Distributed Optimization for Machine Learning

Lecture 4 - Unconstrained Optimization: Gradient Descent

Tianyi Chen

School of Electrical and Computer Engineering
Cornell Tech, Cornell University

September 8, 2025

Different perspectives of (pre)training AI models

1. Given training data $\{\mathbf{x}\}$, develop a Model $h_{\boldsymbol{\theta}}(\mathbf{x})$
2. Define a Loss Function $L(\boldsymbol{\theta})$ based on the format $\{\mathbf{x}\}$
3. Design an Optimization Algorithm to minimize the loss

This three-step framework – **Model, Loss, and Optimization** - is the fundamental blueprint for almost all of supervised machine learning.

Different perspectives of (pre)training AI models

1. Given training data $\{\mathbf{x}\}$, develop a Model $h_{\boldsymbol{\theta}}(\mathbf{x})$

$\downarrow$

2. Define a Loss Function $L(\boldsymbol{\theta})$ based on the format $\{\mathbf{x}\}$

We will further extend to different data modality and different state-of-the-art (SOTA) models in Theme 3 of the class.

Generic (model-agnostic) optimization

3. Design an Optimization Algorithm to minimize the loss $L(\boldsymbol{\theta})$

In the next few lectures, we will first review some generic (model-agnostic and data-agnostic) **optimization** techniques - which are the fundamental for training almost all types of machine learning and AI models.

[†]Since we first consider data-agnostic optimization, the algorithms are all "batch" algorithms, which use the entire dataset to compute each optimization update.

Differentiable unconstrained minimization

For now on, let's use $\mathbf{x}$ to replace $\boldsymbol{\theta}$ as the optimization variable

$$\begin{array}{ll} \min_{\mathbf{x}} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \mathbb{R}^n \end{array}$$

- f (objective or cost function) is differentiable

Why not solve $\nabla f(\mathbf{x}) = 0$ directly?

For simple functions, we can find the **analytical solution** by solving

$$f(x) = (x-3)^2 \implies \nabla f(x) = 2(x-3) = 0 \implies x = 3.$$

Why not solve $\nabla f(\mathbf{x}) = 0$ directly?

For simple functions, we can find the **analytical solution** by solving

$$f(x) = (x-3)^2 \implies \nabla f(x) = 2(x-3) = 0 \implies x = 3.$$

For AI/ML, this direct approach is usually **impossible** for several reasons:

No closed-form solution

For a deep neural network, $\nabla f(\mathbf{x}) = 0$ is a massive system of highly non-linear, coupled equations. No algebraic formula to solve for $\mathbf{x}$.

Prohibitive computational cost

Even when a closed-form solution exists, it requires computationally infeasible operations, like inverting a massive matrix ($O(n^3)$ cost).

Why not solve $\nabla f(\mathbf{x}) = 0$ directly?

For simple functions, we can find the **analytical solution** by solving

$$f(x) = (x-3)^2 \implies \nabla f(x) = 2(x-3) = 0 \implies x = 3.$$

For AI/ML, this direct approach is usually **impossible** for several reasons:

No closed-form solution

For a deep neural network, $\nabla f(\mathbf{x}) = 0$ is a massive system of highly non-linear, coupled equations. No algebraic formula to solve for $\mathbf{x}$.

Prohibitive computational cost

Even when a closed-form solution exists, it requires computationally infeasible operations, like inverting a massive matrix ($O(n^3)$ cost).

Local minima & saddle points

Solving $\nabla f(\mathbf{x}) = 0$ finds all *stationary points*, including maxima and saddle points. Iterative algorithms move "downhill" to find a minimum.

Iterative descent algorithms

Start with a point $\mathbf{x}^0$, and construct a sequence $\{\mathbf{x}^t\}$ s.t.

$$f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t), \quad t = 0, 1, \ldots$$

Iterative descent algorithms

Start with a point $\mathbf{x}^0$, and construct a sequence $\{\mathbf{x}^t\}$ s.t.

$$f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t), \quad t = 0, 1, \ldots$$

- $\mathbf{d}$ is said to be a **descent direction** of f at $\mathbf{x}$ if

$$\underbrace{f'(\mathbf{x}; \mathbf{d}) := \lim_{\tau \downarrow 0} \frac{f(\mathbf{x} + \tau \mathbf{d}) - f(\mathbf{x})}{\tau}}_{\text{directional derivative}} = \nabla f(\mathbf{x})^\top \mathbf{d} < 0 \tag{1}$$

