Lecture 1

Introduction

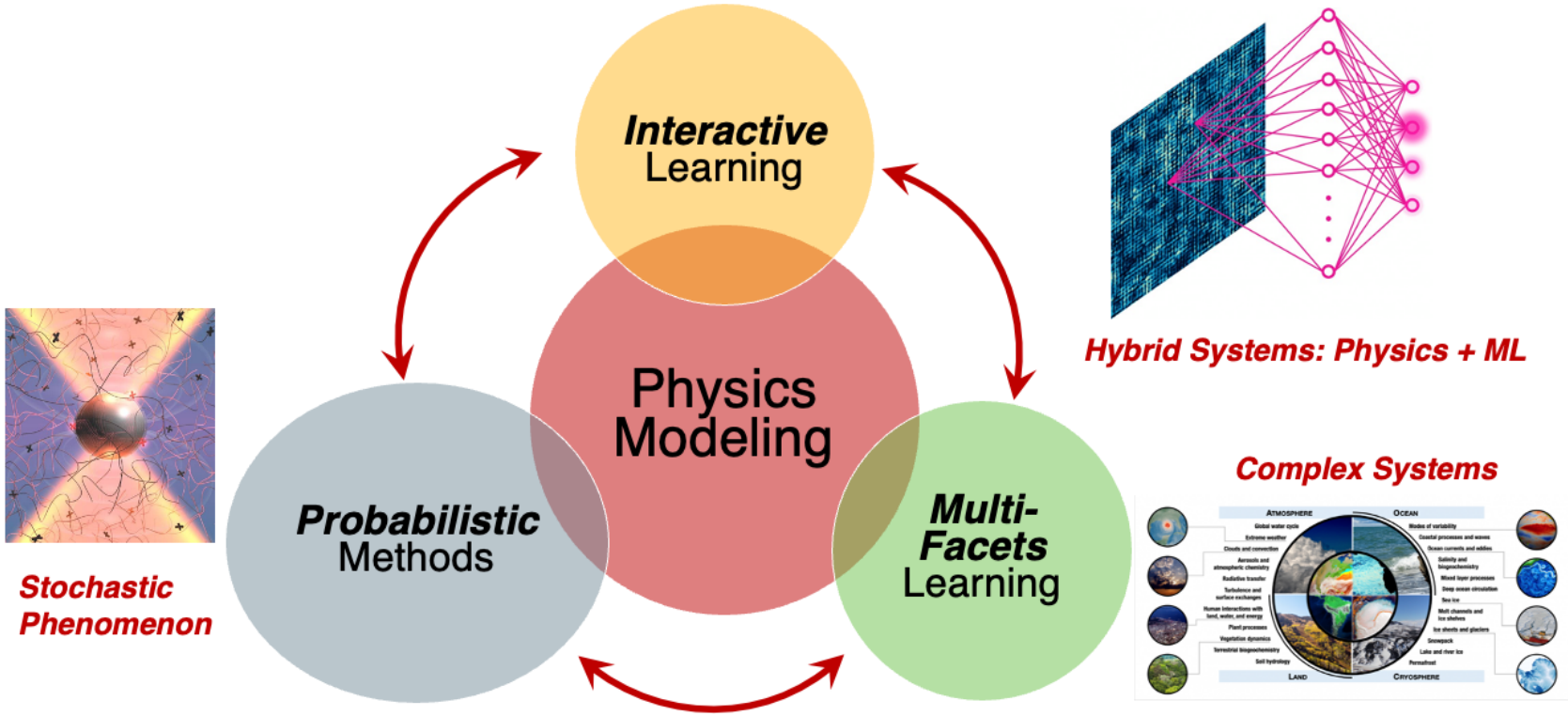
Instructor: Shibo Li

shiboli@cs.fsu.edu

Department of Computer Science
Florida State University

About me

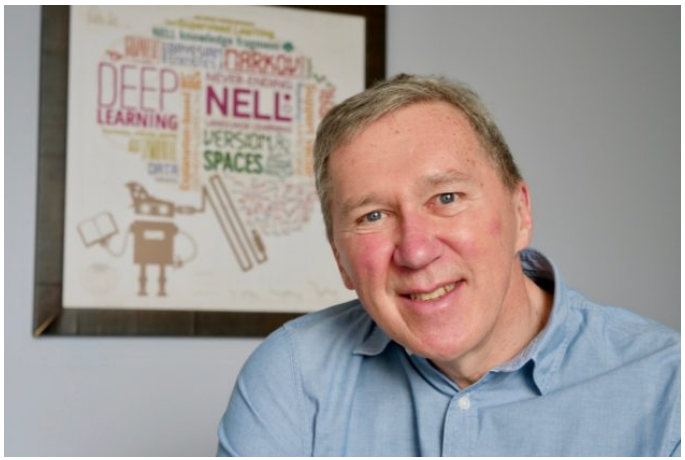
- **Shibo Li**, shiboli@cs.fsu.edu
- Probabilistic Machine Learning
