Non-Research Tips for Information Science Researchers (Summer 2024)

Apr 17, 2024

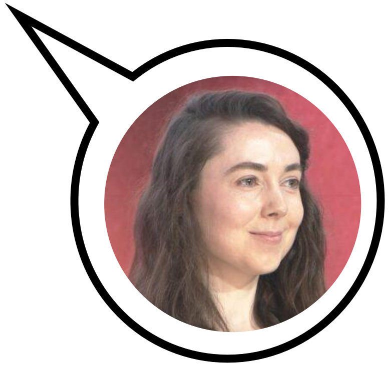
Week 2: Equations and pseudo-codes

https://non-research-tips.github.io/2024

Yusuke Matsui
(UTokyo)

Schedule

Date (2024)	Contents	Presented by
~~Week 1, Apr 10~~	~~Introduction. Review of fundamental concepts~~	~~Yusuke, Koya, Yuki, Jun~~
Week 2, Apr 17	Equations and pseudo-codes	Yusuke Matsui
Week 3, Apr 24	Presentation	Koya Narumi
Week 4, May 1	Tables and plots	Yusuke Matsui
Week 5, May 8	Figures	Koya Narumi
Week 6, May 22	Videos	Koya Narumi
Week 7, May 29	Invited Talk 1	Dr. Yoshiaki Bando (AIST)
Week 8, June 5	Invited Talk 2	Prof. Katie Seaborn (Tokyo Tech)
Week 9, June 12	GitHub in depth	Yusuke Matsui
Week 10, June 19	Automation of research and research dissemination (Web, Cloud, CI/CD)	Jun Kato
Week 11, June 26	Research community	Jun Kato
Week 12, July 3	3DCG illustrations	Yuki Koyama
Week 13, July 10	Final presentations	-



Author

The data x is...

It's obvious from the context what x is.

Author

The data x is...

Reviewer 2

It's obvious from the context what x is.

What is x? A scalar? A vector? Hard to read... Reject!

Author

Reviewer 2

The data x is...

It's obvious from the context what x is.

What is x? A scalar? A vector? Hard to read... Reject!

The data $\boldsymbol{x} \in \mathbb{R}^D$ is...

Let me explicitly define $\boldsymbol{x}$ as a D-dim vector.

Ok, $\boldsymbol{x}$ is a D-dim real-valued vector. This author uses a bold alphabet as a vector. Got it. Keep reading...

Author

Reviewer 2

The data x is...

Writing a clear, simple, and precise mathematical description is extremely important!

The data $\boldsymbol{x} \in \mathbb{R}^D$ is...

Let me explicitly define $\boldsymbol{x}$ as a D-dim vector.

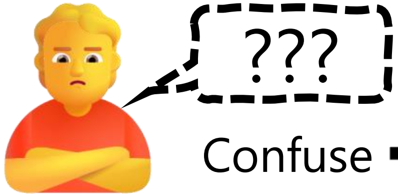
Ok, $\boldsymbol{x}$ is a D-dim real-valued vector. This author uses a bold alphabet as a vector. Got it. Keep reading...

Equations and pseudo-codes

Introduce the guideline of writing equations. Three important things:

1. Describe **precisely what you want to say** without making mistakes.

☹ Wrong equations ➡

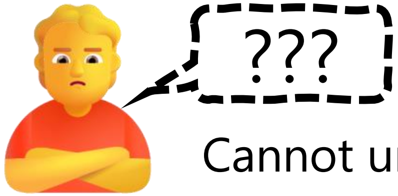


Confuse ➡ Reject!

☹ Wrong equations ➡ ☹☹ Wrong discussions

2. Be **sufficient**, not overly complex or ambiguous.

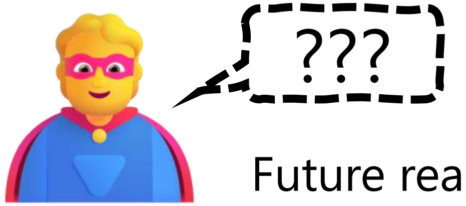
☹ Complex equations ➡



Cannot understand ➡ Reject!

3. Be understandable even if it is read **10 years later**.

☹ Description overfit to the current technology ➡



Future readers

Equations and pseudo-codes

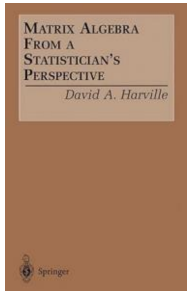
- Main target audience:
  - ✓ Researchers in CV, NLP, or related fields.

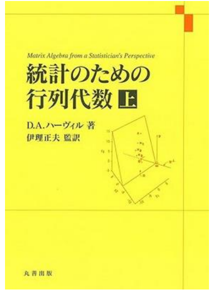
- Different fields have very different conventions.
  - ✓ If the content in this lecture differs from your field's convention, **please always follow your field's convention**.
  - ✓ e.g., I suggest using a bold font for vectors, but your field might never use bold.

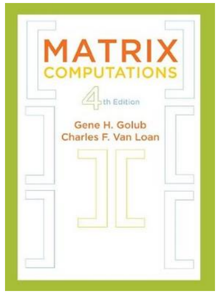
“OK” in US

“OK” in Japan

Reference







Dictionary for notation

- D. A. Harville, “Matrix Algebra From a Statistician's Perspective”, Springer, 2000.
- D. A. ハーヴィル，”統計のための行列代数上・下”，丸善出版，2012.
- G. H. Golub and C. F. Van Loan, “Matrix Computations, 4th edition”, Johns Hopkins University Press, 2012.

Very precise description for Transformer

- M. Phuong and M. Hutter, “Formal Algorithms for Transformers”, arXiv 2022. https://arxiv.org/abs/2207.09238

Higher-order tensor

- T. G. Kolda and B. W. Bader, “Tensor Decompositions and Applications”, SIAM Review, 2009.
- 横田達也，”テンソル分解の基礎と応用”，MIRU チュートリアル，2022.
  https://speakerdeck.com/yokotatsuya/tensorufen-jie-falseji-chu-toying-yong-miru2022tiyutoriaru

The original document for this lecture

- 松井勇佑，”工学系の卒論生のための数式記述入門”，GitHub, 2021. https://github.com/mti-lab/math_writing

工学系の卒論生のための数式記述入門

東京大学　情報理工学系研究科　講師　松井勇佑

初版 2021 年 1 月 7 日 / 最終更新 2023 年 4 月 26 日

- 本資料の GitHub リポジトリ：https://github.com/mti-lab/math_writing
- 著者のウェブページ：http://yusukematsui.me

1　はじめに

本資料は、初めて技術文章を書く学生向けの、数式記述の入門書です。自分のアイデアを式として記述する作業は難しく、最初のうちはうまく書けません。本資料は、そんな学生向けに、数式をどう記述すればいいかのガイドラインを示します。特に、よくありがちな、**変数がゴチャゴチャしてわかりにくく**なったり、**アルファベットを使いすぎて足りなくなったり**する状況の改善を目指します。

重要なことは次の三点です。

- 伝えたいことを間違えることなく正確に記述すること

しょう。

- 東京大学松尾先生による記事：読み物としても面白い、松尾先生による論文執筆指南です。短い文章で重要な事項が凝縮されており、必読です。
  - 松尾ぐみの論文の書き方
  - 松尾ぐみの論文の書き方：英語論文

次に、数式の表記（notation）に関して参考になる資料を紹介します。本記事では、これらを参考に、松井の考える notation のベストプラクティスを紹介します。

- D. A. ハーヴィル、統計のための行列代数　上下：これは線形代数の辞書です。表記はもとより、導出がわからない式に出会った場合に参考になります。
- Golub ら、Matrix Computations：これは実際に計算するこ

I’d like to translate it into English, but I have no time... PR welcome!

Equations

- Basic notation for variables
- String
- Tips for sets
- Inputs/outputs of functions
- Subscript/superscript
- Don’t write numpy
- Misc

Pseudo codes

- Basic
- What do you want to express?
- Misc

Equations

- Basic notation for variables
- String
- Tips for sets
- Inputs/outputs of functions
- Subscript/superscript
- Don’t write numpy
- Misc

Pseudo codes

- Basic
- What do you want to express?
- Misc

Basic notation for variables

- Having a good notation for variables is extremely important.

- The most important rule: **to be consistent**
  - ✓ Bold for vectors (recommended):
    - ☐ This is $\boldsymbol{x} \in \mathbb{R}^3$… Here, $\boldsymbol{y} \in \mathbb{R}^3$…
  - ✓ Non-bold for vectors (ok):
    - ☐ This is $x \in \mathbb{R}^3$… Here, $y \in \mathbb{R}^3$…
  - ✓ Mixed (**REALLY BAD!** 😵😵😵😵):
    - ☐ This is $\boldsymbol{x} \in \mathbb{R}^3$… Here, $y \in \mathbb{R}^3$…

- If you decide your way, keep following the way **throughout the paper.**

Basic notation for variables

➢ For all variables, explicitly write a **domain:**

¥in: "an element of"

domain: a set, e.g., $\mathbb{R}$

$$x \underbrace{\in \mathbb{R}}$$

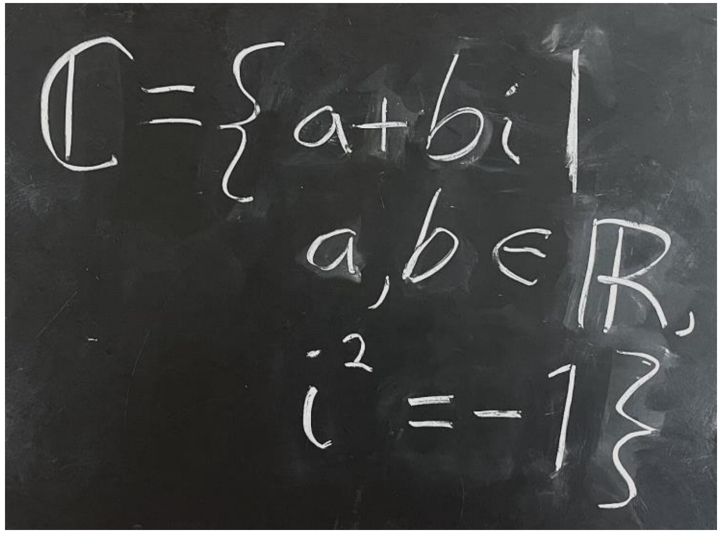
x is an element of real numbers
→ x is a real number

➢ Including a domain is crucial; much easier to understand.
➢ If you think you can skip a domain, you're likely incorrect by 90%.
➢ Writing a domain only requires a bit more space. **No side effects**.

➢ An exceptional case is when you hide a domain **intentionally.**
 ✓ Avoid unnecessarily complex descriptions. (I'll explain later)
 ✓ e.g., a feature volume for time series $\boldsymbol{V} \in \mathbb{R}^{H \times W \times T}$ 🤔 T depends on $\boldsymbol{V}$...

First things first: real numbers, natural numbers, etc

- **Blackboard bold** is used for special symbols such as real numbers.
  - ✓ TeX: `¥mathbb`
  - ✓ Powerpoint: e.g., `¥doubleR`



https://en.wikipedia.org/wiki/Blackboard_bold

- Examples:
  - ✓ $\mathbb{R}$: Real numbers
  - ✓ $\mathbb{N}$: Natural numbers
  - ✓ $\mathbb{Z}$: Integers

- Blackboard bold is typically used only in traditional contexts.
  - ✓ 😵 Don't recommend using it for a vector: $\mathbb{x} \in \mathbb{R}^3$ (although we do so when writing a vector by hand)

$$\boxed{\mathbb{y} = A\mathbb{x} + \mathbb{b}}$$

Summary

Type	How to write	Example of domain	Example of values
Scalar	Lowercase or uppercase	$a \in \mathbb{R}$. $b \in \mathbb{N}$. $K \in \{10, 20, 30\}$.	$a = 3.2$. $b = 13$. $K = 20$.
Vector	Lowercase and bold ➢ TeX: `¥mathbf` or `¥bm` ➢ Powerpoint: bold	$\boldsymbol{x} \in \mathbb{R}^3$. $\boldsymbol{b} \in \{0, 1\}^B$.	$\boldsymbol{x} = [0.1, 0.2, 0.3]^\top$. $\boldsymbol{b} = [0, 1, 1, 0]^\top$.
Matrix and Tensor	Uppercase and bold ➢ TeX: `¥mathbf` or `¥bm` ➢ Powerpoint: bold	$\boldsymbol{A} \in \mathbb{R}^{2\times 3}$. $\boldsymbol{I} \in [0, 1]^{H\times W\times 3}$.	$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 4 \end{bmatrix}$.
Set	Calligraphy font in uppercase ➢ TeX: `¥mathcal` ➢ Powerpoint: e.g., `¥scriptS`	$\mathcal{S} \subset \mathbb{N}$.	$\mathcal{S} = \{2, 4, 8, 16\}$.

Scalar

- <u>Recommend</u>: Lowercase or uppercase (e.g., $a \in \mathbb{R}$, $K \in \mathbb{N}$, $\mu \in \mathbb{C}$)
- If you can specify the range tightly, it's more informative.

