

DSCI 565: OPTIMIZATION ALGORITHMS

*This content is protected and may not
be shared, uploaded, or distributed.*

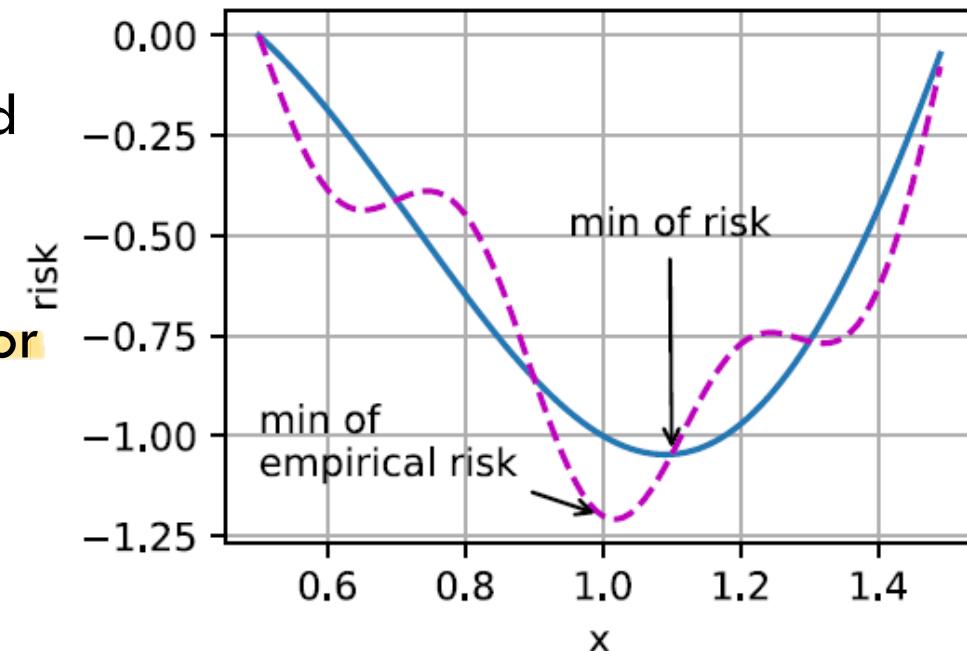
Ke-Thia Yao

Lecture 20: 2025-10-20

Optimization and Deep Learning

2

- Goal of optimization:
 - Given objective function $f(x)$ find x that minimizes the objective function
- Goal of deep learning:
 - Finding a suitable model, given a finite amount of data
- These two goals are not the same
 - If we set objective function to be the loss function and find optimum parameter x^* , then we are minimizing the training error
 - For deep learning we care about generalization error
 - We want select the right model that neither overfits the data, nor underfits the data



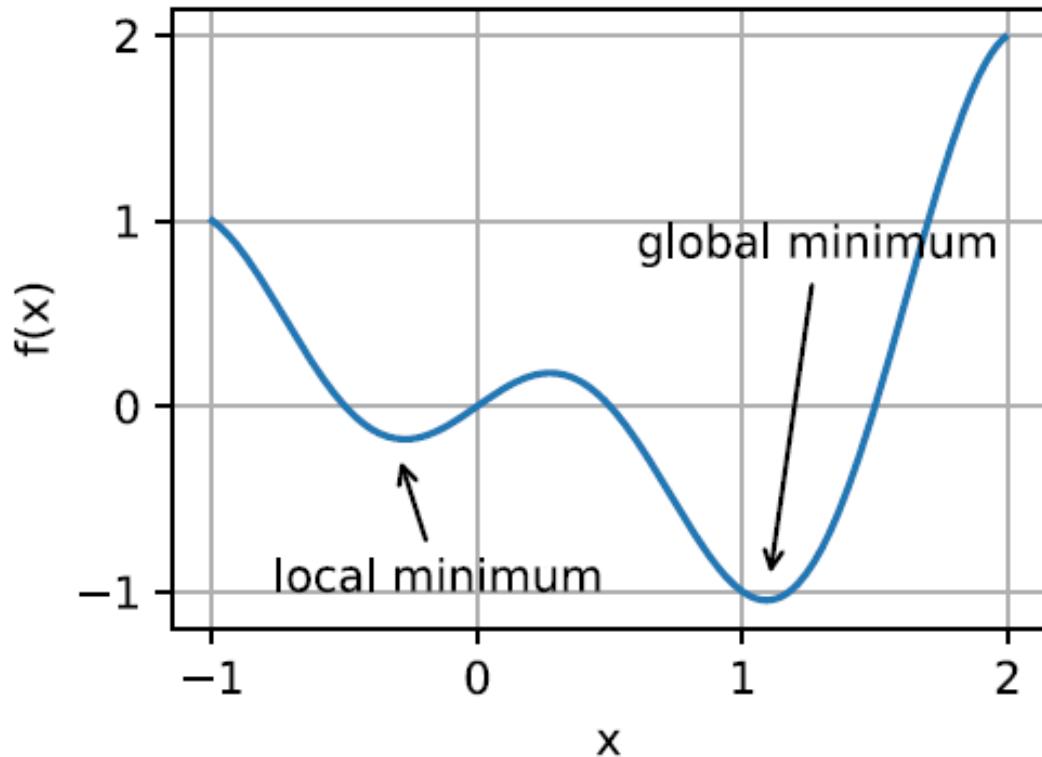
Optimization Challenges in Deep Learning

3

- In deep learning, the objective functions are complicated
- They do not have analytical solutions
- Optimization problems include
 - Local minima
 - Saddle points
 - Vanishing gradients