Iterative descent algorithms

Start with a point $\mathbf{x}^0$, and construct a sequence $\{\mathbf{x}^t\}$ s.t.

$$f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t), \quad t = 0, 1, \dots$$

- $\mathbf{d}$ is said to be a **descent direction** of f at $\mathbf{x}$ if

$$\underbrace{f'(\mathbf{x}; \mathbf{d}) := \lim_{\tau \downarrow 0} \frac{f(\mathbf{x} + \tau \mathbf{d}) - f(\mathbf{x})}{\tau}}_{\text{directional derivative}} = \nabla f(\mathbf{x})^\top \mathbf{d} < 0 \tag{1}$$

In each iteration, search minimum in a descent direction

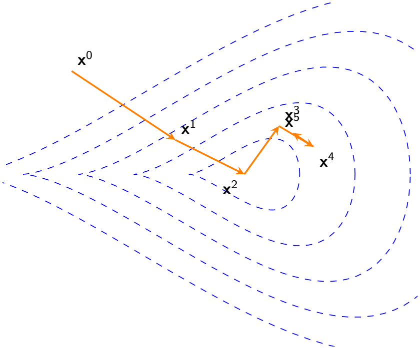
$$\mathbf{x}^{t+1} = \mathbf{x}^t + \eta_t \mathbf{d}^t \tag{2}$$

where $\mathbf{d}^t$: descent direction at $\mathbf{x}^t$; $\eta_t > 0$: stepsize

Gradient descent (GD)

One of the most important examples of (2): **gradient descent**

$$\mathbf{x}^{t+1} = \mathbf{x}^{t} - \eta_t \nabla f(\mathbf{x}^{t}) \tag{3}$$



- traced to Augustin Louis Cauchy '1847 ...

Gradient descent (GD) - steepest descent

One of the most important examples of (2): **gradient descent**

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \eta_t \nabla f(\mathbf{x}^t) \tag{3}$$

- descent direction: $\mathbf{d}^t = -\nabla f(\mathbf{x}^t)$
- a.k.a. **steepest descent**, since from (1) and Cauchy-Schwarz (CS),

$$\underbrace{\arg \min_{\mathbf{d}:||\mathbf{d}||_2 \leq 1} f'(\mathbf{x};\mathbf{d}) = \arg \min_{\mathbf{d}:||\mathbf{d}||_2 \leq 1} \nabla f(\mathbf{x})^\top \mathbf{d} = -\frac{\nabla f(\mathbf{x})}{||\nabla f(\mathbf{x})||_2}}_{\text{direction with the greatest rate of objective value improvement}}$$

Gradient descent (GD) - steepest descent

One of the most important examples of (2): **gradient descent**

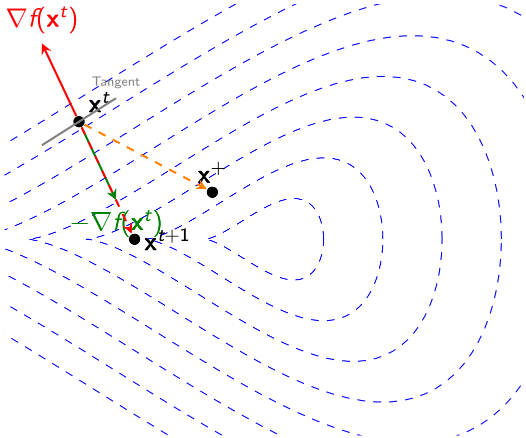
$$\mathbf{x}^{t+1} = \mathbf{x}^t - \eta_t \nabla f(\mathbf{x}^t) \tag{3}$$

- descent direction: $\mathbf{d}^t = -\nabla f(\mathbf{x}^t)$
- a.k.a. **steepest descent**, since from (1) and Cauchy-Schwarz (CS),

$$\underbrace{\arg\min_{\mathbf{d}:\|\mathbf{d}\|_2 \le 1} f'(\mathbf{x};\mathbf{d}) = \arg\min_{\mathbf{d}:\|\mathbf{d}\|_2 \le 1} \nabla f(\mathbf{x})^\top \mathbf{d} = -\frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|_2}}_{\text{direction with the greatest rate of objective value improvement}}$$

The **CS inequality** gives us a lower bound $\mathbf{u}^\top \mathbf{v} \ge -\|\mathbf{u}\|_2\|\mathbf{v}\|_2$, which is achieved when $\mathbf{v} = -c \cdot \mathbf{u}$ for some scalar $c > 0$.

