DSCI 565: PRELIMINARIES

Ke-Thia Yao

Lecture 2: 2025 August 27

Tensor

- For machine learning, tensor is just a **n-dimensional array**
- In physics and engineering, tensor has other interpretations
  - For example, in physics for each point a physical object the stress may have to be represent as 3-d matrix (i.e., stress tensor).
    Tensors over multiple points is called tensor field, but people sometimes just call it tensor

$$x$$

scalar

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

vector

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}$$

matrix

0-dim

1-dim

2-dim

Tensor

- Need to use tensors, because data is high-dimensional
  - An image is three-dimensional height, width and depth (3 colors)
  - A batch of images is four-dimensional
  - A video is four-dimensional height, width and time 5-dimension
  - A batch of videos is five-dimensional

- See notebook: chapter_preliminaries/ndarray.ipynb

Data Preparation

- Numerical packages, like PyTorch and NumPy, are designed for efficiency in terms of speed and memory usage
- They assume all the tensor elements are numbers, which is not the case for real-world data
- The Pandas library is designed data wrangling
- We will not cover Pandas beyond this Jupyter notebook
  - chapter_preliminaries/pandas.ipynb
- A good Pandas book by the creator of Pandas is:
  - McKinney, Wes. 2022. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter*. 3rd edition. O'Reilly Media.

Linear Algebra

- We can define operations over tensors. Two basic operations are:

addition , scalar multiplication

  - Addition

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \boldsymbol{x} + \boldsymbol{y} = \begin{bmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{bmatrix}$$

  - Scalar multiplication

$$\alpha \boldsymbol{x} = \begin{bmatrix} \alpha x_1 \\ \vdots \\ \alpha x_n \end{bmatrix}$$

- Vector Space: Set of objects with defined addition and scalar multiplication and satisfies some minor axioms (such as having element $\mathbf{0} = \boldsymbol{x} + (-\boldsymbol{x})$)
  - n-dimensional tensors are vector spaces

Other Operations

- Element-wise product or Hadamard product

$$\boldsymbol{x} \odot \boldsymbol{y} = \begin{bmatrix} x_1 * y_1 \\ \vdots \\ x_n * y_n \end{bmatrix}$$

Element wise product
Hadamard product

- Summation

$$\sum_{i=1}^{n} x_i \, , \qquad \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}$$

Dot Product (aka Inner Product)

Dot Product

$x^T y = \sum_{i=1}^{d} x_i y_i$

Given two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$

$x^T y$

$$\boldsymbol{x}^T \boldsymbol{y} = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{d} x_i y_i$$

- Dot product have many uses → interpret as weighted average (weigh sum is 1)
  - If $\boldsymbol{x}$ is a weight vector, then the dot product is weighted average of elements of $\boldsymbol{y}$.
  - The length of projection of $\boldsymbol{x}$ onto $\boldsymbol{y}$ is $\|\boldsymbol{x}\| \cos(\theta)$, where $\cos(\theta) = \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|}$

Matrix Vector

$a_{1\,n\times 1}$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_1{}^\top \\ \boldsymbol{a}_2{}^\top \\ \vdots \\ \boldsymbol{a}_m{}^\top \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\mathbf{b} = A\boldsymbol{x} \qquad b_i = \sum_{j=1}^{n} a_{ij} x_j \qquad ik, k \to i$$

$$A = \begin{bmatrix} \boldsymbol{a}_1{}^\top \\ \boldsymbol{a}_2{}^\top \\ \vdots \\ \boldsymbol{a}_m{}^\top \end{bmatrix} \qquad A\boldsymbol{x} = \begin{bmatrix} \boldsymbol{a}_1{}^\top \\ \boldsymbol{a}_2{}^\top \\ \vdots \\ \boldsymbol{a}_m{}^\top \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{a}_1^\top \boldsymbol{x} \\ \boldsymbol{a}_2^\top \boldsymbol{x} \\ \vdots \\ \boldsymbol{a}_m^\top \boldsymbol{x} \end{bmatrix}$$

Matrix-Matrix Multiple

$AB = C$ $\quad c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$

Say we have two matrices $\mathbf{A} \in \mathbb{R}^{n \times \mathrm{p}}$ and $\mathbf{B} \in \mathbb{R}^{p \times m}$

$$\mathbf{A} = \underset{n \times k}{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nk} \end{bmatrix}} \quad \mathbf{B} = \underset{k \times m}{\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{k1} & b_{k2} & \cdots & b_{km} \end{bmatrix}}$$

Then $\mathbf{C} \in \mathbb{R}^{n \times m}$

$$\mathbf{C} = \mathbf{AB} \qquad c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj} \qquad ik, kj \rightarrow ij$$

Matrix-Matrix Multiple

Say we have two matrices $A \in \mathbb{R}^{n \times p}$ and $B \in \mathbb{R}^{p \times m}$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{np} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{p1} & b_{p2} & \cdots & b_{pm} \end{bmatrix}$$

$$\mathbf{C} = \mathbf{A}\mathbf{B} = \begin{bmatrix} \mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \vdots \\ \mathbf{a}_n^\top \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_m \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^\top \mathbf{b}_1 & \mathbf{a}_1^\top \mathbf{b}_2 & \cdots & \mathbf{a}_1^\top \mathbf{b}_m \\ \mathbf{a}_2^\top \mathbf{b}_1 & \mathbf{a}_2^\top \mathbf{b}_2 & \cdots & \mathbf{a}_2^\top \mathbf{b}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_n^\top \mathbf{b}_1 & \mathbf{a}_n^\top \mathbf{b}_2 & \cdots & \mathbf{a}_n^\top \mathbf{b}_m \end{bmatrix}$$

