Spring 2023

SWCON253: Machine Learning

Lecture 03
Gradient Descent

Jinwoo Choi
Assistant Professor
CSE, Kyung Hee University

Contents

References

- *Mathematics for Machine Learning* by Deisenroth, Faisal, and Ong (https://mml-book.com)
- *Intro to Deep Learning & Generative Models* by Sebastian Raschka (http://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2020/)
- *패턴 인식* by 오일석, *기계 학습* by 오일석

1. Vector Calculus

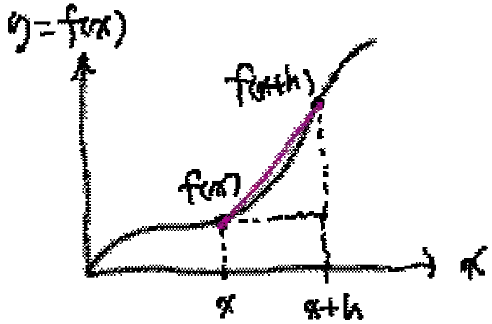
1. Derivative
2. Chain Rule
3. Gradient: *Collection of Partial Derivatives for a Scalar Function*
4. *Multivariate* Chain Rule
5. Jacobian: *Collection of Gradients for a Vector Function*
6. Hessian: *Collection of Second-order Partial Derivatives*
7. Gradients for a *Matrix* Function

Derivative: *Differentiation of univariate function*

- Derivative of $f(x)$ 기울기

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

x a scalar



- Useful Formula

Product Rule: $(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$ 곱미분

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: $(g(f(x)))' = (g \circ f)'(x) = g'(f(x))f'(x)$ $= \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial x}$

합성함수의 미분

그냥 같은말

$x \rightarrow \boxed{f} \xrightarrow{f(x)} \boxed{g} \rightarrow g(f(x))$

$=$

$x \rightarrow \boxed{g \circ f} \rightarrow (g \circ f)(x)$

Derivative (cont'd)

Derivatives of Common Functions

	Function $f(x)$	Derivative with respect to x
1	a	0
2	x	1
3	ax	a
4	x^2	$2x$
5	x^a	ax^{a-1}
6	a^x	$\log(a)a^x$
7	$\log(x)$	$1/x$
8	$\log_a(x)$	$1/(x\log(a))$
9	$\sin(x)$	$\cos(x)$
10	$\cos(x)$	$-\sin(x)$
11	$\tan(x)$	$\sec^2(x)$

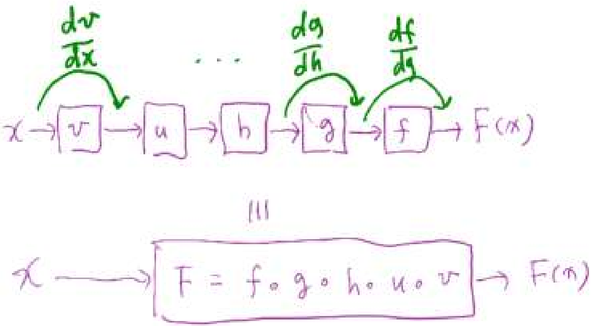
Chain Rule

◆ Composition of Functions

$$F(x) = f(g(x)).$$

$$F'(x) = f'(g(x))g'(x).$$

$$\frac{d}{dx}[f(g(x))] = \frac{df}{dg} \cdot \frac{dg}{dx}.$$



$$F(x) = f(g(h(u(v(x))))).$$

$$\boxed{\begin{aligned} \frac{dF}{dx} &= \frac{d}{dx}F(x) = \frac{d}{dx}f(g(h(u(v(x))))) \\ &= \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{du} \cdot \frac{du}{dv} \cdot \frac{dv}{dx} \end{aligned}}$$

<u>Example:</u> $f(x) = \log(\sqrt{x})$ $g(x)$ 을 $\sqrt{x}$

$$\frac{df}{dx} = \frac{d}{dg}\log(g) \cdot \frac{d}{dx}\sqrt{x}$$

$$= \frac{1}{\sqrt{x}} \cdot \frac{1}{2\sqrt{x}} = \frac{1}{2x}$$

Gradient: *Differentiation of multivariate function*

다변량 함수

- Partial Derivatives of $f(x)$

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

f depends on
one or more variables
vector $\rightarrow$ $\boldsymbol{x} \in \mathbb{R}^n$