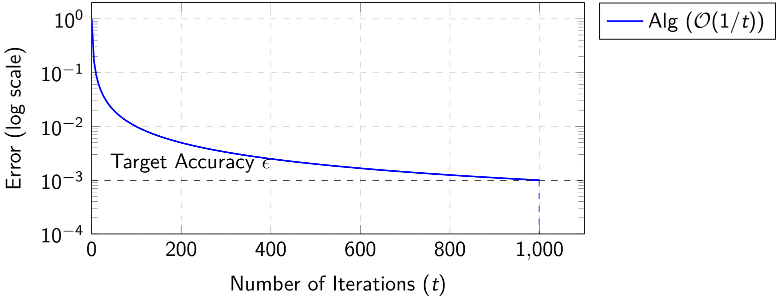
Gradient descent (GD) - steepest descent



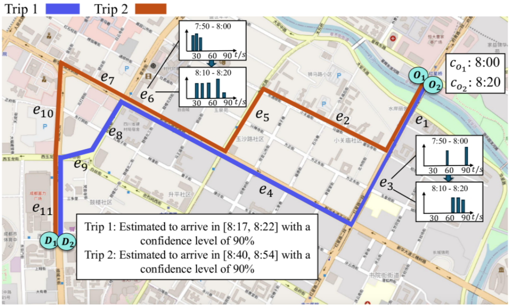
Gradient descent is great. But when it will terminate?

In the real world, we have a limited budget. How many iterations are needed to reach a good-enough solution?

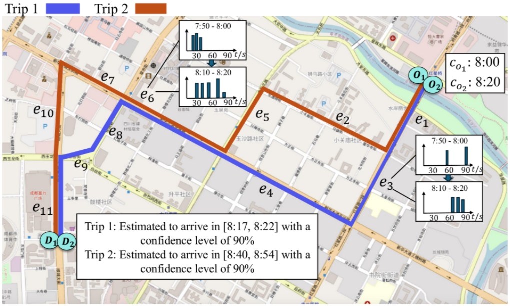
Convergence analysis gives us the answer. It tells us how an algorithm will perform at scale.



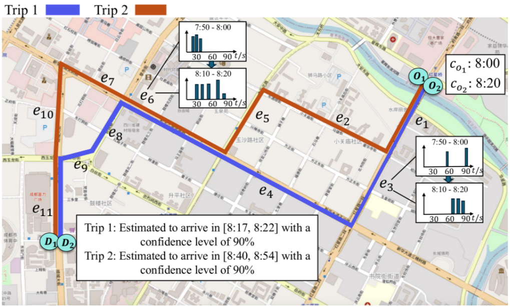
Convergence analysis as means to obtain ETA



Convergence analysis as means to obtain ETA



Convergence analysis as means to obtain ETA



- How many miles I can drive per hour given the total distance?
- How much progress I make per hour given the remaining distance?

Table of Contents

Quadratic minimization

To get a sense of the convergence rate of GD, let's begin with quadratic objective functions (e.g., in linear regression)

$$\min_{\mathbf{x}} \; f(\mathbf{x}) := \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \mathbf{Q}(\mathbf{x} - \mathbf{x}^*)$$

for some $n \times n$ matrix $\mathbf{Q} \succ 0$, where $\nabla f(\mathbf{x}) = \mathbf{Q}(\mathbf{x} - \mathbf{x}^*)$

Quadratic minimization

To get a sense of the convergence rate of GD, let's begin with quadratic objective functions (e.g., in linear regression)

$$\min_{\mathbf{x}} \ f(\mathbf{x}) := \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \mathbf{Q}(\mathbf{x} - \mathbf{x}^*)$$

for some $n \times n$ matrix $\mathbf{Q} \succ 0$, where $\nabla f(\mathbf{x}) = \mathbf{Q}(\mathbf{x} - \mathbf{x}^*)$

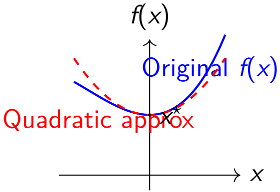
Accordingly, the GD update rule becomes

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \eta_t \nabla f(\mathbf{x}^t) = (\mathbf{I} - \eta_t \mathbf{Q})(\mathbf{x}^t - \mathbf{x}^*) + \mathbf{x}^*!$$

What is unique about quadratic minimization?

