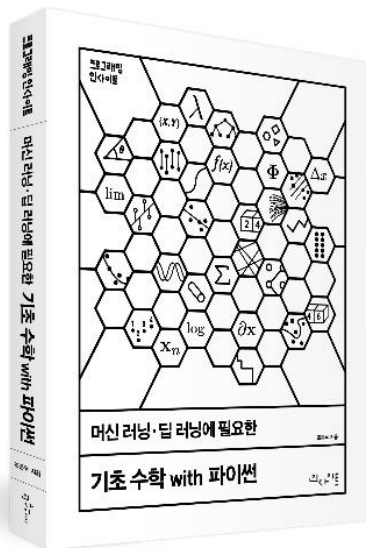
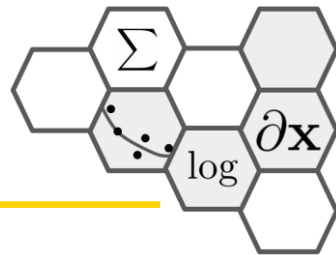


인공지능 이산수학

조준우

metamath@gmail.com

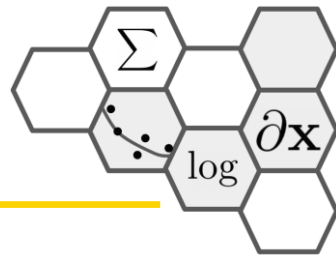
잠깐 홍보 😊



- 2018~2019: 패스트캠퍼스 머신러닝을 위한 기초 수학 강의
- 2020: 머신러닝·딥러닝에 필요한 기초 수학 with 파이썬 출간
- 2021: 서울 이노베이션스퀘어 딥러닝 중급과정 강사 외 다수 강의
- 2022~: 동북권 이노베이션스퀘어 머신러닝·딥러닝 기본과정 강사
- 블로그: metamath1.github.io, 한국어로 찾기 힘든 자료 위주로 정리

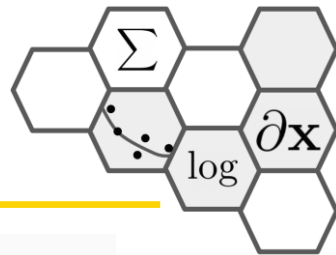
머신러닝·딥러닝에 필요한 기초 수학 with 파이썬,
조준우, 인사이트

What is discrete mathematics?



- Discrete mathematics is the part of mathematics devoted to the study of discrete objects.
- Math courses based on the material studied in discrete mathematics include logic, set theory, number theory, linear algebra, abstract algebra, combinatorics, graph theory, and probability theory (the discrete part of the subject).
- Also, discrete mathematics contains the necessary mathematical background for solving problems in operations research (including discrete optimization), chemistry, engineering, biology, and so on.

Preliminaries for ML and DL



2.1. Data Manipulation

- 2.1.1. Getting Started
- 2.1.2. Operations
- 2.1.3. Broadcasting Mechanism
- 2.1.4. Indexing and Slicing
- 2.1.5. Saving Memory
- 2.1.6. Conversion to Other Python Objects
- 2.1.7. Summary
- 2.1.8. Exercises

2.2. Data Preprocessing

- 2.2.1. Reading the Dataset
- 2.2.2. Handling Missing Data
- 2.2.3. Conversion to the Tensor Format
- 2.2.4. Summary
- 2.2.5. Exercises

2.3. Linear Algebra

- 2.3.1. Scalars
- 2.3.2. Vectors
- 2.3.3. Matrices
- 2.3.4. Tensors
- 2.3.5. Basic Properties of Tensor Arithmetic

2.4. Calculus

- 2.4.1. Derivatives and Differentiation
- 2.4.2. Partial Derivatives
- 2.4.3. Gradients
- 2.4.4. Chain Rule
- 2.4.5. Summary
- 2.4.6. Exercises

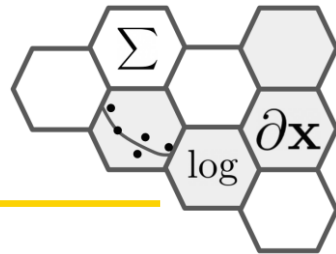
2.5. Automatic Differentiation

- 2.5.1. A Simple Example
- 2.5.2. Backward for Non-Scalar Variables
- 2.5.3. Detaching Computation
- 2.5.4. Computing the Gradient of Python Control Flow
- 2.5.5. Summary
- 2.5.6. Exercises

2.6. Probability

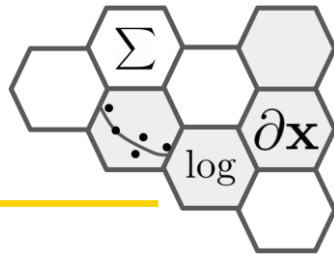
- 2.6.1. Basic Probability Theory
- 2.6.2. Dealing with Multiple Random Variables
- 2.6.3. Expectation and Variance
- 2.6.4. Summary
- 2.6.5. Exercises

Discrete? Continuous?



- Discrete: Countable
 - The number of students in a class
 - The results of rolling dice
- Continuous: Uncountable
 - A person's height and weight
 - Time in a race

수업 진행 개요



집합과 수 체계

- 변수의 종류
- 변수 전처리

논리연산

- 필요조건과 충분조건
- 퍼셉트론

함수

- 다변수 함수와 함수의 합성
- 인공지능과 함수의 관계

행렬

- 행렬곱
- 인공신경망의 행렬표현

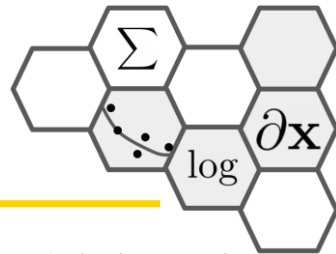
그래프

- 계산 그래프
- 자동 미분
- 그래프 인공신경망

확률

- 평균, 분산, 표준편차
- 공분산과 상관계수
- 인공신경망과 확률분포의 관계

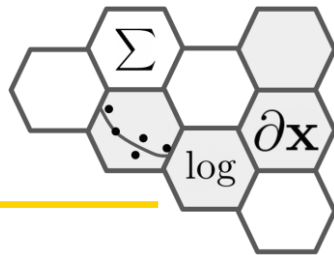
머신러닝이란?



