THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RELAXED SYSTEM LAB

Stochastic Gradient Descent

COMP6211J

Binhang Yuan

Release of the Topics

RELAXED SYSTEM LAB

- Available here:
  - https://github.com/Relaxed-System-Lab/COMP6211J_Course_HKUST/blob/main/topics.md
- 37 topics have been released;
- 75 people registered by 2024/09/04;
- The hope is that at most 4 people choose the same topic.
- Select ***three*** topics you are interested in most:
  - I will do some load balance for the topic allocation.
- Send out a form for you to fill out to indicate your preference on 2024/09/10.
  - You should submit it by ***2024/09/14 - 23:59***

Overview of the Topics

RELAXED
SYSTEM LAB

- Foundation Model Architectures:
  - 2 - Beyond Transformer Architecture
  - 35 - Text Image Generation
  - 36 - Text Video Generation
  - 37 - Text-to-3D Asset Generation

Overview of the Topics

RELAXED
SYSTEM LAB

- Hardware & System:
  - 1 - Tensor Program Optimization
  - 3 - NPU Architecture
  - 4 - TPU Architecture
  - 5 - Collective Communication Optimization
  - 6 - In Network Aggregation
  - 7 - Fairness in ML Service
  - 8 - Multi-Tenant LLM Service

Overview of the Topics

RELAXED
SYSTEM LAB

- ML System – Training Systems and Algorithms:
  - 9 - Data Parallel System Optimization
  - 10 - Data Parallel Algorithmic Optimization - Gradient Compression
  - 11 - Data Parallel Algorithmic Optimization - Asynchronous Training
  - 12 - Data Parallel Algorithmic Optimization - Decentralized Communication
  - 13 - Pipeline Parallel System Optimization
  - 14 - Tensor Model Parallel System Optimization
  - 15 - Optimizer Parallel System Optimization
  - 16 - Long Sequence Parallel Training
  - 17 - Mixture of Expert Parallel Training
  - 18 - Offloading Training System
  - 19 - Auto Parallelism
  - 20 - Robust Training

Overview of the Topics

RELAXED
SYSTEM LAB

- ML System – Inference Systems and Algorithms:
  - 21 - LLM Inference Weight and Activation Compression
  - 22 - LLM Inference KV Cache Compression
  - 23 - LLM Speculative and Parallel Decoding
  - 24 - Offloading Inference System
  - 25 - Disaggregate Generative Inference
  - 26 - Parallel Inference for Diffusion Model

Overview of the Topics

RELAXED
SYSTEM LAB

- Retrieval Augmented Generation:
  - 27 - Retrieval Augmented Generation Embedding Models
  - 28 - Retrieval Augmented Generation Chunk Optimization
  - 29 - Retrieval Augmented Generation Adaptive Retrieval
  - 30 - Retrieval Augmented Generation ReRanking
  - 31 - Security in Retrieval Augmented Generation
  - 33 - Table-Augmented Generation
  - 34 - Multimodal Retrieval Augmented Generation
- Others:
  - 32 - Multi-LLM Interaction

RELAXED
SYSTEM LAB

Preliminary - Linear Algebra

Scalars

RELAXED
SYSTEM LAB

- Sample operations

$$c = a + b$$

$$c = a \cdot b$$

$$c = \sin a$$

- Length

$$|a| = \begin{cases} \text{a} & if \;\; a > 0 \\ -a & otherwise \end{cases}$$

$$|a + b| \leq |a| + |b|$$

$$|a \cdot b| = |a| \cdot |b|$$

Vector

RELAXED
SYSTEM LAB

- Vector in n-dimensions $\boldsymbol{a} \in \mathbb{R}^n$, $\mathbf{a}= \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{bmatrix}$
- Sample operations:

$$\boldsymbol{c} = \boldsymbol{a} + \boldsymbol{b} \ \ where\ c_i = a_i + b_i$$

$$\boldsymbol{c} = \alpha \cdot \boldsymbol{b} \ \ where\ c_i = \alpha b_i$$

$$\boldsymbol{c} = \sin \boldsymbol{a} \ \ where\ c_i = \sin a_i$$

Vector

RELAXED
SYSTEM LAB

- Some properties of vector addition:
  - Commutative:

$$\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{b} + \boldsymbol{a}, \qquad \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$$

  - Associative:

$$(\boldsymbol{a} + \boldsymbol{b}) + \boldsymbol{c} = \boldsymbol{a} + (\boldsymbol{b} + \boldsymbol{c}), \qquad \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^n$$

  - Distributive:

$$\alpha(\boldsymbol{a} + \boldsymbol{b}) = \alpha\boldsymbol{a} + \alpha\boldsymbol{b}, \qquad \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$$

Vector

- p-norm

$$\|\boldsymbol{a}\|_p = \left[\sum_{i=1}^{m} a_i^p\right]^{\frac{1}{p}}$$

- p=1, Manhattan norm

$$\|\boldsymbol{a}\|_1 = \sum_{i=1}^{m} |a_i|$$

- p=2, Euclidean norm

$$\|\boldsymbol{a}\|_2 = \left[\sum_{i=1}^{m} a_i^2\right]^{\frac{1}{2}}$$

$$\|\boldsymbol{a}\| \geq 0 \; \forall \; \boldsymbol{a}$$

$$\|\boldsymbol{a} + \boldsymbol{b}\| \leq \|\boldsymbol{a}\| + \|\boldsymbol{b}\|$$

$$\|\alpha \cdot \boldsymbol{b}\| \leq |\alpha| \|\boldsymbol{b}\|$$

- p= ∞, infinity norm

$$\|\boldsymbol{a}\|_\infty = \max(a_1, a_2, \ldots, a_m)$$

Vector

RELAXED
SYSTEM LAB

- The span of a set of vectors:

$$\text{span}\{\boldsymbol{a_1}, \boldsymbol{a_2}, \dots, \boldsymbol{a_k}\} = \left\{\alpha_1 \boldsymbol{a_1} + \alpha_2 \boldsymbol{a_2} + \cdots + \alpha_k \boldsymbol{a_k} \middle| \alpha_k \in \mathbb{R},\ \boldsymbol{a_k} \in \mathbb{R}^n\right\}$$

- Linear independence:

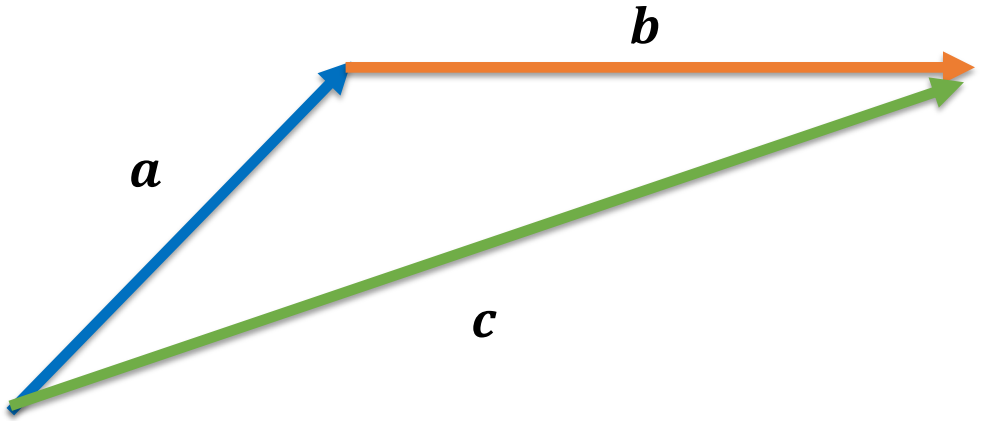
$$\alpha_1 \boldsymbol{a_1} + \alpha_2 \boldsymbol{a_2} + \cdots + \alpha_k \boldsymbol{a_k} = \boldsymbol{0} \Rightarrow\ \alpha_i = 0, \forall i$$

- How does k compare to n, the vector dimension?