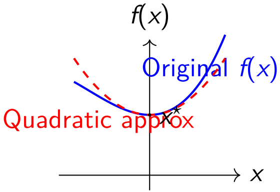
Quadratic minimization

- **Local approximation:** Any general smooth function behaves like a quadratic very close to a minimizer (according to Taylor's theorem).



Quadratic minimization

- **Local approximation:** Any general smooth function behaves like a quadratic very close to a minimizer (according to Taylor's theorem).



- **Tractable analysis:** Allow to derive exact, closed-form convergence rates. The insights we gain apply to more complex problems.

$$\boxed{\mathbf{x}^t - \mathbf{x}^* = \left(\prod_{k=0}^{t-1}(\mathbf{I} - \eta_k \mathbf{Q})\right)(\mathbf{x}^0 - \mathbf{x}^*)}.$$

Convergence for constant stepsizes

Proof: According to the GD update rule,

$$\mathbf{x}^{t+1} - \mathbf{x}^* = \mathbf{x}^t - \mathbf{x}^* - \eta_t \nabla f(\mathbf{x}^t) = (\mathbf{I} - \eta_t \mathbf{Q})(\mathbf{x}^t - \mathbf{x}^*)$$
$$\implies ||\mathbf{x}^{t+1} - \mathbf{x}^*||_2 \leq ||\mathbf{I} - \eta_t \mathbf{Q}|| \cdot ||\mathbf{x}^t - \mathbf{x}^*||_2$$

To get the fastest convergence, we must choose the stepsize η that **minimizes the contraction factor** $||\mathbf{I} - \eta \mathbf{Q}||_2$.

Convergence for constant stepsizes

Proof: According to the GD update rule,

$$\mathbf{x}^{t+1} - \mathbf{x}^* = \mathbf{x}^t - \mathbf{x}^* - \eta_t \nabla f(\mathbf{x}^t) = (\mathbf{I} - \eta_t \mathbf{Q})(\mathbf{x}^t - \mathbf{x}^*)$$
$$\Longrightarrow \; \|\mathbf{x}^{t+1} - \mathbf{x}^*\|_2 \leq \|\mathbf{I} - \eta_t \mathbf{Q}\| \cdot \|\mathbf{x}^t - \mathbf{x}^*\|_2$$

To get the fastest convergence, we must choose the stepsize η that **minimizes the contraction factor** $\|\mathbf{I} - \eta \mathbf{Q}\|_2$.

$$\|\mathbf{I} - \eta \mathbf{Q}\| = \underbrace{\max\{|1 - \eta\lambda_1(\mathbf{Q})|, |1 - \eta\lambda_n(\mathbf{Q})|\}}_{\text{remark: optimal choice is } \eta_t = \frac{2}{\lambda_1(\mathbf{Q}) + \lambda_n(\mathbf{Q})}}$$
$$= 1 - \frac{2\lambda_n(\mathbf{Q})}{\lambda_1(\mathbf{Q}) + \lambda_n(\mathbf{Q})} = \frac{\lambda_1(\mathbf{Q}) - \lambda_n(\mathbf{Q})}{\lambda_1(\mathbf{Q}) + \lambda_n(\mathbf{Q})}$$