<https://medium.com/@ezralazuardy/how-machine-learn-c2f73f60ef14>

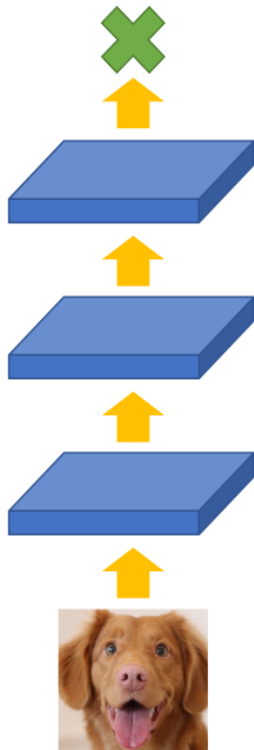
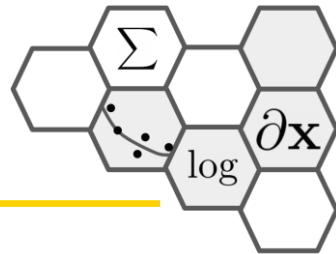
- 넓은 의미 : 컴퓨터를 이용한 문제 해결에 있어서 지식기반, 규칙 기반 방법이 아닌 데이터와 범용 알고리즘으로 퍼포먼스를 개선하는 방법
- Samuel, A. L. (1959) : "Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort"
"Some Studies in Machine Learning Using the Game of Checkers" in IBM Journal of Research and Development (Volume:3, Issue: 3), p. 210, 기계학습, 오일석, 한빛미디어
- Tom Mitchell (1998) : Well-posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P , if its performance on T , as measured by P , improves with experience E .
Mitchell, T., 1997, Machine Learning, McGraw Hill
- 좁은 의미 : 주어진 데이터를 가장 잘 표현하는 함수를 찾는 것

머신러닝 분류: 지도학습

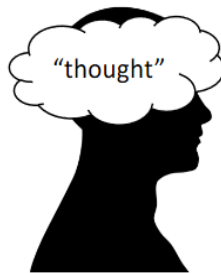


- 지도학습Supervised learning
 - 회귀 문제Regression : 선형 회귀Linear Regression
 - 정답 : 연속된 실수
 - 예 : 대지면적에 따른 집값, 시간에 따른 트랜지스터 집적 개수, 기온에 따른 빙과류 판매량
 - 분류 문제Classification : 로지스틱 회귀Logistic Regression
 - 정답 : (0,1) 또는 (0, 1, \dots , K)
 - 예 : 개-고양이 분류, 양성종양-악성종양 분류, \dots

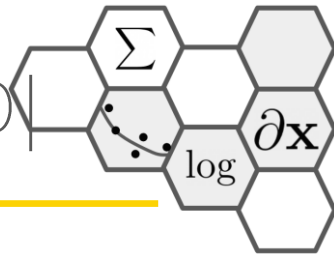
그래서 딥러닝의 도대체 뭐지?



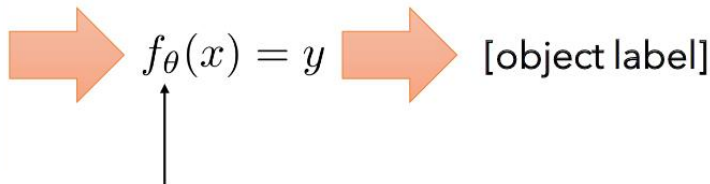
- Machine learning with **multiple layers** of **learned representations**
- The **function** that represents the transformation from input to internal representation to output is usually a deep neural network
 - This is a bit circular, because almost all **multi-layer parametric** functions with **learned parameters** can be called neural networks (more on this later)
- The parameters for every layer are usually (**but not always!**) trained with respect to the overall task objective (**e.g., accuracy**)
 - This is sometimes referred to as **end-to-end** learning



얇은 'Shallow' 학습과 깊은 'Deep' 학습 차이

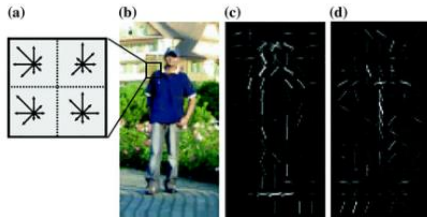


“Shallow” learning



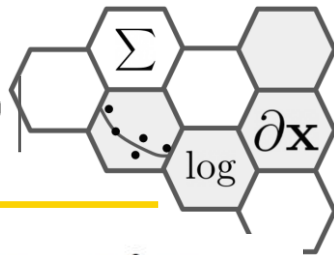
fixed function for extracting *features* from x

$$\phi(x)^T \theta \leq 0$$

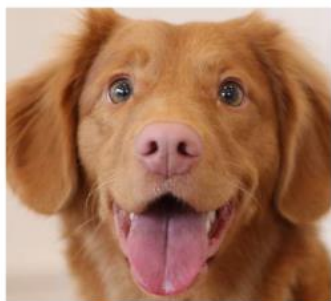


- Kind of a “compromise” solution: don’t hand-program the rules, but hand-program the features
- Learning on top of the features can be simple (just like the 2D example from before!)
- Coming up with good features is very hard!

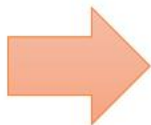
얕은 'Shallow' 학습과 깊은 'Deep' 학습 차이



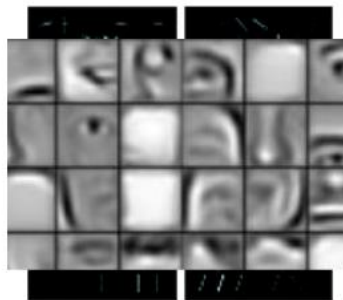
From shallow learning to deep learning



input

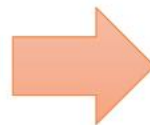


$$\phi(x)$$



~~features~~
learned features

what if we learn parameters here too?