Norms

- Norms measurement the magnitude of a vector
- l_2 norm

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

l_1 norm

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} abs(x_i)$$

- l_p norm

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

Norms

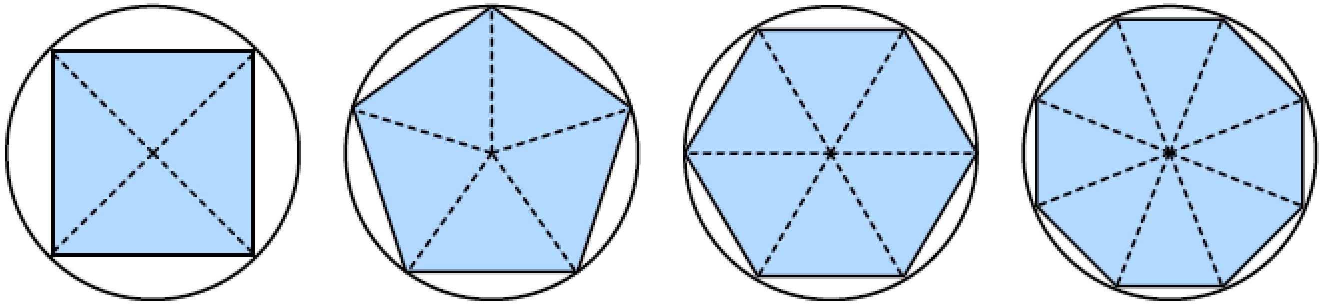
- Frobenius Norm for matrices

$$\|X\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^2}$$

- See Notebook: chapter_preliminaries/linear-algebra.ipynb

Calculus

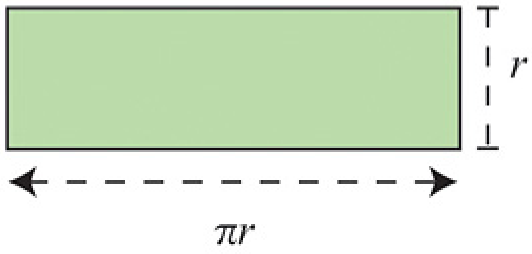
- Idea of limits: approximate area of a circle by n triangles