Apply the above bound recursively to complete the proof. $\square$

Convergence for constant stepsizes

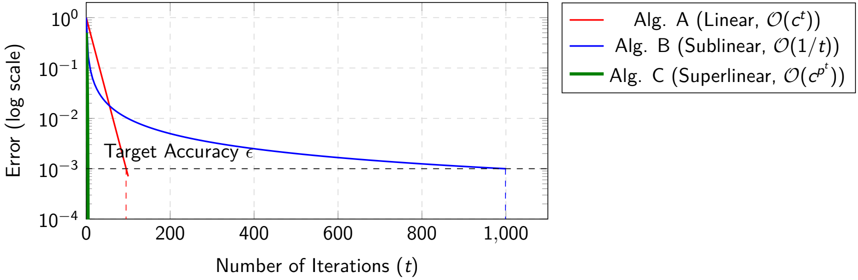
Convergence rate: if $\eta_t \equiv \eta = \frac{2}{\lambda_1(\mathbf{Q})+\lambda_n(\mathbf{Q})}$, then

$$\|\mathbf{x}^t - \mathbf{x}^*\|_2 \le \left(\frac{\lambda_1(\mathbf{Q}) - \lambda_n(\mathbf{Q})}{\lambda_1(\mathbf{Q}) + \lambda_n(\mathbf{Q})}\right)^t \|\mathbf{x}^0 - \mathbf{x}^*\|_2$$

where $\lambda_1(\mathbf{Q})$ (resp. $\lambda_n(\mathbf{Q})$) is the largest (smallest) eigenvalue of $\mathbf{Q}$.

- often called **linear convergence** or **geometric convergence**
- since the error lies below a log-linear plot of error vs. iteration count

Different rates of convergence speed



Convergence for constant stepsizes

Convergence rate: if $\eta_t \equiv \eta = \frac{2}{\lambda_1(\mathbf{Q})+\lambda_n(\mathbf{Q})}$, then

$$\|\mathbf{x}^t - \mathbf{x}^*\|_2 \le \left(\frac{\lambda_1(\mathbf{Q}) - \lambda_n(\mathbf{Q})}{\lambda_1(\mathbf{Q}) + \lambda_n(\mathbf{Q})}\right)^t \|\mathbf{x}^0 - \mathbf{x}^*\|_2$$

where $\lambda_1(\mathbf{Q})$ (resp. $\lambda_n(\mathbf{Q})$) is the largest (smallest) eigenvalue of $\mathbf{Q}$.

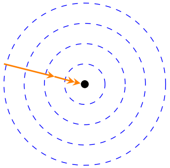
- the convergence rate is dictated by the **condition number**

$$\kappa := \frac{\lambda_1(\mathbf{Q})}{\lambda_n(\mathbf{Q})} \text{ of } \mathbf{Q}\text{, or equivalently, } \kappa := \frac{\max_{\mathbf{x}} \lambda_1(\nabla^2 f(\mathbf{x}))}{\min_{\mathbf{x}} \lambda_n(\nabla^2 f(\mathbf{x}))}$$

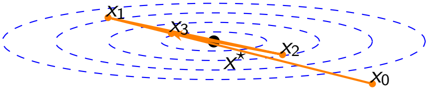
The impact of the condition number κ

The condition number of $\mathbf{Q}$ dictates the geometry of the loss surface.

Well-conditioned ($\kappa \approx 1$)



Ill-conditioned ($\kappa \gg 1$)



$\Rightarrow$ When contours are circular, the negative gradient points directly at the minimum. Convergence is fast.

$\Rightarrow$ When contours are highly elliptical, the gradient is almost orthogonal to the direction of the minimum, causing zig-zags.

Exact line search

The stepsize rule $\eta_t \equiv \eta = \frac{2}{\lambda_1(\mathbf{Q})+\lambda_n(\mathbf{Q})}$ relies on the spectrum of $\mathbf{Q}$, which requires preliminary experimentation.

Another more practical strategy is the **exact line search** rule

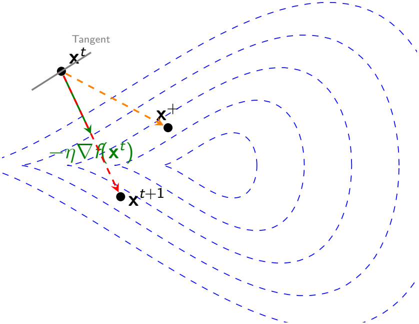
$$\eta_t = \arg\min_{\eta \geq 0} f(\mathbf{x}^t - \eta \nabla f(\mathbf{x}^t)) \tag{4}$$

Exact line search

The stepsize rule $\eta_t \equiv \eta = \frac{2}{\lambda_1(\mathbf{Q})+\lambda_n(\mathbf{Q})}$ relies on the spectrum of $\mathbf{Q}$, which requires preliminary experimentation.