Local Minima

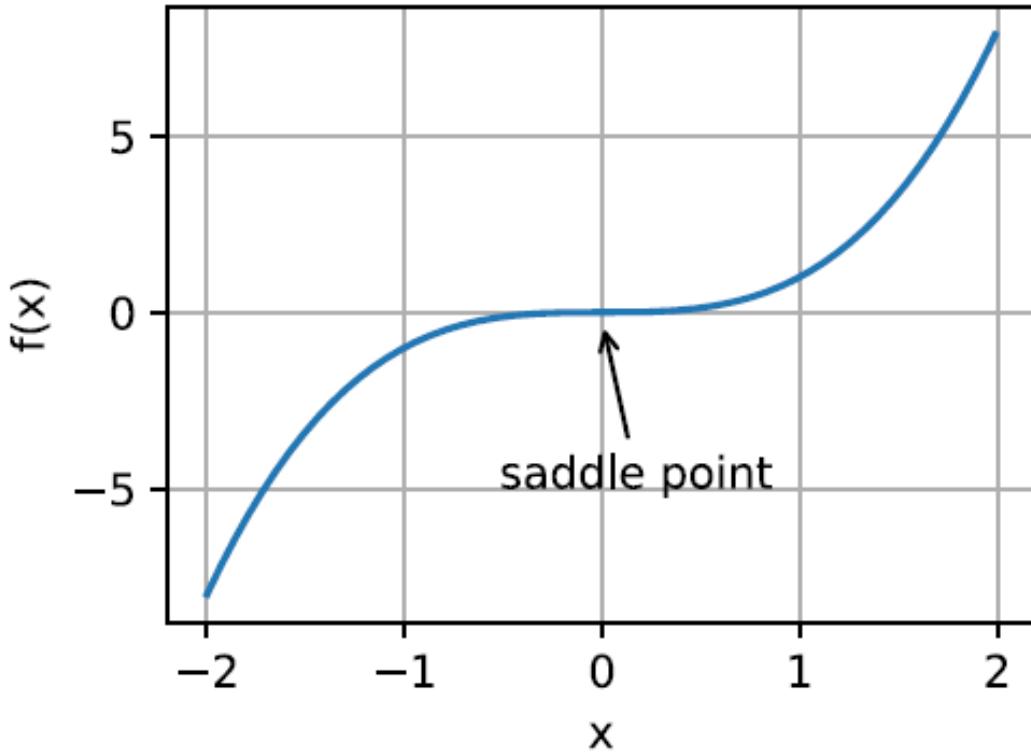
4



- For any objective function $f(x)$, x is a local minima if $f(x)$ is smaller than $f(x + \epsilon)$ for some ϵ
- At a local minima the gradient is zero
- For deep learning objective functions, there are many local minima

Saddle Points

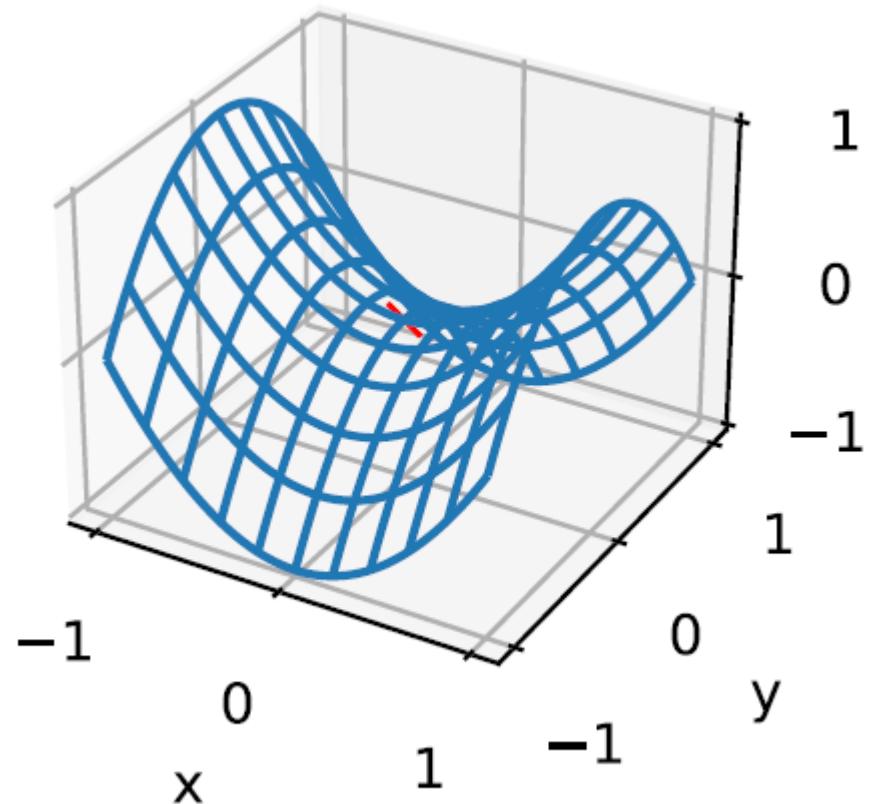
5



- At a saddle point the gradient vanishes, but it is neither a minima nor a maxima
- Simply finding locations where the gradients vanish is not sufficient
- Consider the function $f(x) = x^3$

Saddle Points in Higher Dimensions

6



- In higher dimensions, a saddle point look like
 - A minima in one dimension
 - A maxima in another dimension
- Consider $f(x, y) = x^2 - y^2$
- Use **Hessian matrix** to determine type of critical point

Hessian Matrix

7

- For a function $f: \mathbb{R}^n \rightarrow \mathbb{R}$, the Hessian matrix \mathbf{H} of f is all is second order partial derivatives:

$$\mathbf{H} \equiv \begin{bmatrix} \frac{\partial f}{\partial x_1^2} & \frac{\partial f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial f}{\partial x_1 \partial x_n} \\ \frac{\partial f}{\partial x_2 \partial x_1} & \frac{\partial f}{\partial x_2^2} & \cdots & \frac{\partial f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_n \partial x_1} & \frac{\partial f}{\partial x_n \partial x_2} & \cdots & \frac{\partial f}{\partial x_n^2} \end{bmatrix}$$

