)

- Logistic Regression
- The perceptron
- Neural networks