Example	Meaning	
$a \in [2, 7]$	$2 \leq a \leq 7$	More informative than $a \in \mathbb{R}$
$a \in (2, 7)$	$2 < a < 7$	
$a \in [2, 7)$	$2 \leq a < 7$	
$a \in \{2, 7\}$	$a = 2$ or $a = 7$	
$a \in \{2, \dots, 7\}$	If naturally interpreted, $a = 2$ or $a = 3$ or … or $a = 7$.	

$$[2, 7)$$

Range (remember your high-school math!)

All brackets are different and have various meanings

( ) : Parentheses

[ ] : Square brackets

{ } : Curly brackets

Vector

- Recommend: Lowercase and bold, e.g.,
  - ✓ $\boldsymbol{x} \in \mathbb{R}^3$ 3-dimensional real-valued vector
  - ✓ $\boldsymbol{b} \in \{0, 1\}^B$ B-dimensional binary vector

- How to write
  - ✓ TeX: `¥mathbf` or `¥bm`
  - ✓ Powerpoint: set bold

- `¥mathbf` or `¥bm`?
  - ✓ Depends on the TeX style.
  - ✓ Render both, compare them, and select the best one.

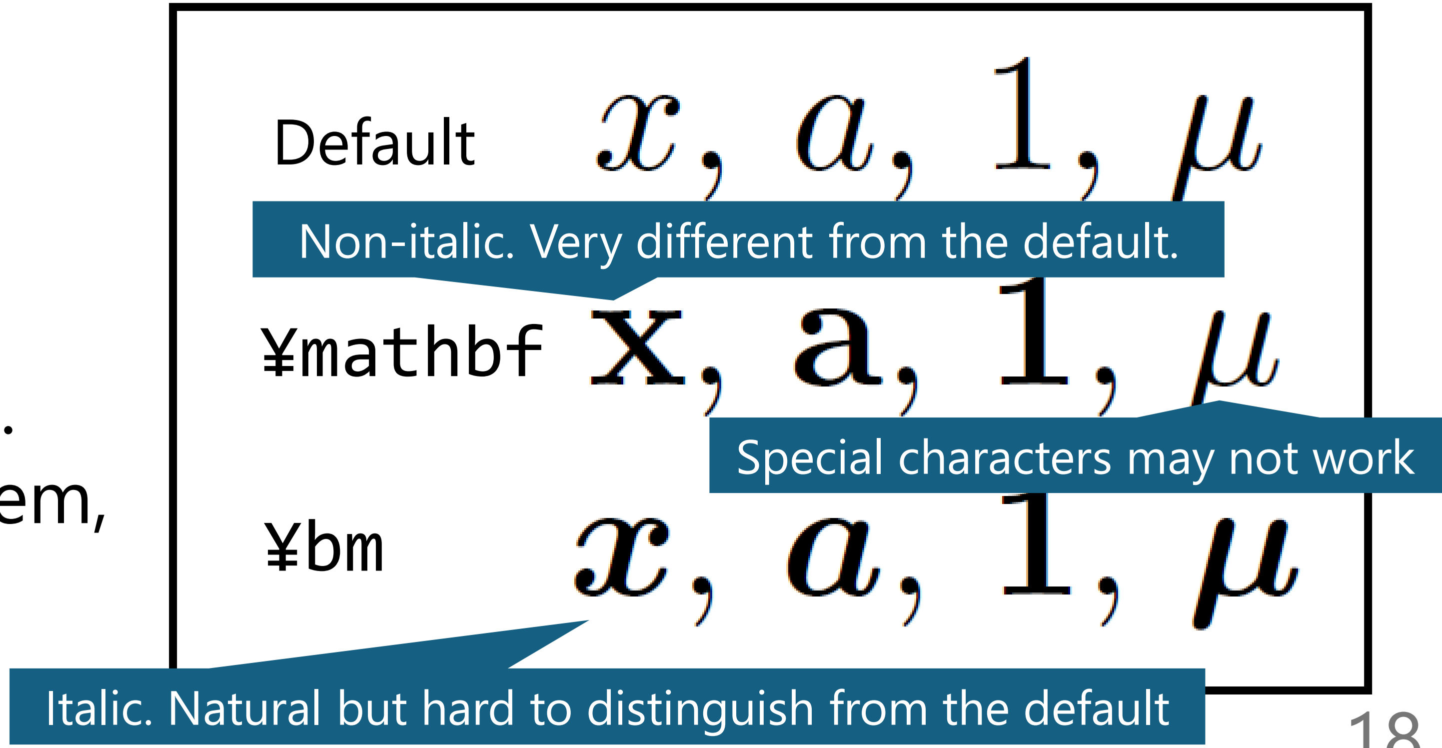
Default	$x,\ a,\ 1,\ \mu$
`¥mathbf`	$\mathbf{x},\ \mathbf{a},\ \mathbf{1},\ \mu$
`¥bm`	$\boldsymbol{x},\ \boldsymbol{a},\ \boldsymbol{1},\ \boldsymbol{\mu}$

Vector

- Recommend: Lowercase and bold, e.g.,
  - ✓ $\boldsymbol{x} \in \mathbb{R}^3$ 3-dimensional real-valued vector
  - ✓ $\boldsymbol{b} \in \{0,1\}^B$ B-dimensional binary vector

- How to write
  - ✓ TeX: `¥mathbf` or `¥bm`
  - ✓ Powerpoint: set bold

- `¥mathbf` or `¥bm`?
  - ✓ Depends on the TeX style.
  - ✓ Render both, compare them, and select the best one.



Vector

- Row-vector or column-vector?
  - ✓ <u>Recommend</u>: If you write a vector (e.g., $\boldsymbol{x} \in \mathbb{R}^3$), assume that all vectors are column-vectors.

$$\boldsymbol{x} = [1, 2, 3]^\top = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$ **Column-vector**. OK.

$$\boldsymbol{y} = [1, 2, 3]$$ **Row-vector**. Only write this if you explicitly need it.

- Why should we be careful?
  - ✓ Need to be careful when performing matrix operations:

Given $\boldsymbol{A} \in \mathbb{R}^{3\times3}$, $\begin{cases} \boldsymbol{Ax} \text{ can be computed} \\ \boldsymbol{Ay} \text{ can}\textbf{not} \text{ be computed } (\boldsymbol{Ay}^\top \text{ can be computed}) \end{cases}$

Vector

- More strictly, just writing "$\mathbb{R}^D$" cannot decide it's colum or row.
  - ✓ Explicitly saying D-dim column-vectors: $\boldsymbol{a} \in \mathbb{R}^{D\times 1}$
  - ✓ Explicitly saying D-dim row-vectors: $\boldsymbol{b} \in \mathbb{R}^{1\times D}$

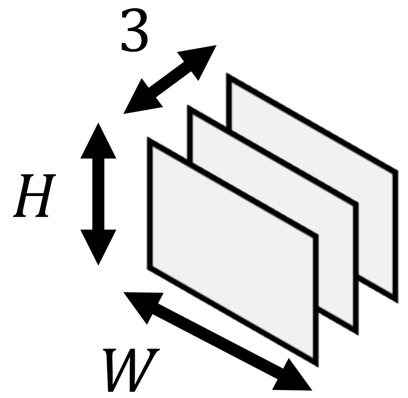
Recommend this way if you need a row vector

- <u>Recommend</u>: Decide in your mind that $\boldsymbol{c} \in \mathbb{R}^D$ is an abbreviated notation of $\boldsymbol{c} \in \mathbb{R}^{D\times 1}$

- Important rule: Again, make it to be **consistent** throughout the paper.
  - ✓ Recommend: "column-vector $\boldsymbol{x} \in \mathbb{R}^3$ and column-vector $\boldsymbol{y} \in \mathbb{R}^3$"
  - ✓ Not recommend but OK: "row-vector $\boldsymbol{x} \in \mathbb{R}^3$ and row-vector $\boldsymbol{y} \in \mathbb{R}^3$"
  - ✓ 😵 NO!!!: "column-vector $\boldsymbol{x} \in \mathbb{R}^3$ and row-vector $\boldsymbol{y} \in \mathbb{R}^3$

Same description but sometimes column, sometimes row??? Confusing. Reject!

Matrix and tensor



- <u>Recommend</u>: Uppercase and bold, e.g.,
  - ✓ $\boldsymbol{A} \in \mathbb{R}^{2\times 3}$ 2x3 real-valued matrix
  - ✓ $\boldsymbol{I} \in [0,1]^{H\times W\times 3}$ HxWx3 3rd-order tensor. Each element is in $[0,1]$

- Bold or not?
  - ✓ Non-bold may be more usual.
  - ✓ But a non-bold uppercase alphabet can be misinterpreted as a scalar (e.g., K is a matrix? Scalar?). Thus, I prefer bold for a matrix.

- Higher-order tensor is often used in CV as an image is 3rd-order tensor.
  - ✓ For higher-order tensors, one may use `¥mathsf` or bold `¥mathcal`

$$\mathsf{A} \in \mathbb{R}^{2\times 3\times 4}$$

$$\boldsymbol{\mathcal{A}} \in \mathbb{R}^{2\times 3\times 4}$$

Matrix and tensor

Example	Meaning
$a \in \mathbb{R}$	Real-valued scalar
$\boldsymbol{a} \in \mathbb{R}^2$	Two-dimensional real-valued vector
$\boldsymbol{A} \in \mathbb{R}^{2\times3}$	2x3 real-valued matrix
$\boldsymbol{B} \in [0,1]^{2\times3}$	2x3 matrix, where each element is in $[0,1]$
$\boldsymbol{C} \in [0,1)^{2\times3\times4}$	2x3x4 tensor, where each element is in $[0,1)$
$\boldsymbol{D} \in \{-1,0,1\}^{2\times3}$	2x3 matrix, where each element is either -1 or 0 or 1
$\boldsymbol{E} \in \{0,\dots,100\}^{2\times3}$	2x3 matrix, where each element is either 0 or … or 100
$\boldsymbol{f} \in \mathbb{N}^{1\times ab}$	(ab)-dimensional row vector of natural numbers
$\boldsymbol{F} \in \mathbb{N}^{a\times b}$	axb matrix of natural numbers

Reshape

Matrix and tensor

- Accessing an element of a vector: Given $\boldsymbol{a} \in \mathbb{R}^3$,
  - ✓ i-th element: $a_i \in \mathbb{R},\ a[i] \in \mathbb{R},\ a(i) \in \mathbb{R}$
  - ✓ Natural choice: a_i
  - ✓ Don't write: $\boldsymbol{a}_i$ (this implies an i-th vector from $\{\boldsymbol{a}_n\}_{n=1}^{10}$)

Several styles

- Accessing an element of a matrix: Given $\boldsymbol{A} \in \mathbb{R}^{3\times 2}$,
  - ✓ (i,j)-th element: $A_{ij} \in \mathbb{R},\ a_{ij} \in \mathbb{R},\ A[i,j] \in \mathbb{R},\ A(i,j) \in \mathbb{R}$
  - ✓ i-th row: $\boldsymbol{A}_{i:} \in \mathbb{R}^{1\times 2},\ \boldsymbol{A}[i,:] \in \mathbb{R}^{1\times 2},\ \boldsymbol{A}(i,:) \in \mathbb{R}^{1\times 2}$
  - ✓ j-th column: $\boldsymbol{A}_{:j} \in \mathbb{R}^3,\ \boldsymbol{A}[:,j] \in \mathbb{R}^3,\ \boldsymbol{A}(:,j) \in \mathbb{R}^3$

Several styles

One may not use bold

Matrix and tensor

- Parenthesis or square brackets?
  - ✓ Both are fine.

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

- Personally, I prefer square brackets as parenthesis have more meanings.

- i.e., parenthesis for vector/matrix may conflict with other usages, e.g.,
  - ✓ Usual sentence: “We think XXX (Note that $\boldsymbol{x} = (1, 2)$)”
  - ✓ Binomial coefficient: $\binom{2}{3}$ Cannot distinguish with $\begin{pmatrix} 2 \\ 3 \end{pmatrix} \in \mathbb{R}^2$
  - ✓ Element-based representation of a matrix: $\boldsymbol{A} = (a_{ij})$
  - ✓ Function: $f((1, 2))$
  - ✓ Tuple: $(1, 2, 3)$ Parenthesis is used for a tuple.
  - ✓ etc…

Set

- <u>Recommend</u>: Calligraphy font in uppercase, e.g.,
  - ✓ $\mathcal{S} = \{2, 13, 5, 7\} \subset \mathbb{N}$ A set of natural numbers.
  - ✓ $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\} \subset \mathbb{R}^D$ A set of N D-dimensional vectors.
    - Can be written as $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^N$.   Here, $\boldsymbol{x}_n \in \mathbb{R}^D$.

- How to write
  - ✓ TeX: `¥mathcal`
  - ✓ Powerpoint: e.g., `¥scriptS`

- Cardinality (the number of elements): $|\mathcal{S}| = 4$

- A set doesn't contain duplicate elements.
  - ✓ If does, it's called multi-set (or bag): $\mathcal{A} = \{a, a, b\}$.