Another more practical strategy is the **exact line search** rule

$$\eta_t = \arg\min_{\eta \geq 0} f(\mathbf{x}^t - \eta \nabla f(\mathbf{x}^t)) \tag{4}$$



Convergence for exact line search*

Convergence rate: if $\eta_t = \arg\min_{\eta \geq 0} f(\mathbf{x}^t - \eta \nabla f(\mathbf{x}^t))$, then

$$f(\mathbf{x}^t) - f(\mathbf{x}^*) \leq \left(\frac{\lambda_1(\mathbf{Q}) - \lambda_n(\mathbf{Q})}{\lambda_1(\mathbf{Q}) + \lambda_n(\mathbf{Q})} \right)^{2t} \left(f(\mathbf{x}^0) - f(\mathbf{x}^*) \right)$$

- stated in terms of the objective values
- convergence rate not faster than the constant stepsize rule

Convergence for exact line search*

Proof: For notational simplicity, let $\mathbf{g}^t = \nabla f(\mathbf{x}^t) = \mathbf{Q}(\mathbf{x}^t - \mathbf{x}^*)$. It can be verified that exact line search gives (only holds for quadratic loss)

$$\eta_t = \frac{(\mathbf{g}^t)^\top \mathbf{g}^t}{(\mathbf{g}^t)^\top \mathbf{Q}\mathbf{g}^t}.$$

Using $f(\mathbf{x}^t) = \frac{1}{2}(\mathbf{x}^t - \mathbf{x}^*)^\top \mathbf{Q}(\mathbf{x}^t - \mathbf{x}^*) = \frac{1}{2}(\mathbf{g}^t)^\top \mathbf{Q}^{-1}\mathbf{g}^t$, this gives

$$\begin{aligned}
f(\mathbf{x}^{t+1}) &= \frac{1}{2}(\mathbf{x}^t - \eta_t \mathbf{g}^t - \mathbf{x}^*)^\top \mathbf{Q}(\mathbf{x}^t - \eta_t \mathbf{g}^t - \mathbf{x}^*) \\
&= \frac{1}{2}(\mathbf{x}^t - \mathbf{x}^*)^\top \mathbf{Q}(\mathbf{x}^t - \mathbf{x}^*) - \eta_t ||\mathbf{g}^t||_2^2 + \frac{\eta_t^2}{2}(\mathbf{g}^t)^\top \mathbf{Q}\mathbf{g}^t \\
&= \frac{1}{2}(\mathbf{x}^t - \mathbf{x}^*)^\top \mathbf{Q}(\mathbf{x}^t - \mathbf{x}^*) - \frac{||\mathbf{g}^t||_2^4}{2(\mathbf{g}^t)^\top \mathbf{Q}\mathbf{g}^t} \\
&= \left(1 - \frac{||\mathbf{g}^t||_2^4}{((\mathbf{g}^t)^\top \mathbf{Q}\mathbf{g}^t)(((\mathbf{g}^t)^\top \mathbf{Q}^{-1}\mathbf{g}^t)}\right) f(\mathbf{x}^t).
\end{aligned}$$

Convergence for exact line search*

Proof (cont.): From Kantorovich's inequality

$$\frac{||\mathbf{y}||_2^4}{(\mathbf{y}^\top \mathbf{Q}\mathbf{y})(\mathbf{y}^\top \mathbf{Q}^{-1}\mathbf{y})} \geq \frac{4\lambda_1(\mathbf{Q})\lambda_n(\mathbf{Q})}{(\lambda_1(\mathbf{Q})+\lambda_n(\mathbf{Q}))^2},$$

we arrive at

$$\begin{aligned} f(\mathbf{x}^{t+1}) &\leq \left(1 - \frac{4\lambda_1(\mathbf{Q})\lambda_n(\mathbf{Q})}{(\lambda_1(\mathbf{Q})+\lambda_n(\mathbf{Q}))^2}\right) f(\mathbf{x}^t) \\ &= \left(\frac{\lambda_1(\mathbf{Q})-\lambda_n(\mathbf{Q})}{\lambda_1(\mathbf{Q})+\lambda_n(\mathbf{Q})}\right)^2 f(\mathbf{x}^t) \end{aligned}$$

This concludes the proof since $f(\mathbf{x}^*) = \min_{\mathbf{x}} f(\mathbf{x}) = 0$ $\square$

Table of Contents

In-Class Lab: Calculating the descent

Goal: To manually compute the first few steps of Gradient Descent and see how the learning rate (η) affects convergence.

