

Lecture 5: Vector Calculus

Yi, Yung (이웅)

Mathematics for Machine Learning
<https://yung-web.github.io/home/courses/mathml.html>
KAIST EE

MONTH DAY, 2021

- Differentiation of Univariate Functions
- Partial Differentiation and Gradients
- Gradients of Vector-Valued Functions
- Gradients of Matrices
- Useful Identities for Computing Gradients
- Backpropagation and Automatic Differentiation
- Higher-Order Derivatives
- Linearization and Multivariate Taylor Series

MONTH DAY, 2021 1 / 35

MONTH DAY, 2021 2 / 35

Summary

- Machine learning is about solving an optimization problem whose variables are the parameters of a given model.
- Solving optimization problems require gradient information.
- Central to this chapter is the concept of the function, which we often write

$$f : \mathbb{R}^D \mapsto \mathbb{R}$$
$$\mathbf{x} \mapsto f(\mathbf{x})$$

- Differentiation of Univariate Functions
- Partial Differentiation and Gradients
- Gradients of Vector-Valued Functions
- Gradients of Matrices
- Useful Identities for Computing Gradients
- Backpropagation and Automatic Differentiation
- Higher-Order Derivatives
- Linearization and Multivariate Taylor Series

MONTH DAY, 2021 3 / 35

MONTH DAY, 2021 4 / 35

- **Difference Quotient.** The average slope of f between x and $x + \partial x$

$$\frac{\partial y}{\partial x} := \frac{f(x + \partial x) - f(x)}{\partial x}$$

- **Derivative.** Pointing in the direction of steepest ascent of f .

$$\frac{df}{dx} := \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

- Unless confusion arises, we often use $f' = \frac{df}{dx}$.

- Representation of a function as an infinite sum of terms, using derivatives of evaluated at x_0 .

- **Taylor polynomial.** The Taylor polynomial of degree n of $f : \mathbb{R} \mapsto \mathbb{R}$ at x_0 is:

$$T_n(x) := \sum_{k=0}^n \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k, \text{ where } f^{(k)}(x_0) \text{ is the } k\text{th derivative of } f \text{ at } x_0.$$

- **Taylor Series.** For a smooth function $f \in \mathcal{C}^\infty$, the Taylor series of f at x_0 is:

$$T_\infty(x) := \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k.$$

- If $f(x) = T_\infty(x)$, f is called **analytic**.

- **Product rule.** $(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$

- **Quotient rule.** $\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2}$

- **Sum rule.** $(f(x) + g(x))' = f'(x) + g'(x)$

- **Chain rule.** $(g(f(x)))' = g'(f(x))f'(x)$

- Differentiation of Univariate Functions
- **Partial Differentiation and Gradients**
- Gradients of Vector-Valued Functions
- Gradients of Matrices
- Useful Identities for Computing Gradients
- Backpropagation and Automatic Differentiation
- Higher-Order Derivatives
- Linearization and Multivariate Taylor Series

- Now, $f: \mathbb{R}^n \mapsto \mathbb{R}$.
- Gradient of f w.r.t. \mathbf{x} $\nabla_{\mathbf{x}} f$: Varying one variable at a time and keeping the others constant.

Partial Derivative. For $f: \mathbb{R}^n \mapsto \mathbb{R}$,

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= \lim_{h \rightarrow 0} \frac{f(x_1 + h, x_2, \dots, x_n) - f(\mathbf{x})}{h} \\ &\vdots \\ \frac{\partial f}{\partial x_n} &= \lim_{h \rightarrow 0} \frac{f(x_1, x_2, \dots, x_n + h) - f(\mathbf{x})}{h} \end{aligned}$$

Gradient. Get the partial derivatives and collect them in the row vector.

$$\nabla_{\mathbf{x}} f = \frac{df}{d\mathbf{x}} = \left(\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \dots \quad \frac{\partial f(\mathbf{x})}{\partial x_n} \right) \in \mathbb{R}^{1 \times n}$$