Set

- Domain??
  - 😔 It's a bit tough to show a domain of a set.

Powerset: Given a set $\mathcal{A}$, a powerset of $\mathcal{A}$ is defined as all subsets of $\mathcal{A}$. E.g.,
- $\mathcal{A} = \{1, 3, 5\}$, then
- $2^{\mathcal{A}} = \{\phi, \{1\}, \{3\}, \{5\}, \{1, 3\}, \{3, 5\}, \{5, 1\}, \{1, 3, 5\}\}$

  - Consider a set $\mathcal{S} = \{5, 3\} \subset \mathcal{A}$, the domain of a set $\mathcal{S}$ is:
    - $\mathcal{S} \in 2^{\mathcal{A}}$
  - In the same way, if a set $\mathcal{B}$ is a subset of $\mathcal{X}$, $\mathcal{B}$'s domain is $2^{\mathcal{X}}$
    - $\mathcal{B} \subset \mathbb{R}$, then $\mathcal{B} \in 2^{\mathbb{R}}$

Usually, "$\mathcal{B} \subset \mathbb{R}$" style is easier to read.

- The powerset may helps when defining a function with a set-input $f(\mathcal{B})$:
  - $f: 2^{\mathbb{R}} \to \mathbb{R}$

Summary (again)

Type	How to write	Example of domain	Example of values
Scalar	Lowercase or uppercase	$a \in \mathbb{R}$. $b \in \mathbb{N}$. $K \in \{10, 20, 30\}$.	$a = 3.2$. $b = 13$. $K = 20$.
Vector	Lowercase and bold ➢ TeX: `¥mathbf` or `¥bm` ➢ Powerpoint: bold	$\boldsymbol{x} \in \mathbb{R}^3$. $\boldsymbol{b} \in \{0, 1\}^B$.	$\boldsymbol{x} = [0.1, 0.2, 0.3]^\top$. $\boldsymbol{b} = [0, 1, 1, 0]^\top$.
Matrix and Tensor	Uppercase and bold ➢ TeX: `¥mathbf` or `¥bm` ➢ Powerpoint: bold	$\boldsymbol{A} \in \mathbb{R}^{2\times3}$. $\boldsymbol{I} \in [0, 1]^{H\times W\times3}$.	$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 4 \end{bmatrix}$.
Set	Calligraphy font in uppercase ➢ TeX: `¥mathcal` ➢ Powerpoint: e.g., `¥scriptS`	$\mathcal{S} \subset \mathbb{N}$.	$\mathcal{S} = \{2, 4, 8, 16\}$.

Equations

- Basic notation for variables
- String
- Tips for sets
- Inputs/outputs of functions
- Subscript/superscript
- Don’t write numpy
- Misc

Pseudo codes

- Basic
- What do you want to express?
- Misc

String 🤔 It's not easy to use strings in equations

- If you use strings for labels, the easiest way is to define them directly.
  - ✓ A label set $\mathcal{L} = \{\text{“dog”}, \text{“cat”}, \text{“horse”}\}$
  - ✓ Image classifier $f: \mathbb{R}^{H\times W\times 3} \to \mathcal{L}$

If you don't run more complex operations on them, this should suffice.

- Alternatively, you can assign an integer for each label.
  - ✓ A label set $\mathcal{L} = \{1, \dots, N\}$
  - ✓ Image classifier $f: \mathbb{R}^{H\times W\times 3} \to \mathcal{L}$

Label	ID	One-hot
“dog”	1	$[1, 0, 0]^\top$
“cat”	2	$[0, 1, 0]^\top$
“horse”	3	$[0, 0, 1]^\top$

- Alternatively, you can use one-hot vectors.
  - ✓ A label set $\mathcal{L} = \{\boldsymbol{b}_n\}_{n=1}^{N},\ \boldsymbol{b}_n \in \{0, 1\}^N$
  - ✓ Image classifier $f: \mathbb{R}^{H\times W\times 3} \to \mathcal{L}$
  - ✓ Easy to integrate in other equations, e.g., $\|f(\boldsymbol{X}) - \boldsymbol{y}\|_2^2$ for some $\boldsymbol{y}$

- The description $\boldsymbol{b}_n \in \{0,1\}^N$ is not tight; could be any N-dim binary vectors.
- If you want to emphasize $\boldsymbol{b}_n$ is one-hot, you can write:
  - ✓ $\boldsymbol{b}_n \in \mathcal{H}_{1/N}$
  - ✓ $\mathcal{H}_{1/N} = \{\boldsymbol{b} \in \{0,1\}^N \mid ||\boldsymbol{b}|| = 1\}$
  - ✓ Here, $\mathcal{H}_{1/N}$ is a set of all 1-of-N binary encoding vectors [1]

[1] M. Norouzi and D. J. Fleet, "Cartesian k-means", CVPR 2013

Label	ID	One-hot
"dog"	1	$[1,0,0]^\top$
"cat"	2	$[0,1,0]^\top$
"horse"	3	$[0,0,1]^\top$

- Alternatively, you can use one-hot vectors
  - ✓ A label set $\mathcal{L} = \{\boldsymbol{b}_n\}_{n=1}^N,\ \boldsymbol{b}_n \in \{0,1\}^N$
  - ✓ Image classifier $f: \mathbb{R}^{H\times W\times 3} \to \mathcal{L}$
  - ✓ Easy to integrate in other equations, e.g., $||f(\boldsymbol{X}) - \boldsymbol{y}||_2^2$ for some $\boldsymbol{y}$

String 🆎 A set of character sequences $\mathcal{V}^*$

- If you need more detailed definition: Kleene closure for characters.
  - ✓ Define a vocabulary $\mathcal{V} = \{$“a”, “b”, …, “z”$\}$.
  - ✓ Here, $\mathcal{V}^2 = \mathcal{V} \times \mathcal{V},\ \mathcal{V}^3 = \mathcal{V} \times \mathcal{V} \times \mathcal{V}, \ldots$ i.e., “a” $\in \mathcal{V}$ and “dog”$\in \mathcal{V}^3$
  - ✓ Define **a set of character sequences** $\mathcal{V}^* = \{\emptyset\} \cup \mathcal{V} \cup \mathcal{V}^2 \cup \cdots$
  - ✓ Any string (character sequence) is in $\mathcal{V}^*$: “a”, “dog”, “hoge” $\in \mathcal{V}^*$

- A label set $\mathcal{L} = \{$“dog”, “cat”, “horse”$\} \subset \mathcal{V}^*$

- Straightforward. But not easy to connect with other equations.

Acknowledgment: I thank Hiroyuki Deguchi for the helpful discussion

String 🔢 A set of token sequences $\mathcal{V}^*$

char	token
"a"	1
"b"	2
…	
"z"	N

N: vocabulary size

- If you need more math-friendly representations: Kleene closure for tokens.
  - ✓ Define a vocabulary $\mathcal{V} = \{1, \dots, N\}$.
  - ✓ Here, $\mathcal{V}^2 = \mathcal{V} \times \mathcal{V},\ \mathcal{V}^3 = \mathcal{V} \times \mathcal{V} \times \mathcal{V}, \dots$
  - ✓ i.e., $3 \in \mathcal{V}$ and $[13, 6, 20] \in \mathcal{V}^3$
  - ✓ Define **a set of token sequences** $\mathcal{V}^* = \{\emptyset\} \cup \mathcal{V} \cup \mathcal{V}^2 \cup \cdots$
  - ✓ Any token sequence (variable length integer-vectors) is in $\mathcal{V}^*$
  - ✓ $3, [13, 6, 20], [6, 18, 16, 7] \in \mathcal{V}^*$

- Now it's just a vector.
- Intentionally use "row-vector" style

- This is an ASCII representation for the c-language.

- [2] uses this notation.

[2] M. Phuong and M. Hutter, "Formal Algorithms for Transformers", arXiv 2022. https://arxiv.org/abs/2207.09238

String 🔢 A set of <u>token</u> sequences

v	$D(v)$
1	"a"
2	"b"
	…
N	"z"

- Decoder
  - ✓ Given $v \in \mathcal{V}$, decoder $D(v)$ returns the original character.
  - ✓ e.g., $D(7) =$ "g".
  - ✓ Extends this to $\boldsymbol{w} \in \mathcal{V}^*$. e.g., $D([4, 15, 7]) =$ "dog".

- Vector-style operations
  - ✓ e.g., $\boldsymbol{w} = [6, 18, 16, 7] \in \mathcal{V}^*$, then $D(\boldsymbol{w}) =$ "frog"
  - ✓ Specify a character: $w_2 = 18$, and $D(w_2) =$ "r"
  - ✓ Slicing: $\boldsymbol{w}_{2:4} = [18, 16, 7]$, and $D(\boldsymbol{w}_{2:4}) =$ "rog"

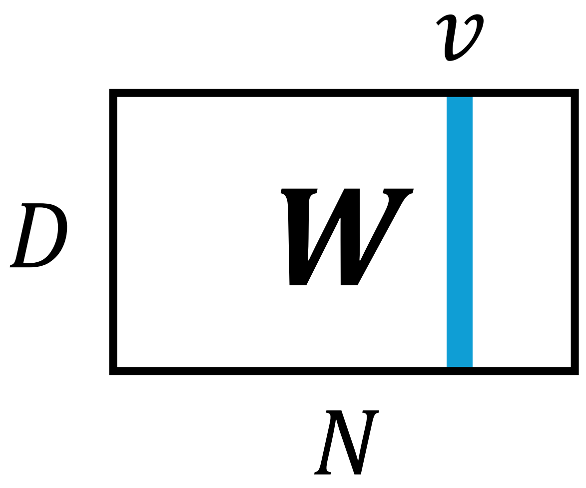
- Difference to a superset? $2^{\mathcal{V}}$ vs $\mathcal{V}^*$
  - ✓ An element of a super set is a set (not a vector). No duplicate.
  - ✓ $\{3, 4\} \in 2^{\mathcal{V}}$ Cannot have duplicates $[3, 4, 4] \in \mathcal{V}^*$ Can have duplicates

String 🔢 A set of token sequences

v	$D(v)$
1	“a”
2	“b”
	…
N	“z”

- Token embedding
  - ✓ Embedding matrix $\boldsymbol{W} \in \mathbb{R}^{D \times N}$

    D: dim of embedding
  - ✓ The vector representation (embedding) of $v \in \mathcal{V}$ is $\boldsymbol{W}[:, v] \in \mathbb{R}^D$



- Can be a tuple?
  - ✓ Yes. You can define so (I’ll discuss later)
  - ✓ $(1, 3, 5) \in \mathcal{V}^*$

Equations

- Basic notation for variables
- String
- Tips for sets
- Inputs/outputs of functions
- Subscript/superscript
- Don’t write numpy
- Misc

Pseudo codes

- Basic
- What do you want to express?
- Misc

Tips for sets

- How to describe $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ in short?
  - ✓ 😐 OK, but a bit long? $\{\boldsymbol{x}_i \mid i = 1, 2, \dots, N\}$
  - ✓ 😐 OK, but a bit long? $\{\boldsymbol{x}_i \mid 1 \leq i \leq N\}$
  - ✓ 😐 OK, but a bit long? $\left\{\boldsymbol{x}_i \mid i \in \{1, 2, \dots, N\}\right\}$
  - ✓ 🙂 Good! Short! $\{\boldsymbol{x}_i\}_{i=1}^{N}$

This may seem obvious? But many junior students struggle with this.

- How to do when I don't want to use an alphabet for #set?
  - ✓ $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\}$
  - ✓ If you need the number of $\mathcal{X}$, use $|\mathcal{X}|$

Tips to hide the number of elements

You don't consume any additional alphabets

✓ Double brackets to enumerate natural numbers
  - ✓ $[\![N]\!] = \{1, 2, \dots, N\}$
  - ✓ 🤔 This form is simple, but not so much intuitive. Be careful.

Set-builder notation

- A way to create a set. e.g.,

Such that

Predicate (rule)

$$\{x \in \mathbb{R} \mid x > 0\}$$

Target variable

Domain of the variable

Real values such that each value is greater than 0
➡ Positive real numbers

- The domain can be written on the right:
  - ✓ $\{x \underline{\in \mathbb{R}} \mid x > 0\} = \{x \mid \underline{x \in \mathbb{R}}, x > 0\}$

- Any function can be applied to the target variable:
  - ✓ $\{3\boldsymbol{x} + 5 \mid \boldsymbol{x} \in \mathbb{R}^4, \|\boldsymbol{x}\|_2 = 1\}$

For all points in the 4-dimensional unit sphere, affine-transform them by scaling 3 and shift 5

Set-builder notation

- Multiple predicates
  - ✓ $\{\boldsymbol{x} \in \mathbb{R}^3 \mid \boldsymbol{x}^\top \boldsymbol{y}_1 = 0,\ \boldsymbol{x}^\top \boldsymbol{y}_2 = 0\}$

All 3-dim vectors that are orthogonal to both $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$

- Multiple variables can be used at the same time (multiple for-loop)
  - ✓ $\{5ij \mid i, j \in \{1, 2, 3\}, i \neq j\}$
- Set-builder notation is equivalent to a list comprehension in Python
  - ✓ $\mathcal{A} = \{1, 2, 3, 4, 5\}$, and $\{a^2 \mid a \in \mathcal{A}, a > 3\}$
  - ✓ `A = [1, 2, 3, 4, 5]` and `{a*a for a in A if a > 3}`

Equations

- Basic notation for variables
- String
- Tips for sets
- Inputs/outputs of functions
- Subscript/superscript
- Don’t write numpy
- Misc

Pseudo codes

- Basic
- What do you want to express?
- Misc

Inputs/outputs of functions

- There are two ways to describe the inputs/outputs of a function
  - ✓ e.g., considering $f(x) = x^2$ for $x \in \mathbb{R}$

$$f: \mathbb{R} \to \mathbb{R}$$ Set-based description. Use `¥to` (→)

$$f: x \mapsto x^2$$ Element-based description. Use `¥mapsto` (↦)

- Describing the inputs/outputs makes functions easier to read.