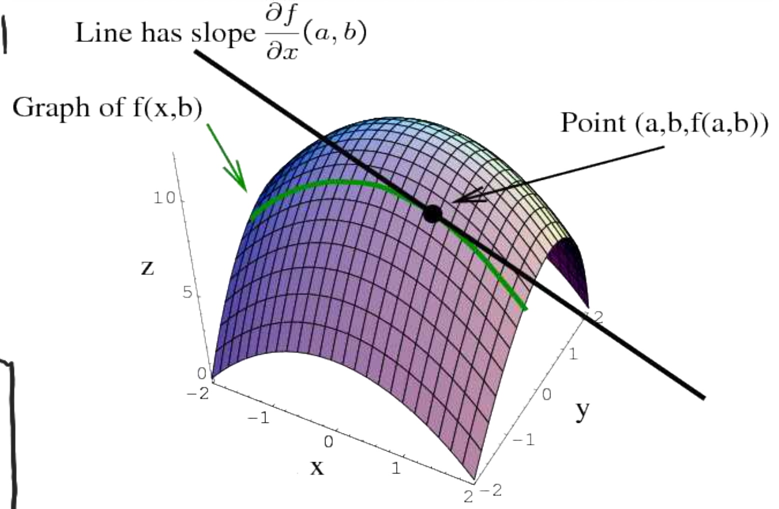
변수마다 미분한거
벡터로 쌓음

$$\frac{\partial f}{\partial x_1} = \lim_{h \to 0} \frac{f(x_1 + h, x_2, \ldots, x_n) - f(\boldsymbol{x})}{h}$$

$$\vdots$$

$$\frac{\partial f}{\partial x_n} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_{n-1}, x_n + h) - f(\boldsymbol{x})}{h}$$

$$\begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix}$$



Gradient: (cont'd)

R

- **Gradient**: Collection of the Partial Derivatives

$$\boxed{\nabla_{\underline{x}} f(\underline{x}) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}} = \frac{d}{d\boldsymbol{x}} f(\boldsymbol{x})$$

Product Rule: $$\frac{\partial}{\partial \boldsymbol{x}}\left(f(\boldsymbol{x})g(\boldsymbol{x})\right) = \frac{\partial f}{\partial \boldsymbol{x}} g(\boldsymbol{x}) + f(\boldsymbol{x}) \frac{\partial g}{\partial \boldsymbol{x}}$$

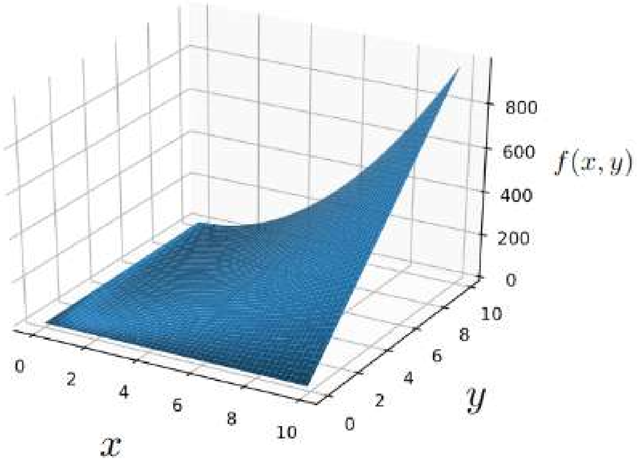
Sum Rule: $$\frac{\partial}{\partial \boldsymbol{x}}\left(f(\boldsymbol{x}) + g(\boldsymbol{x})\right) = \frac{\partial f}{\partial \boldsymbol{x}} + \frac{\partial g}{\partial \boldsymbol{x}}$$

Chain Rule: $$\frac{\partial}{\partial \boldsymbol{x}}(g \circ f)(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}}\left(g(f(\boldsymbol{x}))\right) = \frac{\partial g}{\partial f} \frac{\partial f}{\partial \boldsymbol{x}}$$

Example:

$$f(x, y) = x^2 y + y$$

$$\nabla f(x, y) = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix} = \begin{bmatrix} 2xy \\ x^2 + 1 \end{bmatrix}$$



Multivariate Chain Rule

Two Variables Case

$$f(g(x), h(x))$$

$$\frac{d}{dx}[f(g(x), h(x))] = \frac{\partial f}{\partial g} \cdot \frac{dg}{dx} + \frac{\partial f}{\partial h} \cdot \frac{dh}{dx}$$