- Assistant Professor, Department of Computer Science



What is Machine Learning?

*"A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**."*

– Tom M. Mitchell, Machine Learning (1997)

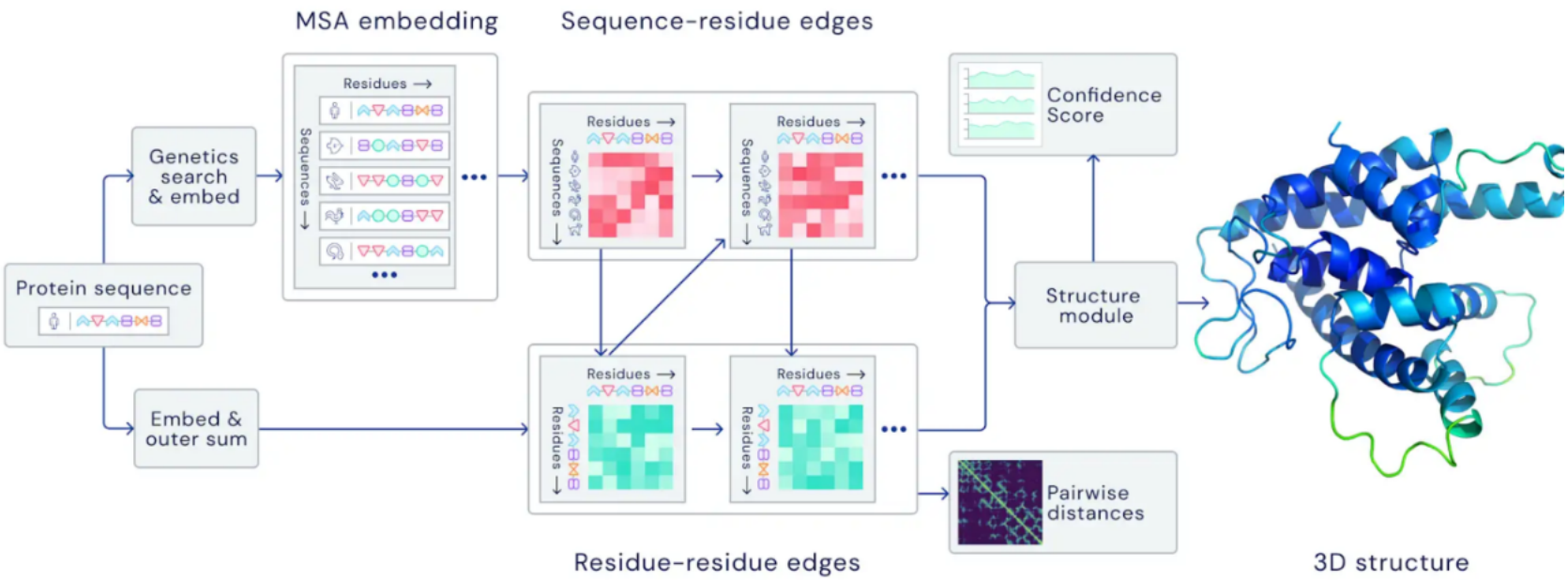


Machine Learning is the driving force of AI!