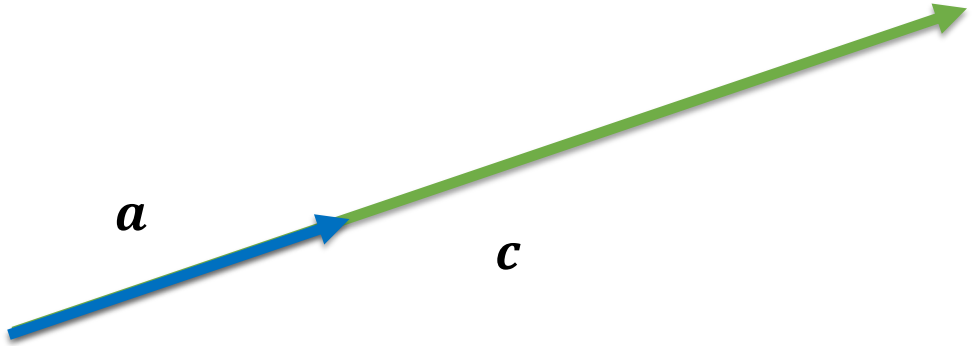
$$k \leq n$$

Vector

RELAXED
SYSTEM LAB



$$c = a + b$$



$$c = \alpha \cdot a$$

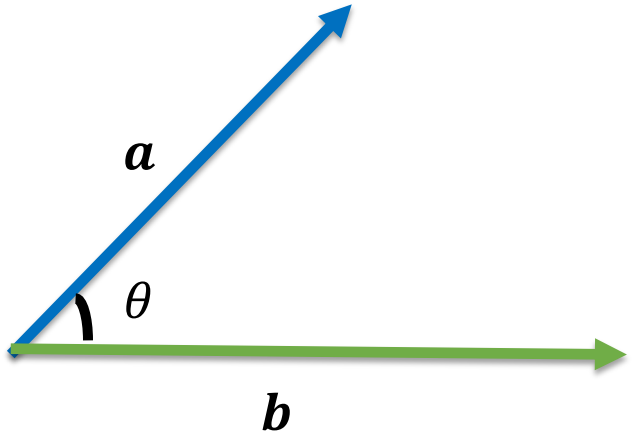
Mathematician's **'parallel for all do'**

Vectors

RELAXED
SYSTEM LAB

- Inner product

$$\boldsymbol{a}^T\boldsymbol{b} = \sum_i a_i b_i = \|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\| \cdot \cos\theta$$



- Orthogonality

$$\boldsymbol{a}^T\boldsymbol{b} = \sum_i a_i b_i = 0$$

  - If we have two vectors that are orthogonal with a third, their linear combination is, too.

Matrices

RELAXED
SYSTEM LAB

- Matrix in m, n-dimensions: $\boldsymbol{A} \in \mathbb{R}^{m \times n}$:

$$\mathbf{A} = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix}$$

- Transpose of matrix:

$$\mathbf{A}^{\mathrm{T}} = \begin{bmatrix} A_{11} & \cdots & A_{m1} \\ \vdots & \ddots & \vdots \\ A_{1n} & \cdots & A_{mn} \end{bmatrix} \in \mathbb{R}^{n \times m}$$

$$(\boldsymbol{A} + \boldsymbol{B})^T = \boldsymbol{A}^T + \boldsymbol{B}^T, \forall \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m \times n}$$

Matrices

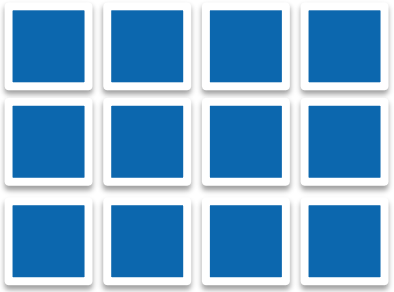
RELAXED
SYSTEM LAB

- Simple operations

$$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B} \;\; where \; C_{ij} = A_{ij} + B_{ij}$$

$$\boldsymbol{C} = \alpha \cdot \boldsymbol{B} \;\; where \; C_{ij} = \alpha B_{ij}$$

$$\boldsymbol{C} = \sin \boldsymbol{A} \;\; where \; C_{ij} = \sin A_{ij}$$



Matrices

RELAXED
SYSTEM LAB

- Some properties of matrix addition:
  - Commutative:

$$\boldsymbol{A} + \boldsymbol{B} = \boldsymbol{B} + \boldsymbol{A}, \qquad \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m \times n}$$

  - Associative:

$$(\boldsymbol{A} + \boldsymbol{B}) + \boldsymbol{C} = \boldsymbol{A} + (\boldsymbol{B} + \boldsymbol{C}), \qquad \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{m \times n}$$

  - Distibutive:

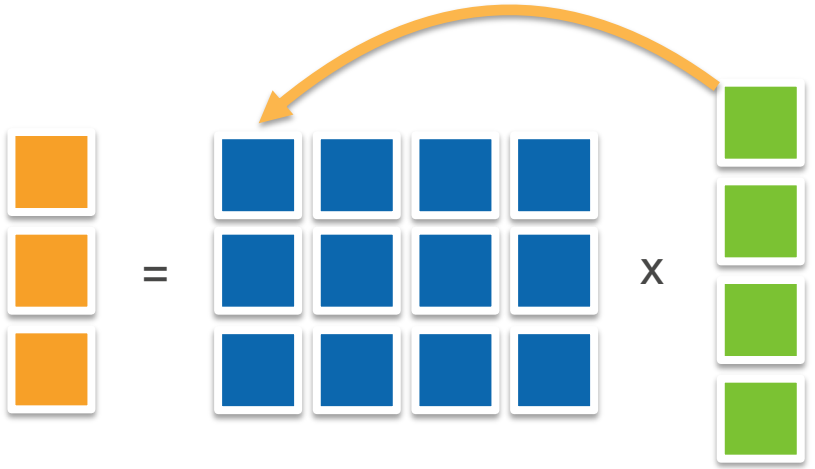
$$\alpha(\boldsymbol{A} + \boldsymbol{B}) = \alpha\boldsymbol{A} + \alpha\boldsymbol{B}, \qquad \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m \times n}$$