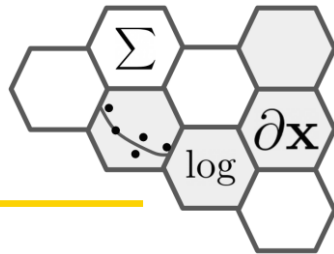
$$\phi(x)^T \theta \leq 0$$







label

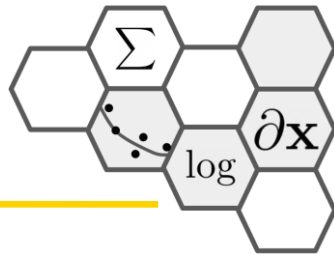
all the **parameters** are here

슬라이드 사용법

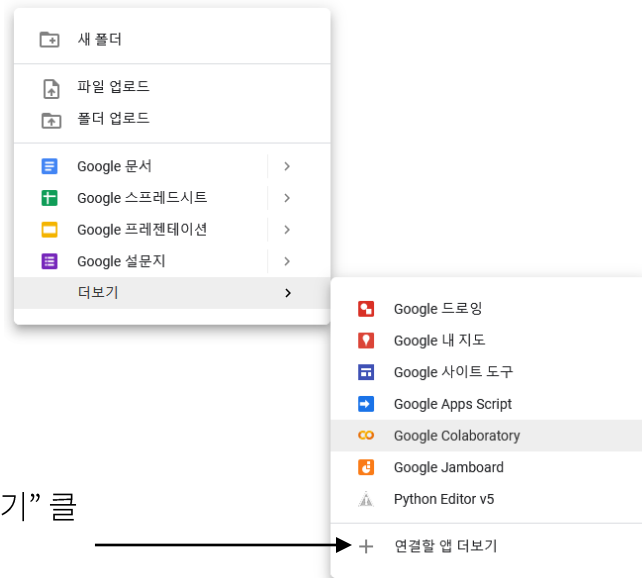


- : From the viewpoint of learning machine learning and deep learning
- : Interactive JavaScript Web Application
-   : google colab 실습 또는 구글 시트
 - 구글 아이디 필요

colab



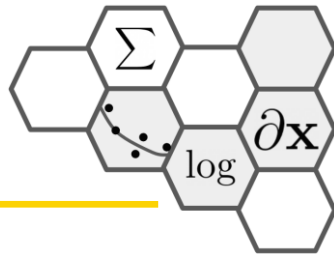
- 소개
 - <https://colab.research.google.com/notebooks/intro.ipynb>
- 장점
 - 구성이 필요하지 않음
 - GPU 무료 액세스
 - 간편한 공유
- 구글 드라이브에서 마우스 오른쪽 클릭
 - Google Colaboratory 클릭



Google Colaboratory가 보이지 않으면 “+ 연결할 앱 더보기” 클릭

‘colab’ 검색하고 설치

colab



Untitled0.ipynb ☆

💬 댓글 👤 공유 ⚙️

파일 수정 보기 삽입 런타임 도구 도움말 [모든 변경사항이 저장됨](#)

+ 코드 + 텍스트

✓ RAM 디스크 수정 가능

```
[1] 1 import numpy as np
```

→ ctrl+enter, shift+enter로 코드 셀 실행

▼ 텍스트 입력가능

```
1 np.__version__
```

```
'1.19.5'
```

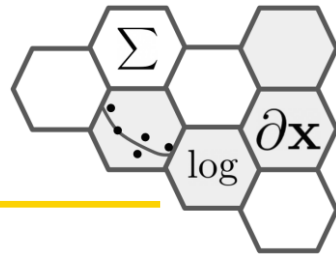
+ 코드

+ 텍스트

```
[ ] 1
```

코드 셀 추가
Ctrl+M B

colab



/content 디렉토리에 파일 업로드

The screenshot displays the Google Colab interface. On the left, the '파일' (Files) sidebar is open, showing a file explorer view of the `/content` directory. It contains a folder named `sample_data` and two files, `bar.txt` and `foo.txt`. An orange circle highlights the 'upload' icon (a document with a plus sign) in the top bar of the sidebar, with an arrow pointing to it from the text above. Another orange circle highlights the 'current directory' icon (a folder) in the sidebar, with an arrow pointing to it from the text below. The main code editor area shows two code cells. The first cell contains Python code for matching a string `s` with a pattern `p`. The second cell, labeled `[91]`, contains a comment in Korean and a file path `/home/metamath/data/oxford-pet/images/train/cat/Abyssinian_1/123.jpg`. The output of the first cell shows 'Match found: Abyssinian_1 Abyssinian_1'.

python-m_.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

RAM 디스크

파일

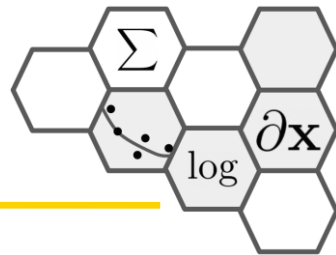
sample_data

bar.txt

foo.txt

현재 실행되는 가상 linux에 /content 디렉토리

colab



colab python-m_ipynb ☆

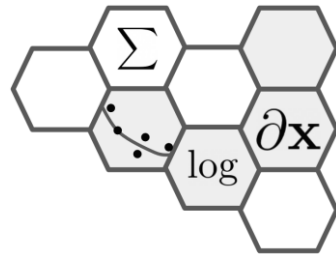
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

파일

- sample_data
- bar.txt
- foo.txt

런타임 메뉴:

- 모두 실행 (Ctrl+F9)
- 이전 셀 실행 (Ctrl+F8)
- 초점이 맞춰진 셀 실행 (Ctrl+Enter)
- 선택항목 실행 (Ctrl+Shift+Enter)
- 이후 셀 실행 (Ctrl+F10)
- 실행 중단 (Ctrl+M |)
- 런타임 다시 시작 (Ctrl+M .) → 컴퓨터를 켜다 켜
- 다시 시작 및 모두 실행 (byssj)
- 런타임 초기화 (apture)
- 런타임 유형 변경 (결에사) → 컴퓨터를 새 컴퓨터로 바꿈
- 세션 관리 (netama)
- 런타임 로그 보기 (w+)_le(pa)