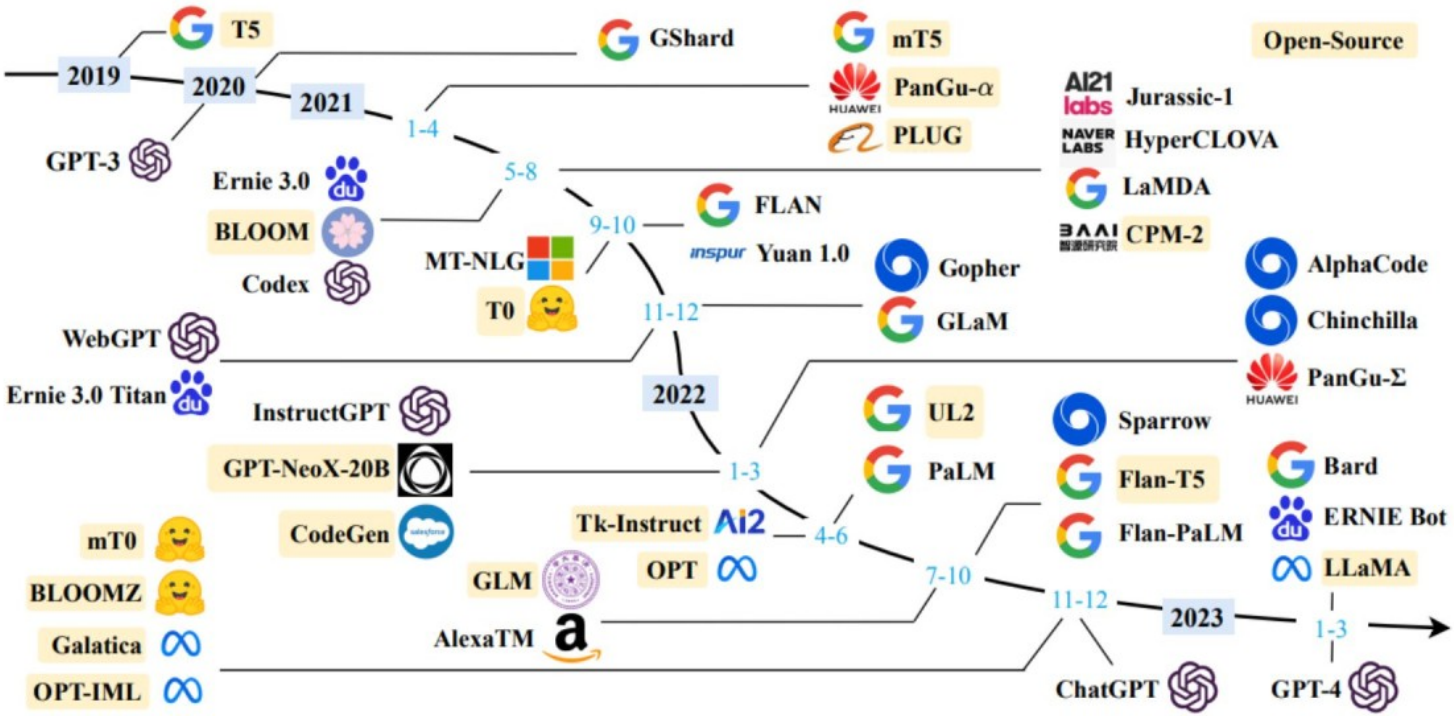
Alpha-Go


柯洁 KE JIE
00:13:28
ALPHAGO
01:30:35
中国围棋协会
Google
浙江省体育局
中国乌镇 围棋峰会
Future of Go Summit in Wuzhen

Alpha-Fold


MSA embedding
Sequence-residue edges
Genetics search & embed
Protein sequence
Embed & outer sum
Residues →
Sequences
Residue-residue edges
Confidence Score
Structure module
Pairwise distances
3D structure

Large-Language Models



Generative Models (GenAI)

Sprouts in the shape of text 'Imagen' coming out of a fairytale book.

