

# Optimization and Gradient Descent<sup>1</sup>

Summer 2023

University of Toronto

---

<sup>1</sup>based on slides by Eleni Triantafillou, Ladislav Rampasek, Jake Snell, Kevin Swersky, Shenlong Wang and others

# What is optimization?

Informally:

Minimizing (or maximizing) some quantity of interest.

# Example Applications

- Engineering: Minimize fuel consumption of an automobile.
- Economics: Maximize returns on an investment.
- Supply Chain Logistics: Minimize time taken to fulfill an order.
- Life: Maximize happiness.

# Formal definition of Optimization

Goal: find  $\theta^* = \arg \min_{\theta} f(\theta)$ , (possibly subject to constraints on  $\theta$ ).

- $\theta \in \mathbb{R}^n$ : optimization variable
- $f : \mathbb{R}^n \rightarrow \mathbb{R}$ : objective function

Maximizing  $f(\theta)$  is equivalent to minimizing  $-f(\theta)$ , so we can treat everything as a minimization problem.

# Assumptions

We make some assumptions to find the best method for solving an optimization problem:

- Is  $\theta$  discrete or continuous?
- What form do constraints on  $\theta$  take (if any)?
- Is  $f$  “well-behaved” (linear, differentiable, convex, etc.)?

# Optimization for Machine Learning

Often in machine learning, we are interested in learning the parameters,  $\theta$  of a model.

Goal: minimize some loss function.

- If we have data  $(x, y)$ , we may want to maximize the probability  $P(y|x, \theta)$ .
- Equivalently, we can minimize  $-P(y|x, \theta)$ .

We can solve the same optimization problem equivalently by applying any monotonic transformation to the objective function.

- So equivalently, we can minimize  $-\log P(y|x, \theta)$ .
- Taking log can help for numerical reasons.

# Gradient Descent

Gradient Descent is one method for solving an optimization problem.

# Gradient Descent: Motivation

From calculus, we know that the minimum of  $f$  must lie at a point where its derivative vanishes, i.e.  $\frac{\partial f(\theta^*)}{\partial \theta} = 0$ .

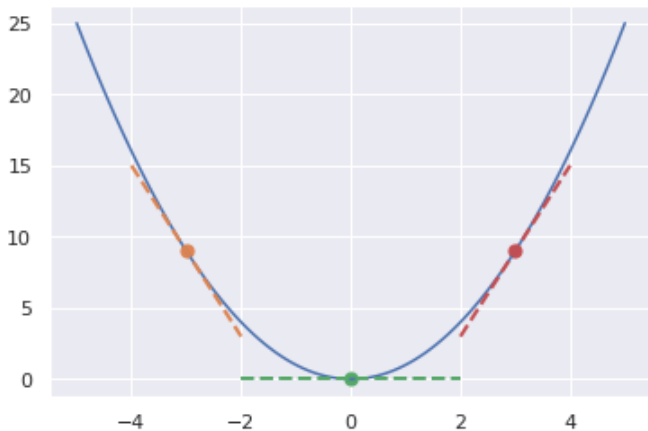
- Sometimes, we can solve this equation analytically for  $\theta$ .
- Mostly, we are not so lucky and must resort to iterative methods.

Recall the Gradient:

$$\nabla_{\theta} f = \left( \frac{\partial f(\theta)}{\partial \theta_1}, \frac{\partial f(\theta)}{\partial \theta_2}, \dots, \frac{\partial f(\theta)}{\partial \theta_n} \right)$$



# Gradient Descent: Motivation



# Gradient Descent Algorithm Review

Let  $\eta$  be the learning rate and  $T$  be the number of iterations:

- Initialize  $\theta_0$  randomly.
- For  $t = 1 : T$ 
  - $\delta_t = -\eta \nabla_{\theta_{t-1}} f$
  - $\theta_t \leftarrow \theta_{t-1} + \delta_t$

Choice of learning rate matters:

- Too big: the objective function will blow up.
- Too small: the algorithm will take a long time to converge.

# Gradient Descent with Line Search

Let  $\eta$  be the learning rate and  $T$  be the number of iterations:

- Initialize  $\theta_0$  randomly.
- For  $t = 1 : T$ 
  - Find a step size  $\eta_t$  such that  $f(\theta_t - \eta_t \nabla_{\theta_{t-1}}) < f(\theta_t)$
  - $\delta_t = -\eta_t \nabla_{\theta_{t-1}} f$
  - $\theta_t \leftarrow \theta_{t-1} + \delta_t$

Requires a line-search step at every iteration.

# Gradient Descent with Momentum

Let  $\eta$  be the learning rate and  $T$  be the number of iterations. We introduce a momentum coefficient  $\alpha \in [0, 1)$  so that the updates have “memory”:

- Initialize  $\theta_0$  randomly.
- For  $t = 1 : T$ 
  - $\delta_t = -\eta \nabla_{\theta_{t-1}} f + \alpha \delta_{t-1}$
  - $\theta_t \leftarrow \theta_{t-1} + \delta_t$

Momentum is a nice trick that can help speed up convergence.