- **Example.** $f(x, y) = (x + 2y^3)^2$

$$\frac{\partial f(x, y)}{\partial x} = 2(x + 2y^3) \frac{\partial x + 2y^3}{\partial x} = 2(x + 2y^3)$$

$$\frac{\partial f(x, y)}{\partial y} = 2(x + 2y^3) \frac{\partial x + 2y^3}{\partial y} = 12(x + 2y^3)y^2$$

- **Example.** $f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3$

$$\nabla_{(x_1, x_2)} f = \frac{df}{d\mathbf{x}} = \left(\frac{\partial f(x_1, x_2)}{\partial x_1} \quad \frac{\partial f(x_1, x_2)}{\partial x_2} \right) = (2x_1 x_2 + x_2^3 \quad x_1^2 + 3x_1 x_2^2)$$

- **Product rule.**

$$\frac{\partial}{\partial \mathbf{x}} (f(\mathbf{x})g(\mathbf{x})) = \frac{\partial f}{\partial \mathbf{x}} g(\mathbf{x}) + f(\mathbf{x}) \frac{\partial g}{\partial \mathbf{x}}$$

- **Sum rule.**

$$\frac{\partial}{\partial \mathbf{x}} (f(\mathbf{x}) + g(\mathbf{x})) = \frac{\partial f}{\partial \mathbf{x}} + \frac{\partial g}{\partial \mathbf{x}}$$

- **Chain rule.**

$$\frac{\partial}{\partial \mathbf{x}} g(f(\mathbf{x})) = \frac{\partial g}{\partial f} \frac{\partial f}{\partial \mathbf{x}}$$

- $f: \mathbb{R}^2 \mapsto \mathbb{R}$ of two variables x_1 and x_2 . $x_1(t)$ and $x_2(t)$ are functions of t .

$$\frac{df}{dt} = \left(\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \right) \begin{pmatrix} \frac{\partial x_1(t)}{\partial t} \\ \frac{\partial x_2(t)}{\partial t} \end{pmatrix} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t}$$

- **Example.** $f(x_1, x_2) = x_1^2 + 2x_2$, where $x_1(t) = \sin(t)$, $x_2(t) = \cos(t)$

$$\frac{df}{dt} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t} = 2\sin(t)\cos(t) - 2\sin t = 2\sin(t)(\cos(t) - 1)$$

- $f: \mathbb{R}^2 \mapsto \mathbb{R}$ of two variables x_1 and x_2 . $x_1(s, t)$ and $x_2(s, t)$ are functions of s, t .

$$\begin{aligned} \frac{\partial f}{\partial s} &= \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial s} \\ \frac{\partial f}{\partial t} &= \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t} \end{aligned} \quad \left| \quad \frac{df}{d(s, t)} = \frac{\partial f}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial (s, t)} = \left(\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \right) \begin{pmatrix} \frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t} \end{pmatrix}$$

- Differentiation of Univariate Functions
- Partial Differentiation and Gradients
- **Gradients of Vector-Valued Functions**
- Gradients of Matrices
- Useful Identities for Computing Gradients
- Backpropagation and Automatic Differentiation
- Higher-Order Derivatives
- Linearization and Multivariate Taylor Series

- For a function $f: \mathbb{R}^n \mapsto \mathbb{R}^m$ and vector $\mathbf{x} = (x_1 \dots x_n)^T \in \mathbb{R}^n$, the vector-valued function is:

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix}$$

- Partial derivative w.r.t. x_i is a column vector: $\frac{\partial \mathbf{f}}{\partial x_i} = \begin{pmatrix} \frac{\partial f_1}{\partial x_i} \\ \vdots \\ \frac{\partial f_m}{\partial x_i} \end{pmatrix}$
- Gradient (or Jacobian): $\frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} = \left(\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1} \dots \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_n} \right)$

$$\begin{aligned} \mathbf{J} = \nabla_{\mathbf{x}} \mathbf{f} &= \frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} = \left(\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1} \dots \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_n} \right) \\ &= \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{pmatrix} \end{aligned}$$

- For a $\mathbb{R}^n \mapsto \mathbb{R}^m$ function, its Jacobian is a $m \times n$ matrix.