- It's ok to directly write the description in a sentence. **No side effects**.
  - ✓ 😳 "Let us define a function f as follows …"
  - ✓ 😊 "Let us define a function $f: \mathbb{N} \to \{1, \dots, 10\}$ as follows …"

Inputs/outputs of functions

Domain re two way Codomain e the inputs/outputs of a function

✓ e.g. considering $f(x) = x^2$ for $x \in \mathbb{R}$

$$f: \mathbb{R} \to \mathbb{R}$$

Set-based description. Use `¥to` (→)

$$f: x \mapsto x^2$$

Element-based description. Use `¥mapsto` (↦)

➢ Describing the inputs/outputs makes functions easier to read.

➢ It's ok to directly write the description in a sentence. **No side effects**.

✓ "Let us define a function f as follows …"

✓ "Let us define a function $f: \mathbb{N} \to \{1, \dots, 10\}$ as follows …"

Inputs/outputs of functions

- Multiple inputs: $z = f(x, y) = x^2 + y + 1$
  - ✓ $f: \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
  - ✓ $f: (x, y) \mapsto z$ or $f: (x, y) \mapsto x^2 + y + 1$

- Vector input: $f(\boldsymbol{x}) = \boldsymbol{a}^\top \boldsymbol{x} + b$, where $\boldsymbol{a}, \boldsymbol{x} \in \mathbb{R}^D$ and $b \in \mathbb{R}$
  - ✓ $f: \mathbb{R}^D \to \mathbb{R}$
  - ✓ $f: \boldsymbol{x} \mapsto \boldsymbol{a}^\top \boldsymbol{x} + b$

- Vector input and vector output: for $\boldsymbol{x} = [x_1, x_2]^\top \in \mathbb{R}^2$, f is defined as
  - ✓ $f: \boldsymbol{x} \mapsto \begin{bmatrix} x_1 + x_2 \\ 3x_1 + \log x_2 \\ x_2^3 \end{bmatrix}$ by the element-based description.
  - ✓ Then, the set-based description is $f: \mathbb{R}^2 \to \mathbb{R}^3$

Inputs/outputs of functions

- When the output is a vector, should the function itself be bold?
  - ✓ Depends.

$$f: \mathbb{R}^2 \to \mathbb{R}^3$$
$$\boldsymbol{f}: \mathbb{R}^2 \to \mathbb{R}^3$$

- 😊 With this, the equation may be beautiful: $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{a}$
- 🤨 In modern CV, all functions may have a vector-output, thus all functions may be bold. It may seem "heavy"?
- In this document, I use non-bold for vector-functions.

Set-based? Element-based?

- Usually, the set-based description is more informative.
- Readers have an interest in what are the possible values of the inputs and outputs.
- e.g., for $f(x) = x^2$,

$$f: \mathbb{R} \to \mathbb{R}$$

$$f: x \mapsto x^2$$

😊 Both an input and an output are real-values.

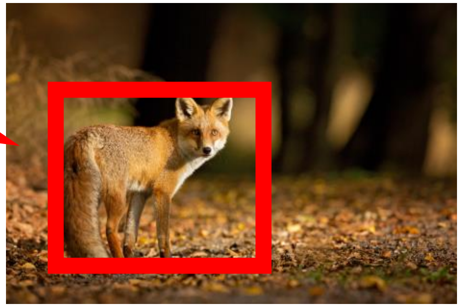
😳 This is just repeating the definition.

- The element-based description is useful if you want to **intentionally hide** the complex relationships.

Set-based? Element-based?

➢ Consider an object detector f, which inputs $H \times W \times 3$ 8-bit image (each pixels is in $\{0, \dots, 255\}$) and returns the followings.

Label $l \in \{1, \dots, K\}$
Bounding box $\boldsymbol{b} = [y, x, h, w]^\top \in \mathbb{N}^4$
Confidence score $\alpha \in \mathbb{R}$



➢ Set-based precise description is complex:

✓ $f: \{0, \dots, 255\}^{H \times W \times 3} \to \{1, \dots, K\} \times \mathbb{N}^4 \times \mathbb{R}$

🤔 The pixel value range is not important.
Don't want to use alphabets H and W here.

🤔 Not so much clear about the right side.
What is each variable?

➢ May be better to moderately hide the detail:

"We consider a K-class object detector $f: \boldsymbol{I} \mapsto (l, \boldsymbol{b}, \alpha)$. This function inputs an image $\boldsymbol{I}$, and returns a label $l \in \{1, \dots, K\}$, a bbox $\boldsymbol{b} = [y, x, h, w]^\top \in \mathbb{N}^4$, and a confidence $\alpha \in \mathbb{R}$"

Equations

- Basic notation for variables
- String
- Tips for sets
- Inputs/outputs of functions
- Subscript/superscript
- Don’t write numpy
- Misc

Pseudo codes

- Basic
- What do you want to express?
- Misc

Subscript/superscript

- Superscripts and subscripts are used to provide additional information to a variable.

$$a_n^k$$

Superscript

Subscript

- Principal: **As little use as possible!**

- $x_{i,j}^k$ Three variables: 😵 No! Hard to follow…
- x_{a_i} Subscript of subscript: 😵 No! Hard to follow…

- Alternatives for subscript/superscript: decorations
  - e.g., $\hat{x}, \bar{x}$
  - No: $x_{\text{mean}} = \frac{1}{N}\sum x_n$ Recommend: $\bar{x} = \frac{1}{N}\sum x_n$

Subscript/superscript

- Tips: a loop index can be removed by re-define a variable.
- Consider $\mathcal{V} = \{v_n\}_{n=1}^{N}$
- 😐 "Here, $f(v_n)$ is …"

n is not important

```python
for n in range(len(V)):
    f(V[n])
```

- 😊 "Let us consider $v \in \mathcal{V}$. Here, $f(v)$ is …"

Remove it!

```python
for v in V:
    f(v)
```

Subscript/superscript

- Don’t make the subscript bold (very typical mistake)
  - ✓ 😵 `¥mathbf{x_i}`: $\boldsymbol{x_i}$
  - ✓ 😊 `¥mathbf{x}_i`: $\boldsymbol{x}_i$

> - Natural interpretation is that $\boldsymbol{i}$ is a vector to indicate identifiers, e.g., $\boldsymbol{i} = [1, 2]$, results in $x_{1,2}$
> - Moreover, this mistake smells amateurish.

- If two+ letters are used in the meaning of a label, make it Roman.
  - ✓ `¥mathbf{x}_¥mathrm{in}`: x_{in} OK
  - ✓ `¥mathbf{x}_{in}`: x_{in} Not recommend

> - This can be interpreted as
>   - ✓ $j = i * n$
>   - ✓ x_j

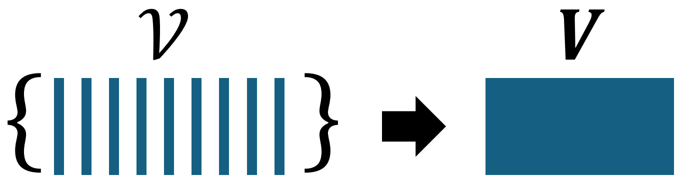
Subscript/superscript

- Don't put variables with different meanings in sub/superscript.

- Consider $\mathcal{V} = \{\boldsymbol{v}_n\}_{n=1}^{N}$, where $\boldsymbol{v}_n \in \mathbb{R}^D$

- How to denote d-th element of n-th vector?
  - ✓ 😵 $v_{n,d} \in \mathbb{R}$ Not recommend! n and d have different meanings!
  - ✓ 🤔 $v_n[d] \in \mathbb{R}$: OK. Combining square brackets.
  - ✓ 😊 $v_d \in \mathbb{R}$ or $v[d] \in \mathbb{R}$ for $\boldsymbol{v} \in \mathcal{V}$: Clear.

- Tips: You can stack $\mathcal{V}$ to form a matrix $\boldsymbol{V} \in \mathbb{R}^{D \times N}$



  - ✓ Then, $v_{d,n} \in \mathbb{R}$ or $v[d, n] \in \mathbb{R}$ is OK (usual matrix element access)

(n, d) is now (d, n). Be careful!

- Vector-set ($\mathcal{V}$) to matrix ($\boldsymbol{V}$) doesn't consume an alphabet.

Equations

- Basic notation for variables
- String
- Tips for sets
- Inputs/outputs of functions
- Subscript/superscript
- Don’t write numpy
- Misc

Pseudo codes

- Basic
- What do you want to express?
- Misc

Don't write numpy

- Many people write numpy descriptions directly in equations. It's wrong.

```
a = np.array([5, 4, 3])
b = a - 3
```

$$a = [5, 4, 3]$$
$$b = a - 3$$

A horrible mistake!

- Numpy can broadcast a scalar
- We can get b=`[2, 1, 0]`

- Math equations don't broadcast.
- This operation is undefined!

- Recommend:
  - $\boldsymbol{a} = [5, 4, 3]^\top$. $\boldsymbol{b} = \boldsymbol{a} - 3\mathbf{1}$

- Use bold
- Column-vectors

- Use all-one vector $\mathbf{1} = [1, 1, \ldots]^\top$

Specific examples of failures

- Let X and Y be 3-channel images.
- Consider a mask matrix B whose elements are 0 or 1.
- We adopt X when the value of the mask is one and Y when the value of the mask is zero.
- Z, the resulting image combining X and Y, is computed as follows.

$$Z = BX + (1 - B)Y$$

Z B X $1 - B$ Y

There are three big mistakes... Can you guess? 🤔

Specific examples of failures

- Let X and Y be 3-channel images.
- Consider a mask matrix B whose elements are 0 or 1.
- We adopt X when the value of the mask is one and Y when the value of the mask is zero.
- Z, the resulting image combining X and Y, is computed as follows.

$$\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{X} + (1 - \boldsymbol{B})\boldsymbol{Y}$$

$\boldsymbol{Z} \in \mathbb{R}^{H \times W \times 3}$ $=$ $\boldsymbol{B} \in \{0,1\}^{H \times W}$ $\boldsymbol{X} \in \mathbb{R}^{H \times W \times 3}$ $+$ $1 - \boldsymbol{B}$ $\boldsymbol{Y} \in \mathbb{R}^{H \times W \times 3}$

First, make the variables bold, and write domains

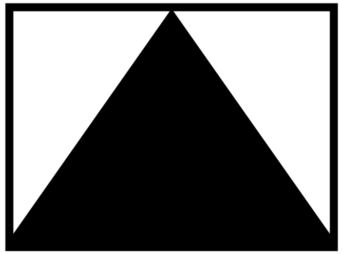
Specific examples of failures

- Let X and Y be 3-channel images.
- Consider a mask matrix B whose elements are 0 or 1.
- We adopt X when the value of the mask is one and Y when the value of the mask is zero.
- Z, the resulting image combining X and Y, is computed as follows.

$$\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{X} + (1 - \boldsymbol{B})\boldsymbol{Y}$$

$\boldsymbol{Z} \in \mathbb{R}^{H \times W \times 3}$

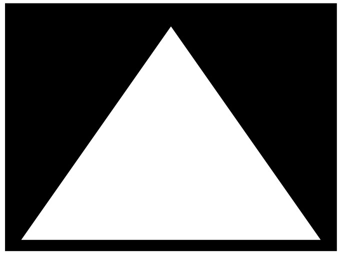
$=$



$\boldsymbol{B} \in \{0, 1\}^{H \times W}$

$\boldsymbol{X} \in \mathbb{R}^{H \times W \times 3}$

$+$



$1 - \boldsymbol{B}$

$\boldsymbol{Y} \in \mathbb{R}^{H \times W \times 3}$

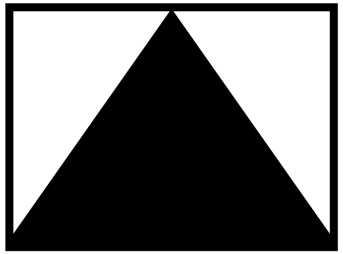
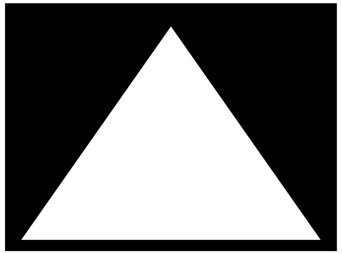
First, make the variables bold, and write domains

Mistake1: Broadcast happens here! We need to use an all-one matrix.