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

A high contrast portrait of a very happy fuzzy panda dressed as a chef in a high end kitchen making dough. There is a painting of flowers on the wall behind him.

Teddy bears swimming at the Olympics 400m Butterfly event.

A cute corgi lives in a house made out of sushi.

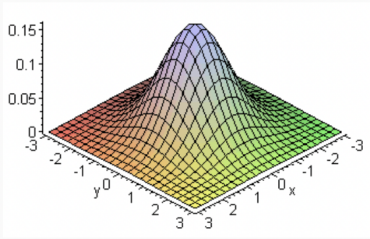
A cute sloth holding a small treasure chest. A bright golden glow is coming from the chest.

Machine Learning is EVERYWHERE!

What is Probabilistic Learning

In a nutshell, probabilistic learning is branch of ML that uses probabilistic (or Bayesian) principles for model design and algorithm development.!

What is Probabilistic Learning

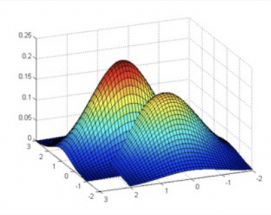


Prior Distribution

$$p(\boldsymbol{\theta})$$

Data Likelihood

$$p(\mathcal{D}|\boldsymbol{\theta})$$



Posterior Distribution

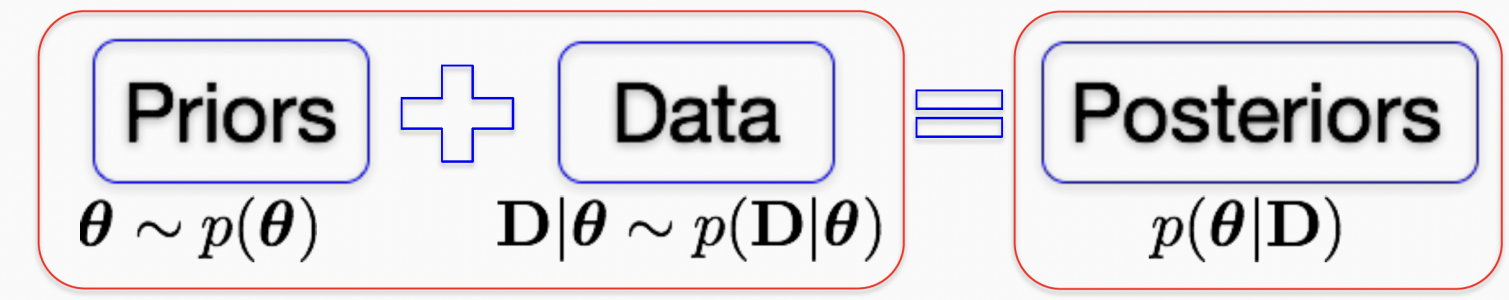
$$p(\boldsymbol{\theta}|\mathcal{D})$$

Baye's Rule:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\boldsymbol{\theta}, \mathcal{D})}{p(\mathcal{D})} = \frac{p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})}{\int p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}}$$

Why Probabilistic Learning

- Unified, principled mathematical framework



- Uncertainty reasoning

Asthma: 60%

Heart disease: 30%

Healthy: 10%

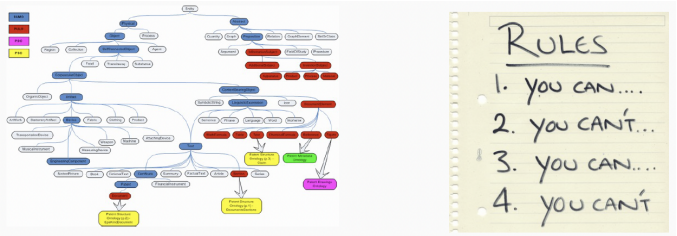
Raining: 70%

Sunny: 30%

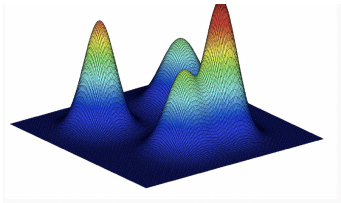
How Important is the Uncertainty?