Example:

$$f(g,h) = g^2 h + h$$

$$\text{where } g(x) = 3x, \ \text{ and } h(x) = x^2$$

$$\frac{\partial f}{\partial g} = 2gh \qquad \frac{\partial f}{\partial h} = g^2 + 1$$

$$\frac{dg}{dx} = \frac{d}{dx}3x = 3 \qquad \frac{dh}{dx} = \frac{d}{dx}x^2 = 2x$$

$$\cancel{\frac{d}{dx}[f(g(x))]} = [2gh \cdot 3] + [(g^2+1) \cdot 2x]$$

$$= 2xg^2 + 6gh + 2x$$

$$\frac{d}{dx}[f(g(x), h(x))]$$

$f(\mathbf{v}(x))$ For $\mathbf{v}(x) = \begin{bmatrix} g(x) \\ h(x) \end{bmatrix}$

v(x)

$$\frac{df(\boldsymbol{v})}{dx} = \frac{\partial f(\boldsymbol{v})}{\partial \boldsymbol{v}} \odot \frac{\partial \boldsymbol{v}}{\partial x} = \nabla_{\boldsymbol{v}} f(\boldsymbol{v}) \odot \frac{\partial \boldsymbol{v}}{\partial x}$$

$$= \begin{bmatrix} \partial f / \partial g \\ \partial f / \partial h \end{bmatrix} \odot \begin{bmatrix} dg/dx \\ dh/dx \end{bmatrix} = \frac{\partial f}{\partial g} \cdot \frac{dg}{dx} + \frac{\partial f}{\partial h} \cdot \frac{dh}{dx}$$

Jacobian: *Gradients of vector-valued function*

R

◆ Multivariate *Vector-Valued* Function

$$\underbrace{\mathbf{f}}_{\text{vector}}(\underbrace{x_1, x_2, \ldots, x_n}_{\text{vector}}) = \begin{bmatrix} f_1(x_1, x_2, x_3, \cdots x_n) \\ f_2(x_1, x_2, x_3, \cdots x_n) \\ f_3(x_1, x_2, x_3, \cdots x_n) \\ \vdots \\ f_m(x_1, x_2, x_3, \cdots x_n) \end{bmatrix}$$

Input, output 다 벡터
→ vector function

Example: $f_1(x)$

$\mathbf{f}: \mathbb{R}^2 \mapsto \mathbb{R}^3$

$$\mathbf{f}(\mathbf{x}) = (\underbrace{2x_1 + x_2^2}_{= f_1(x)}, \underbrace{-x_1^2 + 3x_2}_{= f_2(x)}, \underbrace{4x_1x_2}_{= f_3(x)})^{\mathrm{T}}$$

$$\mathbf{J} = \begin{pmatrix} 2 & 2x_2 \\ -2x_1 & 3 \\ 4x_2 & 4x_1 \end{pmatrix} \qquad \mathbf{J}|_{(2,1)^{\mathrm{T}}} = \begin{pmatrix} 2 & 2 \\ -4 & 3 \\ 4 & 8 \end{pmatrix}$$

◆ *Jacobian*: Gradients of Vector-Valued Function

$$\mathbf{J}(\boldsymbol{f}(\boldsymbol{x})) = \nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) = \frac{d\boldsymbol{f}(\boldsymbol{x})}{d\boldsymbol{x}}$$

- Matrix of the partial derivatives

$$J(x_1, x_2, x_3, \cdots x_m) = \begin{bmatrix} \boxed{\frac{\partial f_1}{\partial x_1} \quad \frac{\partial f_1}{\partial x_2} \quad \frac{\partial f_1}{\partial x_3} \quad \cdots \quad \frac{\partial f_1}{\partial x_n}} \\ \frac{\partial f_2}{\partial x_1} \quad \frac{\partial f_2}{\partial x_2} \quad \frac{\partial f_2}{\partial x_3} \quad \cdots \quad \frac{\partial f_2}{\partial x_n} \\ \frac{\partial f_3}{\partial x_1} \quad \frac{\partial f_3}{\partial x_2} \quad \frac{\partial f_3}{\partial x_3} \quad \cdots \quad \frac{\partial f_3}{\partial x_n} \\ \vdots \quad \vdots \quad \vdots \quad \ddots \quad \vdots \\ \frac{\partial f_m}{\partial x_1} \quad \frac{\partial f_m}{\partial x_2} \quad \frac{\partial f_m}{\partial x_3} \quad \cdots \quad \frac{\partial f_m}{\partial x_n} \end{bmatrix} \begin{array}{l} (\nabla f_1)^{\top} \\ \\ \\ \\ \\ \end{array} = \begin{bmatrix} \nabla_x f_1(x)^{T} \\ \vdots \\ \nabla_x f_m(x)^{T} \end{bmatrix}$$