Hessian Matrix Interpretation

8

- At a **critical point** of function f , i.e., gradient of f is zero
 - When the **eigenvalues** of the function's Hessian matrix are **all positive**, we have a **local minimum** for the function
 - When the eigenvalues of the function's Hessian matrix are **all negative**, we have a **local maximum** for the function
 - When the eigenvalues of the function's Hessian matrix are **negative and positive**, we have a **saddle point** for the function

Hessian Matrix Example

9

- For $f(x, y) = \frac{1}{2}(x^2 - y^2)$

$$H = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{1}{2} \cdot 2 = 1$$

$$\frac{\partial^2 f}{\partial y^2} = \frac{1}{2} \cdot (-2) = -1$$

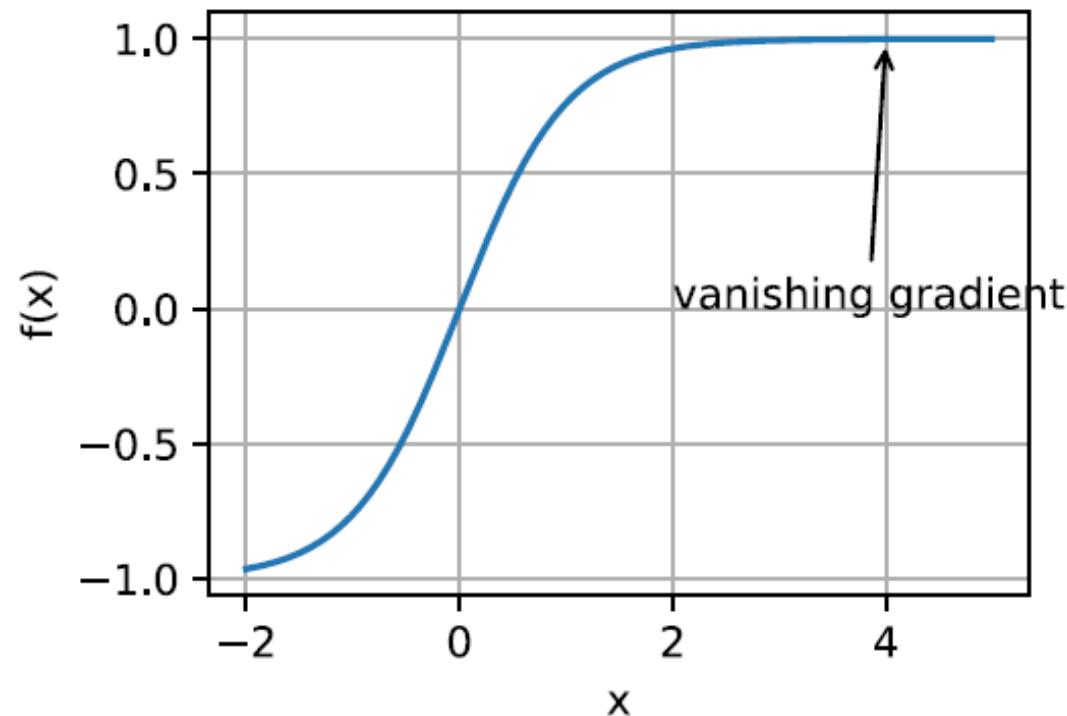
- The eigenvalues of its Hessian matrix is 1 and -1:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = -1 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- Critical point (0,0) is a saddle point

Vanishing Gradients

10



- At these points, the gradients become close to zero
- Optimization gets stuck and causes the learning to be slow

Convexity

11

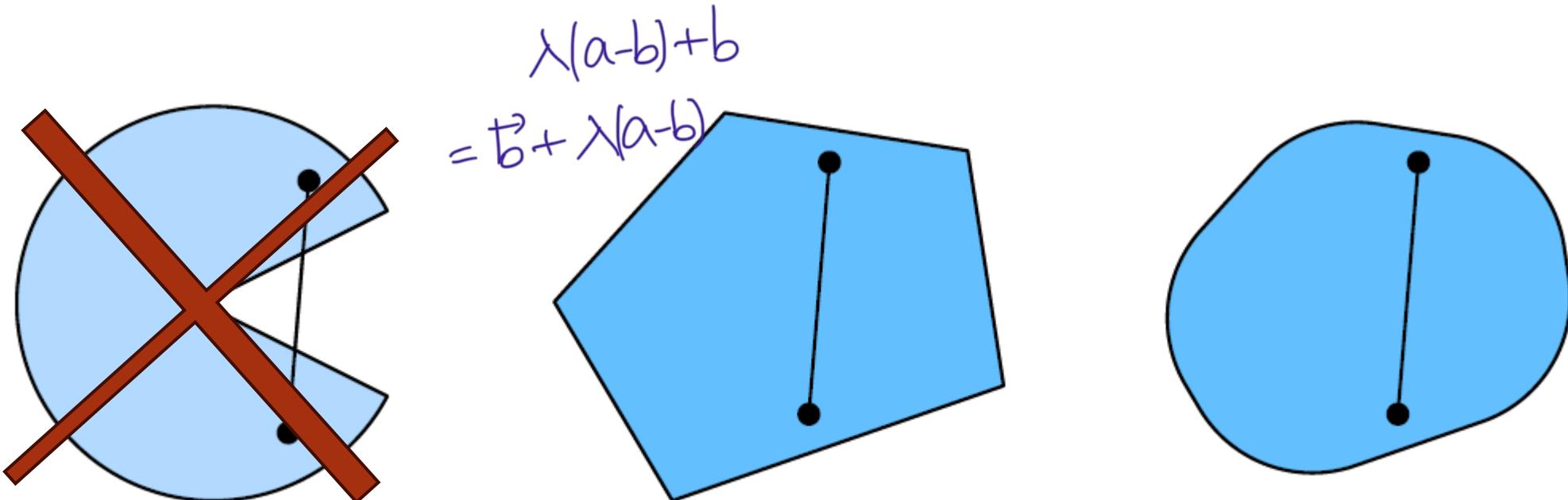
- Optimization problems that obey convexity properties are easier to analyze and solve
- If an algorithm performs poorly for convex problems, then it is unlikely to do well for nonconvex problems
- Deep learning optimization problems are typically nonconvex, but near local optima they look like convex problems