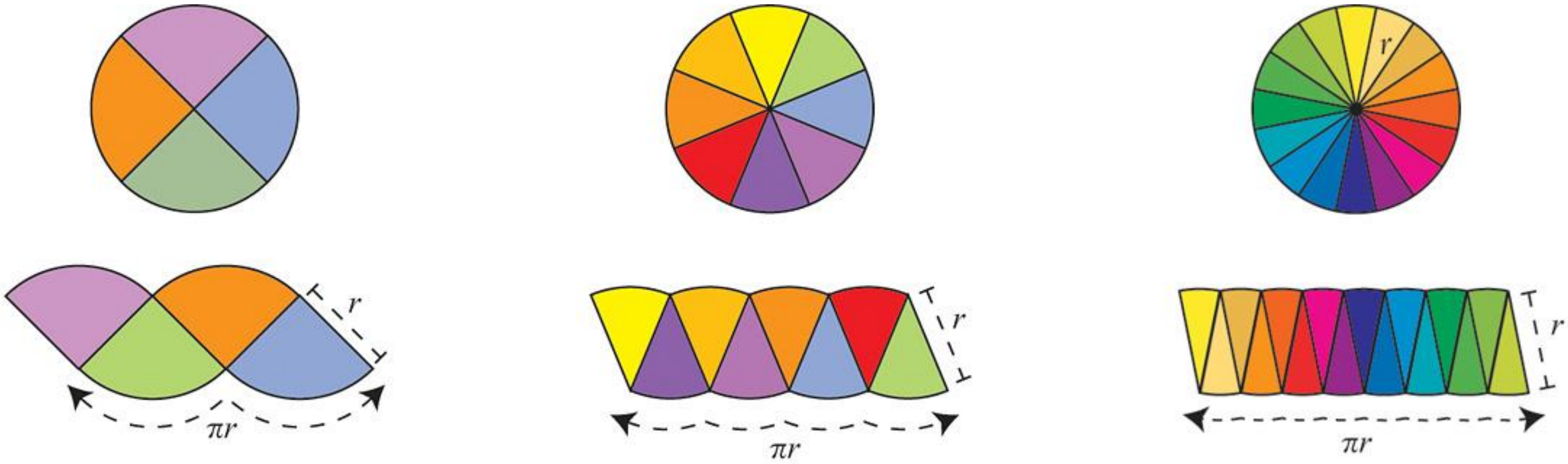


$$n \times \frac{r}{2} \times \frac{2\pi r}{n}$$

Calculus

- Approximate circle area by a rectangle



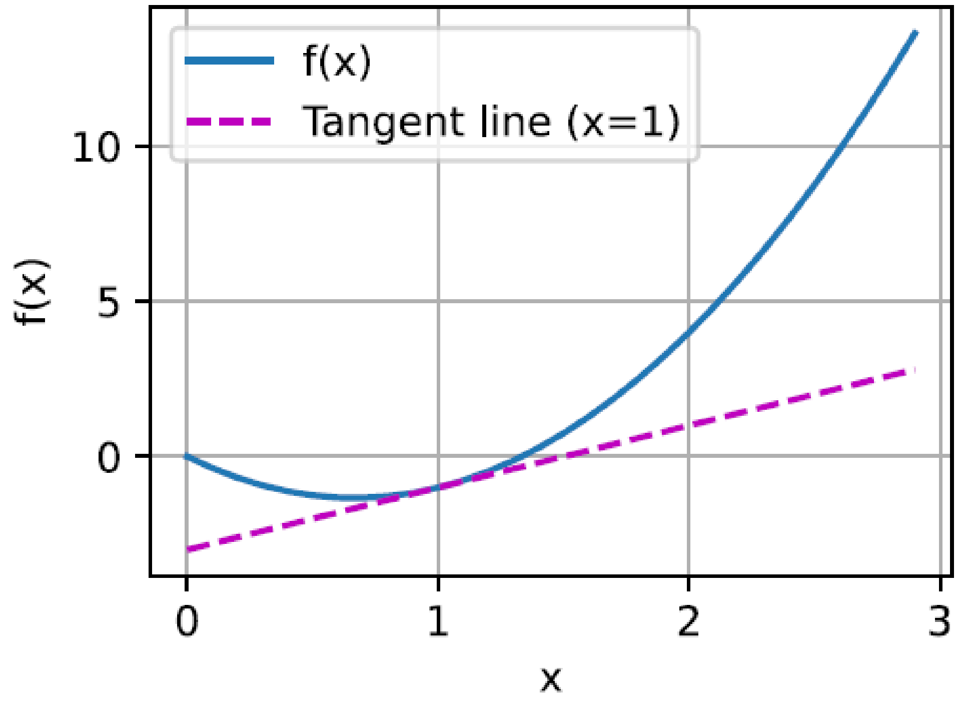


https://archive.nytimes.com/opinionator.blogs.nytimes.com/2010/04/04/take-it-to-the-limit/

Derivatives

- Derivative of a function $f(x)$

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$



- Other ways to write derivatives

Given $y = f(x)$

$$f'(x) = y' = \frac{dy}{dx} = \frac{df}{dx} = \frac{d}{dx} f(x) = Df(x) = D_x f(x)$$

Computing Derivatives Numerically

- Say we want to know $f'(10)$
  - Evaluate f twice: $f(10)$ and $f(10+h)$
- For h in $[0.01, 0.001, 0.0001, 0.00001]$
  - Compute $\frac{f(10+h)-f(10)}{h}$

Compute Derivatives Symbolically

- This is what you learned in Calculus class

$\frac{d}{dx}C = 0$	for any constant C	$\frac{d}{dx}[Cf(x)] = C\frac{d}{dx}f(x)$	Constant multiple rule
$\frac{d}{dx}x^n = nx^{n-1}$	for $n \neq 0$	$\frac{d}{dx}[f(x)+g(x)] = \frac{d}{dx}f(x) + \frac{d}{dx}g(x)$	Sum rule
$\frac{d}{dx}e^x = e^x$		$\frac{d}{dx}[f(x)g(x)] = f(x)\frac{d}{dx}g(x) + g(x)\frac{d}{dx}f(x)$	Product rule
$\frac{d}{dx}\ln x = x^{-1}$		$\frac{d}{dx}\frac{f(x)}{g(x)} = \frac{g(x)\frac{d}{dx}f(x) - f(x)\frac{d}{dx}g(x)}{g^2(x)}$	Quotient rule

- Example:

$$\frac{d}{dx}[3x^2 - 4x] = 3\frac{d}{dx}x^2 - 4\frac{d}{dx}x = 6x - 4.$$

Chain Rule for Composition of Functions

- Given $y = f(g(x))$ decompose to underlying functions

  $y = f(u)$ and $u = g(x)$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

- Or, sometime people write

$$\frac{dy}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

Chain Rule (cont)

- Given $y = h(f(x), g(x))$ then
$$\frac{dy}{dx} = \frac{dh}{df}\frac{df}{dx} + \frac{dh}{dg}\frac{dg}{dx}$$
- Use chain rule to derive:
  - Sum rule: let $h(f, g) = f + g$
  - Product rule: let $h(f, g) = f * g$
  - Quotient rule: let $h(f, g) = f/g$

Partial Derivatives

- Given multivariate function $y = f(x_1, x_2, \dots, x_n)$

$$\frac{\partial y}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, \dots, x_{i-1}, x_i + h, x_{i+1}, \dots, x_n) - f(x_1, \dots, x_i, \dots, x_n)}{h}$$