↓

m x n 행렬

Hessian: 2^{nd}–order differentiation of *multivariate* function

R

◆ **Partial Derivatives of Gradient**

2차미분 1차 미분

2차 미분

$$\mathbf{H}(f(\boldsymbol{x})) = \frac{\partial}{\partial \boldsymbol{x}} \underline{\nabla_{\boldsymbol{x}} f(\boldsymbol{x})} = \mathbf{J}(\underline{\nabla_{\boldsymbol{x}} f(\boldsymbol{x})})$$

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

↳ 대칭 행렬

Note that the Hessian is always symmetric, since

$$\frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \frac{\partial^2 f(x)}{\partial x_j \partial x_i}.$$

Example:

$$f(\mathbf{x}) = f(x_1, x_2)$$

$$= \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1 x_2 + (-4 + 4x_2^2)x_2^2$$

x_1 2번

x_1 1번, x_2 1번

$$\mathbf{H} = \begin{pmatrix} 10x_1^4 - 25.2x_1^2 + 8 & 1 \\ 1 & 48x_2^2 - 8 \end{pmatrix}$$

x_2 2번

$$\mathbf{H}|_{(0,1)^T} = \begin{pmatrix} 8 & 1 \\ 1 & 40 \end{pmatrix}$$

Useful Formula

R

For Linear Functions

$$\nabla_{\underline{x}} b^T \underline{x} = \underline{b} \iff \frac{d}{dx} ax = a$$

$$\nabla^2_{\underline{x}} b^T \underline{x} = 0 \iff \frac{d^2}{dx^2} ax = 0$$

proof)

For $x \in \mathbb{R}^n$, let $f(x) = b^T x$ for some known vector $b \in \mathbb{R}^n$. Then

$$f(x) = \sum_{i=1}^{n} b_i x_i \qquad \frac{\partial f(x)}{\partial x_k} = \frac{\partial}{\partial x_k} \sum_{i=1}^{n} b_i x_i = b_k.$$

대칭행렬

$$\nabla_{\underline{x}} \underline{x}^T A \underline{x} = 2A\underline{x} \text{ (if } A \text{ symmetric)} \iff \frac{d}{dx} ax^2 = 2ax$$

$$\nabla^2_{\underline{x}} \underline{x}^T A \underline{x} = 2A \text{ (if } A \text{ symmetric)} \iff \frac{d^2}{dx^2} ax^2 = 2a$$

※ If A is not symmetric,

$$\nabla_{\underline{x}} \underline{x}^T A \underline{x} = (A + A^T)\underline{x}$$

proof)

$f(x) = x^T A x$ for $A \in \mathbb{S}^n$. $\qquad f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} x_i x_j.$

$$\begin{aligned}
\frac{\partial f(x)}{\partial x_k} &= \frac{\partial}{\partial x_k} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} x_i x_j \\
&= \frac{\partial}{\partial x_k} \left[\sum_{i \neq k} \sum_{j \neq k} A_{ij} x_i x_j + \sum_{i \neq k} A_{ik} x_i x_k + \sum_{j \neq k} A_{kj} x_k x_j + A_{kk} x_k^2 \right] \\
&= \sum_{i=1}^{n} A_{ik} x_i + \sum_{j=1}^{n} A_{kj} x_j = 2 \sum_{i=1}^{n} A_{ki} x_i
\end{aligned}$$

Useful Formula (cont'd)

R

For Linear Functions

$$\nabla_{\underline{x}} \underline{b}^T \underline{x} = \underline{b}$$
$$\nabla^2_{\underline{x}} b^T \underline{x} = 0$$

$$\nabla_{\underline{x}} A\underline{x} = \nabla_{\underline{x}} \begin{bmatrix} a_1^T \underline{x} \\ \vdots \\ a_n^T \underline{x} \end{bmatrix} = \begin{bmatrix} - a_1^T - \\ \vdots \\ - a_n^T - \end{bmatrix} = A$$

$$A = \begin{bmatrix} - a_1^T - \\ \vdots \\ - a_n^T - \end{bmatrix}$$

$$\nabla_{\underline{x}} \begin{bmatrix} f_1(\underline{x}) \\ \vdots \\ f_n(\underline{x}) \end{bmatrix} \quad \text{[illegible] (Jacobian)} \quad = \begin{bmatrix} \nabla f_1(\underline{x})^T \\ \vdots \\ \nabla f_n(\underline{x})^T \end{bmatrix}$$