Matrices

RELAXED SYSTEM LAB

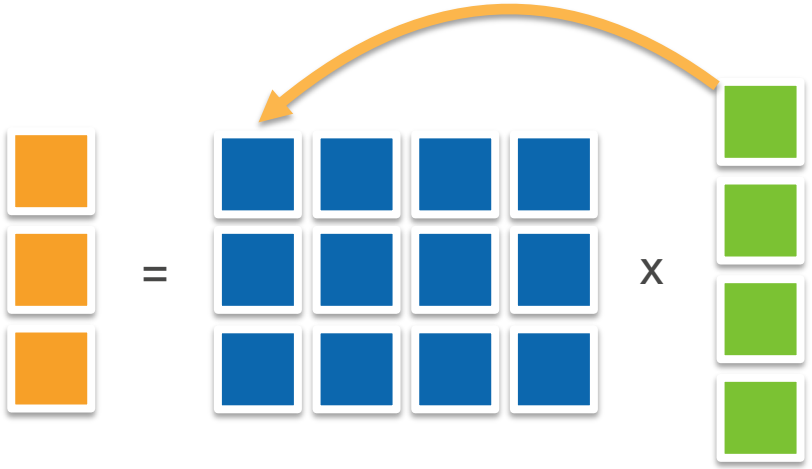
- Multiplications (matrix-vector), $\boldsymbol{c} = \boldsymbol{Ab}, \boldsymbol{c} \in \mathbb{R}^m, \boldsymbol{A} \in \mathbb{R}^{m \times n}, \boldsymbol{B} \in \mathbb{R}^n$

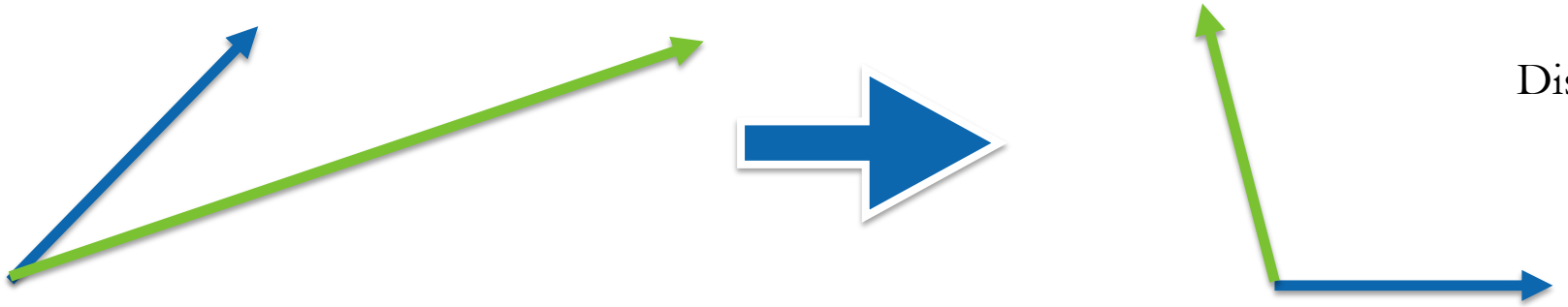
$$\begin{bmatrix} c_1 \\ \dots \\ c_m \end{bmatrix} = \boldsymbol{c} = \boldsymbol{Ab} = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix} \quad where\ c_i = \sum_{j=1}^{n} A_{ij} b_j$$



Matrices

RELAXED
SYSTEM LAB





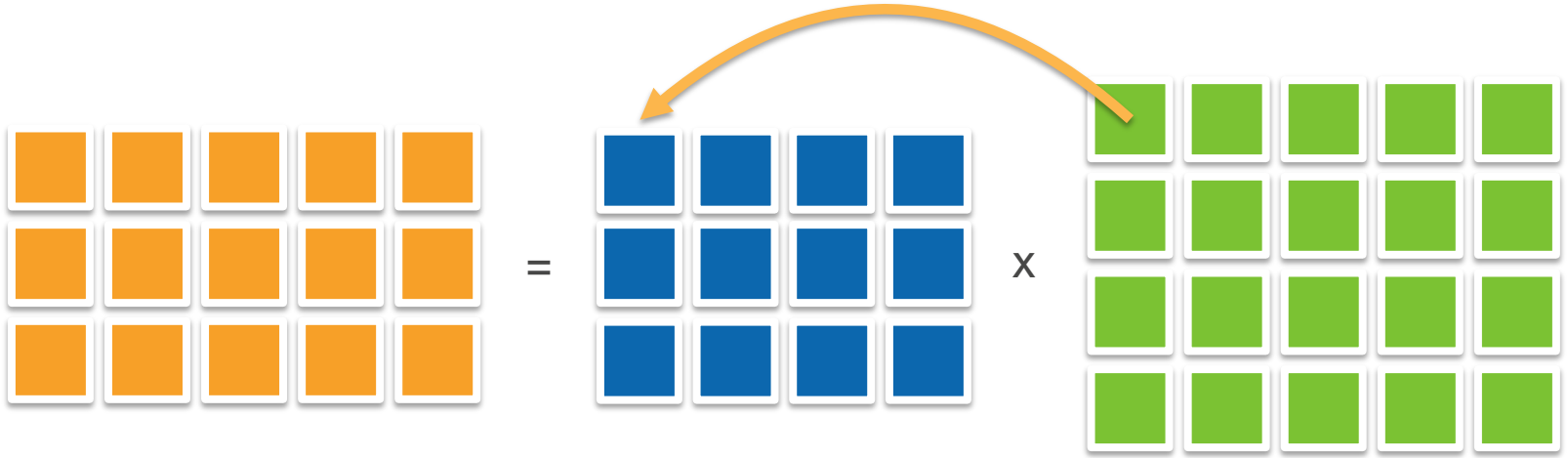
Distort the space.