Convex Sets

12

- A set \mathcal{X} in a vector space is convex if for any $a, b \in \mathcal{X}$ the line segment connecting a and b is also in \mathcal{X}

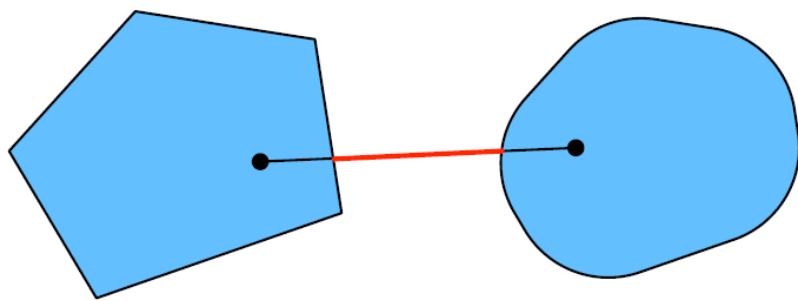
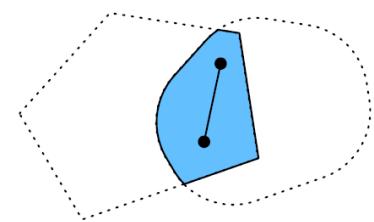
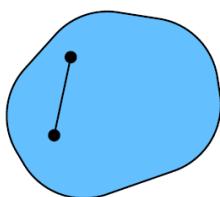
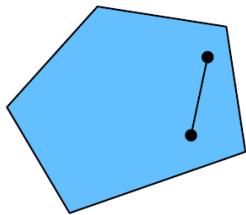
$$\lambda a + (1 - \lambda)b \in \mathcal{X}, \quad \text{for } \lambda \in [0, 1]$$



Convex Sets

13

- Intersection of convex sets is convex
- Union of convex sets need not be convex

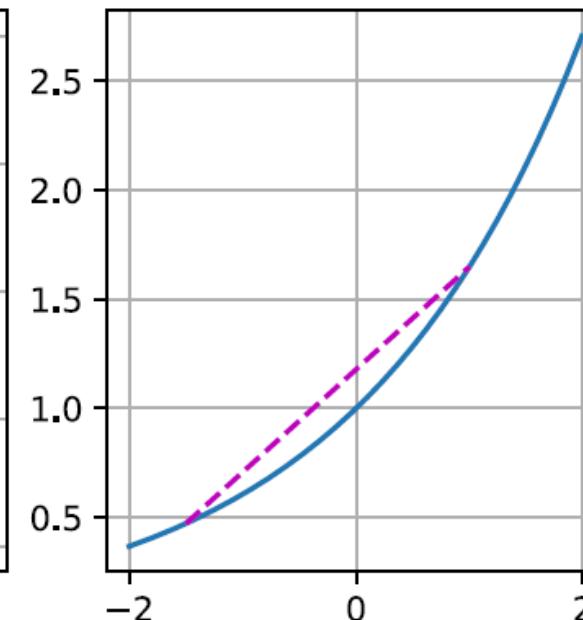
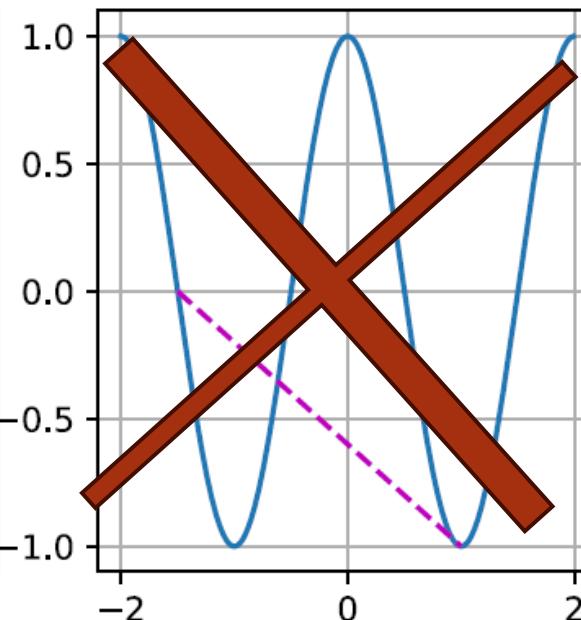
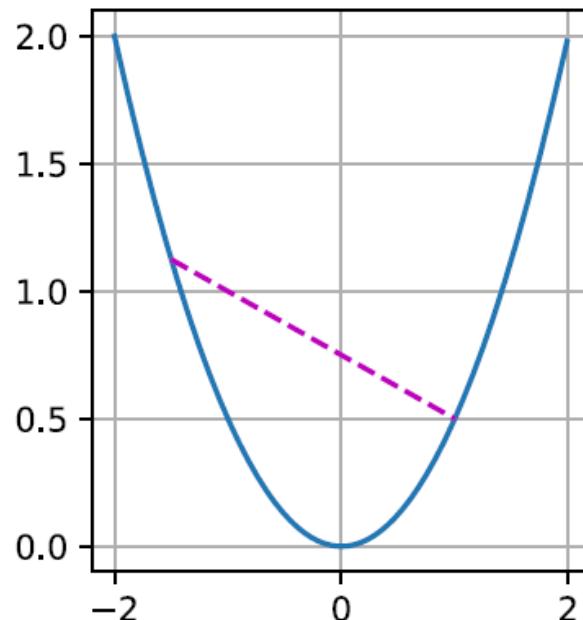