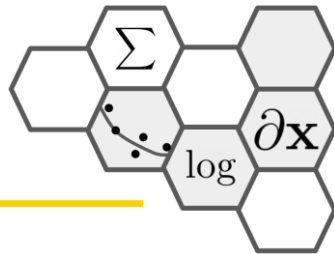
인공지능 이산수학

집합과 수 체계

조준우

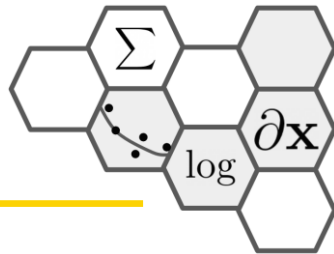
metamath@gmail.com

집합의 개념

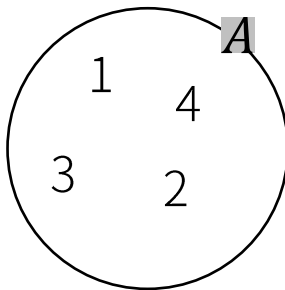


- 집합
 - 어떤 조건에 의해 대상을 분명하게 정할 수 있을 때, 그 대상들의 모임*
- 원소
 - 집합을 이루는 개별 대상
 - $a \in A, a \notin A$
- 집합의 표현
 - 원소나열법
 - $\{1, 2, 3, 4, 5\}$: 순서 상관 없음, $\{2, 3, 1, 4, 5\}$ 와 같음
 - 조건제시법
 - $\{x \mid x \text{는 } 5\text{이하의 자연수}\}$

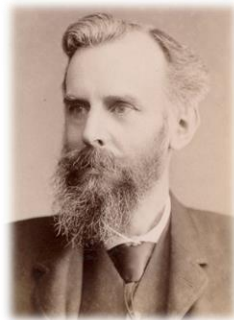
집합의 표현



- 벤 다이어그램
 - 집합을 그림으로 나타낸 것
 - $A = \{1, 2, 3, 4\}$

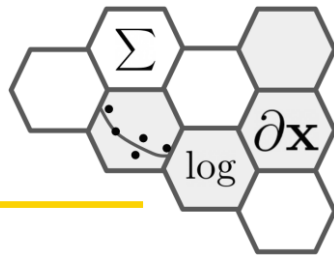


- $B = \{x | x^2 - 4x - 5 = 0\}$ 의 벤 다이어그램
- 원소의 개수 $n(A)$
- 원소가 없는 집합: 공집합 \emptyset



벤 John Venn 1834~1923
Public domain

부분집합

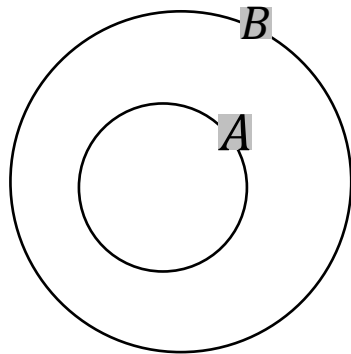


- 집합 A의 모든 원소가 집합 B에 속할 때, A가 집합 B의 부분집합
 $A \subset B$

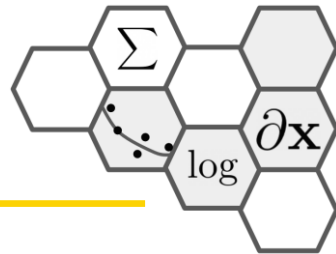
- 집합 A가 집합 B의 부분집합이 아님
 $A \not\subset B$

- 공집합과 자기 자신에 대해
 $A \subset A$ $\emptyset \subset A$

- 부분집합에서 자기 자신이 빠지면 ‘진부분집합’



부분집합

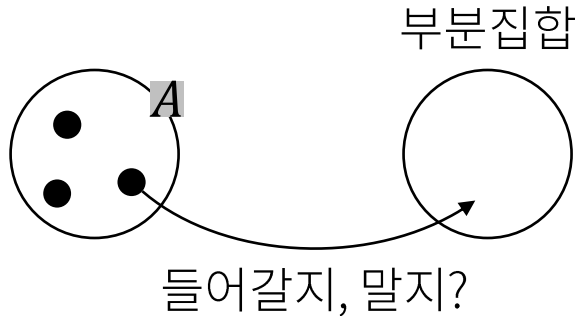


- 다음 집합의 부분집합?

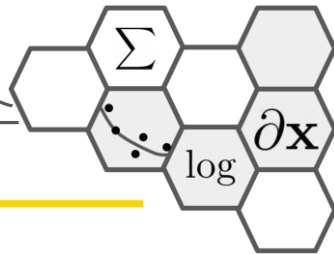
$$A = \{1, 2, 3\}$$

- 부분집합의 개수?

$$2^n$$

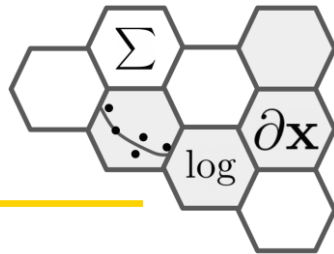


수 체계: 자연수, 정수, 유리수



- 자연수 \mathbb{N}
 - 사물을 셀 때나 순서를 매길 때 사용하는 수, 1, 2, 3, \dots
- 정수 \mathbb{Z}
 - 자연수에 0과 음수를 더한 것으로 $\dots, -2, -1, 0, 1, 2, \dots$ 같은 수
- 유리수 \mathbb{Q}
 - 분자, 분모로 정수를 갖는 분수로 나타낼 수 있는 수
 - Rational number: 이치에 맞는 수?