Matrices

RELAXED SYSTEM LAB

- Multiplications (matrix-matrix) $\boldsymbol{C} = \boldsymbol{AB}, \boldsymbol{C} \in \mathbb{R}^{m \times p}, \boldsymbol{A} \in \mathbb{R}^{m \times n}, \boldsymbol{B} \in \mathbb{R}^{n \times p}$

$$\begin{bmatrix} C_{11} & \cdots & C_{1p} \\ \vdots & \ddots & \vdots \\ C_{m1} & \cdots & C_{mp} \end{bmatrix} = \boldsymbol{C} = \boldsymbol{AB} = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix} \begin{bmatrix} B_{11} & \cdots & A_{1p} \\ \vdots & \ddots & \vdots \\ B_{n1} & \cdots & A_{np} \end{bmatrix} \text{where } C_{ik} = \sum_{j=1}^{n} A_{ij} B_{jk}$$



Matrices

RELAXED SYSTEM LAB

- Some properties of matrix multiplication:
  - Non-commutative!

$$\boldsymbol{AB} \neq \boldsymbol{BA}$$

  - Associative:

$$(\boldsymbol{AB})\boldsymbol{C} = \boldsymbol{A}(\boldsymbol{BC}), \qquad \forall \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$$

$$\alpha(\boldsymbol{AB}) = (\alpha\boldsymbol{A})\boldsymbol{B}, \qquad \forall \boldsymbol{A}, \boldsymbol{B}$$

  - Distributive:

$$\boldsymbol{A}(\boldsymbol{B} + \boldsymbol{C}) = \boldsymbol{AB} + \boldsymbol{AB}, \qquad \forall \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$$

  - Transpose:

$$(\boldsymbol{AB})^T = \boldsymbol{B}^T \boldsymbol{A}^T$$

Matrix

RELAXED SYSTEM LAB

- Norm: $$\|\boldsymbol{A}\| = \sup\left\{\frac{\|\boldsymbol{Ax}\|_p}{\|\boldsymbol{x}\|_p}, \forall\, \boldsymbol{x} \neq 0\right\}$$
- Interpretation: how much can the mapping induced by $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ can stretch vectors?
- Choices depend on how to measure the length of vectors.

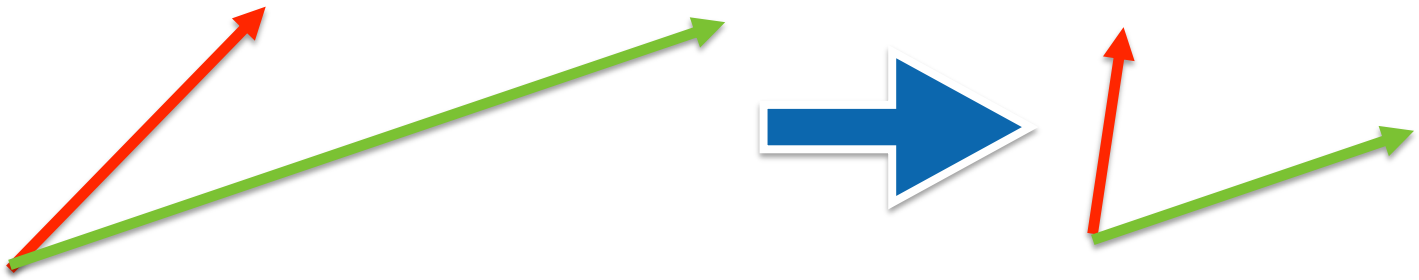
- Popular norms:
  - p=1, $\|\boldsymbol{A}\|_1 = \max\limits_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}|$
  - p=∞, $\|\boldsymbol{A}\|_\infty = \max\limits_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}|$
  - p=2, $\|\boldsymbol{A}\|_2 = \sigma_{max}(\boldsymbol{A})$ (the largest singular value of matrix.)
  - Frobenius norm: $\|\boldsymbol{A}\|_F = \left[\sum_{ij} A_{ij}^2\right]^{\frac{1}{2}}$

Matrix

RELAXED
SYSTEM LAB

- Eigenvectors and egienvalue
  - Vectors that are not changed by the matrix ($\boldsymbol{x}$ is the vector, λ is the eginvalue):

$$\boldsymbol{Ax} = \lambda \boldsymbol{x}$$



  - For symmetric matrices, we can always find the eigenvalue and eigenvector.

Special Matrices $\boldsymbol{A} \in \mathbb{R}^{n \times n}$

RELAXED
SYSTEM LAB

- Symmetric matrix: $\boldsymbol{A}^T = \boldsymbol{A}$

$$A_{ij} = A_{ji}$$

- Antisymmetric matrix: $\boldsymbol{A}^T = -\boldsymbol{A}$

$$A_{ij} = -A_{ji}$$

- Positive semi-definite:

$$\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \geq 0, \qquad \forall \boldsymbol{x}$$

Special Matrices

RELAXED
SYSTEM LAB

- Orthogonal Matrices:
  - All rows of the matrix are orthogonal to each other;
  - All rows of the matrix have unit length.
  - Rewrite in matrix form:

$$\sum_j U_{ij}U_{kj} = \delta_{ik}$$

$$\boldsymbol{UU^T = I}$$

- Permutation Matrices:
  - There is only one 1 in each row or column:

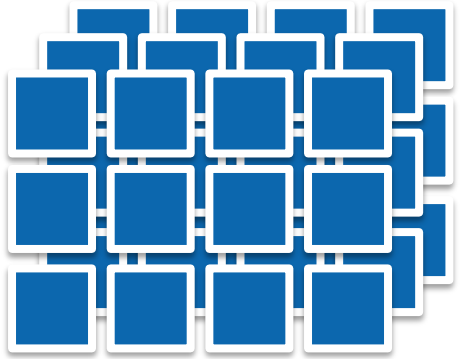
$$P_{ij} = \begin{cases} 1 \; if \; and \; only \; if \; j = \pi(i) \\ 0. \qquad\qquad\qquad otherwise \end{cases}$$

Tensor

RELAXED
SYSTEM LAB

- A tensor is a collection of numbers labelled by indices.

- The rank of a tensor is the number of indices required to specify an entry in the tensor:
  - A vector is a rank–1 tensor;
  - A matrix is a rank–2 tensor.



Tensor

RELAXED SYSTEM LAB

- Einstein summation convention:
  - Each index can appear at most twice in any term.
  - Repeated indices are implicitly summed over.
  - Each term must contain identical non-repeated indices.

$$M_{ij}v_j \equiv \sum_i M_{ij}v_j$$

Matrix multiplication between matrix M and vector v

$$M_{ij}u_jv_j$$

The index j appears three times in the first term.

$$T_{ijk}u_k + M_{ip}$$

The first term contains the non-repeated index j whereas the second term contains p.

RELAXED
SYSTEM LAB

Empirical Risk

Define the Empirical Risk

RELAXED
SYSTEM LAB

- Suppose we have:
  - a dataset $\mathcal{D} = \{(x_1, y_1), (x_1, y_2), \ldots, (x_N, y_N)\}$, where
  - $x_i \in \mathcal{X}$ is the input and
  - $y_i \in \mathcal{Y}$ is the output.
  - Let $h: \mathcal{X} \to \mathcal{Y}$ be a hypothesized model (mapping from input to output) we are trying to evaluate, which is parameterized by $w \in \mathbb{R}^d$.
  - Let $L: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ be a non-negative loss function which measures how different two outputs are
- The emprical risk R is defined as:

$$R(h_w) = \frac{1}{N} \sum_{i=1}^{N} L(h_w(x_i), y_i)$$

Common Loss Functions

RELAXED
SYSTEM LAB

- Mean squared error loss.
- L1 Loss.
- Negative log-likelihood loss.
- Cross entropy loss.
- KL divergence loss.