Convex Functions

14

- Given a convex set \mathcal{X} , a function $f: \mathcal{X} \rightarrow \mathbb{R}$ is convex if for all $x, x' \in \mathcal{X}$ and for all $\lambda \in [0,1]$

$$\lambda f(x) + (1 - \lambda)f(x') \geq f(\lambda x + (1 - \lambda)x')$$



Convex functions

15

□ Examples of convex functions

- $f(\mathbf{x}) = \mathbf{a}^T \mathbf{x} + \mathbf{b}$

- $f(\mathbf{x}) = \mathbf{x}^T \mathbf{x}$

- $f(\mathbf{x}) = \|\mathbf{x}\|_1$

Jensen's Inequality

16

- For convex functions, Jensen's inequality states

$$\sum_i \alpha_i f(x_i) \geq f\left(\sum_i \alpha_i x_i\right) \text{ and } E_X[f(X)] \geq f(E_X[X]),$$

- Where $\alpha_i \geq 0$ such that $\sum_i \alpha_i = 1$, and X is random variable
- Jensen's inequality provides a lower bound, which is usually simpler than original expression

Properties of Convex Functions

17

- Local minima are global minima
- Below sets of convex functions are convex
- A function is convex if its Hessian matrix H is semi-positive definite

Convexity: Local Minima is Global Minima

18

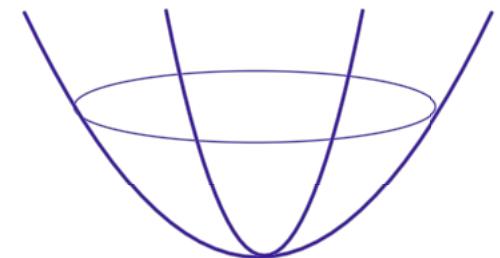
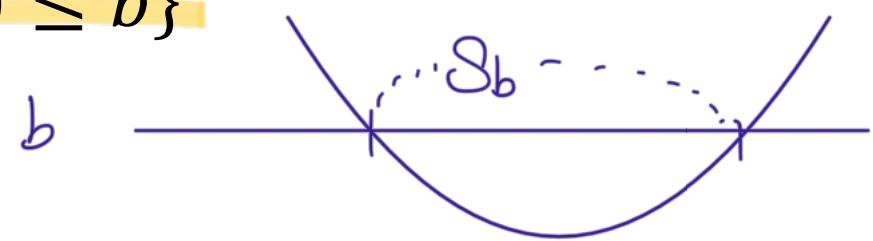
- Let f be a convex function defined on a convex set \mathcal{X}
If $x^* \in \mathcal{X}$ is local minima, then x^* is a global minima
- Proof by contradiction
 - Suppose there exists $x' \in \mathcal{X}$, where $f(x') < f(x^*)$
 - If x^* is local minima, then there is a neighbor near x^* , $0 < |x - x^*| \leq P$, such that $f(x^*) < f(x)$
 - We can find a λ such that $\lambda x^* + (1 - \lambda)x'$ is in this neighborhood
 - But this contradicts that x^* is a local minima

$$\begin{aligned}f(\lambda x^* + (1 - \lambda)x') &\leq \lambda f(x^*) + (1 - \lambda)f(x') \\&< \lambda f(x^*) + (1 - \lambda)f(x^*) \\&= f(x^*),\end{aligned}$$

Below Sets of Convex Functions are Convex

19

- We can define convex sets using *below sets* of convex functions
- Given a convex function f defined on a convex set \mathcal{X} , any below set
$$\mathcal{S}_b \equiv \{x | x \in \mathcal{X} \text{ and } f(x) \leq b\}$$
is convex.
- Proof:
 - For any $x, x' \in \mathcal{S}_b$ we need to show $\lambda x + (1 - \lambda)x' \in \mathcal{S}_b$ for $\lambda \in [0, 1]$
 - But $f(\lambda x + (1 - \lambda)x') \leq \lambda f(x) + (1 - \lambda)f(x') \leq b$
 - By definition of convexity and $f(x) \leq b$ and $f(x') \leq b$



Convexity and Second Derivative

20

- A function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if and only if the Hessian H of f is positive semi-definite
- A matrix H is positive semi-definite if for all $\mathbf{x} \in \mathbb{R}^n$

$$\mathbf{x}^T H \mathbf{x} \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n$$

- Proof is in Section 12.2.2

i.g. $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{x}\|^2 \quad H = I$

$$\therefore \mathbf{x}^T H \mathbf{x} = \mathbf{x}^T \mathbf{x} > 0$$

Constrained Optimization

21

- Constrained optimization problems

$$\min_x f(x)$$

subject to $c_i(x) \leq 0$ for all $i \in \{1, \dots, n\}$

↑ constrain

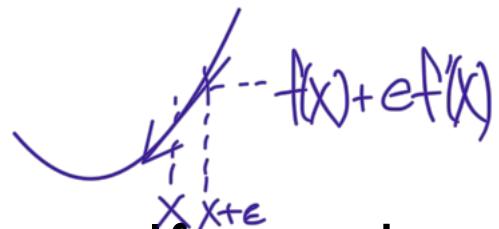
where f is the objective function and c_i are constraint functions

- Convex optimization: If f is convex and c_i define convex sets then there are polynomial algorithms to find optimal x
- Example convex constraints: $c_1(x) = \|x\|_2 - 1$ or $c_2(x) = v^T x + b$
- In general optimization problems are NP-hard