수 체계: 무리수, 실수



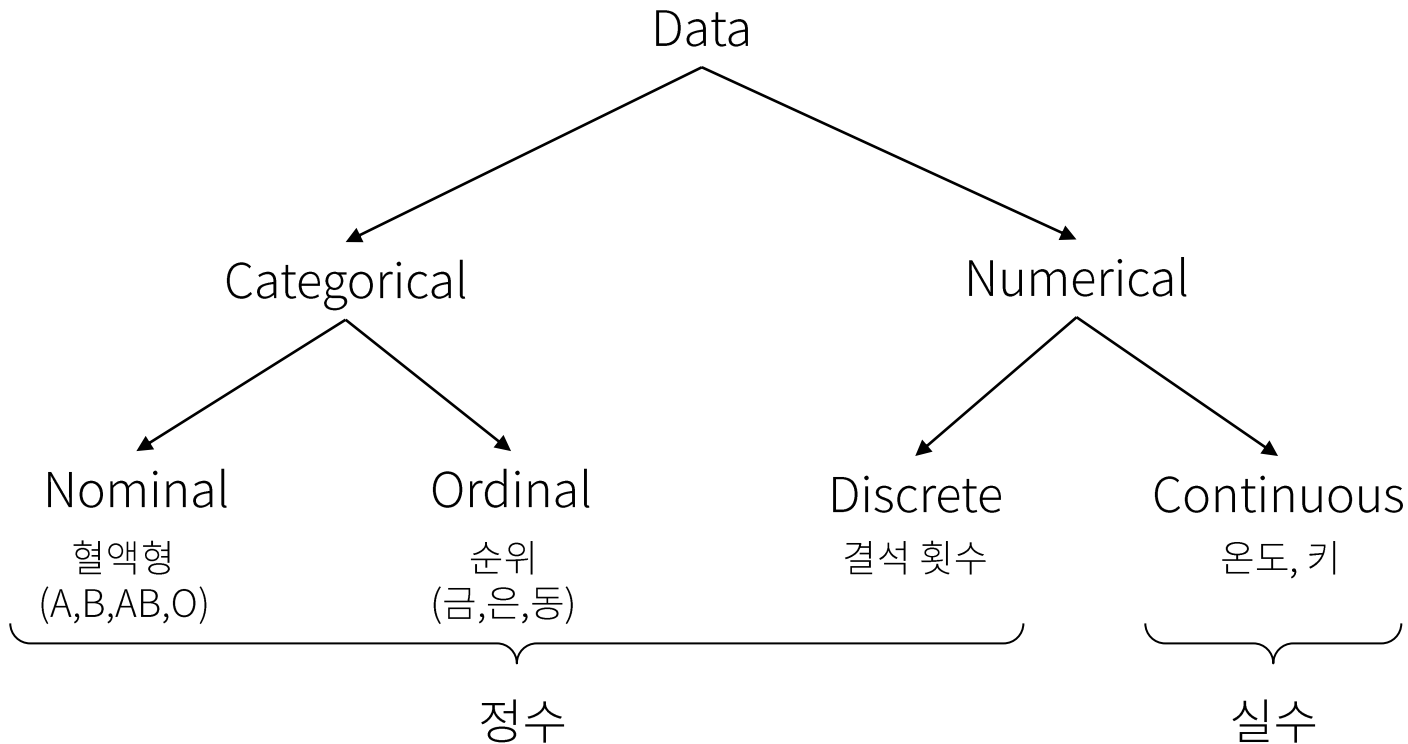
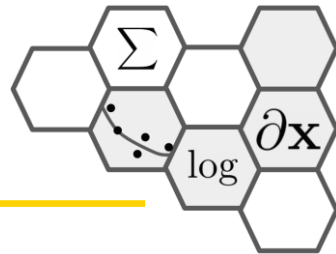
- 무리수 \mathbb{I}

- Irrational number : 이치에 맞지 않는 수?
- ratio: 비율
- ir- 접두어로 반대의 의미 \rightarrow 비율이 없는 수 또는 비율로 나타낼 수 없는 수
- $e: 2.718\cdots, \pi: 3.14\cdots$

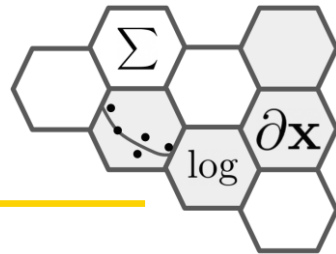
- 실수 \mathbb{R}

- 유리수 + 무리수

👁️: Types of Data



CO 변수 종류 확인



Categorical: Ordinal



	Name	Country	Age	Job	Hand	Height	Capital
0	John	USA	31	Student	L	T	48.35
1	Sabre	France	33	NaN	R	S	150.80
2	Kim	Korea	None	Developer	R	M	99.00
3	Sato	None	40	Chef	B	S	100.00
4	Lee	Korea	36	Professor	L	T	182.30
5	Smith	UK	55	CEO	L	S	1101.65
6	David	USA	NaN	Banker	R	S	131.87
7	Park	Korea	35	Student	R	T	65.80



Numerical: Discrete

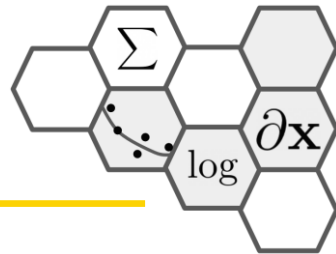


Categorical
Nominal



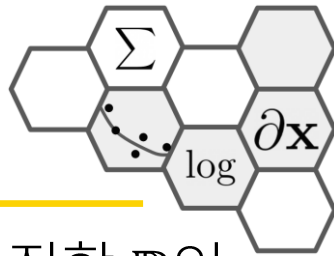
Numerical: Continuous

👁️: 어떤 인코딩이 적합?



	순서 의미 있음	순서 의미 없음
Tree 기반 모델	OrdinalEncoder	OrdinalEncoder or OneHotEncoder
선형 모델 Linear Regression Logistic Regression Support Vector Machine	OrdinalEncoder (단 순서 주의해서)	OneHotEncoder

합의 기호 Σ : 시그마sigma



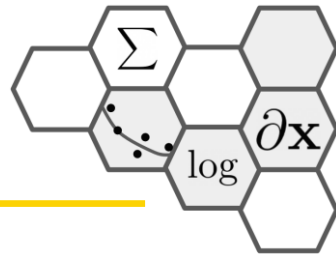
- 수열 : 정의역이 자연수 전체 집합 \mathbb{N} 이고 공역이 실수 전체 집합 \mathbb{R} 인 함수

$$f: \mathbb{N} \rightarrow \mathbb{R}$$

- 수열 $2, 4, 6, 8, \dots$ 는 $f(1) = 2, f(2) = 4, f(3) = 6$ 처럼 숫자의 위치, 출력이 주어진 숫자인 함수
- 수열의 첫째 항부터 제 n 항까지의 합

$$\sum_{k=1}^n a_k = a_1 + a_2 + a_3 + \dots + a_n$$

Σ 의 성질



$$\sum_{k=1}^n (a_k + b_k) = \sum_{k=1}^n a_k + \sum_{k=1}^n b_k$$

$$\sum_{k=1}^n (a_k - b_k) = \sum_{k=1}^n a_k - \sum_{k=1}^n b_k$$

$$\sum_{k=1}^n c a_k = c \sum_{k=1}^n a_k$$

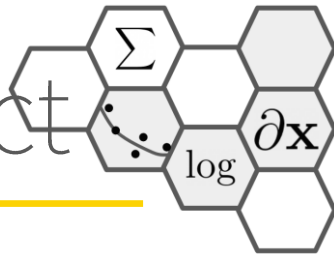
$$\sum_{k=1}^n c = cn$$

$$\sum_{i=1}^m \sum_{j=1}^n a_i b_j = \sum_{i=1}^m a_i \left(\sum_{j=1}^n b_j \right)$$