Computational Cost of the Emprical Risk

- The number of training examples N, the cost will be proportional to N.
- The cost to compute the loss function L.
- The cost to evaluate the hypothesis h_w.

Minimize the Emprical Risk

RELAXED
SYSTEM LAB

- Don't just want to calculate the empirical risk;
- Let $f\colon \mathbb{R}^d \to \mathbb{R}_+$ be the optimization object, which is formulated by the empirical risk;
- Let $\mathcal{D} = \{(x_1, y_1), (x_1, y_2), \dots, (x_N, y_N)\} = \{\xi_1, \xi_2, \dots, \xi_N\}$ be the training set;
- **The training computation is solving the following optimization problem:**

$$mimimize\colon\ R(h_w) = \frac{1}{N}\sum_{i=1}^{N} L(h_w(x_i), y_i) = f(w) = \frac{1}{N}\sum_{i=1}^{N} f(w; \xi_i)$$
$$over\ w \in \mathbb{R}^d$$

RELAXED
SYSTEM LAB

Gradient Descent

Gradient Descent Algorithm

RELAXED SYSTEM LAB

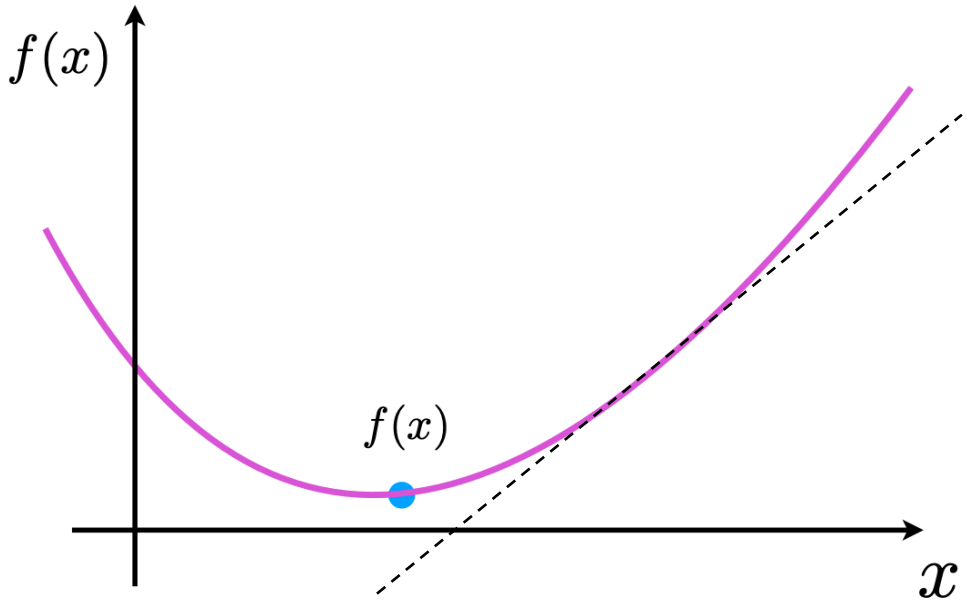
- Suppose we have:
  - w_0 denotes the value of the initalized parameter;
  - w_t denotes the value of the parameter at iteration t;
  - $\alpha_t \in \mathbb{R}$ denotes the learning rate at iteration t;
  - ∇f denotes the gradient (*vector of partial derivatives*) of function f.
- **The gradient decent algorithm is defined by**:

$$w_{t+1} = w_t - \alpha_t \cdot \nabla f(w_t) = w_t - \alpha_t \cdot \frac{1}{N} \sum_{i=1}^{N} \nabla f(w_t; \xi_i)$$

Definition of a Derivative

RELAXED
SYSTEM LAB

- First, suppose we have:
  - $f: \mathbb{R} \to \mathbb{R}$;
- Definition of a derivative:
  - $f'(x) = \frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$