- $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$, $\mathbf{f}: \mathbb{R}^n \mapsto \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$

- Partial derivatives: $f_i(\mathbf{x}) = \sum_{j=1}^n A_{ij}x_j \implies \frac{\partial f_i}{\partial x_j} = A_{ij}$

- Gradient

$$\frac{d\mathbf{f}}{d\mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} = \begin{pmatrix} A_{11} & \dots & A_{1n} \\ \vdots & & \vdots \\ A_{m1} & \dots & A_{mn} \end{pmatrix} = \mathbf{A}$$

- $h: \mathbb{R} \mapsto \mathbb{R}$, $h(t) = (f \circ g)(t)$ with

$$f: \mathbb{R}^2 \mapsto \mathbb{R}, f(\mathbf{x}) = \exp(x_1 x_2^2), \quad g: \mathbb{R} \mapsto \mathbb{R}^2, \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = g(t) = \begin{pmatrix} t \cos(t) \\ t \sin(t) \end{pmatrix}$$

- (Note) $\frac{\partial f}{\partial \mathbf{x}} \in \mathbb{R}^{1 \times 2}$ and $\frac{\partial g}{\partial t} \in \mathbb{R}^{2 \times 1}$

- Using the chain rule,

$$\frac{dh}{dt} = \frac{\partial f}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t} = \left(\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \right) \begin{pmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{pmatrix} = (\exp(x_1 x_2^2) x_2^2 \quad 2 \exp(x_1 x_2^2) x_1 x_2) \begin{pmatrix} \cos(t) - t \sin(t) \\ \sin(t) + t \cos(t) \end{pmatrix}$$

- A linear model: $\mathbf{y} = \Phi \boldsymbol{\theta}$, $\boldsymbol{\theta} \in \mathbb{R}^D$: parameter vector, $\Phi \in \mathbb{R}^{N \times D}$: input features, and $\mathbf{y} \in \mathbb{R}^N$: observations.
 - Goal: Find a good parameter vector that provides the best-fit, formulated by minimizing the following loss $L: \mathbb{R}^D \mapsto \mathbb{R}$ over the parameter vector $\boldsymbol{\theta}$.

$$L(\mathbf{e}) := \|\mathbf{e}\|^2, \text{ where } \mathbf{e}(\boldsymbol{\theta}) = \mathbf{y} - \Phi \boldsymbol{\theta}$$

- $\frac{\partial L}{\partial \boldsymbol{\theta}} = \frac{\partial L}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial \boldsymbol{\theta}}$ (Note: $\frac{\partial L}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{1 \times D}$, $\frac{\partial L}{\partial \mathbf{e}} \in \mathbb{R}^{1 \times N}$, $\frac{\partial \mathbf{e}}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{N \times D}$)

- Using that $\|\mathbf{e}\|^2 = \mathbf{e}^T \mathbf{e}$, $\frac{\partial L}{\partial \mathbf{e}} = 2\mathbf{e}^T \in \mathbb{R}^{1 \times N}$