- Other ways to write partial derivatives

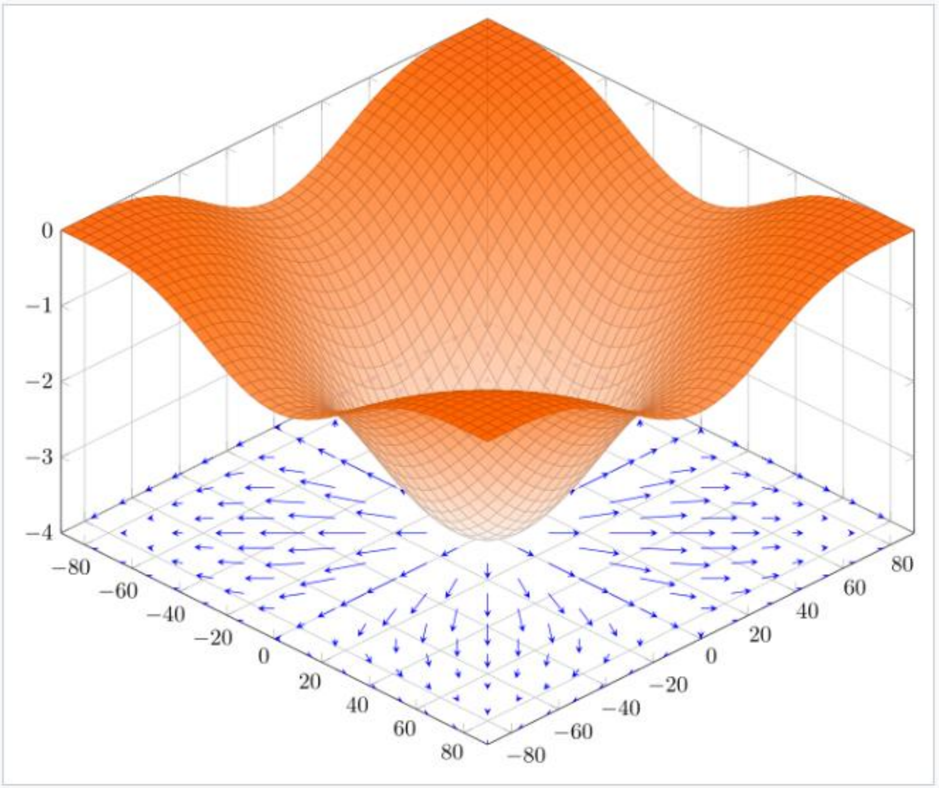
$$\frac{\partial y}{\partial x_i} = \frac{\partial f}{\partial x_i} = \partial_{x_i} f = \partial_i f = f_{x_i} = f_i = D_i f = D_{x_i} f.$$

Gradient : direction of greatest change

- Partial derivative wrt to each variable of $f(x_1, x_2, \ldots, x_n)$
- Gradient

$$\nabla_{\mathrm{x}} \mathrm{f(x)} = \left[\partial_{\mathrm{x}_1} \mathrm{f(x)},\ \partial_{\mathrm{x}_2} \mathrm{f(x)},\ \ldots \partial_{\mathrm{x}_n} \mathrm{f(x)}\right]^{\mathrm{T}}$$

- Gradient represent the direction of greatest change



The gradient of the function $f(x,y) = -(\cos^2 x + \cos^2 y)^2$ depicted as a projected vector field on the bottom plane.

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

Chain Rule for Multivariate Functions

- Suppose $y = f(\boldsymbol{u})$ has variable $u_1, u_2, \dots, u_m$ where $u_i = g_i(\boldsymbol{x})$ has variables $x_1, x_2, \dots, x_n$, i.e., $\boldsymbol{u} = g(\boldsymbol{x})$
- Chain rule

$$\frac{\partial y}{\partial x_i} = \frac{\partial y}{\partial u_1}\frac{\partial u_1}{\partial x_i} + \frac{\partial y}{\partial u_2}\frac{\partial u_2}{\partial x_i} + \cdots + \frac{\partial y}{\partial u_m}\frac{\partial u_m}{\partial x_i}$$

- Rewriting using vectors

$$\frac{\partial y}{\partial x_i} = \left(\frac{\partial u_1}{\partial x_i}, \dots, \frac{\partial u_m}{\partial x_i}\right)\begin{pmatrix}\frac{\partial y}{\partial u_1} \\ \vdots \\ \frac{\partial y}{\partial u_m}\end{pmatrix}$$

Chain Rule for Multivariate Functions

- Gradient for multivariate functions

$$\nabla_x y = \begin{pmatrix} \frac{\partial u_1}{\partial x_1} & \cdots & \frac{\partial u_m}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial u_1}{\partial x_n} & \cdots & \frac{\partial u_m}{\partial x_n} \end{pmatrix} \begin{pmatrix} \frac{\partial y}{\partial u_1} \\ \vdots \\ \frac{\partial y}{\partial u_m} \end{pmatrix} = J^T \nabla_{\boldsymbol{u}} y$$