Definition of a Derivative

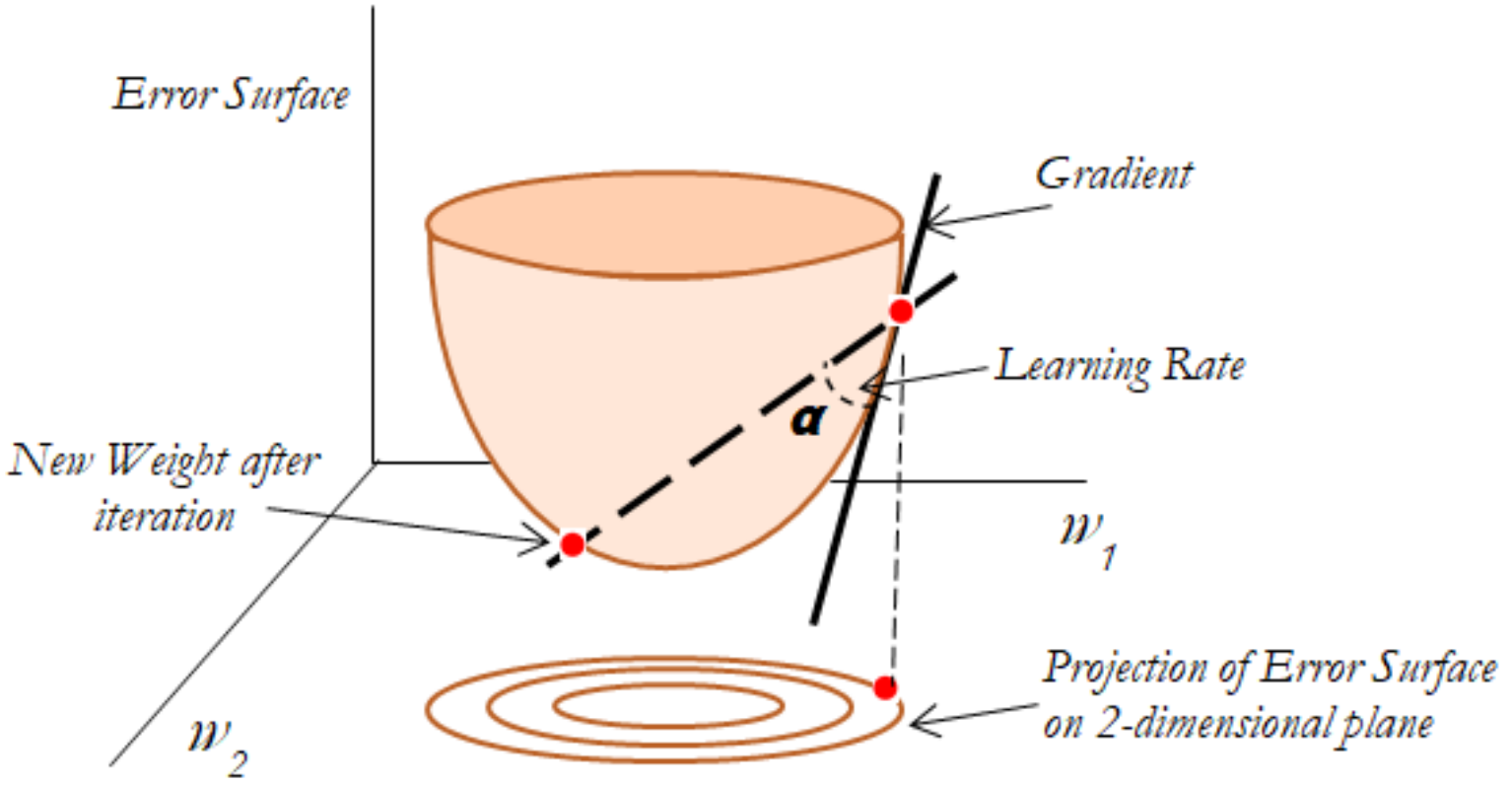
RELAXED
SYSTEM LAB

- Then, suppose we have:
  - $f: \mathbb{R}^d \rightarrow \mathbb{R}$;
- Definition of a derivative/gradient :
  - $\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \cdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix} \in \mathbb{R}^d$
- Where:
  - $\frac{\partial f}{\partial x_i} = \lim_{\epsilon \to 0} \frac{f(x_1, x_2, \ldots, x_i + \epsilon, x_{i+1}, \ldots, x_d) - f(x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_d)}{\epsilon} = \lim_{\epsilon \to 0} \frac{f(x + \epsilon e_i) - f(x)}{\epsilon}$

Why Does Gradient Descent Work?

RELAXED SYSTEM LAB

- **Intuition:**
  - If the learning rate is small enough and the value of the gradient is nonzero;
  - Gradient descent decreases the value of the objective at each iteration;
  - Eventually, gradient descent comes close to a point where the gradient is zero.



Stochastic Gradient Descent

Basic Idea

RELAXED SYSTEM LAB

- Calculating the gradient over the whole dataset is computationally expensive!
  - LLM pretraining corpus can include trillions of tokens!
- How to reduce this cost?
  - Replace the full gradient (which is a sum) with ***a single gradient example***.
  - iteratively by sampling a random example ξ_t uniformly from the training set and then updating the w_t.

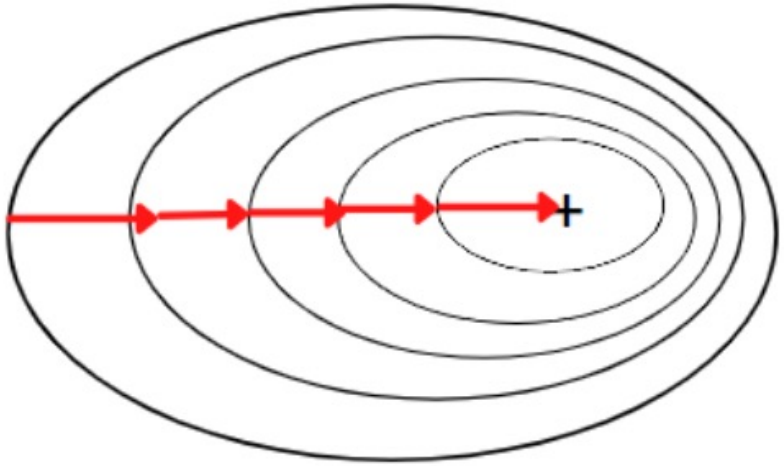
$$w_{t+1} = w_t - \alpha_t \cdot \nabla f(w_t)$$

$$w_{t+1} = w_t - \alpha_t \cdot \nabla f(w_t; \xi_t)$$

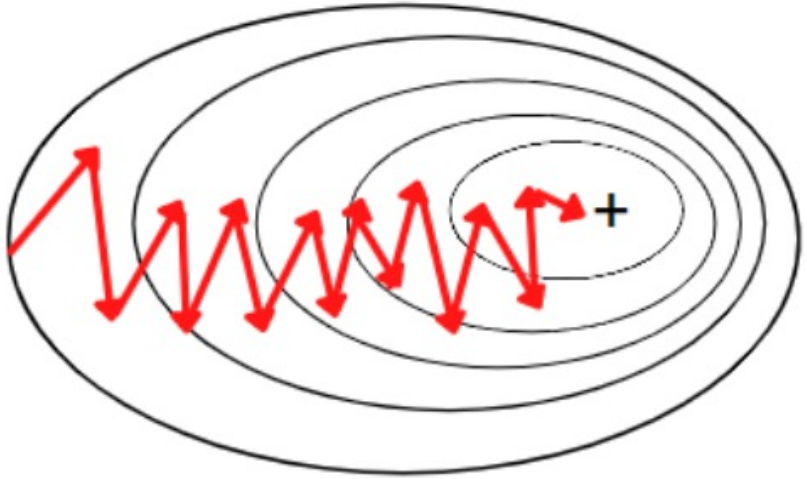
Stochastic Gradient Descent

RELAXED SYSTEM LAB

- Stochastic gradient descent won't necessarily decrease the total loss at every iteration!
  - But it runs much faster!
- Why is it fine to get an approximate solution for training?
  - In machine learning, generalization matters more than optimization.



Gradient Descent



Stochastic Gradient Descent

Mini-Batch Stochastic Gradient Descent

RELAXED SYSTEM LAB

- Basic ideas:
  - To reduce the variance of stochastic gradients;
  - Split the training data into smaller batches;
  - Sampling batch (usually without replacement)
- Suppose we have:
  - B is the batch size;
  - Replace the single gradient with a batch of gradients:

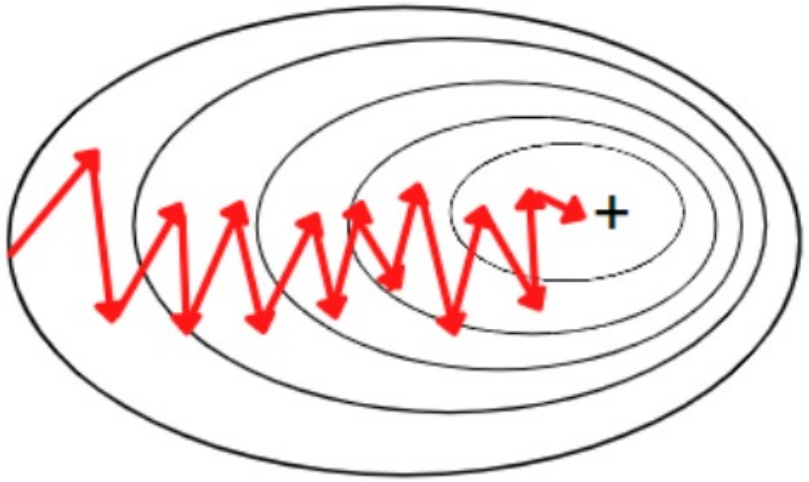
$$w_{t+1} = w_t - \alpha_t \cdot \nabla f(w_t; \xi_t)$$

$$w_{t+1} = w_t - \alpha_t \cdot \frac{1}{\mathrm{B}} \sum_{i=1}^{B} \nabla f(w_t; \xi_i)$$