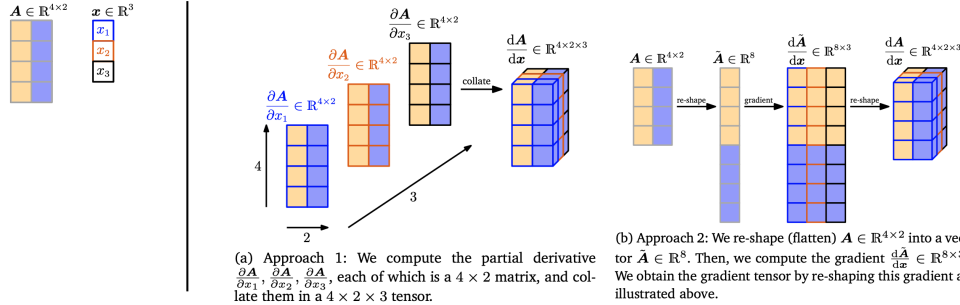
- $\frac{\partial \mathbf{e}}{\partial \boldsymbol{\theta}} = -\Phi \in \mathbb{R}^{N \times D}$

$$\text{Finally, we get: } \frac{\partial L}{\partial \boldsymbol{\theta}} = -2\mathbf{e}^T \Phi = -2 \underbrace{(\mathbf{y}^T - \boldsymbol{\theta}^T \Phi^T)}_{1 \times N} \underbrace{\Phi}_{N \times D}$$

- Differentiation of Univariate Functions
- Partial Differentiation and Gradients
- Gradients of Vector-Valued Functions
- Gradients of Matrices
- Useful Identities for Computing Gradients
- Backpropagation and Automatic Differentiation
- Higher-Order Derivatives
- Linearization and Multivariate Taylor Series

- Differentiation of Univariate Functions
- Partial Differentiation and Gradients
- Gradients of Vector-Valued Functions
- Gradients of Matrices
- Useful Identities for Computing Gradients
- Backpropagation and Automatic Differentiation
- Higher-Order Derivatives
- Linearization and Multivariate Taylor Series

- Gradient of matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ w.r.t. matrix $\mathbf{B} \in \mathbb{R}^{p \times q}$
- Jacobian: A four-dimensional tensor¹ $\mathbf{J} = \frac{d\mathbf{A}}{d\mathbf{B}} \in \mathbb{R}^{(m \times n) \times (p \times q)}$



¹A multidimensional array

MONTH DAY, 2021 21 / 35

- $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$, $\mathbf{f} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$. What is $\frac{d\mathbf{f}}{d\mathbf{A}}$?
- Dimension: If we consider $\mathbf{f} : \mathbb{R}^{m \times n} \mapsto \mathbb{R}^m$, $\frac{d\mathbf{f}}{d\mathbf{A}} \in \mathbb{R}^{m \times (m \times n)}$

- Partial derivatives: $\frac{\partial f_i}{\partial \mathbf{A}} \in \mathbb{R}^{1 \times (m \times n)}$, $\frac{d\mathbf{f}}{d\mathbf{A}} = \begin{pmatrix} \frac{\partial f_1}{\partial \mathbf{A}} \\ \vdots \\ \frac{\partial f_m}{\partial \mathbf{A}} \end{pmatrix}$

$$f_i = \sum_{j=1}^n A_{ij}x_j, \quad i = 1, \dots, m \implies \frac{\partial f_i}{\partial A_{ij}} = x_j,$$

$$\frac{\partial f_i}{\partial A_{ij}} = \mathbf{x}^T \in \mathbb{R}^{1 \times 1 \times n} \quad (\text{for } i\text{th row vector})$$

$$\frac{\partial f_i}{\partial A_{k \neq i}} = 0^T \in \mathbb{R}^{1 \times 1 \times n} \quad (\text{for } k\text{th row vector, } k \neq i)$$

$$\frac{\partial f_i}{\partial \mathbf{A}} = \begin{pmatrix} 0^T \\ \vdots \\ 0^T \\ \mathbf{x}^T \\ 0^T \\ \vdots \\ 0^T \end{pmatrix} \in \mathbb{R}^{1 \times (m \times n)}$$