One-Dimensional Gradient Descent

22

- Let $f: \mathbb{R} \rightarrow \mathbb{R}$ be continuously differentiable, using Taylor expansion



$$f(x + \epsilon) = f(x) + \epsilon f'(x) + O(\epsilon^2)$$

- If we take a “small” step, $\eta > 0$, in the negative gradient direction

$$f(x - \eta f'(x)) = f(x) - \eta f'^2(x) + O(\eta^2 f'^2(x)).$$

- If the derivative $f'(x) \neq 0$, then $\eta f'^2(x) > 0$. For small enough η

$$f(x - \eta f'(x)) \lesssim f(x)$$

- Gradient descent iterate using

$$x \leftarrow x - \eta f'(x)$$

Learning Rate

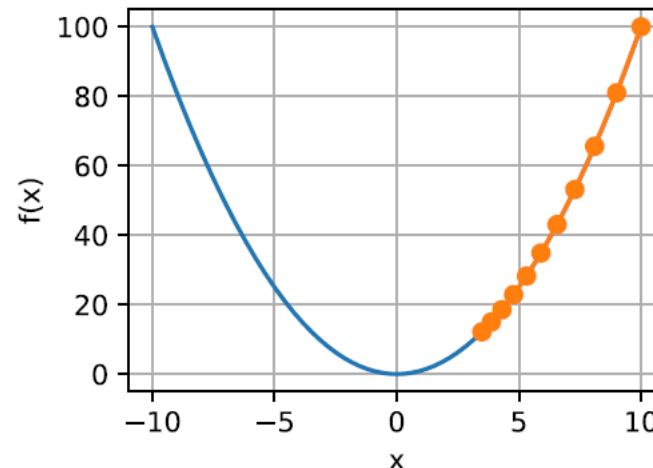
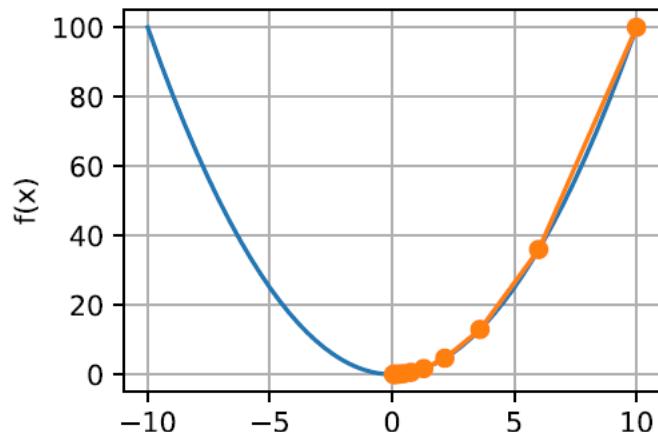
$$f(x - \eta f'(x)) \leq f(x)$$

23

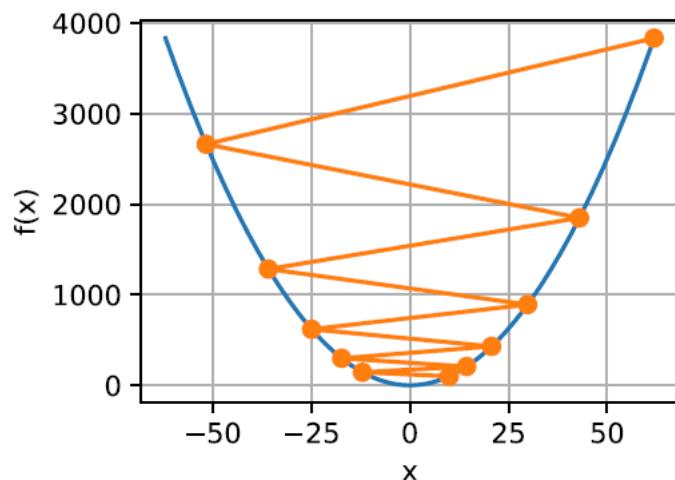
- The learning rate η controls the size of the step
- If η is too small, then convergence to local minima is slow
- If η is too large, then higher order terms of the Taylor expansion $\mathcal{O}(\eta^2 f''^2(x))$ may become significant
- The gradient descent may overshoot and diverge

Learning Rate and Convergence

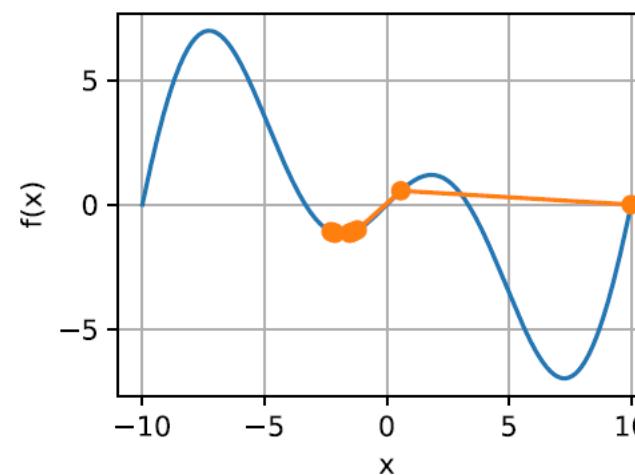
24



η too small



η too large



local minimum

Multivariate Gradient Descent

25

- Let $f: \mathbb{R}^d \rightarrow \mathbb{R}$, then the gradient of f is

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_d} \right]^T$$

 \vec{x}

- The Taylor expansion is

$$f(\mathbf{x} + \boldsymbol{\epsilon}) = f(\mathbf{x}) + \boldsymbol{\epsilon}^T \nabla f(\mathbf{x}) + \mathcal{O}(\|\boldsymbol{\epsilon}\|^2)$$

- Gradient descent iterate using

$$\mathbf{x} := \mathbf{x} - \eta \nabla f(\mathbf{x})$$

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f(\mathbf{x})$$

Newton's Method

26

- For function $f: \mathbb{R}^d \rightarrow \mathbb{R}$, we add another term to the Taylor expansion

$$f(\mathbf{x} + \boldsymbol{\epsilon}) = f(\mathbf{x}) + \boldsymbol{\epsilon}^T \nabla f(\mathbf{x}) + \frac{1}{2} \boldsymbol{\epsilon}^T \nabla^2 f(\mathbf{x}) \boldsymbol{\epsilon} + \mathcal{O}(\|\boldsymbol{\epsilon}\|^3)$$