Specific examples of failures

- Let X and Y be 3-channel images.
- Consider a mask matrix B whose elements are 0 or 1.
- We adopt X when the value of the mask is one and Y when the value of the mask is zero.
- Z, the resulting image combining X and Y, is computed as follows.

$$\boldsymbol{Z} = \boldsymbol{BX} + (1 - \boldsymbol{B})\boldsymbol{Y}$$

 =   +  

$\boldsymbol{Z} \in \mathbb{R}^{H \times W \times 3}$ $\boldsymbol{B} \in \{0, 1\}^{H \times W}$ $\boldsymbol{X} \in \mathbb{R}^{H \times W \times 3}$ $\boldsymbol{1} - \boldsymbol{B}$ $\boldsymbol{Y} \in \mathbb{R}^{H \times W \times 3}$

Specific examples of failures

- Let X and Y be 3-channel images.
- C
- W hen the value of the mask is zero.
- Z, the resulting image combining X and Y, is computed as follows.

$$\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{X} + (1 - \boldsymbol{B})\boldsymbol{Y}$$

Mistake2: $\boldsymbol{BX}$ is matrix-multiplication! We need per-element multiplication (Hadamard product) $\odot$

$\boldsymbol{Z} \in \mathbb{R}^{H\times W\times 3}$ $=$ $\boldsymbol{B} \in \{0,1\}^{H\times W}$ $\boldsymbol{X} \in \mathbb{R}^{H\times W\times 3}$ $+$ $\boldsymbol{1} - \boldsymbol{B}$ $\boldsymbol{Y} \in \mathbb{R}^{H\times W\times 3}$

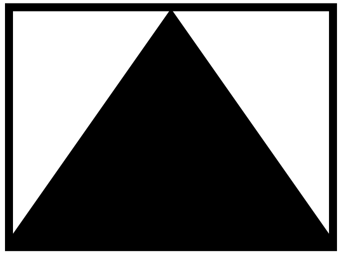
Specific examples of failures

- Let X and Y be 3-channel images.
- Consider a mask matrix B whose elements are 0 or 1.
- We adopt X when the value of the mask is one and Y when the value of the mask is zero.
- Z, the resulting image combining X and Y, is computed as follows.

$$\boldsymbol{Z} = \boldsymbol{B} \odot \boldsymbol{X} + (\mathbf{1} - \boldsymbol{B}) \odot \boldsymbol{Y}$$

$\boldsymbol{Z} \in \mathbb{R}^{H\times W\times 3}$

$=$

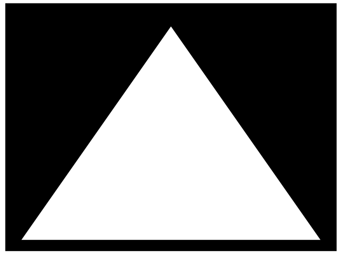


$\boldsymbol{B} \in \{0, 1\}^{H\times W}$

$\odot$

$\boldsymbol{X} \in \mathbb{R}^{H\times W\times 3}$

$+$



$\mathbf{1} - \boldsymbol{B}$

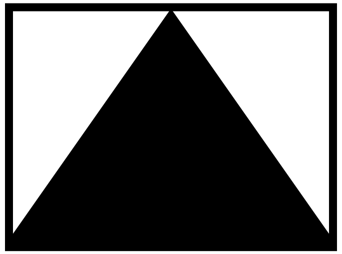
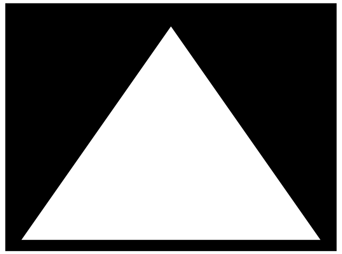
$\odot$

$\boldsymbol{Y} \in \mathbb{R}^{H\times W\times 3}$

Specific examples of failures

- Let X and Y be 3-channel images.
- Consider a mask matrix B whose elements are 0 or 1.
- We adopt X when the value of the mask is one and Y when the value of the mask is zero.
- Z, the resulting image combining X and Y, is computed as follows.

$$\boldsymbol{Z} = \boldsymbol{B} \odot \boldsymbol{X} + (1 - \boldsymbol{B}) \odot \boldsymbol{Y}$$

 =  $\odot$  +  $\odot$ 

$\boldsymbol{Z} \in \mathbb{R}^{H\times W\times 3}$ $\quad$ $\boldsymbol{B} \in \{0,1\}^{H\times W}$ $\quad$ $\boldsymbol{X} \in \mathbb{R}^{H\times W\times 3}$ $\quad$ $\mathbf{1} - \boldsymbol{B}$ $\quad$ $\boldsymbol{Y} \in \mathbb{R}^{H\times W\times 3}$

Mistake3: $\boldsymbol{B} \odot \boldsymbol{X}$ still doesn't work because the tensor shape is different! $H \times W$ vs $H \times W \times 3$

80% computer vision papers ignore this issue....

Specific examples of failures

- Let X and Y be 3-channel images.
- Consider a mask matrix B whose elements are 0 or 1.
- We adopt X when the value of the mask is one and Y when the value of the mask is zero.
- Z, the resulting image combining X and Y, is computed as follows.

$$\boldsymbol{Z} = \boldsymbol{B} \odot \boldsymbol{X} + (\mathbf{1} - \boldsymbol{B}) \odot \boldsymbol{Y}$$

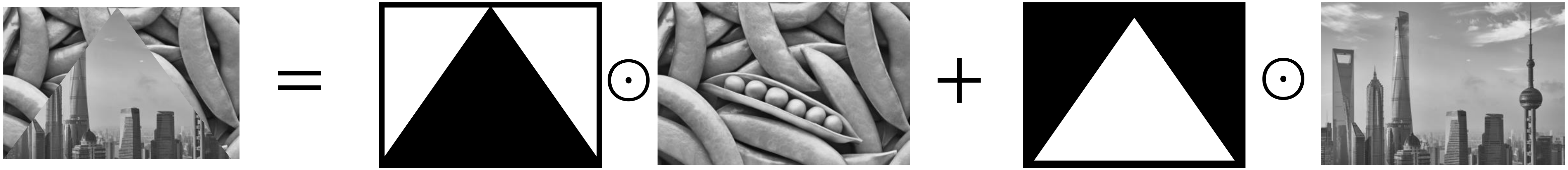
$\boldsymbol{Z} \in \mathbb{R}^{H\times W\times 3}$ $\boldsymbol{B} \in \{0,1\}^{H\times W\times 3}$ $\boldsymbol{X} \in \mathbb{R}^{H\times W\times 3}$ $\mathbf{1} - \boldsymbol{B}$ $\boldsymbol{Y} \in \mathbb{R}^{H\times W\times 3}$

Solution 1: Define $\boldsymbol{B}$ as a stack of the original $\boldsymbol{B}$, making the shape same.

Specific examples of failures

- Let X and Y be 3-channel images.
- Consider a mask matrix B whose elements are 0 or 1.
- We adopt X when the value of the mask is one and Y when the value of the mask is zero.
- Z, the resulting image combining X and Y, is computed as follows.

$$\boldsymbol{Z} = \boldsymbol{B} \odot \boldsymbol{X} + (1 - \boldsymbol{B}) \odot \boldsymbol{Y}$$



$\boldsymbol{Z} \in \mathbb{R}^{H\times W\times \cancel{3}}$ $=$ $\boldsymbol{B} \in \{0,1\}^{H\times W}$ $\odot$ $\boldsymbol{X} \in \mathbb{R}^{H\times W\times \cancel{3}}$ $+$ $\boldsymbol{1} - \boldsymbol{B}$ $\odot$ $\boldsymbol{Y} \in \mathbb{R}^{H\times W\times \cancel{3}}$

Solution 2: Descript everything per channel, then combine the results.

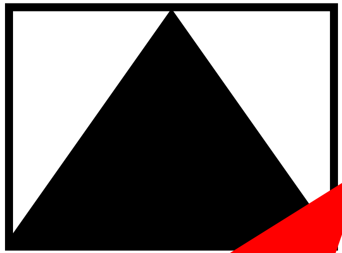
Specific examples of failures

- Let X and Y be 3-channel images.
- Consider a mask matrix B whose elements are 0 or 1.
- We adopt X when the value of the mask is one and Y when the value of the mask is zero.
- Z, the resulting image combining X and Y, is computed as follows.

$$\boldsymbol{Z} = \boldsymbol{B} \odot \boldsymbol{X} + (1 - \boldsymbol{B}) \odot \boldsymbol{Y}$$

$\boldsymbol{Z} \in \mathbb{R}^{H\times W\times 3}$

$=$

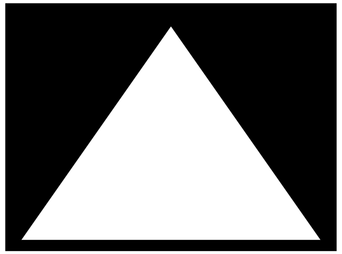


$\boldsymbol{B} \in \{0$

$\odot$

$\boldsymbol{X} \in \mathbb{R}^{H\times W\times 3}$

$+$



$\mathbf{1} - \boldsymbol{B}$

$\odot$

$\boldsymbol{Y} \in \mathbb{R}^{H\times W\times 3}$

Solution 3: Define $\odot$ loosely, e.g.,

[2]Similar to many papers, we also abuse this operator slightly; $X \odot (1 - t(\phi(V)))$ computes the element-wise product between $(1 - t(\phi(V)))$ and each channel of X.

https://arxiv.org/abs/2311.15994

Equations

- Basic notation for variables
- String
- Tips for sets
- Inputs/outputs of functions
- Subscript/superscript
- Don't write numpy
- Misc

Pseudo codes

- Basic
- What do you want to express?
- Misc

Misc

- Don’t write English words in equations
  - 😵 $y = score(x) + 10$
  - 😊 $y = s(x) + 10$

- This can be interpreted as $s * core(x)$

- If you’re not 100% confident, don’t use quantifiers (∀, ∃).

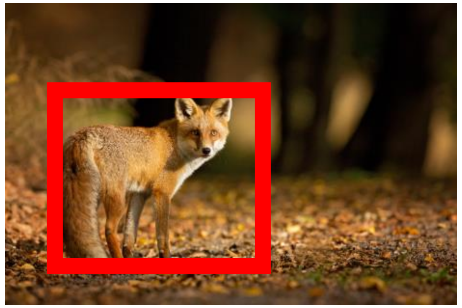
- If you use variables, please always define and explain them. E.g., if you write $y = ax + b$, then explain y, a, x, b.

- If it is too difficult to describe what you are trying to explain in mathematical equations, please explain them in writing and figures.
  - ✓ 😵 Writing half-wrong equations are terrible.

Tuple

- (something like) an ordered-set: e.g., $a = (10, 20, 30)$
- Similar to a set:
  - ✓ Can contain any elements: e.g., $b = ([1, 2, 3]^\top, \text{"a"}, 25)$
  - ✓ Remember: $f: \boldsymbol{I} \mapsto (l, \boldsymbol{b}, \alpha)$

This was a tuple



- But the order is decided:
  - ✓ $(1, 2, 3) \neq (2, 1, 3)$
  - ✓ $\{1, 2, 3\} = \{2, 1, 3\}$
- Can contain duplicates: $a = (1, 1, 3)$
- Similar to a vector:
  - ✓ $\boldsymbol{c} = [1, 2, 3]^\top$ vs $d = (1, 2, 3)$
- But (usually) no mathematical structures:
  - ✓ E.g., the addition is not defined: $(1, 2, 3) + (4, 5, 6)$

No!

Cheat-sheet

Not-recommend	**OK**	**Reason**
Write a vector as x	Write a vector as $\boldsymbol{x}$	Use a bold font for a vector.
$\boldsymbol{x} = [1, 2, 3]$	$\boldsymbol{x} = [1, 2, 3]^\top$	Use column-vectors.
Consider D-dim vector $\boldsymbol{x}$	Consider D-dim vector $\boldsymbol{x} \in \mathbb{R}^D$	Show the domain.
$x_{i,j}^k$ Many sub-/superscripts	Don't use so much.	Hard to understand.
x_{a_i} subscript of subscript	Avoid.	Hard to understand.
$\boldsymbol{x_i}$	$\boldsymbol{x}_i$	Don't make an index bold.
x_{in}	x_{in}	Labels should be roman.
$score(x) + 10$	$s(x) + 10$	Don't use English words.
$a \in \mathbb{R}^3, b \in \mathbb{R}$, then $a + b$	$\boldsymbol{a} + b\mathbf{1}$	Don't broadcast.
For element-wise product, $\boldsymbol{AB}$	$\boldsymbol{A} \odot \boldsymbol{B}$	Element-wise product is Hadamard product.
Using $\forall, \exists$ but not 100% confident	Don't use	You are wrong.