- $$\nabla_{\underline{x}} \underline{x}^T A\underline{x} = 2A\underline{x} \text{ (if } A \text{ symmetric)}$$
  $$\nabla^2_{\underline{x}} \underline{x}^T A\underline{x} = 2A \text{ (if } A \text{ symmetric)}$$

$$\nabla_{\underline{x}} \|\underline{x}\|_2^2 = \nabla_{\underline{x}} \underline{x}^T \underline{x} = 2\underline{x}$$
$$(= 2I\underline{x} = 2\underline{x})$$

$$\|X\|_2^2 = X^T X$$

※ If A is not symmetric,

$$\nabla_{\underline{x}} \underline{x}^T A \underline{x} = (A + A^T)\underline{x}$$
$$\nabla^2_{\underline{x}} \underline{x}^T A \underline{x} = (A + A^T)$$

Exercises :

$$\nabla_{\underline{x}} (\underline{x}^T A^T A \underline{x}) = 2A^T A \underline{x}$$
$$\nabla_{\underline{x}} (\underbrace{\underline{b}^T A}_{\triangleq \underline{c}^T} \underline{x}) = \nabla_{\underline{x}} \underline{c}^T \underline{x} = \underline{c} = (\underline{b}^T A)^T = A^T \underline{b}$$

↳ 행 벡터

Gradient of a Matrix variables

Suppose that $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ is a function that takes as input a matrix A of size $m \times n$ and returns a real value. Then the ***gradient*** of f (with respect to $A \in \mathbb{R}^{m \times n}$) is the matrix of partial derivatives, defined as:

스칼라 값

$$\nabla_A f(A) \in \mathbb{R}^{m \times n} = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \frac{\partial f(A)}{\partial A_{12}} & \cdots & \frac{\partial f(A)}{\partial A_{1n}} \\ \frac{\partial f(A)}{\partial A_{21}} & \frac{\partial f(A)}{\partial A_{22}} & \cdots & \frac{\partial f(A)}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(A)}{\partial A_{m1}} & \frac{\partial f(A)}{\partial A_{m2}} & \cdots & \frac{\partial f(A)}{\partial A_{mn}} \end{bmatrix}$$

	input Scalar	input Vector
O: S	1변수함수	다변수 → gradient
O: V	X	벡터 → Jacobian

i.e., an $m \times n$ matrix with

$$(\nabla_A f(A))_{ij} = \frac{\partial f(A)}{\partial A_{ij}}$$

ij 번째 원소로 편미분

$f\left(\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}\right)$

ex) 100X100 이미지를 10000 차원 벡터로 처리 = 벡터 함수

→ 그대로 처리 (벡터 함수와 원론적으로 같음)

2. Iterative Optimization & Gradient Descent

1. **ML as an Optimization Problem**
2. **Iterative Optimization**
   - [illegible]
   - [illegible]
   - [illegible]
3. **Gradient Descent (GD)**
4. **Stochastic Gradient Descent (SGD)**
5. **Minibatch Gradient Descent (Minibatch GD)**

ML as an Optimization Problem

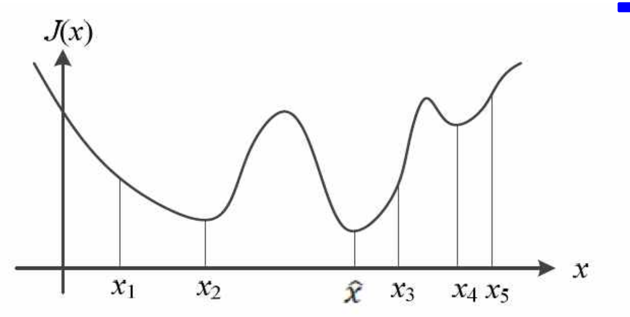
- 기계 학습이 해야 할 일을 식으로 정의하면,

  주어진 Cost Function $J(\Theta)$에 대해 θ: 파라미터(벡터, 행렬)

  $J(\Theta)$를 최소로 하는 최적해 $\hat{\Theta}$을 찾아라. 즉, $\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} J(\Theta)$