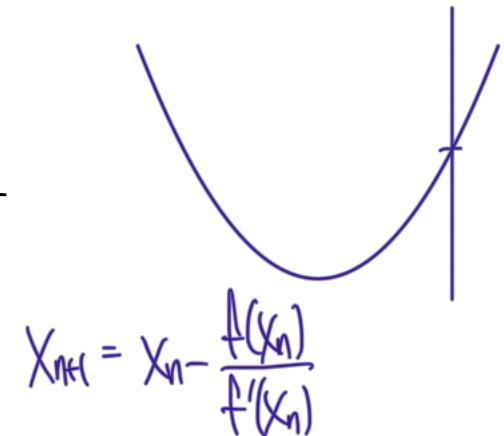
- Where Hessian $\mathbf{H} \equiv \nabla^2 f(\mathbf{x})$
- Take derivative with respect to $\boldsymbol{\epsilon}$, then $\nabla f(\mathbf{x}) + \mathbf{H}\boldsymbol{\epsilon} = 0$

$$\boldsymbol{\epsilon} = -\mathbf{H}^{-1} \nabla f(\mathbf{x})$$

- For $f(x) = \frac{1}{2}x^2$, gradient is $\nabla f(x) = x$ and $H = 1$

- For any x , $\boldsymbol{\epsilon} = -\frac{f'}{f''} = -x$

- Only need one step to reach the global minimum



Notebook

27

- chapter_optimization/gd.ipynb

Newton's Method Convergence

28

- If we are sufficiently close to the minimum, then the error decreases quadratically with each iteration
- Let $x^{(k)}$ be the value of x at the k^{th} iteration, and let error $e^{(k)} \equiv x^{(k)} - x^*$, where x^* is the minimum, then

$$|e^{(k+1)}| \leq c(e^{(k)})^2$$

- Where $\frac{|f'''(\xi^{(k)})|}{2f''(x^{(k)})} \leq c$, $\xi^{(k)} \in [x^{(k)} - e^{(k)}, x^{(k)}]$

Preconditioning

29

- Storing the full Hessian is very expensive, especially for deep learning
- One workaround is to only compute the **diagonal** of the Hessian

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \text{diag}(\mathbf{H})^{-1} \nabla f(\mathbf{x})$$

- This is called **preconditioning**
- Effectively preconditioning selects a **different learning rate for each variable**
- For example, consider if one variable is in millimeters and another in meters

Gradient Descent with Line Search

30

- Instead of fixing the learning rate η , at each step find the best rate by performing a binary search on η that minimized

$$f(\mathbf{x} - \eta \nabla f(\mathbf{x}))$$

- This approach has good convergence
- But it is very expensive for gradient descent, since each step of binary search requires evaluating the entire dataset

Stochastic Gradient Descent

31

- Let $f_i(x)$ be the loss with respect to training example $i \in [1, n]$ and x is the parameter vector
- The objective function and its gradient are:

$$f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) \quad \nabla f(x) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x)$$

- Stochastic gradient update:

$$x \leftarrow x - \eta \nabla f_i(x)$$

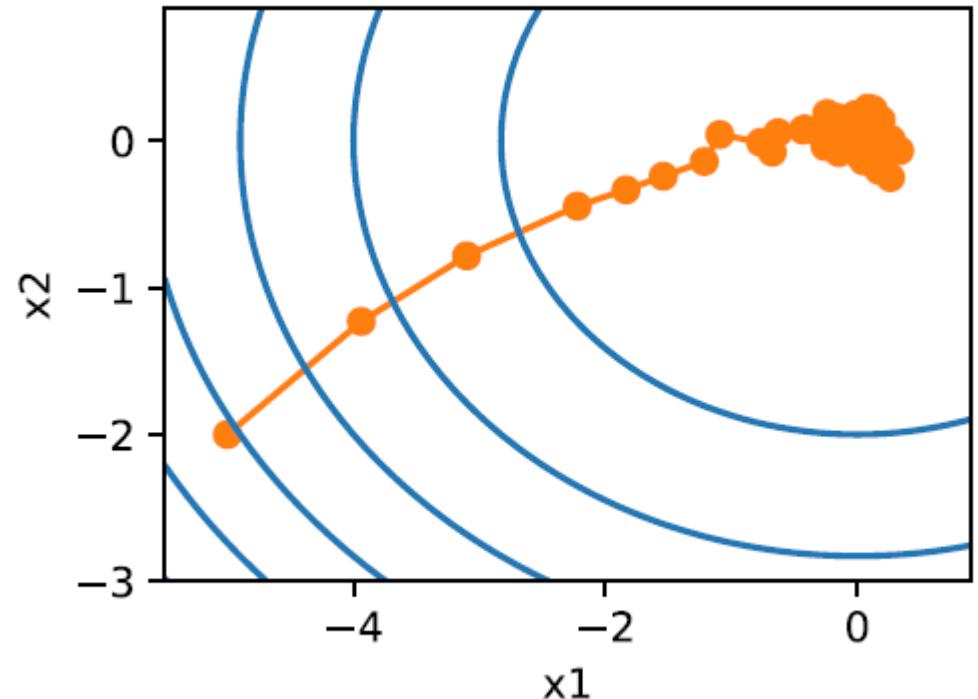
Stochastic Gradient Descent

32

- On average, $\nabla f_i(\mathbf{x})$ is a good estimate of the gradient

$$\mathbb{E}_i \nabla f_i(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\mathbf{x}) = \nabla f(\mathbf{x}).$$

- But it has too much variance
- See notebook
 - chapter_optimization/sgd.ipynb
 - Tend to wander, especially near optima