- Where J is called the Jacobian Matrix

Automatic Differentiation

- Problem with previous methods

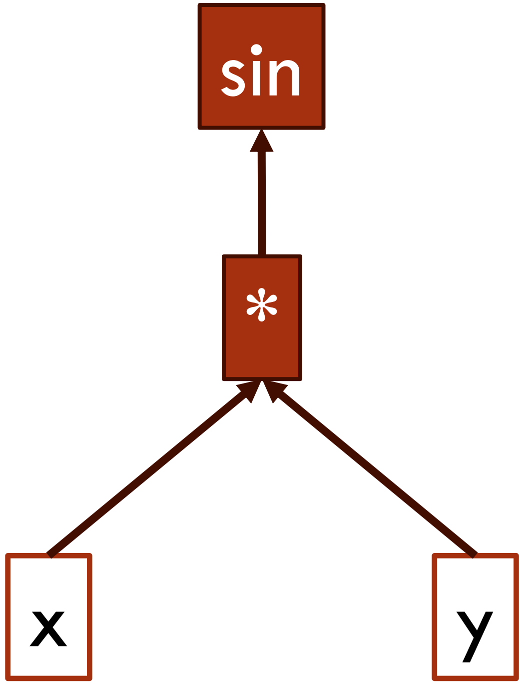
Symbolic differenciation & Numeric differenciation

  - Symbolic differentiation is too difficult if the function is a computer program
  - Numerical differentiation requires discretization which introduce numerical errors
  - Both methods are slow for multivariate functions
- Key observation: computer program is composed of elementary function
  - From elementary functions (exp, log, sin, cos)
  - Using addition, multiplication, division and function composition
- But we know how to differentiate elementary functions, and we have differential rules (sum, product, quotient, chain rules)

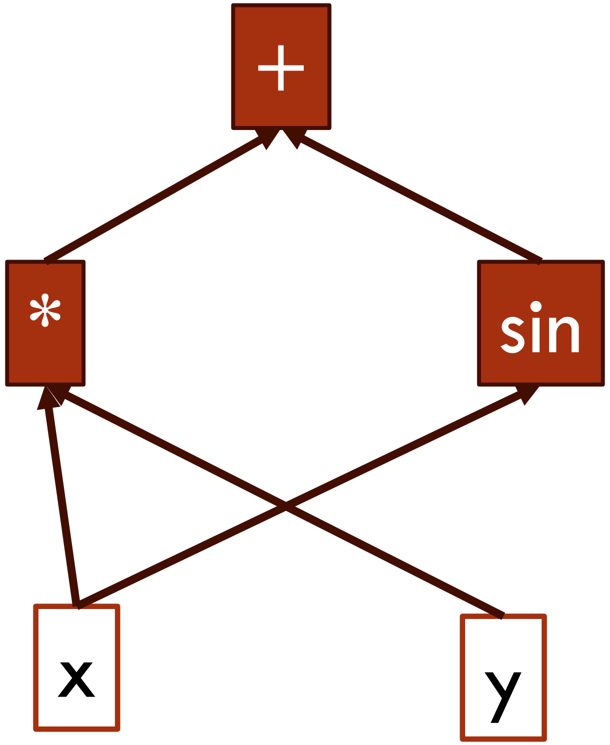
chapter_preliminaries/autograd.ipynb

Parse Tree

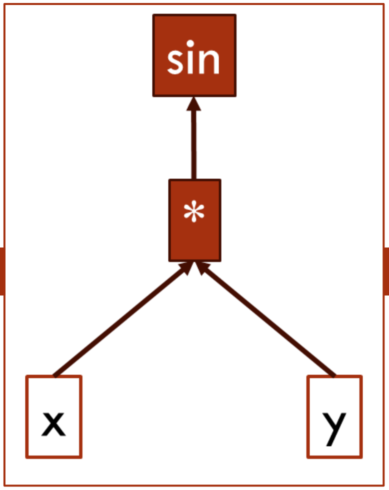
- $\sin(xy)$



- $xy + \sin(x)$



Forward Accumulation Example: $\sin(xy)$



- Differentiate wrt x at $x = 2, y = 3$:

$$\frac{\partial}{\partial x}[\sin(xy)]$$

Apply chain rule and evaluate

$$= \cos(xy)\frac{\partial}{\partial x}[xy] = \cos(2 * 3)\frac{\partial}{\partial x}[xy]$$

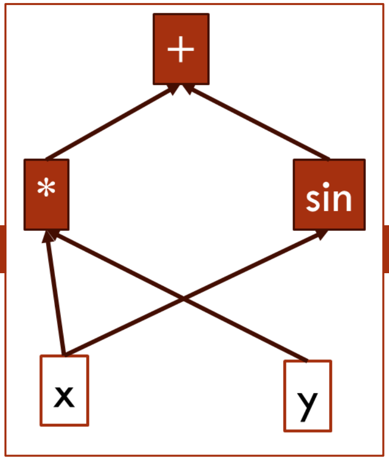
Apply product rule and evaluate

$$= \cos(6)\left[y\frac{\partial}{\partial x}x + x\frac{\partial}{\partial x}y\right] = \cos(6)\left[3\frac{\partial}{\partial x}x + 2\frac{\partial}{\partial x}y\right]$$

$$= \cos(6) * 3 \ * 1$$

Multiple all the factors

Forward Accumulation: $xy + \sin(x)$



- Differentiate wrt to x at $x = 2, y = 3$:

$$\frac{\partial}{\partial x}[xy + \sin(x)]$$

Apply sum rule and evaluate

$$= \frac{\partial}{\partial x}[xy] + \frac{\partial}{\partial x}[\sin(x)] = 1\frac{\partial}{\partial x}[xy] + 1\frac{\partial}{\partial x}[\sin(x)]$$