Generally, it is useful to try values between 0.8 and 0.95, but the choice is problem dependent.

# Convergence Criterion

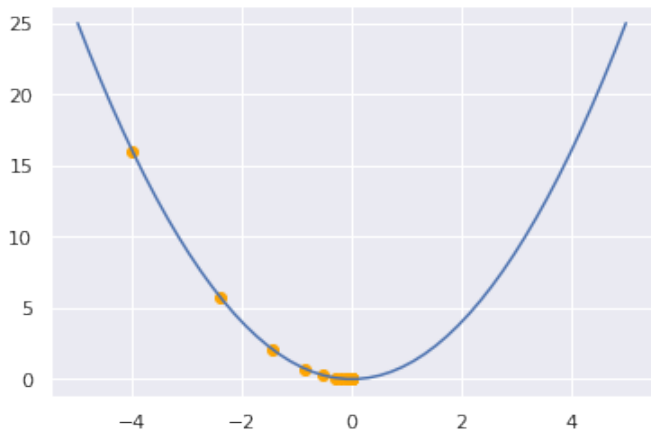
Instead of choosing a fixed number of iterations, we can define some convergence criterion, which is a condition upto which we would like to run the algorithm.

- Initialize  $\theta_0$  randomly.
- Until convergence criterion is satisfied
  - $\delta_t = -\eta \nabla_{\theta_{t-1}} f$
  - $\theta_t \leftarrow \theta_{t-1} + \delta_t$

## Example Convergence Criteria

- Change in objective function value is close to zero (or less than some threshold):  $|f(\theta_{t+1}) - f(\theta_t)| < \epsilon$ .
- Gradient norm is smaller than some threshold:  $\|\nabla_{\theta} f\| < \epsilon$ .
- Validation error starts to increase: also known as Early Stopping.

# Gradient Descent Updates



## Exercise: Gradient Exercise Intuition

Suppose we are trying to optimize the loss function  $f(x) = \frac{1}{2}x^T Ax$ , where  $x \in \mathbb{R}^2$ . Let  $A = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$  and  $x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ . What are the first two iterates of gradient descent, with a learning rate  $\eta = 0.1$ ?



# Stochastic Gradient Descent (SGD)

- Each iteration of Gradient Descent requires that we sum over the entire dataset to compute the gradient.
- SGD idea: at each iteration, sub-sample a small (mini-)batch of data (even just 1 point can work) and use that to estimate the gradient.
- Each update is noisy, but very fast!
- It can be shown that this method produces an unbiased estimate of the true gradient.

# Stochastic Gradient Descent (SGD)

- Batch-learning: computing gradients using the full dataset (which can be a huge, very high-dimensional matrix, e.g. 1 million images of size  $224 \times 224 \times 3$ ).
- Mini-batch learning: computing gradients using subsets of data at every iteration.

# SGD Intuition

- SGD works because similar data yields similar gradients.
- If there is enough redundancy in the data, the noise from subsampling isn't too bad.

Tips:

- Step sizes need to be tuned to different problems.
- Divide the log-likelihood estimate by the mini-batch size. Then learning rate is invariant to mini-batch size.
- Subsample without replacement so that each point is visited during an epoch of training.

# Convexity

A function  $f$  is convex if for any two points  $\theta_1$  and  $\theta_2$  and any  $t \in [0, 1]$ ,

$$f(t\theta_1 + (1 - t)\theta_2) \leq tf(\theta_1) + (1 - t)f(\theta_2)$$

Geometric Intuition: If you draw a line segment between the two points and it lies above the function curve, then the function is said to be convex.

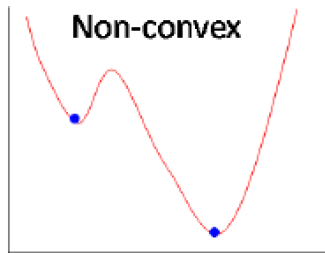
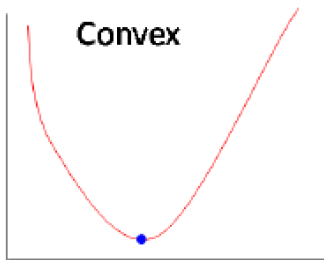
# Compositions of Convex Functions

We can compose convex functions such that the resulting function is also convex:

- If  $f$  is convex, then so is  $\alpha f$  for  $\alpha \geq 0$ .
- If  $f_1$  and  $f_2$  are both convex, then so is  $f_1 + f_2$ .

# Why do we care about convexity?

- Any local minimum is a global minimum.
- This makes optimization a lot easier because we don't have to worry about getting stuck in a local minimum.



# Examples of Convex Functions

- Quadratic Functions
- Negative Logarithms
- Cross-entropy Loss Function

Check out the colab!

## Exercise: Sum of Convex Functions

Prove that the sum of two convex functions is convex.