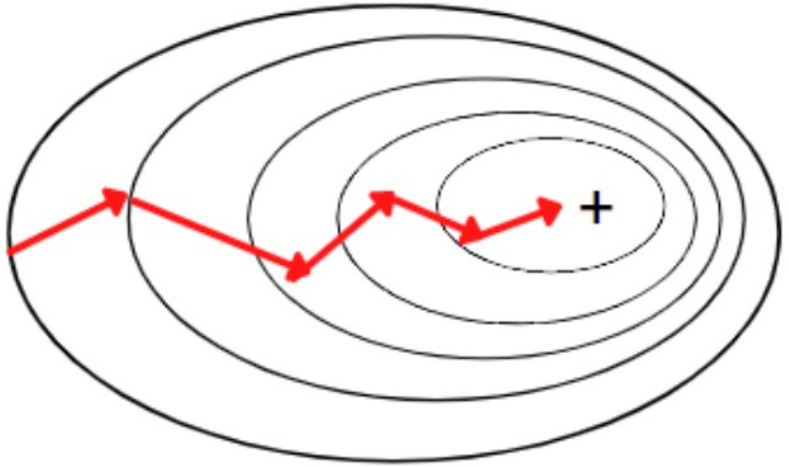
Mini-Batch Stochastic Gradient Descent

RELAXED SYSTEM LAB

- Mini-batch stochastic gradient descent reduces the variance of stochastic gradients!



Stochastic Gradient Descent



Mini-Batch Stochastic Gradient Descent

Acceleration of SGD 1: (Polyak's)Momentum

RELAXED SYSTEM LAB

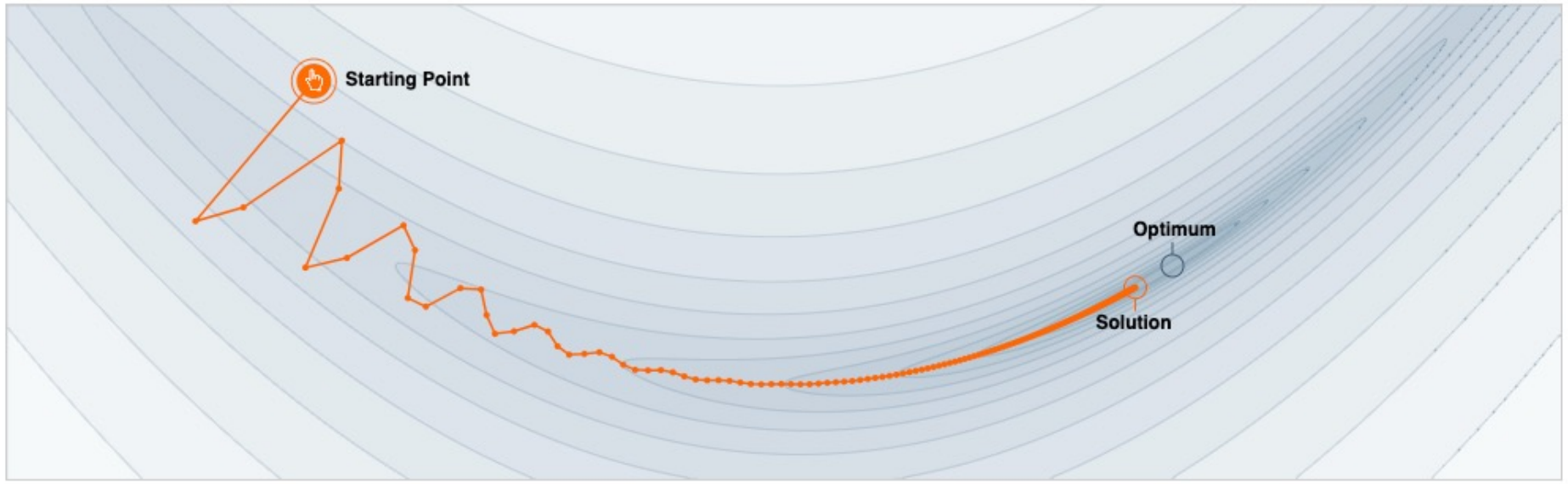
- Basic idea:
  - In SGD or mini-batch SGD the updates at each step is only based on current gradients, which can be unstable.
  - Momentum: exponentially weighted average of gradients.
  - The moving average method should be able to *denoise* the gradients computed at each step.
- Formal equation:

$$w_{t+1} = w_t - \alpha_t \cdot \sum_{i=1}^{B} \nabla f(w_t; \xi_i) + \beta(w_t - w_{t-1})$$

Standard gradient step Momentum step

Why Momentum Really Works?

RELAXED SYSTEM LAB



Step-size α = 0.02

0 0.003 0.006

Momentum β = 0.99

0.00 0.500 0.990

We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

GABRIEL GOH
UC Davis

April. 4
2017

Citation:
Goh, 2017

https://distill.pub/2017/momentum/

Acceleration of SGD 2: (Nesterov's) Momentum

RELAXED SYSTEM LAB

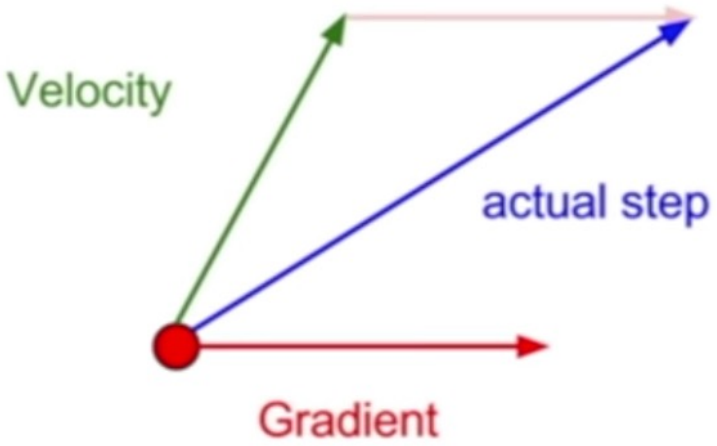
- Basic idea:
  - Polyak's momentum algorithm can fail to converge for some carefully built convex optimization problems.
  - Nesterov's Momentum: evaluates the gradient after applying momentum (at a point closer to the minimum point).
  - Works better for some cases in practice.
- Formal equation:

$$w_{t+1} = w_t - \alpha_t \cdot \sum_{i=1}^{B} \nabla f(w_t + \beta(w_t - w_{t-1}); \xi_i) + \beta(w_t - w_{t-1})$$