MONTH DAY, 2021 22 / 35

- $\mathbf{R} \in \mathbb{R}^{m \times n}$ and $\mathbf{f} : \mathbb{R}^{m \times n} \mapsto \mathbb{R}^{n \times n}$ with $\mathbf{f}(\mathbf{R}) = \mathbf{K} := \mathbf{R}^T \mathbf{R} \in \mathbb{R}^{n \times n}$. What is $\frac{d\mathbf{K}}{d\mathbf{R}} \in \mathbb{R}^{(n \times n) \times (m \times n)}$?
- $\frac{dK_{pq}}{d\mathbf{R}} \in \mathbb{R}^{1 \times m \times n}$. Let \mathbf{r}_i be the i th column of \mathbf{R} . Then $K_{pq} = \mathbf{r}_p^T \mathbf{r}_q = \sum_{k=1}^m R_{kp} R_{kq}$.
- Partial derivative $\frac{\partial K_{pq}}{\partial R_{ij}}$

$$\frac{\partial K_{pq}}{\partial R_{ij}} = \sum_{k=1}^m \frac{\partial}{\partial R_{ij}} R_{kp} R_{kq} = \partial_{pqij}, \quad \partial_{pqij} = \begin{cases} R_{iq} & \text{if } j = p, p \neq q \\ R_{ip} & \text{if } j = q, p \neq q \\ 2R_{iq} & \text{if } j = p, p = q \\ 0 & \text{otherwise} \end{cases}$$

MONTH DAY, 2021 23 / 35

$$\frac{\partial}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^T = \left(\frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right)^T \quad (5.99)$$

$$\frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{f}(\mathbf{X})) = \text{tr} \left(\frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right) \quad (5.100)$$

$$\frac{\partial}{\partial \mathbf{X}} \det(\mathbf{f}(\mathbf{X})) = \det(\mathbf{f}(\mathbf{X})) \text{tr} \left(\mathbf{f}(\mathbf{X})^{-1} \frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right) \quad (5.101)$$

$$\frac{\partial}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^{-1} = -\mathbf{f}(\mathbf{X})^{-1} \frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^{-1} \quad (5.102)$$

$$\frac{\partial \mathbf{a}^T \mathbf{X}^{-1} \mathbf{b}}{\partial \mathbf{X}} = -(\mathbf{X}^{-1})^T \mathbf{a} \mathbf{b}^T (\mathbf{X}^{-1})^T \quad (5.103)$$

$$\frac{\partial \mathbf{x}^T \mathbf{a}}{\partial \mathbf{x}} = \mathbf{a}^T \quad (5.104)$$

$$\frac{\partial \mathbf{a}^T \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}^T \quad (5.105)$$

$$\frac{\partial \mathbf{a}^T \mathbf{X} \mathbf{b}}{\partial \mathbf{X}} = \mathbf{a} \mathbf{b}^T \quad (5.106)$$

$$\frac{\partial \mathbf{x}^T \mathbf{B} \mathbf{x}}{\partial \mathbf{x}} = \mathbf{x}^T (\mathbf{B} + \mathbf{B}^T) \quad (5.107)$$

$$\frac{\partial}{\partial \mathbf{s}} (\mathbf{x} - \mathbf{A} \mathbf{s})^T \mathbf{W} (\mathbf{x} - \mathbf{A} \mathbf{s}) = -2(\mathbf{x} - \mathbf{A} \mathbf{s})^T \mathbf{W} \mathbf{A} \quad \text{for symmetric } \mathbf{W} \quad (5.108)$$

MONTH DAY, 2021 24 / 35

- Differentiation of Univariate Functions
- Partial Differentiation and Gradients
- Gradients of Vector-Valued Functions
- Gradients of Matrices
- Useful Identities for Computing Gradients
- **Backpropagation and Automatic Differentiation**
- Higher-Order Derivatives
- Linearization and Multivariate Taylor Series