  $\min J(\theta)$: $J(\theta)$의 최솟값

  $\operatorname{argmin} J(\theta)$: $J(\theta)$가 최소일 때의 θ값



- Global Optimum vs. Local Optima
  - $\hat{x}$은 전역 최적해
  - x_2와 x_4는 지역 최적해 (근처에서만 최소)

Iterative Optimization – *General Principles*

- Training Dataset: $\mathcal{D}$ → 이진 분류
  - E.g., for binary classification case ($y \in \{0, 1\}$),

$$\mathcal{D} = (\langle \mathbf{x}^{[1]}, y^{[1]} \rangle, \langle \mathbf{x}^{[2]}, y^{[2]} \rangle, \dots, \langle \mathbf{x}^{[n]}, y^{[n]} \rangle) \in (\mathbb{R}^m \times \{0,1\})^n$$

데이터 레이블 / 벡터 스칼라

- Model & Predicted Output: $\hat{y} = h_{\theta}(x)$

linear model

$y = \theta_0 + \theta_1 x$

- Cost Function: $J(\boldsymbol{\theta})$

- General Principles
  1) Initialize parameters ($\boldsymbol{\theta}$) $\mathcal{D}$ → 일단 랜덤하게 초기화
  2) For every training **epoch** ( ): → 데이터를 한번씩 봄?
     - ★ For every [some set of training set]:
       - ① Predict output ($\hat{y}$) & Calculate the cost ($J(\boldsymbol{\theta})$)
       - ② If the cost is satisfactory, then terminates. ★
       - ③ Otherwise, ***update parameters*** ($\boldsymbol{\theta}$) and repeat

만족스러울때까지

↳ 어떻게 Update 할거냐?

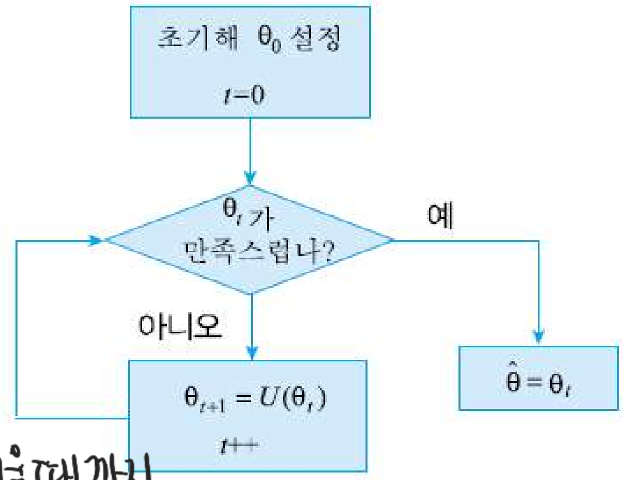


그림 11.6 최적해를 찾기 위한 반복 알고리즘

Iterative Optimization – *Parameter Update*

- Typical Form of Parameter Update

1) $J(\boldsymbol{\theta})$가 작아지도록 $\Delta\boldsymbol{\theta}$를 구한다.
2) $\boldsymbol{\theta}$를 업데이트 한다: $\boldsymbol{\theta} = \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$ → Optimization 방법
   - E.g., Gradient Descent: $\Delta\boldsymbol{\theta} = -\alpha\, \nabla J(\boldsymbol{\theta})$

     → Learning Rate

Iterative Optimization – *Learning Modes*

Training Dataset

$$\mathcal{D} = (\langle \mathbf{x}^{[1]}, y^{[1]} \rangle, \langle \mathbf{x}^{[2]}, y^{[2]} \rangle, \ldots, \langle \mathbf{x}^{[n]}, y^{[n]} \rangle) \in (\mathbb{R}^m \times \{0,1\})^n$$

Learning Modes (Update Modes)

- ***On-line*** mode: Update parameters for *every training* ***example*** 하나 볼 때마다
- ***Batch*** mode: Update parameters for *every training* ***epoch*** 다 보고 평균
- f ***training set*** → Mini-Batch

1) Initialize parameters ($\boldsymbol{\theta}$)
2) For every training **epoch** ($\mathcal{D}$):
   - ★ For every [***some set of training set***]:
     - ① Predict output ($\hat{y}$) & Calculate the cost ($J(\boldsymbol{\theta})$)
     - ② If the cost is satisfactory, then terminates.
     - ③ Otherwise, ***update parameters*** ($\boldsymbol{\theta}$) and repeat ★.