Dynamic Learning Rate for Stochastic Gradient Descent

33

- Adjust learning rate $\eta(t)$

$$\eta(t) = \eta_i \text{ if } t_i \leq t \leq t_{i+1} \quad \text{piecewise constant}$$

$$\eta(t) = \eta_0 \cdot e^{-\lambda t} \quad \text{exponential decay}$$

$$\eta(t) = \eta_0 \cdot (\beta t + 1)^{-\alpha} \quad \text{polynomial decay}$$

- Piecewise constant drop rate when progress stalls
 - Popular choice for deep learning
- Exponential decay maybe too drastic
- For polynomial decay, $\alpha = 0.5$ is a popular choice

Minibatch Stochastic Gradient Descent

34

- Instead of training one example at a time, train in small batches of examples of size b
- If we draw batches uniformly at random from training set, then
 - The expectation of the gradient is unchanged
 - The variance is reduced by a factor of $b^{-\frac{1}{2}}$
- In practice choose batch size that fits into GPU memory

Memory Bottleneck

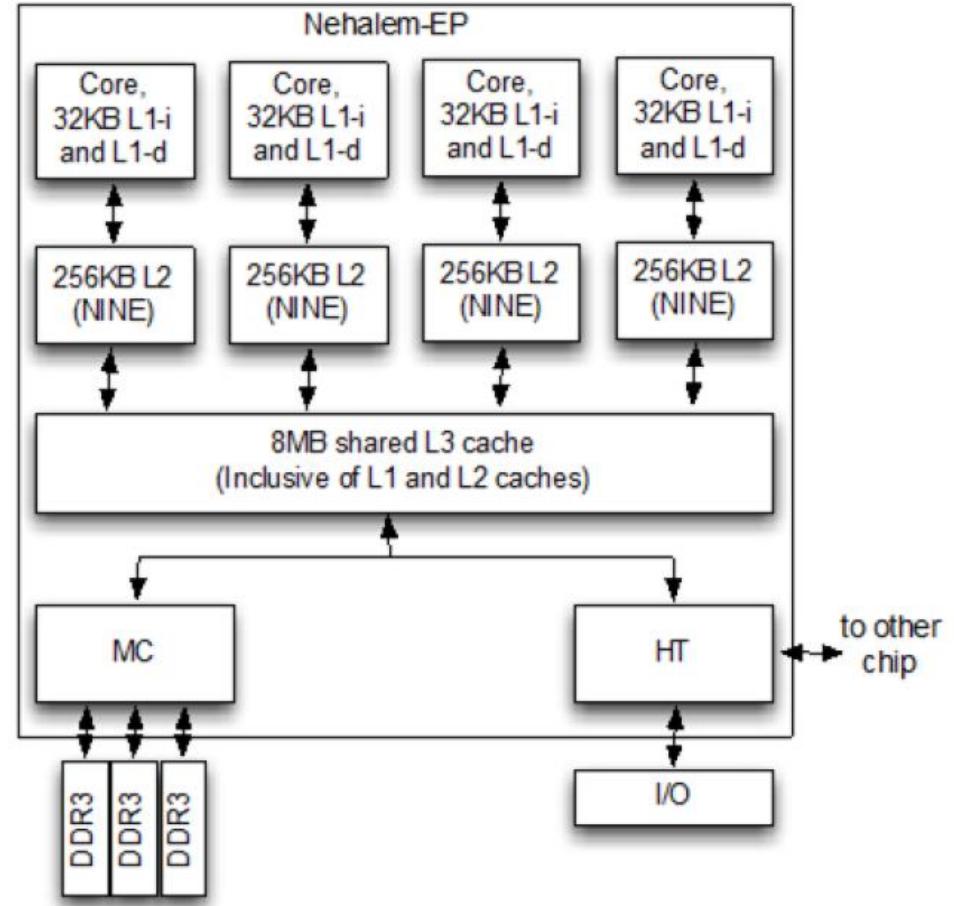
35

- CPUs and GPUs can process faster than memory can deliver the data
- Using vectorization (Single Instruction Multiple Data, SIMD), 2GHz CPU with 16 cores can process $2 * 10^9 * 16 * 32 = 10^{12} = 1000 \text{ GB/s}$
- Main memory of a midrange server can only deliver 100 GB/s
- With just main memory CPU is idle 90% of the time
- GPUs have many more cores

Cache Hierarchy

36

- Cache hierarchy provides multiple levels of memory stores, with varying access speed and size
- Level-one (L1) cache is fast but small
- L2 cache is slower and larger
- L3 cache is even slower and larger



Notebook

37

- chapter_optimization/minibatch-sgd.ipynb
 - Vectorization and caches
 - https://en.wikipedia.org/wiki/Cache_hierarchy

Momentum

38

- Instead of using $\mathbf{g}_{t,t-1}$, the gradient of the minibatch at time t , use a **leaky average** (aka exponentially weighted average, aka exponential moving average) of the past gradients
- Let \mathbf{v}_t be the velocity, $\mathbf{v}_t = \beta\mathbf{v}_{t-1} + \mathbf{g}_{t,t-1}$

$$\mathbf{v}_t = \beta^2\mathbf{v}_{t-2} + \beta\mathbf{g}_{t-1,t-2} + \mathbf{g}_{t,t-1} = \dots = \sum_{\tau=0}^{t-1} \beta^\tau \mathbf{g}_{t-\tau,t-\tau-1}.$$

- Up

- Advan $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \eta_t \mathbf{v}_t$.
 - Reduction in variance beyond a single batch
 - “Rolling down hill”
 - Helps with ill-conditioned problems

III-Conditioned Problem

39

- For $f(x, y) = 0.1x^2 + 2y^2$, the Hessian matrix is

$$H = \begin{bmatrix} 0.2 & 0 \\ 0 & 4 \end{bmatrix}$$

- The eigenvalues of its Hessian matrix is 0.2 and 4:

$$\begin{bmatrix} 0.2 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0.2 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0.2 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 4 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- The condition number is $\frac{4}{0.2} = 20$
- Gradient much larger direction y than direction x
- Larger number implies ill-conditioned

Notebook

40

- See chapter_optimization/momentum.ipynb