- In a neural network with many layers, the function \mathbf{y} is a many-level function compositions

$$\mathbf{y} = (f_K \circ f_{K-1} \circ \cdots \circ f_1)(\mathbf{x}),$$

where, for example,

- \mathbf{x} : images as inputs, \mathbf{y} : class labels (e.g., cat or dog) as outputs
- each f_i has its own parameters
- In neural networks, with the model parameters $\theta = \{\mathbf{A}_0, \mathbf{b}_0, \dots, \mathbf{A}_{K-1}, \mathbf{b}_{K-1}\}$
 - Minimizing the loss function over θ :
$$\min_{\theta} L(\theta),$$
where $L(\theta) = \|\mathbf{y} - \mathbf{f}_K(\theta, \mathbf{x})\|^2$

$$\begin{cases} \mathbf{f}_0 &:= \mathbf{x} \\ \mathbf{f}_1 &:= \sigma_1(\mathbf{A}_0 \mathbf{f}_0 + \mathbf{b}_0) \\ \vdots & \\ \mathbf{f}_K &:= \sigma_K(\mathbf{A}_{K-1} \mathbf{f}_{K-1} + \mathbf{b}_{K-1}) \end{cases}$$

◦ σ_i is called the **activation function** at i -th layer

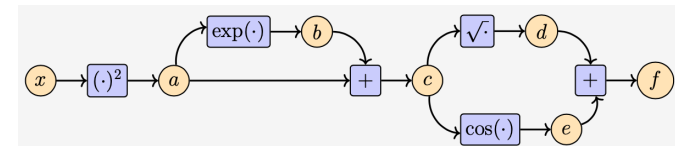
- In neural networks, with the model parameters $\theta = \{\mathbf{A}_0, \mathbf{b}_0, \dots, \mathbf{A}_{K-1}, \mathbf{b}_{K-1}\}$
 - Minimizing the loss function over θ :
$$\min_{\theta} L(\theta),$$
where $L(\theta) = \|\mathbf{y} - \mathbf{f}_K(\theta, \mathbf{x})\|^2$

$$\begin{cases} \mathbf{f}_0 &:= \mathbf{x} \\ \mathbf{f}_1 &:= \sigma_1(\mathbf{A}_0 \mathbf{f}_0 + \mathbf{b}_0) \\ \vdots & \\ \mathbf{f}_K &:= \sigma_K(\mathbf{A}_{K-1} \mathbf{f}_{K-1} + \mathbf{b}_{K-1}) \end{cases}$$

◦ σ_i is called the activation function at i -th layer

- **Question.** How can we efficiently compute $\frac{dL}{d\theta}$ in computers?

- $f(x) = \sqrt{x^2 + \exp(x^2)} + \cos(x^2 + \exp(x^2))$
- Computation graph: Connect via “elementary” operations



$$a = x^2, \quad b = \exp(a), \quad c = a + b, \quad d = \sqrt{c}, \quad e = \cos(c), \quad f = d + e$$

- Automatic Differentiation
 - A set of techniques to **numerically** (not symbolically) evaluate the gradient of a function by working with **intermediate variables** and applying the **chain rule**.

- $a = x^2$, $b = \exp(a)$, $c = a + b$, $d = \sqrt{c}$, $e = \cos(c)$, $f = d + e$

- Derivatives of the intermediate variables with their inputs

$$\frac{\partial a}{\partial x} = 2x, \quad \frac{\partial b}{\partial a} = \exp(a), \quad \frac{\partial c}{\partial a} = 1 = \frac{\partial c}{\partial b}, \quad \frac{\partial d}{\partial c} = \frac{1}{2\sqrt{c}}, \quad \frac{\partial e}{\partial c} = -\sin(c), \quad \frac{\partial f}{\partial d} = 1 = \frac{\partial f}{\partial e}$$