Equations

- Basic notation for variables
- String
- Tips for sets
- Inputs/outputs of functions
- Subscript/superscript
- Don’t write numpy
- Misc

Pseudo codes

- Basic
- What do you want to express?
- Misc

Pseudo-code: Basic

- What is pseudo-code?
  - ✓ Describe an algorithm
  - ✓ OK to use mathematical equations
  - ✓ OK to use data structures
  - ✓ Highlight the key idea
  - ✓ Many ways to explain the idea

procedure CP-ALS($\mathcal{X}$,R)
  initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \dots, N$
  repeat
    for $n = 1, \dots, N$ **do**
      $\mathbf{V} \leftarrow \mathbf{A}^{(1)\mathsf{T}}\mathbf{A}^{(1)} * \cdots * \mathbf{A}^{(n-1)\mathsf{T}}\mathbf{A}^{(n-1)} * \mathbf{A}^{(n+1)\mathsf{T}}\mathbf{A}^{(n+1)} * \cdots * \mathbf{A}^{(N)\mathsf{T}}\mathbf{A}^{(N)}$
      $\mathbf{A}^{(n)} \leftarrow \mathbf{X}_{(n)}(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{V}^{\dagger}$
      normalize columns of $\mathbf{A}^{(n)}$ (storing norms as $\boldsymbol{\lambda}$)
    end for
  until fit ceases to improve or maximum iterations exhausted
  return $\boldsymbol{\lambda}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \dots, \mathbf{A}^{(N)}$
end procedure

Fig. 3.3 *ALS algorithm to compute a CP decomposition with R components for an Nth-order tensor $\mathcal{X}$ of size $I_1 \times I_2 \times \cdots \times I_N$.*

Kolda and Bader, "Tensor Decompositions and Applications", SIAM Review, 2009.

Algorithm 3 SVD1 : QB-backed low-rank SVD (see [HMT11] and [RST10])

function SVD1($\mathbf{A}, k, \epsilon, s$)
  Inputs:
    $\mathbf{A}$ is an $m \times n$ matrix. The returned approximation will have rank *at most* k. The approximation produced by the randomized phase of the algorithm will attempt to $\mathbf{A}$ to within ϵ error, but will not produce an approximation of rank greater than $k + s$.
  Output:
    The compact SVD of a low-rank approximation of $\mathbf{A}$.
  Abstract subroutines:
    QBDecomposer generates a QB decomposition of a given matrix; it tries to reach a prescribed error tolerance but may stop early if it reaches a prescribed rank limit.
  $\mathbf{Q}, \mathbf{B} = \texttt{QBDecomposer}(\mathbf{A}, k + s, \epsilon)$ # $\mathbf{QB} \approx \mathbf{A}$
  $r = \min\{k,$ number of columns in $\mathbf{Q}\}$
  $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}^* = \texttt{svd}(\mathbf{B})$
  $\mathbf{U} = \mathbf{U}[: , :r]$
  $\mathbf{V} = \mathbf{V}[: , :r]$
  $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}[:r , :r]$
  $\mathbf{U} = \mathbf{QU}$
  return $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}^*$

Murray+, "Randomized Numerical Linear Algebra : A Perspective on the Field With an Eye to Software", arXiv 2023

Pseudo-code: Basic

- What is pseudo-code?
  - ✓ Describe an algorithm
  - ✓ OK to use mathematical equations
  - ✓ OK to use data structures
  - ✓ Highlight the key idea
  - ✓ Many ways to explain the idea

Inputs / outputs description

procedure CP-ALS($\mathcal{X}$,R)
  $\boxed{\text{initialize } \mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R} \text{ for } n = 1, \ldots, N}$
  repeat
    for $n = 1, \ldots, N$ **do**
      $\mathbf{V} \leftarrow \mathbf{A}^{(1)\mathsf{T}}\mathbf{A}^{(1)} * \cdots * \mathbf{A}^{(n-1)\mathsf{T}}\mathbf{A}^{(n-1)} * \mathbf{A}^{(n+1)\mathsf{T}}\mathbf{A}^{(n+1)} * \cdots * \mathbf{A}^{(N)\mathsf{T}}\mathbf{A}^{(N)}$
      $\mathbf{A}^{(n)} \leftarrow \mathbf{X}_{(n)}(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{V}^{\dagger}$
      normalize columns of $\mathbf{A}^{(n)}$ (storing norms as $\boldsymbol{\lambda}$)
    end for
  until fit ceases to improve or maximum iterations exhausted
  return $\boldsymbol{\lambda}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$
end procedure

Fig. 3.3 *ALS algorithm to compute a CP decomposition with R components for an Nth-order tensor $\mathcal{X}$ of size $I_1 \times I_2 \times \cdots \times I_N$.*

Kolda and Bader, "Tensor Decompositions and Applications", SIAM Review, 2009.

…rithm 3 S… QB-backed low-rank SVD (see [HMT11] and [RST10])

function S…($\mathbf{A}, k, \epsilon, s$)

Inputs:
  $\mathbf{A}$ is an $m \times n$ matrix. The returned approximation will have rank *at most* k. The approximation produced by the randomized phase of the algorithm will attempt to $\mathbf{A}$ to within ϵ error, but will not produce an approximation of rank greater than $k + s$.

Output:
  The compact SVD of a low-rank approximation of $\mathbf{A}$.

Abstract subroutines:
  `QBDecomposer` generates a QB decomposition of a given matrix; it tries to reach a prescribed error tolerance but may stop early if it reaches a prescribed rank limit.

  $\mathbf{Q}, \mathbf{B} = \mathtt{QBDecomposer}(\mathbf{A}, k + s, \epsilon)$ # $\mathbf{QB} \approx \mathbf{A}$
  $r = \min\{k,$ number of columns in $\mathbf{Q}\}$
  $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}^* = \mathtt{svd}(\mathbf{B})$
  $\mathbf{U} = \mathbf{U}[:\, ,\, :r]$
  $\mathbf{V} = \mathbf{V}[:\, ,\, :r]$
  $\mathbf{\Sigma} = \mathbf{\Sigma}[:r\, ,\, :r]$
  $\mathbf{U} = \mathbf{QU}$
  return $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}^*$

Murray+, "Randomized Numerical Linear Algebra : A Perspective on the Field With an Eye to Software", arXiv 2023

Pseudo-code: Basic

- What is pseudo-code?
  - ✓ Describe an algorithm
  - ✓ OK to use mathematical equations
  - ✓ OK to use data structures
  - ✓ Highlight the key idea
  - ✓ Many ways to explain the idea

procedure CP-ALS($\mathcal{X}$,R)
    initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$
    repeat
        for $n = 1, \ldots, N$ **do**
            $\mathbf{V} \leftarrow \mathbf{A}^{(1)\mathsf{T}}\mathbf{A}^{(1)} * \cdots * \mathbf{A}^{(n-1)\mathsf{T}}\mathbf{A}^{(n-1)} * \mathbf{A}^{(n+1)\mathsf{T}}\mathbf{A}^{(n+1)} * \cdots * \mathbf{A}^{(N)\mathsf{T}}\mathbf{A}^{(N)}$
            $\mathbf{A}^{(n)} \leftarrow \mathbf{X}_{(n)}(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{V}^{\dagger}$
            normalize columns of $\mathbf{A}^{(n)}$ (storing norms as $\boldsymbol{\lambda}$)
        end for
    until fit ceases to improve or maximum iterations exhausted
    return $\boldsymbol{\lambda}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$
end procedure

Fig. 3.3 *ALS algorithm to compute a CP decomposition with R components for an Nth-order tensor $\mathcal{X}$ of size $I_1 \times I_2 \times \cdots \times I_N$.*

Kolda and Bader, "Tensor Decompositions and Applications", SIAM Review, 2009.

Can run other functions or pseudo-codes

Algorithm 3 `SVD1` : QB-backed low-rank SVD (see [HMT11] and [RST10])

function `SVD1`($\mathbf{A}, k, \epsilon, s$)
    Inputs:
        $\mathbf{A}$ is an $m \times n$ matrix. The returned approximation will have rank *at most* k. The approximation produced by the randomized phase of the algorithm will attempt to $\mathbf{A}$ to within ϵ error, but will not produce an approximation of rank greater than $k + s$.
    Output:
        The compact SVD of a low-rank approximation of $\mathbf{A}$.
    Abstract subroutines:
        `QBDecomposer` generates a QB decomposition of a given matrix; it tries to reach a prescribed error tolerance but may stop early if it reaches a prescribed rank limit.
    $\boxed{\mathbf{Q}, \mathbf{B} = \texttt{QBDecomposer}(\mathbf{A}, k+s, \epsilon)}$ # $\mathbf{QB} \approx \mathbf{A}$
    $r = \min\{k,$ number of columns in $\mathbf{Q}\}$
    $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}^* = \texttt{svd}(\mathbf{B})$
    $\mathbf{U} = \mathbf{U}[:\,,\, :r]$
    $\mathbf{V} = \mathbf{V}[:\,,\, :r]$
    $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}[:r\,,\, :r]$
    $\mathbf{U} = \mathbf{QU}$
    return $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}^*$

Murray+, "Randomized Numerical Linear Algebra : A Perspective on the Field With an Eye to Software", arXiv 2023

Pseudo-code: Basic

- What is pseudo-code?
  - ✓ Describe an algorithm
  - ✓ OK to use mathematical equations
  - ✓ OK to use data structures
  - ✓ Highlight the key idea
  - ✓ Many ways to explain the idea

procedure CP-ALS($\mathcal{X}$,R)
  initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$
  repeat
    for $n = 1, \ldots, N$ **do**
      $\mathbf{V} \leftarrow \mathbf{A}^{(1)\mathsf{T}}\mathbf{A}^{(1)} * \cdots * \mathbf{A}^{(n-1)\mathsf{T}}\mathbf{A}^{(n-1)} * \mathbf{A}^{(n+1)\mathsf{T}}\mathbf{A}^{(n+1)} * \cdots * \mathbf{A}^{(N)\mathsf{T}}\mathbf{A}^{(N)}$
      $\mathbf{A}^{(n)} \leftarrow \mathbf{X}_{(n)}(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{V}^{\dagger}$
      normalize columns of $\mathbf{A}^{(n)}$ (storing norms as $\boldsymbol{\lambda}$)
    end for
  until fit ceases to improve or maximum iterations exhausted
  return $\boldsymbol{\lambda}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$
end procedure

Fig. 3.3 *ALS algorithm to compute a CP decomposition with R components for an Nth-order tensor $\mathcal{X}$ of size $I_1 \times I_2 \times \cdots \times I_N$.*

Kolda and Bader, "Tensor Decompositions and Applications", SIAM Review, 2009.

Algorithm 3 SVD1 : QB-backed low-rank SVD (see [HMT11] and [RST10])

function SVD1($\mathbf{A}, k, \epsilon, s$)
  Inputs:
    $\mathbf{A}$ is an $m \times n$ matrix. The returned approximation will have rank *at most* k. The approximation produced by the randomized phase of the algorithm will attempt to $\mathbf{A}$ to within ϵ error, but will not produce an approximation of rank greater than $k + s$.
  Output:
    The compact SVD of a low
  Abstract subroutines:
    QBDecomposer generates a QB position of a given matrix; it tries to reach a prescribed error rance but may stop early if it reaches a prescribed rank limit.
  $\mathbf{Q}, \mathbf{B} = \texttt{QBDecomposer}(\mathbf{A}, k + s, \epsilon)$ # QB ≈ A
  $r = \min\{k,$ number of columns in $\mathbf{Q}\}$
  $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}^{*} = \texttt{svd}(\mathbf{B})$
  $\mathbf{U} = \mathbf{U}[:\,,\,:r]$
  $\mathbf{V} = \mathbf{V}[:\,,\,:r]$
  $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}[:r\,,\,:r]$
  $\mathbf{U} = \mathbf{Q}\mathbf{U}$
  return $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}^{*}$

Comments

Murray+, "Randomized Numerical Linear Algebra : A Perspective on the Field With an Eye to Software", arXiv 2023

Pseudo-code: Basic

- What is pseudo-code?
  - ✓ Describe an algorithm
  - ✓ OK to use mathematical equations
  - ✓ OK to use data structures

> - Can use usual sentences to represent complex operations.
> - If you think it's hard to write your operations by equations, use sentences.

procedure CP-ALS($\mathcal{X}$,R)
  initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n =$ …
  repeat
    for $n = 1, \ldots, N$ **do**
      $\mathbf{V} \leftarrow \mathbf{A}^{(1)\mathsf{T}}\mathbf{A}^{(1)} * \cdots * \mathbf{A}^{(n-1)}$ … $* \mathbf{A}^{(n+1)\mathsf{T}}\mathbf{A}^{(n+1)} * \cdots * \mathbf{A}^{(N)\mathsf{T}}\mathbf{A}^{(N)}$
      $\mathbf{A}^{(n)} \leftarrow \mathbf{X}_{(n)}(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{V}^{\dagger}$
      normalize columns of $\mathbf{A}^{(n)}$ (storing norms as $\boldsymbol{\lambda}$)
    end for
  until fit ceases to improve or maximum iterations exhausted
  return $\boldsymbol{\lambda}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$
end procedure

Fig. 3.3 *ALS algorithm to compute a CP decomposition with R components for an Nth-order tensor $\mathcal{X}$ of size $I_1 \times I_2 \times \cdots \times I_N$.*

Kolda and Bader, "Tensor Decompositions and Applications", SIAM Review, 2009.