Gradient Descent (GD)

◆ *Parameter Update*

- *based on the* ***Gradient*** *of the Cost Function:* $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \nabla J(\boldsymbol{\theta})$

◆ *Batch Learning Modes*

- [illegible] *every training* ***epoch***
  - ★ Training set에 속한 모든 training example의 Gradient를 평균한 후 한꺼번에 갱신

알고리즘 2-4 배치 경사 하강 알고리즘(BGD)

입력 : 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 학습률 ρ
출력 : 최적해 $\widehat{\boldsymbol{\Theta}}$

난수를 생성하여 초기해 $\boldsymbol{\Theta}$를 설정한다.
repeat
  $\mathbb{X}$에 있는 샘플의 그레이디언트 $\nabla_1, \nabla_2, \cdots, \nabla_n$을 계산한다.
  $\nabla_{total} = \frac{1}{n}\sum_{i=1,n} \nabla_i$ // 그레이디언트 평균을 계산
  $\boldsymbol{\Theta} = \boldsymbol{\Theta} - \rho\nabla_{total}$
until(멈춤 조건)
$\widehat{\boldsymbol{\Theta}} = \boldsymbol{\Theta}$

업데이트 1번

Stochastic Gradient Descent (SGD)

↳ Online or Mini-Batch

◆ ***On-line** Learning Modes*

- [illegible] *every training* ***example***
  - ★ 각 training example의 Gradient를 계산한 후 즉시 갱신
  - ★ 결과가 training example들의 순서에 의존하지 않도록 example들의 순서를 "**임의로(stochastic)**" 선택한다.
    - shuffling (중복 X)
    - sampling (중복 O)

알고리즘 2-5 스토케스틱 경사 하강 알고리즘(SGD)

입력 : 훈련집합 $\mathbb{X}$와 $\mathbb{Y}$, 학습률 ρ

출력 : 최적해 $\hat{\mathbf{\Theta}}$

```
난수를 생성하여 초기해 Θ를 설정한다.
repeat
    𝕏의 샘플의 순서를 섞는다.
    for (i=1 to n)
        i번째 샘플에 대한 그레이디언트 ∇_i를 계산한다.
        Θ = Θ − ρ∇_i
until(멈춤 조건)
Θ̂ = Θ
```

Shuffling 매 epoch마다

업데이트 n번

Sampling

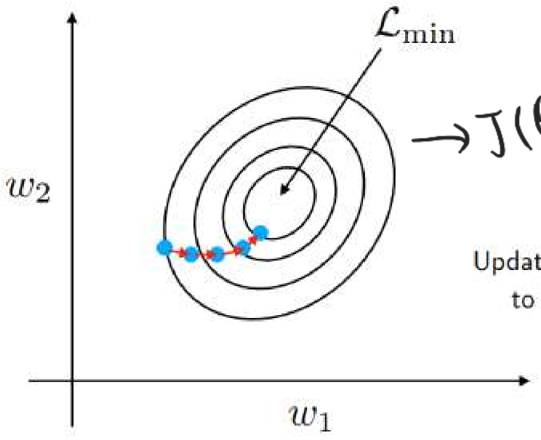
Sampling

다른 방식의 구현 (독립 샘플링)

```
𝕏에서 임의로 샘플 하나를 뽑는다.
뽑힌 샘플의 그레이디언트 ∇를 계산한다.
Θ = Θ − ρ∇
```

GD vs. SGD

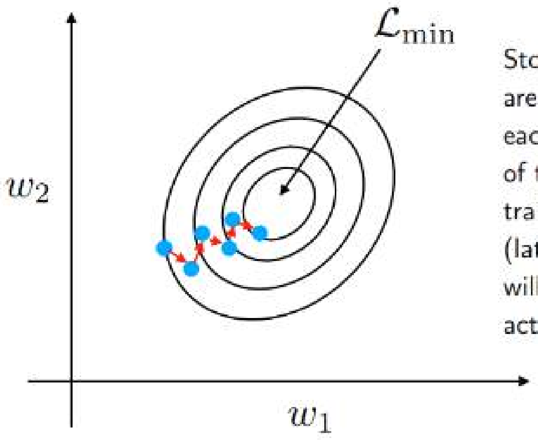
- **Batch GD**



$\rightarrow J(\theta)$

Updates perpendicular
to contour lines

- **SGD** → Deep Learning에서는 더 도움 됨



Stochastic updates
are a bit noisier, because
each batch is an approximation
of the overall loss on the
training set
(later, in deep neural nets, we
will see why noisier updates are
actually helpful)

Minibatch Gradient Descent (Minibatch GD)