Figure: "Neither Autopilot nor the driver noticed the white side of the tractor trailer against a brightly lit sky, so the brake was not applied. The high ride height of the trailer combined with its positioning across the road and the extremely rare circumstances of the impact caused the Model S to pass under the trailer, with the bottom of the trailer impacting the windshield," said Tesla in a statement after the crash last year.

- **Modeling**



Complex Knowledge/Assumptions



Valid Prior Distribution

- **Calculation**

$$p(\boldsymbol{\theta}|\mathbf{D}) = \frac{p(\boldsymbol{\theta})p(\mathbf{D}|\boldsymbol{\theta})}{\int p(\boldsymbol{\theta})p(\mathbf{D}|\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}}$$

High dimensional integration

- MCMC sampling
- Variational approximations
- Belief propagation

We will cover both the classical and state-of-the-art approaches to deal with these challenges!

Warning

- This course is **math intensive** and requires **a certain level of programming** (with Matlab, R or Python). Python components may require TensorFlow and/or PyTorch. The coding workload is not heavy, but requests **mathematical derivations and careful debugging**.
- The workload is heavy (6-10 hours per-week)

How will you learn?

- Attendance is **Required**. Take classes to follow the math, understand the models and algorithms
- **Derive the math details by yourself!**
- Finish the homework assignments to deepen your understanding
- **Implement and debug the models and algorithms by yourself!**
- Using **proper help is encouraged!** (ChatGPT, Claude, StackOverflow, MathStacks...). **But plagiarism is prohibited!**

Overview of This Course

This course focuses on the mathematic foundations, modeling and algorithmic ideas in probabilistic learning

This course is **not** about

- Applying ML to specific tasks (e.g., image tagging and autonomous driving)
- Using specific ML tools/libraries, e.g., scikit-learn and PyTorch
- How to program and debug, e.g., with Python, R or Matlab

Overview of This Course

This course is an **advanced** course for students who want to study ML **in depth** or quickly get to the **frontier** research of probabilistic learning

This course is **not** about a preliminary course, *e.g.*, entry-level introduction of statistics. That means,

The content can be hard for some ones

Do NOT take this course if

- You are struggling with linear algebra, calculus or basic statistical concepts
- You are sick of mathematical symbols, derivations, proofs and calculations
- You do NOT feel good in programming and debugging

We Assume

- You are **not** scared of math, statistics, calculus and calculations; you are happy with them!
- You are **comfortable** with abstract symbols and matrices operations
- You can pick-up Matlab/Python/R quickly (even if you have never used them before)
- You enjoy debugging, step in, step out, print, etc.
- You can quickly learn how to use TensorFlow or PyTorch or Jax by following the documentation and searching for the online examples

and Most Important

- You have planed for **enough efforts** for this class (e.g., 6-10 hours per-week)

You feel **NOT** right about any of these assumptions

- Seriously consider whether to take this course
- We want you to succeed. We do not want to make you feel tortured.

Course Information

- The course website contains all the detailed information
- The course website is linked to my homepage
  - My homepage: `https://imshibo.com/`
  - Course Website: `https://cis5930.github.io/`

Note: this review is neither compressive nor in depth. Due to time limit, this review is just to point out **key concepts and computational rules as the guidance.** We list the references for you to check out details for future usage.