Apply product rule and evaluate

Apply chain rule and evaluate

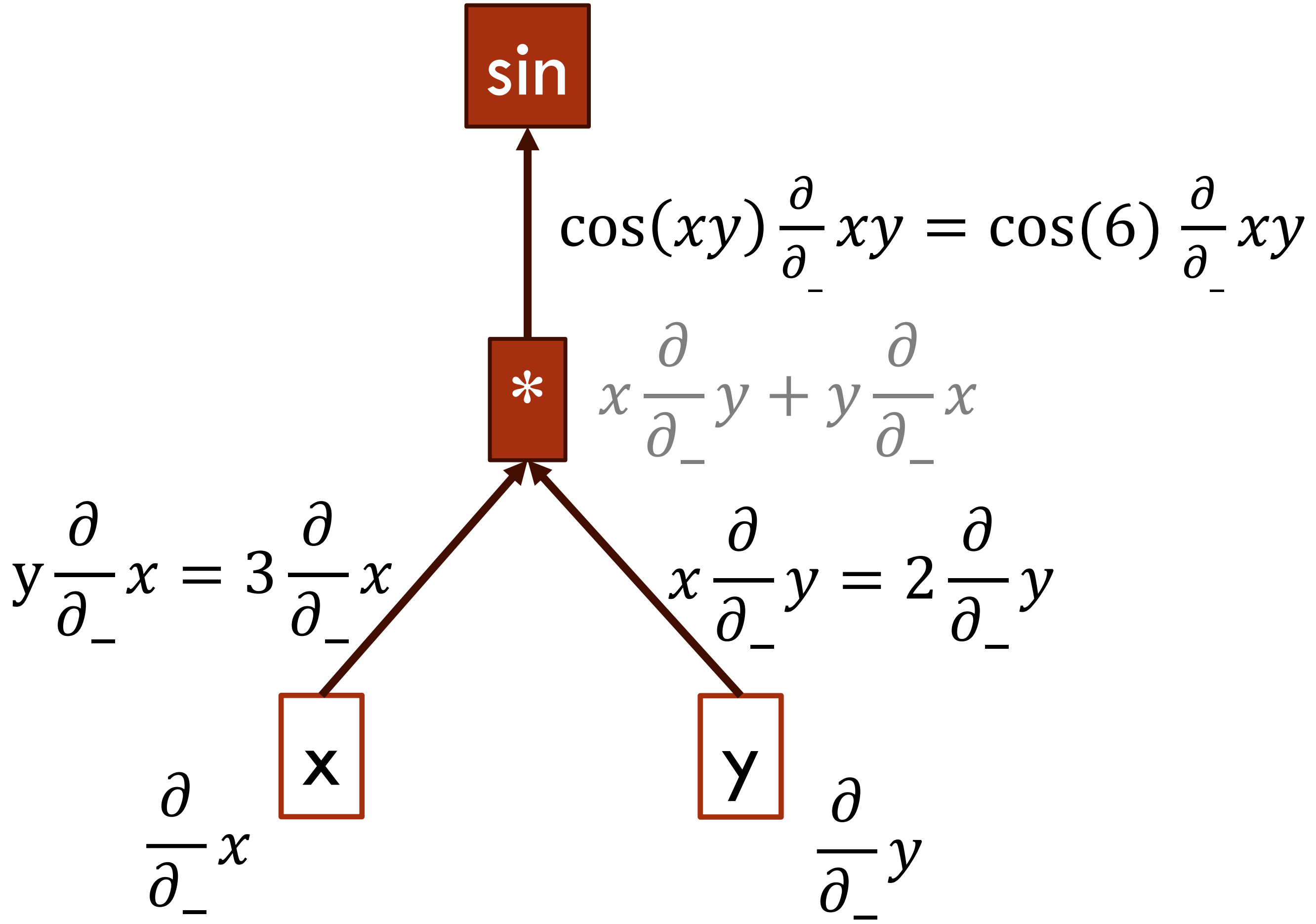
$$= 1\left[y\frac{\partial}{\partial x}x + x\frac{\partial}{\partial x}y\right] + 1\left[\cos(x)\frac{\partial}{\partial x}x\right] = 1\left[3\frac{\partial}{\partial x}x + 2\frac{\partial}{\partial x}y\right] + 1\left[\cos(2)\frac{\partial}{\partial x}x\right]$$