- ***Minibatch** Learning Modes* 가장 많이 씀
  - Update parameters for *every* ***subset (minibatch)*** *of* ***training set***
    - ★ Training set을 여러 개의 minibatch로 나누고,
      minibatch에 속한 모든 training example의 Gradient를 평균한 후 한꺼번에 갱신
  - [illegible]와 Batch mode의 중간으로 볼 수 있음

minibatch size: hyperparameter
얘도 shuffle 해야 함

- **Minibatch mode is *most commonly* used in ML & DL**
  - Choosing a subset takes advantage of vectorization (faster than "on-line") 빠르다
  - [illegible] noise가 줄어든다.
  - Makes more updates/epoch than "batch" and thus converges faster

3. Automatic Differentiation

실제 미분

Automatic Differentiation

Automatic Differentiation

derivative of, gradient

- [illegible] *numerically.*
  수치적으로
- Note that it is ***not a formular*** but a procedure and it is ***not symbolic*** differentiation.
- *Backpropagation* is a special case of it.

역전파

Most automatic differentiation systems, including *Autograd*, construct the ***computation graph*** .

It has two modes: ***Reverse mode*** & ***Forward mode***:

$$x \to a \to b \to y$$

합성함수

$y = F(x) = b(a(x))$

reverse mode $$\frac{dy}{dx} = \left(\frac{dy}{db}\frac{db}{da}\right)\frac{da}{dx},$$

forward mode $$\frac{dy}{dx} = \frac{dy}{db}\left(\frac{db}{da}\frac{da}{dx}\right).$$

https://arxiv.org/pdf/1502.05767.pdf

https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/slides/lec10.pdf

Automatic Differentiation – *Illustration*

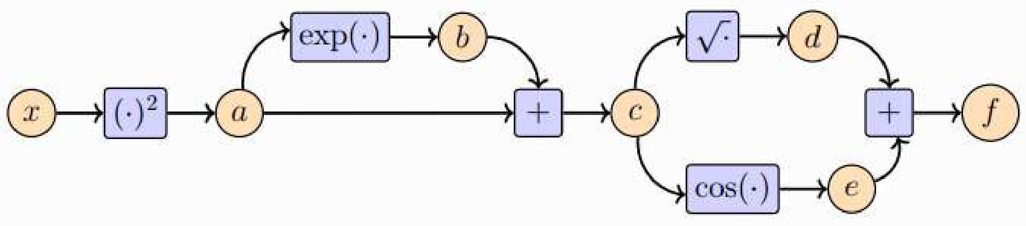
Consider a function

$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos\left(x^2 + \exp(x^2)\right)$$

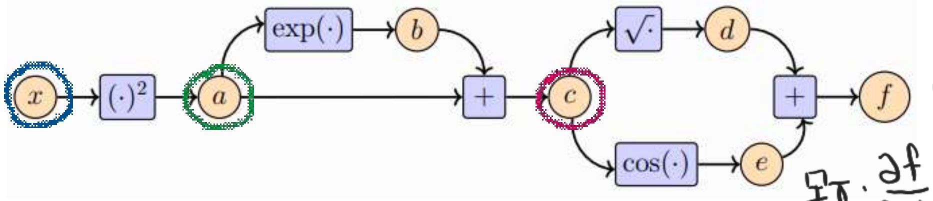
1) Build the *Computation Graph*

노드

- If we were to implement the function on a computer,



Automatic Differentiation – *Illustration (cont'd)*



(Foward)

목표: $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a} \cdot \frac{\partial a}{\partial x}$

2) Calculate the values and the derivatives

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

$$\frac{\partial a}{\partial x} = 2x \qquad \frac{\partial b}{\partial a} = \exp(a)$$

$$\frac{\partial d}{\partial c} = \frac{1}{2\sqrt{c}} \qquad \frac{\partial e}{\partial c} = -\sin(c)$$

각 편미분을 변수로 생각

- we observe that the computation required for calculating the derivatives is ***of similar complexity*** as the computation of the function itself.

3) Apply *Chain Rule* (in reverse mode)

$$\frac{\partial f}{\partial c} = \frac{\partial d}{\partial c} + \frac{\partial e}{\partial c}$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial c} \cdot \frac{\partial c}{\partial a} = \frac{\partial f}{\partial c} \cdot \left(\frac{\partial b}{\partial a} + 1\right)$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a} \cdot \frac{\partial a}{\partial x}$$

"Backpropagation"

Complexity 는 Foward나 Reserve 나 비슷