The Setup

- **Our function:** A simple parabola, $f(x) = x^2$.
- **Its gradient:** $f'(x) = 2x$. The minimum is at $x = 0$.
- **The GD update rule:** $x_{t+1} = x_t - \eta \cdot f'(x_t)$.

Part 1: A "Good" learning rate

Let's start at $\mathbf{x_0} = \mathbf{4}$ with a learning rate of $\eta = \mathbf{0.1}$.

Instructions: Fill out the table below for the first 3 steps of GD.

$\mathbf{t}$	$\mathbf{x_t}$	$\mathbf{f'(x_t) = 2x_t}$	$\eta \cdot \mathbf{f'(x_t)}$	$\mathbf{x_{t+1}}$
0	4	8	0.8	3.2
1	3.2			
2				

Question

What do you observe about the value of x_t? Is it approaching the minimum at $x = 0$?

Part 2: A “Bad” learning rate

Let's see what happens if the learning rate is too large. Start again at $\mathbf{x_0} = \mathbf{4}$ but with $\eta = \mathbf{1.1}$.

Instructions: Fill out the table for the first 3 steps.

$\mathbf{t}$	$\mathbf{x_t}$	$\mathbf{f'(x_t) = 2x_t}$	$\eta \cdot \mathbf{f'(x_t)}$	$\mathbf{x_{t+1}}$
0	4	8	8.8	-4.8
1	-4.8			
2				

Question

What is happening to the value of x_t now? Is the algorithm converging?

Part 3: Backtracking line search

Instead of a fixed η, let's find one automatically at the first step ($t = 0$).

- Start at $\mathbf{x_0} = \mathbf{4}$ and backtracking parameters: $\alpha = \mathbf{0.5}$, $\beta = \mathbf{0.5}$.
- Start with an initial guess of $\eta = \mathbf{1.0}$.

Instructions: Check the Armijo condition. If it's true, update $\eta \leftarrow \beta\eta$ and check again. Stop when the condition is false.

$$\textbf{Check if}: \quad f(x_t - \eta \nabla f(x_t)) > f(x_t) - \alpha\eta ||\nabla f(x_t)||_2^2$$

Current η	**LHS**: $f(4 - \eta \cdot 8)$	**RHS**: $16 - 0.5 \cdot \eta \cdot 64$	**Is LHS > RHS?**
1.0	$(4-8)^2 = 16$	$16 - 32 = -16$	Yes
0.5			
...			

What is the final step size η_0 accepted by the algorithm? Using this η_0, what is the next point, x_1?

Solutions: The effect of the learning rate

Part 1: Converging ($\eta = 0.1$)

$\mathbf{t}$	$\mathbf{x_t}$	$\mathbf{f'(x_t)}$	$\eta \cdot \mathbf{f'(x_t)}$	$\mathbf{x_{t+1}}$
0	4.0	8.0	0.8	3.2
1	3.2	6.4	0.64	2.56
2	2.56	5.12	0.512	2.048

Part 2: Diverging ($\eta = 1.1$)

$\mathbf{t}$	$\mathbf{x_t}$	$\mathbf{f'(x_t)}$	$\eta \cdot \mathbf{f'(x_t)}$	$\mathbf{x_{t+1}}$
0	4.0	8.0	8.8	-4.8
1	-4.8	-9.6	-10.56	5.76
2	5.76	11.52	12.672	-6.912

A learning rate that is too large can cause the algorithm to overshoot the minimum and diverge completely.

Part 3: Backtracking solution: The final accepted step size is $\eta_0 = \mathbf{0.5}$. The next point is:

$$x_1 = x_0 - \eta_0 \cdot f'(x_0) = 4 - 0.5 \cdot 8 = 0$$

Backtracking prevents divergence by finding a safe stepsize automatically.

Recap and fine-tuning

- What we have talked about today?
  - $\Rightarrow$ What is gradient descent and why it works?
  - $\Rightarrow$ What is its performance on quadractic minimization?

Welcome anonymous survey!