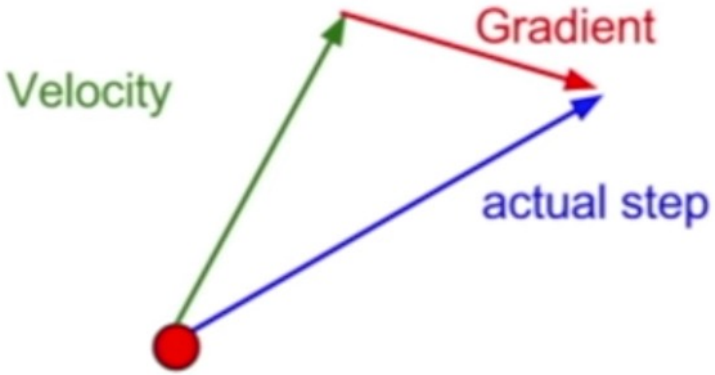
Estimate gradient after applying momentum — Momentum step

Polyak's Momentum vs. Nesterov's Momentum

RELAXED SYSTEM LAB



Polyak's Momentum



Nesterov's Momentum

(Approximated) Second Order Method

Definition of Second-Order Derivative

RELAXED SYSTEM LAB

- Suppose we have:
  - $f: \mathbb{R} \to \mathbb{R}$;
- Definition of a derivative:
  - $f'(x) = \frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$;
- Definition of second-order derivative:
  - $f''(x) = \frac{\partial^2 f}{\partial x^2} = \lim_{\epsilon \to 0} \frac{f'(x+\epsilon) - f'(x)}{\epsilon}$;
- Represent the **local curvature**: how the slope of the function changes.

Definition of Second-Order Derivative

RELAXED SYSTEM LAB

- Suppose we have:
  - $f: \mathbb{R}^d \to \mathbb{R}$;
- Definition of a gradient:
  - $\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \cdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix} \in \mathbb{R}^d$

- Second-order derivative **Hessian matrix**:

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_p} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_p \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1^2} \end{bmatrix} \in \mathbb{R}^{d \times d}$$

Netown's Method

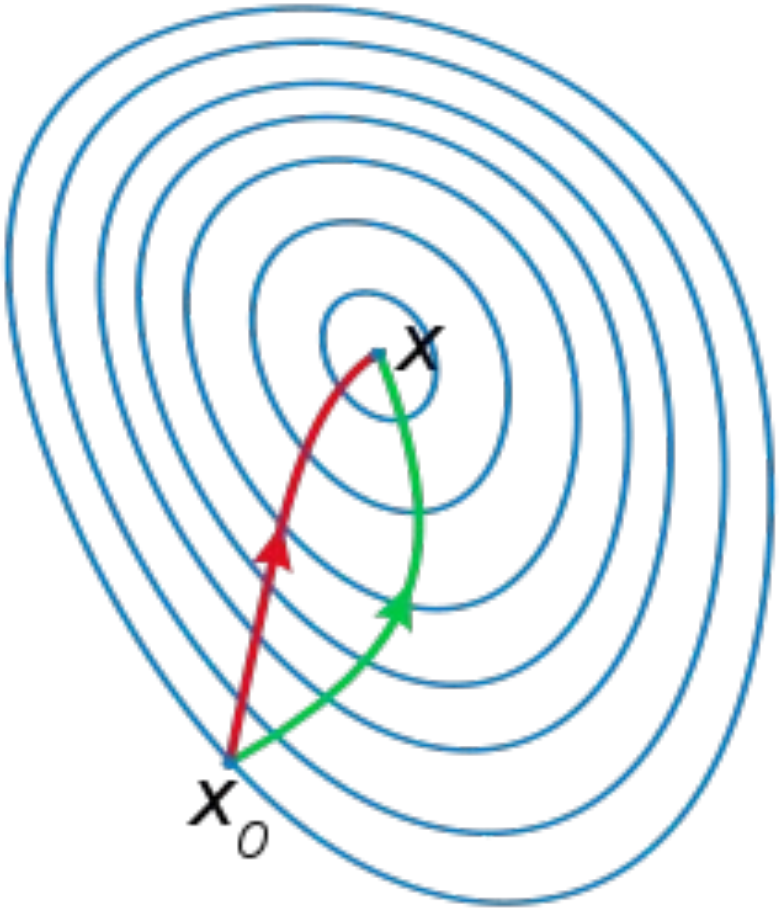
RELAXED
SYSTEM LAB

- Suppose we have:
  - w_0 denotes the value of the initalized parameter;
  - w_t denotes the value of the parameter at iteration t;
  - ∇f denotes the gradient of function f;
  - $\nabla^2 f$ denotes the Hessian matrix of function f.
- **The Newton's method is defined by**:

$$w_{t+1} = w_t - \left(\nabla^2 f(w_t)\right)^{-1} \nabla f(w_t)$$

Netwon's Method

- Benefits:
  - Superlinear (quadratic) convergence rate!
- Problem:
  - To compute the Hessian matrix is too computationally expensive!
  - Even storing the Hessian matrix is impossible for most ML models.



Gradient descent (green) v.s. Newton's method (red): Newton's method uses curvature information to take a more direct route.

Adaptive Moment Estimation (Adam)

RELAXED SYSTEM LAB

- Suppose we have:
  - w_0 denotes the value of the initalized parameter;
  - w_t denotes the value of the parameter at iteration t;
  - ∇f denotes the gradient of function f;
  - m_t denotes the first order moment;
  - v_t denotes the second order moment;
  - β_1, β_2 denotes two hyper-parameters;

- **Adam is defined by**:
  - $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla f(w_t)$
  - $v_t = \beta_2 v_{t-1} + (1-\beta_2)\left(\nabla f(w_t)\right)^2$
  - $\widehat{m_t} = \frac{m_t}{1-\beta_1^t}$
  - $\widehat{v_t} = \frac{v_t}{1-\beta_2^t}$
  - $w_{t+1} = w_t - \alpha_t \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}}+\epsilon}$

Further Reading (Optional)

RELAXED SYSTEM LAB

Optimization Methods for Large-Scale Machine Learning

Léon Bottou* Frank E. Curtis† Jorge Nocedal‡

June 16, 2016

Abstract

This paper provides a review and commentary on the past, present, and future of numerical optimization algorithms in the context of machine learning applications. Through case studies on text classification and the training of deep neural networks, we discuss how optimization problems arise in machine learning and what makes them challenging. A major theme of our study is that large-scale machine learning represents a distinctive setting in which the stochastic gradient (SG) method has traditionally played a central role while conventional gradient-based nonlinear optimization techniques typically falter. Based on this viewpoint, we present a comprehensive theory of a straightforward, yet versatile SG algorithm, discuss its practical behavior, and highlight opportunities for designing algorithms with improved performance. This leads to a discussion about the next generation of optimization methods for large-scale machine learning, including an investigation of two main streams of research on techniques that diminish noise in the stochastic directions and methods that make use of second-order derivative approximations.

https://arxiv.org/abs/1606.04838