- Standard notations
  - non-bold letters: scalars

$$a, b, x, y, B, D, G, \alpha, \gamma, \ldots$$

  - Bold small letters: vectors

$$\mathbf{a}, \mathbf{b}, \mathbf{x}, \mathbf{y}, \boldsymbol{\gamma}, \boldsymbol{\eta}, \ldots$$

  - Bold capital: matrices

$$\mathbf{A}, \mathbf{X}, \mathbf{Z}, \boldsymbol{\Gamma}, \ldots$$

- scalar input, scalar output

$$y(x + \mathrm{d}x) = y(x) + a \cdot \mathrm{d}x + (\text{high-order terms})$$

$$\frac{\partial y}{\partial x} = a \longleftrightarrow \mathrm{d}y = a \cdot \mathrm{d}x$$

- vector input, scalar output

$$\mathbf{x} = (x_1, \ldots, x_n)^\top, \quad \mathrm{d}\mathbf{x} = (\mathrm{d}x_1, \ldots, \mathrm{d}x_n)^\top$$

$$y(\mathbf{x} + \mathrm{d}\mathbf{x}) = y(\mathbf{x}) + \mathbf{a}\mathrm{d}\mathbf{x} + (\text{high-order terms})$$

We use row-vector to represent gradient

$$\frac{\partial y}{\partial \mathbf{x}} = \left(\frac{\partial y}{\partial x_1}, \ldots, \frac{\partial y}{\partial x_n}\right) = \mathbf{a} \longleftrightarrow \mathrm{d}y = \mathbf{a}\mathrm{d}\mathbf{x}$$

- In general, vector input, vector output

$$\mathbf{y} = (y_1, \ldots, y_m)^\top \qquad \mathbf{x} = (x_1, \ldots, \mathbf{x}_n)^\top, \quad \mathrm{d}\mathbf{x} = (\mathrm{d}x_1, \ldots, \mathrm{d}x_n)^\top$$

$$\mathbf{y}(\mathbf{x} + \mathrm{d}\mathbf{x}) = \mathbf{y}(\mathbf{x}) + \mathbf{A}\mathrm{d}\mathbf{x} + (\text{high order terms})$$

What is this? What's size? $m \times n$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} = \mathbf{A} \qquad \longleftrightarrow \qquad \mathrm{d}\mathbf{y} = \mathbf{A}\mathrm{d}\mathbf{x}$$

$$y(x + \mathrm{d}x) = y(x) + a \cdot \mathrm{d}x + (\text{high-order terms})$$

$$y(\mathbf{x} + \mathrm{d}\mathbf{x}) = y(\mathbf{x}) + \mathbf{a}\mathrm{d}\mathbf{x} + (\text{high-order terms})$$

$$\mathbf{y}(\mathbf{x} + \mathrm{d}\mathbf{x}) = \mathbf{y}(\mathbf{x}) + \mathbf{A}\mathrm{d}\mathbf{x} + (\text{high order terms})$$

In all the cases, {a, **a**, **A**} are derivatives. We define (partial) gradient as the derivative

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{A}$$

This is consistent with the definition of Jacobian. However, we need to be aware if output is scalar, the gradient is a row vector

- What is the benefit of this notation? We can apply the chain-rule in a natural way

$$\mathbf{y} = \mathbf{f}(\mathbf{x}), \quad \mathbf{x} = \mathbf{g}(\mathbf{z})$$

$$\mathbf{y} : m \times 1 \qquad \mathbf{x} : n \times 1 \qquad \mathbf{z} : q \times 1$$

$$\boxed{\frac{\partial \mathbf{y}}{\partial \mathbf{z}}} = \boxed{\frac{\partial \mathbf{y}}{\partial \mathbf{x}}} \cdot \boxed{\frac{\partial \mathbf{x}}{\partial \mathbf{z}}}$$

$m \times q$ $\quad m \times n$ $\quad n \times q$

- Some literature uses the notation of derivative transpose

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{A}^\top$$

The benefit is for scalar y, the gradient is a column vector. The cons is when doing the chain rule, you have to multiply from right to left. Why?

Matrix/Vector Derivative

- In whichever case, the key to derive/compute the derivative!

$$\mathbf{y}(\mathbf{x}+\mathrm{d}\mathbf{x}) = \mathbf{y}(\mathbf{x}) + \mathbf{A}\mathrm{d}\mathbf{x} + (\text{high order terms})$$

$$\mathrm{d}\mathbf{y} = \mathbf{A}\mathrm{d}\mathbf{x}$$

- The general idea: recursively apply the chain rule to get the target derivative!