Algorithm 3 `SVD1` : QB-backed low-rank SVD (see [HMT11] and [RST10])

1: **function** `SVD1`($\mathbf{A}, k, \epsilon, s$)
  Inputs:
  Output:
   The compact SVD of a low-rank approximation of $\mathbf{A}$.
  Abstract subroutines:
   `QBDecomposer` generates a QB decomposition of a given matrix; it tries to reach a prescribed error tolerance but may stop early if it reaches a prescribed rank limit.
2:   $\mathbf{Q}, \mathbf{B} =$ `QBDecomposer`$(\mathbf{A}, k+s, \epsilon)$ # $\mathbf{QB} \approx \mathbf{A}$
3:   $r = \min\{k,$ number of columns in $\mathbf{Q}\}$
4:   $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}^* =$ `svd`$(\mathbf{B})$
5:   $\mathbf{U} = \mathbf{U}[:\,,\ :r]$
6:   $\mathbf{V} = \mathbf{V}[:\,,\ :r]$
7:   $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}[:r\,,\ :r]$
8:   $\mathbf{U} = \mathbf{QU}$
9:   return $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}^*$

Murray+, "Randomized Numerical Linear Algebra : A Perspective on the Field With an Eye to Software", arXiv 2023

Pseudo-code: Basic

- Look very different
- Left: more pseudo-code-ish
- Right: more like equations (this is actually MATLAB-ish)

✓ OK to u… …tures

✓ Highligh… …idea

✓ Many way… …explain the idea

procedure CP-ALS($\mathcal{X}$,R)
  initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$
  repeat
    for $n = 1, \ldots, N$ **do**
      $\mathbf{V} \leftarrow \mathbf{A}^{(1)\mathsf{T}}\mathbf{A}^{(1)} * \cdots * \mathbf{A}^{(n-1)\mathsf{T}}\mathbf{A}^{(n-1)} * \mathbf{A}^{(n+1)\mathsf{T}}\mathbf{A}^{(n+1)} * \cdots * \mathbf{A}^{(N)\mathsf{T}}\mathbf{A}^{(N)}$
      $\mathbf{A}^{(n)} \leftarrow \mathbf{X}_{(n)}(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{V}^{\dagger}$
      normalize columns of $\mathbf{A}^{(n)}$ (storing norms as $\boldsymbol{\lambda}$)
    end for
  until fit ceases to improve or maximum iterations exhausted
  return $\boldsymbol{\lambda}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$
end procedure

Fig. 3.3 *ALS algorithm to compute a CP decomposition with R components for an Nth-order tensor $\mathcal{X}$ of size $I_1 \times I_2 \times \cdots \times I_N$.*

Kolda and Bader, "Tensor Decompositions and Applications", SIAM Review, 2009.

…SVD (see [HMT11] and [RST10])

…eturned approximation will have rank *at* …n produced by the randomized phase of the …o $\mathbf{A}$ to within ϵ error, but will not produce a… …nk greater than $k + s$.

Outpu…
The c… …of a low-rank approximation of $\mathbf{A}$.

Abstract s… …s:
QBDecomp… …generates a QB decomposition of a given matrix; it tries to rea… a prescribed error tolerance but may stop early if it reaches a prescribed rank limit.

$\mathbf{Q}, \mathbf{B} = \mathtt{QBDecomposer}(\mathbf{A}, k+s, \epsilon)$ # $\mathbf{QB} \approx \mathbf{A}$
$r = \min\{k,$ number of columns in $\mathbf{Q}\}$
$\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}^* = \mathtt{svd}(\mathbf{B})$
$\mathbf{U} = \mathbf{U}[:\ ,\ :r]$
$\mathbf{V} = \mathbf{V}[:\ ,\ :r]$
$\boldsymbol{\Sigma} = \boldsymbol{\Sigma}[:r\ ,\ :r]$
$\mathbf{U} = \mathbf{QU}$
return $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}^*$

Murray+, "Randomized Numerical Linear Algebra : A Perspective on the Field With an Eye to Software", arXiv 2023

Pseudo-code: Basic

- Compared to equations, it's more like coding.
  - ✓ Assigning operation is usual: $a \leftarrow a + 1$

- Fonts
  - ✓ `¥textsc{Clustering}` $\textsc{Clustering}$
  - ✓ `¥texttt{Clustering}` `Clustering`

Can be used for function names

In Powerpoint, use "Consolas" font. `Like this.`

- Several TeX packages https://www.overleaf.com/learn/latex/Algorithms

```
i ← 10
if i ≥ 5 then
    i ← i − 1
else
    if i ≤ 3 then
        i ← i + 2
    end if
end if
```

```
Algorithm 1 An algorithm with caption
Require: n ≥ 0
Ensure: y = x^n
  y ← 1
  X ← x
  N ← n
  while N ≠ 0 do
    if N is even then
      X ← X × X
      N ← N/2                          ▷ This is a comment
    else if N is odd then
      y ← y × X
      N ← N − 1
    end if
  end while
```

```
Algorithm 1: An algorithm with caption
  Data: n ≥ 0
  Result: y = x^n
  y ← 1;
  X ← x;
  N ← n;
  while N ≠ 0 do
    if N is even then
      X ← X × X;
      N ← N/2 ; /* This is a comment */
    else
      if N is odd then
        y ← y × X;
        N ← N − 1;
      end
    end
  end
```

Pseudo-code: Basic

- Again, consistency is important!
  - ✓ 😊 OK: $a \leftarrow a + 1$ … $b \leftarrow f(x)$
  - ✓ 😵 NO!: $a \leftarrow a + 1$ … $b = f(x)$

- You can mix a mathematical way and a coding way
  - ✓ $x \leftarrow \frac{1}{\mathrm{Pop}(v)} \int_0^a f(\theta) d\theta$

- Don't write too much! Pseudo-code should be simple.

- Several styles
  - $\boldsymbol{v}$.push_back(a), Append($\boldsymbol{v}, a$), $\boldsymbol{v} \leftarrow [\boldsymbol{v}|a]$

    Assuming a row vector
  - 😵 Again, don't use italic for function: $\boldsymbol{v}.push\_back(a)$

Equations

- Basic notation for variables
- String
- Tips for sets
- Inputs/outputs of functions
- Subscript/superscript
- Don’t write numpy
- Misc

Pseudo codes

- Basic
- What do you want to express?
- Misc

What do you want to express?

- The design of pseudo-code depends on what you want to express.
- Consider `std::vector<int> arr` in your code

- If you use arr to represent a set of integers,
  - ✓ You can use a set:
  - ✓ $\mathcal{A} \leftarrow \emptyset$: initialization
  - ✓ $\mathcal{A} \leftarrow \mathcal{A} \cup \{x\}$: add
  - ✓ $\mathcal{A} \leftarrow \mathcal{A} \setminus \{y\}$: delete

- If $O(1)$ access is important, or use it in a mathematical context,
  - ✓ You should use an array (vector)
  - ✓ $\boldsymbol{a} \in \mathbb{R}^D$
  - ✓ $\boldsymbol{a}[i]$ or a_i : Implies an $O(1)$ access.

Case study: Beam search for neighbor search

Algorithm 1. Search-on-Graph(G, $\mathbf{p}$, $\mathbf{q}$, l)

Require: graph G, start node $\mathbf{p}$, query point $\mathbf{q}$, candidate pool size l
Ensure: k nearest neighbors of $\mathbf{q}$

$i = 0$, candidate pool $S = \emptyset$
S.add($\mathbf{p}$)
while $i < l$ **do**
  i = the id of the first unchecked node p_i in S
  mark $\mathbf{p_i}$ as checked
  for all neighbor $\mathbf{n}$ of $\mathbf{p_i}$ in G **do**
    if $\mathbf{n}$ has not been visited **then**
      S.add($\mathbf{n}$)
    end if
  end for
  sort S in ascending order of the distance to $\mathbf{q}$
  if S.size() $> l$ **then**
    S.resize(l) / / remove nodes from back of S to keep its size no larger than l
  end if
end while
return the first k nodes in S

NSG [Cong+, VLDB 19]

Algorithm 1: GreedySearch(s, x_q, k, L)

Data: Graph G with start node s, query x_q, result size k, search list size $L \geq k$
Result: Result set $\mathcal{L}$ containing k-approx NNs, and a set $\mathcal{V}$ containing all the visited nodes
begin
  initialize sets $\mathcal{L} \leftarrow \{s\}$ and $\mathcal{V} \leftarrow \emptyset$
  while $\mathcal{L} \setminus \mathcal{V} \neq \emptyset$ **do**
    let $p* \leftarrow \arg\min_{p \in \mathcal{L} \setminus \mathcal{V}} ||\mathrm{x}_p - \mathrm{x}_q||$
    update $\mathcal{L} \leftarrow \mathcal{L} \cup N_{\mathrm{out}}(p^*)$ and $\mathcal{V} \leftarrow \mathcal{V} \cup \{p^*\}$
    if $|\mathcal{L}| > L$ **then**
      update $\mathcal{L}$ to retain closest L points to x_q
  return [closest k points from $\mathcal{L}$; $\mathcal{V}$]

DiskANN [Subramanya+, NeurIPS 19]

Algorithm 1 Beam search

Data: graph G, query q, initial vertex v_0, output size k
Initialization:
V = $\{v_0\}$ // a set of visited vertices
H = $\{v_0 : d(v_0, q)\}$ // a heap of candidates
while *has runtime budget* **do**
  v_i = SelectNearest(H, q)
  for $\hat{v} \in Expand(v_i, G)$ **do**
    if $\hat{v} \notin V$ **then**
      $V := Add(V, \hat{v})$
      $H := Insert(H, \hat{v}, d(\hat{v}, q))$
    end
  end
end
return TopK(V, q, k)

Learning to route [Baranchuk+, ICML 19]

- ➢ All papers have totally different pseudo code for the almost same algorithm 😵
- ➢ Hint: Explicitly state the data structure or not

Full details: https://speakerdeck.com/matsui_528/cvpr23-tutorial-theory-and-applications-of-graph-based-search

Case study: Beam search for neighbor search

Candidates are stored in an array

Algorithm 1. Search-on-Graph(G, p, q, l)

Require: graph G, start node p, query point q, candidate pool size l

Ensure: k nearest neighbors of q

```
i = 0, candidate pool S = ∅
S.add(p)
while i < l do
  i = the id of the first unchecked node p_i in S
  mark p_i as checked
  for all neighbor n of p_i in G do
    if n has not been visited then
      S.add(n)
    end if
  end for
  sort S in ascending order of the distance to q
  if S.size() > l then
    S.resize(l) // remove nodes from back of S to keep its
    size no larger than l
  end if
end while
return the first k nodes in S
```

NSG [Cong+, VLDB 19]

Sort the array explicitly

Candidates are stored in a set

Algorithm 1: GreedySearch(s, x_q, k, L)

Data: Graph G with start node s, query x_q, result size k, search list size $L \geq k$

Result: Result set $\mathcal{L}$ containing k-approx NNs, and a set $\mathcal{V}$ containing all the visited nodes

```
begin
  initialize sets L ← {s} and V ← ∅
  while L \ V ≠ ∅ do
    let p* ← arg min_{p∈L\V} ||x_p − x_q||
    update L ← L ∪ N_out(p*) and
      V ← V ∪ {p*}
    if |L| > L then
      update L to retain closest L
        points to x_q
  return [closest k points from L; V]
```

DiskANN [Subramanya+, NeurIPS 19]

When need to sort, say "closest L points"

Algorithm 1 Beam search

Data: graph G, query q, initial vertex v_0, output size k

Initialization:

V = $\{v_0\}$ // a set of visited vertices

H = $\{v_0 : d(v_0, q)\}$ // a heap of candidates

```
while has runtime budget do
  v_i = SelectNearest(H, q)
  for v̂ ∈ Expand(v_i, G) do
    if v̂ ∉ V then
      V := Add(V, v̂)
      H := Insert(H, v̂, d(v̂, q))
    end
  end
end
return TopK(V, q, k)
```

Learning to route [Baranchuk+, ICML 19]

Candidates are stored in a heap; automatically sorted

- All papers have totally different pseudo code for the almost same algorithm 😵
- Hint: Explicitly state the data structure or not

Full details: https://speakerdeck.com/matsui_528/cvpr23-tutorial-theory-and-applications-of-graph-based-search

Case study: Beam search for neighbor search

Algorithm 1. Search-on-Graph(G, **p**, **q**, l)