✓NOTE Σ 활용

딥러닝에서 합의 기호를 3개, 4개씩 겹쳐 사용하는 경우가 많고, 이 상태에서 미분까지 하게 되면???

곱의 기호 \prod : 프로덕트product



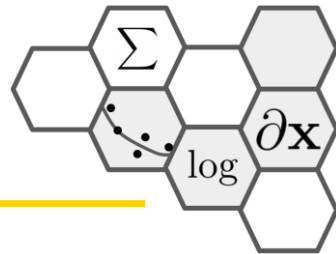
- 합의 기호처럼 순서대로 모두 곱함

$$\prod_{k=1}^n a_k = a_1 \times a_2 \times a_3 \times \cdots \times a_n$$

$$\prod_{k=1}^n c = c^n$$

$$\prod_{k=1}^n a_k b_k = \prod_{k=1}^n a_k \times \prod_{k=1}^n b_k$$

Σ, Π 예제



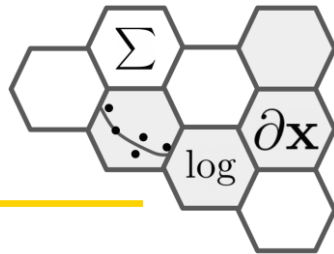
$$2 + 4 + 6 + 8$$

$$\sum_{i=1}^4 2i = 2 \sum_{i=1}^4 i$$

$$\prod_{i=1}^2 \sum_{j=1}^2 x_i y_j$$

$$\prod_{i=1}^2 \sum_{j=1}^2 x_i y_j = (x_1 y_1 + x_1 y_2) \times (x_2 y_1 + x_2 y_2)$$

계승 !: 팩토리얼factorial



- 1부터 n 까지 1씩 증가하는 수열의 곱

$$n! = 1 \times 2 \times 3 \times \cdots \times n = \prod_{k=1}^n k$$

$$0! = 1 \quad \text{🤔}$$

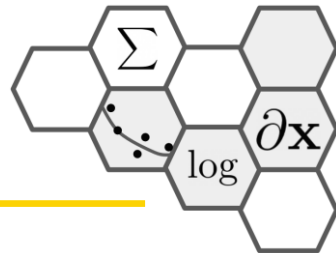
$$n! = (n - 1)! \times n$$

$$1! = (1 - 1)! \times 1 = 1$$

$$1! = 0! \times 1 = 1$$

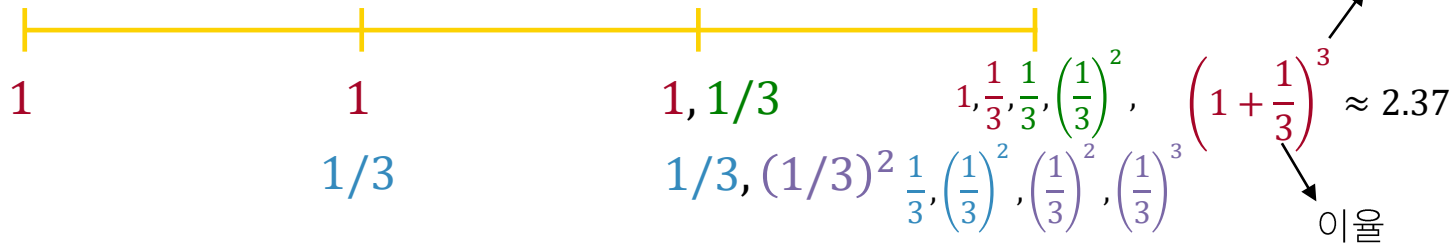
$$\therefore 0! = 1$$

복리 이자



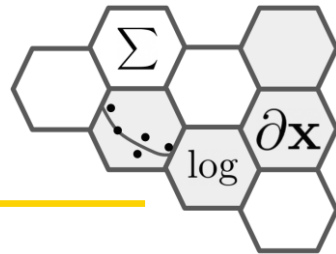
- 특정 기간동안 2배가 붙어나는 예금에서 인출과 입금을 반복

베르누이 JAKOB BERNOULLI (1655~1705)



$$\lim_{x \rightarrow \infty} \left(1 + \frac{1}{x}\right)^x = ?$$

자연상수 e



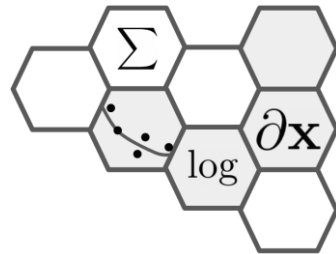
- 야코프 베르누이에 의해 아래식이 수렴함이 발견
- 라이프니츠에 의해 상수 b 로 처음 사용
- 오일러에 의해 e 로 처음 표기

$$\lim_{x \rightarrow 0} (1 + x)^{\frac{1}{x}} = e \approx 2.718$$

$$\lim_{x \rightarrow \infty} \left(1 + \frac{1}{x}\right)^x = e \approx 2.718$$



오일러 LEONHARD EULER (1707~1783)