$$= 1 * 3 + 1 \ * \cos(2)$$

Multiple all the factors, then add

Backward Accumulation: $\sin(xy)$

- At $x = 2, y = 3$ differentiate $\frac{\partial}{\partial \_}[\sin xy]$

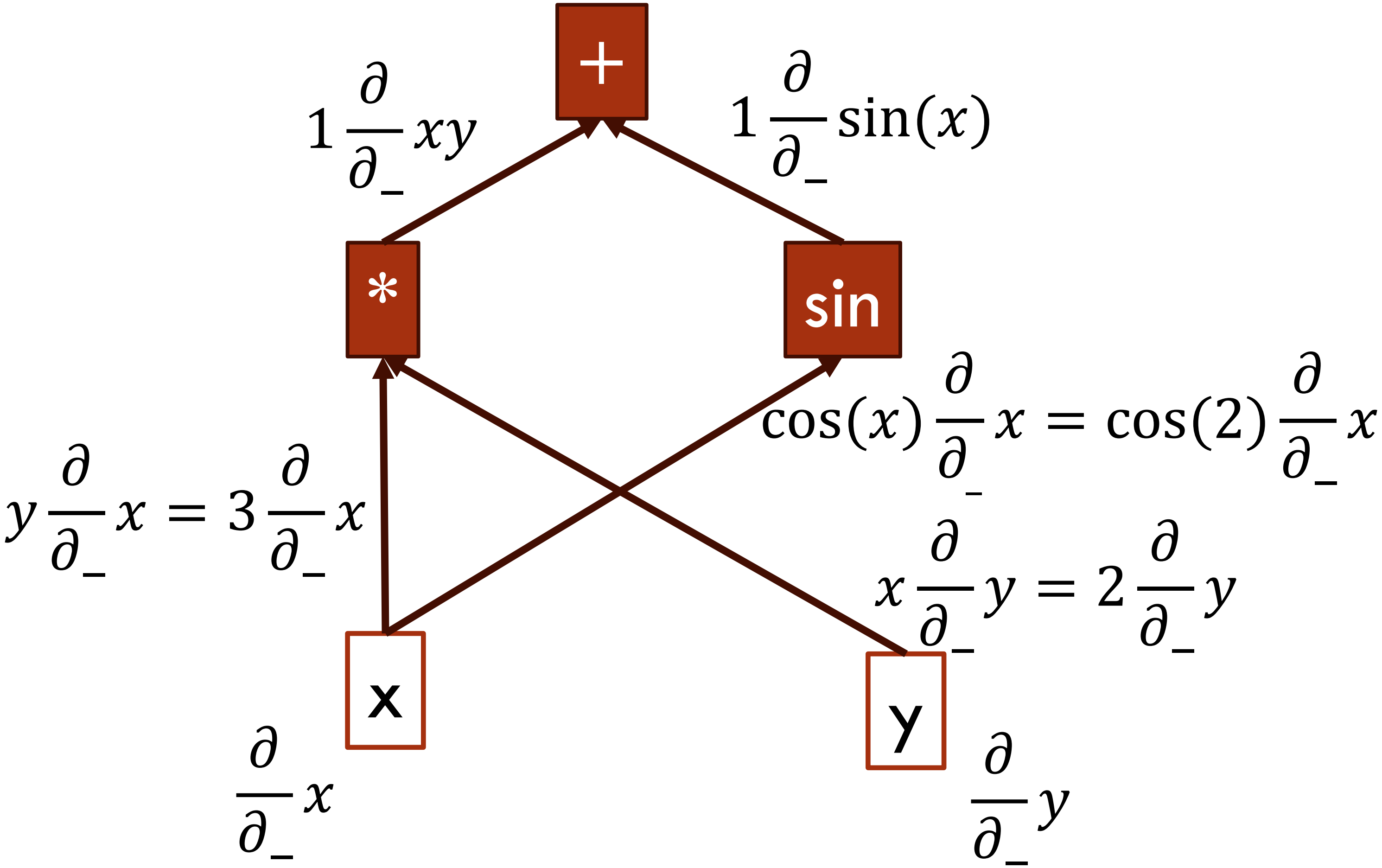


$$\frac{\partial}{\partial x}\sin(xy) = \cos(6)(3)$$

$$\frac{\partial}{\partial y}\sin(xy) = \cos(6)(2)$$

Backward Accumulation: $xy + \sin(x)$

- At $x = 2, y = 3$ differentiate $\frac{\partial}{\partial\_}\{xy + \sin(x)\}$



$$\frac{\partial}{\partial y}xy + \sin(x) = 2$$

$$\frac{\partial}{\partial x}xy + \sin(x) = 3 + \cos(2)$$

Automatic Differentiation Backward Accumulation with Python


```
class Var:
    def __init__(self, value, children=None, op='Var'):
        self.value = value
        self.grad = 0
        self.children = children or []
        self.op = op
    def __add__(self, other):
        return Var(self.value + other.value,
                   [(1, self), (1, other)], 'add')
    def __mul__(self, other):
        return Var(self.value * other.value,
                   [(other.value, self), (self.value, other)], 'mul')
    def sin(self):
        return Var(math.sin(self.value),
                    [(math.cos(self.value), self)], 'sin')
    def backward(self, grad=1):
        self.grad += grad
        for coef, child in self.children:
            child.backward(grad * coef)
    def __repr__(self):
        return f'({self.op}, ' + str(self.value) + ')'
```

```
# Example: f(x, y) = x * y + sin(x)
x = Var(2, op='x')
y = Var(3, op='y')
z = x * y + x.sin()
```

- See Brightspace
  - content/notebooks/AutomaticDifferentiationBackwardAccumulation.ipynb
  - content/notebooks/AutomaticDifferentiationForwardAccumulation.ipynb