Require: graph G, start node **p**, query point **q**, candidate pool size l
Ensure: k nearest neighbors of **q**

```
i = 0, candidate pool S = ∅
S.add(p)
while i < l do
    i = the id of the first unchecked node p_i in S
    mark p_i as checked
    for all neighbor n of p_i in G do
        if n has not been visited then
            S.add(n)
        end if
    end for
    sort S in ascending order of the distance to q
    if S.size() > l then
        S.resize(l) // remove nodes from back of S to keep its
        size no larger than l
    end if
end while
return the first k nodes in S
```

NSG [Cong+, VLDB 19]

Just "check"

Algorithm 1: GreedySearch(s, x_q, k, L)

Data: Graph G with start node s, query x_q, result size k, search list size $L \geq k$
Result: Result set $\mathcal{L}$ containing k-approx NNs, and a set $\mathcal{V}$ containing all the visited nodes

```
begin
    initialize sets L ← {s} and V ← ∅
    while L \ V ≠ ∅ do
        let p* ← arg min_{p ∈ L\V} ||x_p − x_q||
        update L ← L ∪ N_out(p*) and
        V ← V ∪ {p*}
        if |L| > L then
            update L to retain closest L
            points to x_q
    return [closest k points from L; V]
```

DiskANN [Subramanya+, NeurIPS 19]

Checked items are stored in a set

Algorithm 1 Beam search

Data: graph G, query q, initial vertex v_0, output size k
Initialization:
V = $\{v_0\}$ // a set of visited vertices
H = $\{v_0 : d(v_0, q)\}$ // a heap of candidates

```
while has runtime budget do
    v_i = SelectNearest(H, q)
    for v̂ ∈ Expand(v_i, G) do
        if v̂ ∉ V then
            V := Add(V, v̂)
            H := Insert(H, v̂, d(v̂, q))
        end
    end
end
return TopK(V, q, k)
```

Learning to route [Baranchuk+, ICML 19]

- All papers have totally different pseudo code for the almost same algorithm 😵
- Hint: Explicitly state the data structure or not

Full details: https://speakerdeck.com/matsui_528/cvpr23-tutorial-theory-and-applications-of-graph-based-search

Case study: Beam search for neighbor search

Algorithm 1. Search-on-Graph(G, **p**, **q**, l)

Require: graph G, start node **p**, query point **q**, candidate pool size l
Ensure: k nearest neighbors of **q**

```
i = 0, candidate pool S = ∅
S.add(p)
while i < l do
    i = the id of the first unchecked node p_i in S
    mark p_i as checked
    for all neighbor n of p_i in G do
        if n has not been visited then
            S.add(n)
        end if
    end for
    sort S in ascending order of the distance to q
    if S.size() > l then
        S.resize(l) // remove nodes from back of S to keep its
        size no larger than l
    end if
end while
return the first k nodes in S
```

NSG [Cong+, VLDB 19]

Algorithm 1: GreedySearch(s, x_q, k, L)

Data: Graph G with start node s, query x_q, result size k, search list size $L \geq k$
Result: Result set $\mathcal{L}$ containing k-approx NNs, and a set $\mathcal{V}$ containing all the visited nodes

```
begin
    initialize sets L ← {s} and V ← ∅
    while L \ V ≠ ∅ do
        let p* ← arg min_{p∈L\V} ||x_p − x_q||
        update L ← L ∪ N_out(p*) and
         V ← V ∪ {p*}
        if |L| > L then
            update L to retain closest L
             points to x_q
    return [closest k points from L; V]
```

DiskANN [Subramanya+, NeurIPS 19]

Algorithm 1 Beam search

Data: graph G, query q, initial vertex v_0, output size k
Initialization:

```
V = {v_0} // a set of visited vertices
H = {v_0 : d(v_0, q)} // a heap of candidates
while has runtime budget do
    v_i = SelectNearest(H, q)
    for v̂ ∈ Expand(v_i, G) do
        if v̂ ∉ V then
            V := Add(V, v̂)
            H := Insert(H, v̂, d(v̂, q))
        end
    end
end
return TopK(V, q, k)
```

Learning to route [Baranchuk+, ICML 19]

Visited item are simply said to be "visited"; implying an additional hidden data structure (array)

Visited items are stored in a set

- All papers have totally different pseudo code for the almost same algorithm 😮
- Hint: Explicitly state the data structure or not

Full details: https://speakerdeck.com/matsui_528/cvpr23-tutorial-theory-and-applications-of-graph-based-search

Case study: Beam search for neighbor search

Algorithm 1. Search-on-Graph(G, **p**, **q**, l)

Require: graph G, start node **p**, query point **q**, candidate pool size l
Ensure: k nearest neighbors of **q**

```
i = 0, candidate pool S = ∅
S.add(p)
while i < l do
    i = the id of the first unchecked node p_i in S
    mark p_i as checked
    for all neighbor n of p_i in G do
        if n has not been visited then
            S.add(n)
        end if
    end for
    sort S in ascending order of the distance to q
    if S.size() > l then
        S.resize(l) // remove nodes from back of S to keep its
        size no larger than l
    end if
end while
return the first k nodes in S
```

NSG [Cong+, VLDB 19]

Algorithm 1: GreedySearch(s, x_q, k, L)

Data: Graph G with start node s, query x_q, result size k, search list size $L \geq k$
Result: Result set $\mathcal{L}$ containing k-approx NNs, and a set $\mathcal{V}$ containing all the visited nodes

```
begin
    initialize sets L ← {s} and V ← ∅
    while L \ V ≠ ∅ do
        let p* ← arg min_{p∈L\V} ||x_p − x_q||
        update L ← L ∪ N_out(p*) and
        V ← V ∪ {p*}
        if |L| > L then
            update L to retain closest L
```

$\mathcal{L}$; $\mathcal{V}$]

S 19]

Algorithm 1 Beam search

Data: graph G, query q, initial vertex v_0, output size k
Initialization:
V = $\{v_0\}$ // a set of visited vertices
H = $\{v_0 : d(v_0, q)\}$ // a heap of candidates

```
while has runtime budget do
    v_i = SelectNearest(H, q)
    for v̂ ∈ Expand(v_i, G) do
        if v̂ ∉ V then
            V := Add(V, v̂)
            H := Insert(H, v̂, d(v̂, q))
        end
    end
end
return TopK(V, q, k)
```

Learning to route [Baranchuk+, ICML 19]

Termination condition??

- All papers have totally different pseudo code for the almost same algorithm 😵
- Hint: Explicitly state the data structure or not

Full details: https://speakerdeck.com/matsui_528/cvpr23-tutorial-theory-and-applications-of-graph-based-search

Case study

```
function CSGVERTICES
    Input: 𝒱, ℱ: set of vertices and facets of input polyhedra
    Output: 𝒱_f: corresponding set of final output vertices
    for F_1 in ℱ do                                                  ▷ Enumerate input faces
        for v in F_1 do                                              ▷ Order-1 candidates
            if ISFINAL1(v) then 𝒱_f ← 𝒱_f ∪ {v}
        end for
        for F_2 in ℱ do
            v_1,v_2 ← INTERSECT2FACETS(F_1,F_2)                      ▷ Order-2 candidates
            if {v_1, v_2} = ∅ then continue F_2 loop                 ▷ No intersection
            if ISFINAL2(v_1) then 𝒱_f ← 𝒱_f ∪ {v_1}
            if ISFINAL2(v_2) then 𝒱_f ← 𝒱_f ∪ {v_2}
            for F_3 in ℱ do                                          ▷ Order-3 candidates
                v ← INTERSECTSEGMENTFACET(v_1, v_2, F_3)
                if v ≠ ∅ and ISFINAL3(v) then 𝒱_f ← 𝒱_f ∪ {v}
            end for
        end for
    end for
end function
```

M. Douze+, “QuickCSG: Fast Arbitrary Boolean Combinations of N Solids”, arXvi 2017.

Case study

Enumeration for a set

Comments

function CSGVERTICES

Input: $\mathcal{V}$, $\mathcal{F}$: set of vertices and facets of input polyhedra

Output: $\mathcal{V}_f$: corresponding set of final output vertices

$\boxed{\textbf{for } F_1 \textbf{ in } \mathcal{F} \textbf{ do}}$ ▷ Enumerate input faces

  for v **in** F_1 **do** ▷ Order-1 candidates

    if ISFINAL1(v) **then** $\boxed{\mathcal{V}_f \leftarrow \mathcal{V}_f \cup \{v\}}$

  end for

  for F_2 **in** $\mathcal{F}$ **do** ▷ Order-2 candidates

    $\boxed{v_1, v_2 \leftarrow}$ INTERSECT2FACETS(F_1, F_2)

    if $\{v_1, v_2\} = \emptyset$ **then continue** F_2 **loop** ▷ No intersection

    if ISFINAL2(v_1) **then** $\mathcal{V}_f \leftarrow \mathcal{V}_f \cup \{v_1\}$

    if ISFINAL2(v_2) **then** $\mathcal{V}_f \leftarrow \mathcal{V}_f \cup \{v_2\}$

    fo... **in** $\mathcal{F}$ **do** ▷ Order-3 candidates

      ... $\boxed{\text{INTERSECTSEGMENTFACET}}$($v_1$, v_2, F_3)

      ... $\neq \emptyset$ **and** ISFINAL3(v) **then** $\mathcal{V}_f \leftarrow \mathcal{V}_f \cup \{v\}$

    e...

  end...

end fo...

end funct...

Set update

¥textsc for a function name

Returning two variables
(This is often strange in a usual math,
but it's ok in pseudo-codes)

M. Douze+, "QuickCSG: Fast Arbitrary Boolean Combinations of N Solids", arXvi 2017.

OOP style?

Algorithm 2: **Lookup**($\mathcal{T}$, k)

Input: $\mathcal{T}$: the LIPP index, k: the lookup key
Output: *isFound*: indicates whether k is found,
e: the entry containing k if found

```
begin
    n ← the root node of T;
    e ← n.E[n.M(k)];
    while type(e) == NODE do
        n ← the node pointed to from entry e;
        e ← n.E[n.M(k)];
    if type(e) == DATA then
        k' ← the key in entry e;
        if k == k' then
            return ⟨True, e⟩;
    return ⟨False, e⟩;
end
```

- It's ok to define a class as well
- Member variables can be accessed via a usual "." notation.
- But don't make the thing complex.

Wu+, "Updatable Learned Index with Precise Positions", VLDB 2021

Equations

- Basic notation for variables
- String
- Tips for sets
- Inputs/outputs of functions
- Subscript/superscript
- Don't write numpy
- Misc

Pseudo codes

- Basic
- What do you want to express?
- Misc

Misc

- Consider what are global variables!

- You can use a sentence to describe a difficult concept, e.g.,
  - $\mathcal{J} \leftarrow$ Identifiers of top K smallest values of $\boldsymbol{x}$
  - $\mathrm{T} \leftarrow (N_1 \times N_2 \times N_3)$ empty arrays of B-Trees.

- Intentionally, you can use **almost-code** style.

Algorithm 1 SimSiam Pseudocode, PyTorch-like

```
# f: backbone + projection mlp
# h: prediction mlp

for x in loader: # load a minibatch x with n samples
   x1, x2 = aug(x), aug(x) # random augmentation
   z1, z2 = f(x1), f(x2) # projections, n-by-d
   p1, p2 = h(z1), h(z2) # predictions, n-by-d

   L = D(p1, z2)/2 + D(p2, z1)/2 # loss

   L.backward() # back-propagate
   update(f, h) # SGD update

def D(p, z): # negative cosine similarity
   z = z.detach() # stop gradient

   p = normalize(p, dim=1) # l2-normalize
   z = normalize(z, dim=1) # l2-normalize
   return -(p*z).sum(dim=1).mean()
```

- Their contribution is it's easy to implement the algorithm in PyTotch
- Complicated idea, such as "detach", can be explained in one line.
- But be careful! Can future readers understand?

Chen and He, "Exploring Simple Siamese Representation Learning", CVPR 2021

Misc

- ➢ If you use almost-code style, use the minted package.
- ➢ https://www.overleaf.com/learn/latex/Code_Highlighting_with_minted

```latex
\documentclass{article}
\usepackage{minted}
\begin{document}
\begin{minted}{python}
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)
```

➡

```python
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
```

Schedule

Date (2024)	Contents	Presented by
~~Week 1, Apr 10~~	~~Introduction. Review of fundamental concepts~~	~~Yusuke, Koya, Yuki, Jun~~
~~Week 2, Apr 17~~	~~Equations and pseudo-codes~~	~~Yusuke Matsui~~
Week 3, Apr 24	Presentation	Koya Narumi
Week 4, May 1	Tables and plots	Yusuke Matsui
Week 5, May 8	Figures	Koya Narumi
Week 6, May 22	Videos	Koya Narumi
Week 7, May 29	Invited Talk 1	Dr. Yoshiaki Bando (AIST)
Week 8, June 5	Invited Talk 2	Prof. Katie Seaborn (Tokyo Tech)
Week 9, June 12	GitHub in depth	Yusuke Matsui
Week 10, June 19	Automation of research and research dissemination (Web, Cloud, CI/CD)	Jun Kato
Week 11, June 26	Research community	Jun Kato
Week 12, July 3	3DCG illustrations	Yuki Koyama
Week 13, July 10	Final presentations	-