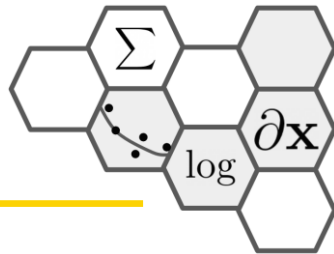
인공지능 이산수학

논리 연산

조준우

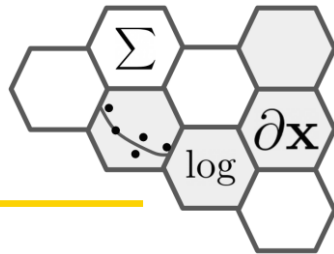
metamath@gmail.com

명제proposition



- 진릿값truth value: 참, 거짓을 나타내는 값
 - True: T, 1
 - False: F, 0
- 명제proposition: 진릿값(참, 거짓)을 부여할 수 있는 문장이나 식
 - 주로 p, q, r
 - $5+4=9$: 참 명제
 - 제주는 한국의 수도다: 거짓 명제
 - 머신러닝은 어렵다: 명제 아님

논리 연산자: 부정

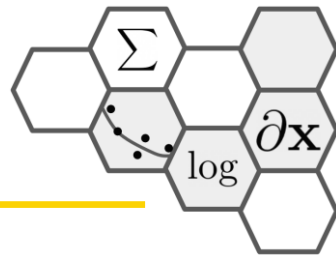


- 논리 연산자: 명제에 적용되는 연산자
- 부정 연산자 \neg
 - 명제 p 에 대해 ‘ p 가 아니다’를 의미하는 연산자
 - 진릿값은 반대

p	$\neg p$
T	F
F	T

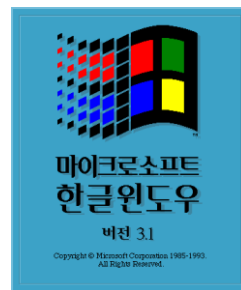
진리표true table

논리 연산자: 논리곱

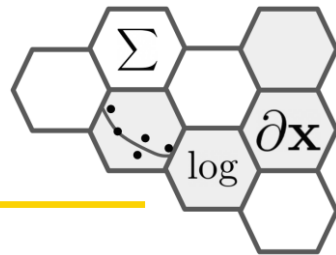


- $p \wedge q$: 명제 p 와 q 에 대해 ‘ p 그리고 q ’를 의미
- p 와 q 모두 참일때만 결과가 참
 - p : 철수의 PC는 컬러 모니터이다.
 - q : 그 PC의 운영체제는 windows 3.1이다.
 - $p \wedge q$: 철수의 PC 모니터는 컬러이고 운영체제는 windows 3.1이다.

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

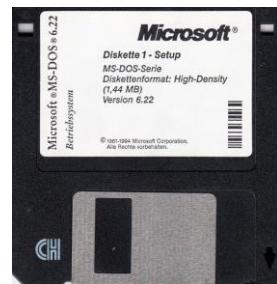


논리 연산자: 논리합

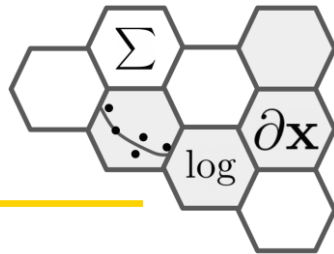


- $p \vee q$: 명제 p 와 q 에 대해 ' p 또는 q '를 의미
- p 와 q 둘 중 하나만 참이면 결과가 참
 - p : 철수의 PC는 컬러 모니터이다.
 - q : 그 PC의 운영체제는 windows 3.1이다.
 - $p \wedge q$: 철수의 PC는 컬러 모니터이거나 운영체제는 windows 3.1이다.

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F



논리 연산자: 배타적 논리합

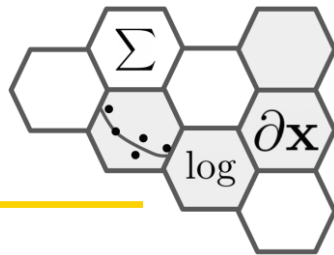


- $p \oplus q$: 명제 p 와 q 에 대해 둘 중 하나만 참일 때 참이고 나머지는 거짓이 되는 연산
 - p : 철수는 한 해 저축으로 차를 산다.
 - q : 철수는 한 해 저축으로 유럽 여행을 간다.
 - $p \oplus q$: 철수는 한 해 저축으로 차를 사거나 유럽 여행을 간다. 하지만 둘 다는 불가능하다.

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

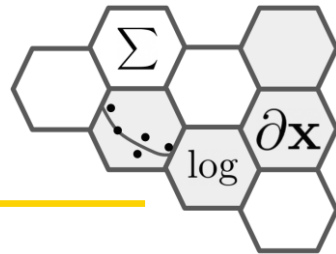


조건과 진리집합



- 조건: x 에 따라 참, 거짓을 판별할 수 있는 명제
 - $x < 10$: 참, 거짓 판단 불가능
 - 전체 집합 U 가 \mathbb{N} 이면 $x = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ 인 경우 참
 - 나머지는 거짓
- 진리집합
 - 조건을 참으로 만드는 원소의 집합
 - $x = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
 - 명제 p 가 거짓이라면 진리집합이 공집합이라는 의미