- Compute $\frac{\partial f}{\partial x}$ by working backward from the output

$$\begin{aligned} \frac{\partial f}{\partial c} &= \frac{\partial f}{\partial d} \frac{\partial d}{\partial c} + \frac{\partial f}{\partial e} \frac{\partial e}{\partial c}, & \frac{\partial f}{\partial b} &= \frac{\partial f}{\partial c} \frac{\partial c}{\partial b} \\ \frac{\partial f}{\partial a} &= \frac{\partial f}{\partial b} \frac{\partial b}{\partial a} + \frac{\partial f}{\partial c} \frac{\partial c}{\partial a}, & \boxed{\frac{\partial f}{\partial x}} &= \frac{\partial f}{\partial a} \frac{\partial a}{\partial x} \end{aligned} \quad \left| \quad \begin{aligned} \frac{\partial f}{\partial c} &= 1 \cdot \frac{1}{2\sqrt{c}} + 1 \cdot (-\sin(c)) \\ \frac{\partial f}{\partial b} &= \frac{\partial f}{\partial c} \cdot 1, & \frac{\partial f}{\partial a} &= \frac{\partial f}{\partial b} \exp(a) + \frac{\partial f}{\partial c} \cdot 1 \\ \boxed{\frac{\partial f}{\partial x}} &= \frac{\partial f}{\partial a} \cdot 2x \end{aligned}$$

- Implementation of gradients can be very expensive, unless we are careful.
- Using the idea of automatic differentiation, the whole gradient computation is decomposed into a set of gradients of elementary functions and application of the chain rule.
- Why **backward**?
 - In neural networks, the input dimensionality is often much higher than the dimensionality of labels.
 - In this case, the backward computation (than the forward computation) is much cheaper.
- Works if the target is expressed as a computation graph whose elementary functions are differentiable. If not, some care needs to be taken.

- Differentiation of Univariate Functions
- Partial Differentiation and Gradients
- Gradients of Vector-Valued Functions
- Gradients of Matrices
- Useful Identities for Computing Gradients
- Backpropagation and Automatic Differentiation
- **Higher-Order Derivatives**
- **Linearization and Multivariate Taylor Series**

- Some optimization algorithms (e.g., Newton's method) require second-order derivatives, if they exist.
- (Truncated) Taylor series is often used as an approximation of a function.
- For $f : \mathbb{R}^n \mapsto \mathbb{R}$ of variable $\mathbf{x} \in \mathbb{R}^n$, $\nabla_{\mathbf{x}} f = \frac{df}{d\mathbf{x}} = \left(\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \dots \quad \frac{\partial f(\mathbf{x})}{\partial x_n} \right) \in \mathbb{R}^{1 \times n}$
 - If f is twice-differentiable, the order doesn't matter.

$$H_{\mathbf{x}} f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \frac{\partial^2 f}{\partial x_2 \partial x_n} & \dots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

- For $f : \mathbb{R}^n \mapsto \mathbb{R}^m$, $\nabla_{\mathbf{x}} f \in \mathbb{R}^{m \times n}$
 - Thus, $H_{\mathbf{x}} f \in \mathbb{R}^{m \times n \times n}$ (a tensor)

- First-order approximation of $f(\mathbf{x})$ (i.e., linearization by taking the first two terms of Taylor Series)

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + (\nabla_{\mathbf{x}} f)(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$$

- Multivariate Taylor Series for $f : \mathbb{R}^D \mapsto \mathbb{R}$ at \mathbf{x}_0

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \frac{D_{\mathbf{x}}^k f(\mathbf{x}_0)}{k!} \delta^k,$$

where $D_{\mathbf{x}}^k f(\mathbf{x}_0)$ is the k th derivative of f w.r.t. \mathbf{x} , evaluated at \mathbf{x}_0 , and $\delta := \mathbf{x} - \mathbf{x}_0$.

- Partial sum up to, say n , can be an approximation of $f(\mathbf{x})$.
- $D_{\mathbf{x}}^k f(\mathbf{x}_0)$ and δ^k are k th order tensors, i.e., k -dimensional array.
- δ^k is a k -fold outer product \otimes . For example, $\delta^2 = \delta \otimes \delta = \delta \delta^T$. $\delta^3 = \delta \otimes \delta \otimes \delta$.

Questions?

1)