Automatic Differentiation

- Previous examples use scaler functions: $f: \mathcal{R}^n \to \mathcal{R}$ loss
- Automatic differentiation can be extended to vector functions: $f: \mathcal{R}^n \to \mathcal{R}^m$ weights
- Forward accumulation differentiation is more efficient if $n \ll m$
- Backward accumulation differentiation is more efficient if $n \gg m$
- In deep learning
  - The output is usually a scaler loss function, $m = 1$
  - The input is huge, i.e., n is the number of weights in the neural network

Probabilities

SKIP

Observable and Unobservable Variables

- Data usually comes from a process that is not completely known
- We model this as a random process and analyze it using probability theory
- Example
  - Tossing a coin as a random process, as we cannot predict at any toss on which side the con will land. Heads or Tails?
  - We can only talk about the probability of the outcome (*observable variable*) for the next toss.
  - In case we have all the information (e.g. speed of throw, angle, wind, etc) we might be able to precisely predict outcome. These are the *unobservable variables*.

Random Variable

- A *random variable* X is a function that assigns a number to each outcome in the sample space of a random experiment
- The probability of a particular outcome x is defined by probability distribution P(X=x)
- Coin toss example:
  - Let X=1 denote the outcome is Heads, and X=0 is Tails
    - Random variable $X \in \{1,0\}$
  - Let p_0 be the probability the outcome is Heads: $P(X = 1) = p_0$ and $P(X = 0) = 1 - P(X = 1) = 1 - p_0$
    - Bernoulli: $P\ (X = i) = p_o^i(1 - p_o)^{1-i}$

Sample

- If we do not know P(X), then we need a sample $\mathcal{X}$ (e.g. our training data) that contains examples dawn from the probability distribution of the observable variables x^t denoted as $p(x)$.
- The goal is to build an approximator $\hat{p}(x)$ using sample $\mathcal{X}$
- Coin toss example:
  - Sample: $\mathcal{X} = \{x^t\}^N_{t=1}$
    Estimation: $\hat{p}_0 = \#\{\text{Heads}\}/\#\{\text{Tosses}\} = \sum_t x^t / N$
  - If we observe: {heads, heads, heads, tails, heads, tails, tails, heads, heads} we have:
    - $X = \{1,1,1,0,1,0,0,1,1\}$, $\hat{p}_0 = \frac{6}{9}$
  - Prediction of next toss:

    Heads if $\hat{p}_0 > \frac{1}{2}$, Tails otherwise

Brief Review of Probabilities

- Sum rule

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- Product rule

$$P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$$

- If A and B are independent

$$P(A \wedge B) = P(A)P(B); \quad A \perp B$$

  - Conditional independence
  - $P(A \wedge B \,|C) = P(A|C)P(B|C); \quad A \perp B|C$

Bayes' Rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Derivation using the product rule:

$$\begin{aligned} P(A|B) &= \frac{P(A \wedge B)}{P(B)} \\ &= \frac{P(B|A)P(A)}{P(B)} \end{aligned}$$

Bayes' Rule

prior *likelihood*

posterior

$$P(C \mid \mathbf{x}) = \frac{P(C)\,p(\mathbf{x} \mid C)}{p(\mathbf{x})}$$

evidence

$$P(C=0) + P(C=1) = 1$$

$$p(\mathbf{x}) = p(\mathbf{x} \mid C=1)P(C=1) + p(\mathbf{x} \mid C=0)P(C=0)$$

$$p(C=0 \mid \mathbf{x}) + P(C=1 \mid \mathbf{x}) = 1$$

Bayes' Rule: $K>2$ Classes

- In the general case we K mutually exclusive classes, $C_i, i = 1, \dots, K$
- Prior probability satisfy:

$$P(C_i) \geq 0 \text{ and } \sum_{i=1}^{K} P(C_i) = 1$$

- Posterior probability of class C_i is calculated as:

$$P(C_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_i)P(C_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x} \mid C_i)P(C_i)}{\sum_{k=1}^{K} p(\mathbf{x} \mid C_k)P(C_k)}$$

- Bayes' classifier

$$\text{choose } C_i \text{ if } P(C_i \mid \mathbf{x}) = \max_k P(C_k \mid \mathbf{x})$$

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20

- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M|S) = \frac{P(S|M)P(M)}{P(S)} = \frac{0.5 \times \frac{1}{50000}}{\frac{1}{20}} = \frac{1}{5000}$$

Another Example

- Want to decide between two hypothesis, for a particular location
  - There is oil underground
  - There is no oil underground
- Suppose there is oil test: pos or neg

$$P(oil) = .008 \qquad P(\neg oil) = .992$$

$$P(pos \mid oil) = .98 \qquad P(neg \mid oil) = .02$$

$$P(pos \mid \neg oil) = .03 \qquad P(neg \mid \neg oil) = .97$$

Another Example (cont)

- Give test is positive, compute posteriori probability for oil and not oil

$$P(oil \mid pos) \propto P(pos \mid oil)P(oil) = .98(.008) = .0078$$

$$P(\neg oil \mid pos) \propto P(pos \mid \neg oil)P(\neg oil) = .03(.992) = .0298$$

$$P(\neg oil \mid pos) = \frac{.0298}{.0078 + .0298} = .79$$

- Given positive test, probability of not oil is 0.79