조건 명제conditional proposition



$$\underline{x = 20 \text{이면 } x^2 = 40 \text{ 이다.}}$$

p

\rightarrow

q

전제premise
가정hypothesis

결론conclusion
결과consequence

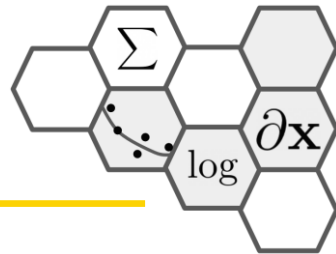
implies



힐베르트David Hilbert 1862~1943

Public domain

조건 명제|conditional proposition



$$\underline{x = 2 \text{이면 } x^2 = 4 \text{ 이다.}}$$

$$p \rightarrow q$$

전제premise
가정hypothesis

진리집합

P

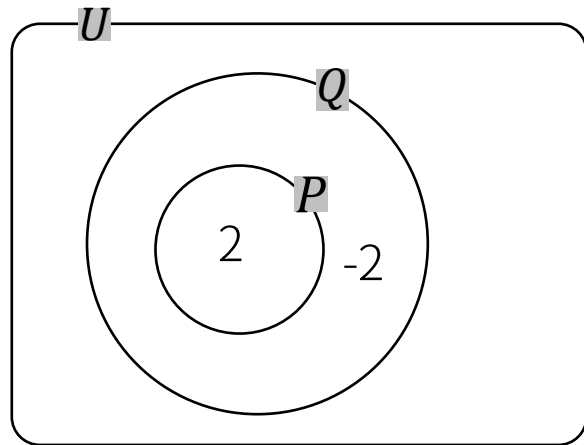
결론conclusion
결과consequence

진리집합

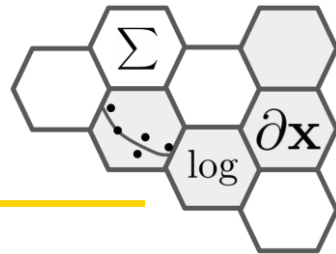
Q

$P \subset Q$ 이면 $p \rightarrow q$ 는 참

$P \not\subset Q$ 이면 $p \rightarrow q$ 는 거짓

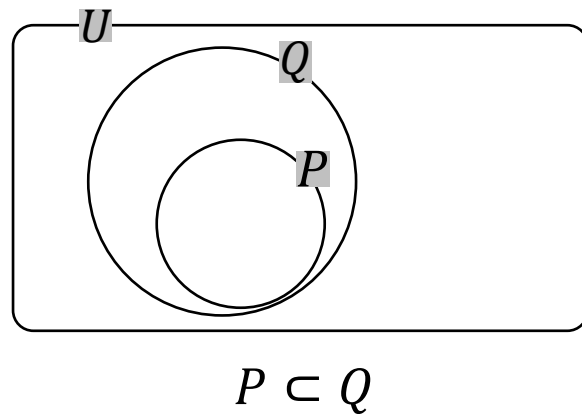


조건 명제의 진리표

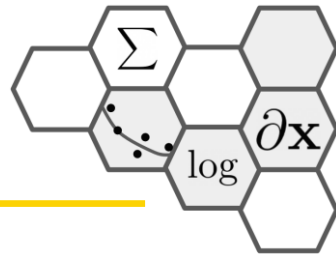


- $p \rightarrow q$ 의 진리 판단은 $P \subset Q$, P 가 Q 의 부분집합인가에 따라 판단

p	q	$p \rightarrow q$
T	T	T
F	T	T
F	F	T
T	F	F

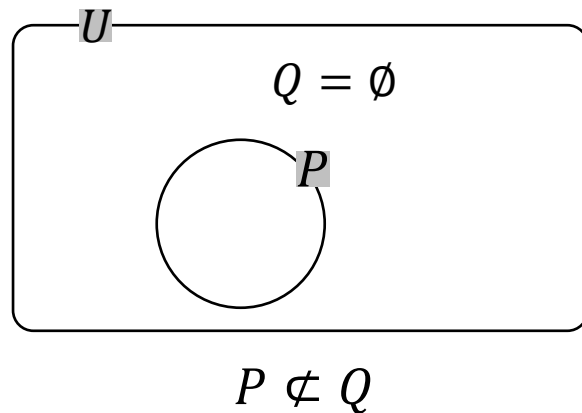


조건 명제의 진리표

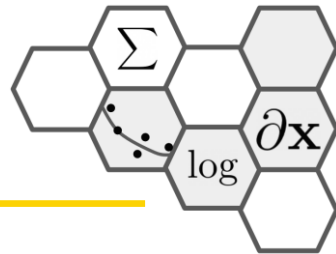


- $p \rightarrow q$ 의 진리 판단은 $P \subset Q$, P 가 Q 의 부분집합인가에 따라 판단

p	q	$p \rightarrow q$
T	T	T
F	T	T
F	F	T
T	F	F

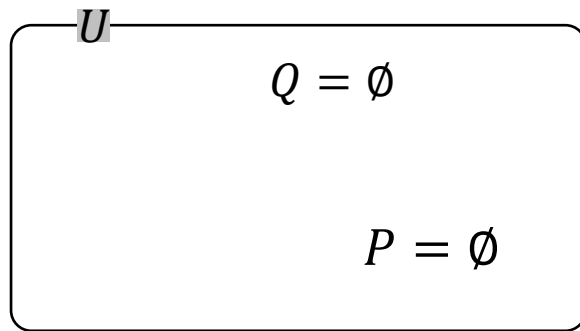


조건 명제의 진리표



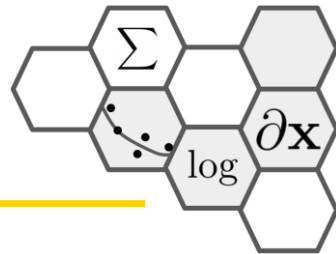
- $p \rightarrow q$ 의 진리 판단은 $P \subset Q$, P 가 Q 의 부분집합인가에 따라 판단

p	q	$p \rightarrow q$
T	T	T
F	T	T
F	F	T
T	F	F



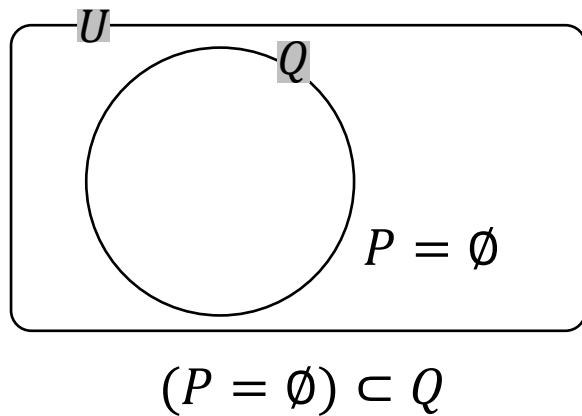
$$(P = \emptyset) \subset (Q = \emptyset)$$

조건 명제의 진리표

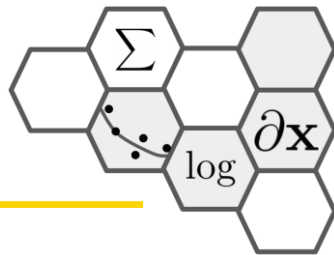


- $p \rightarrow q$ 의 진리 판단은 $P \subset Q$, P 가 Q 의 부분집합인가에 따라 판단

p	q	$p \rightarrow q$
T	T	T
F	T	T
F	F	T
T	F	F

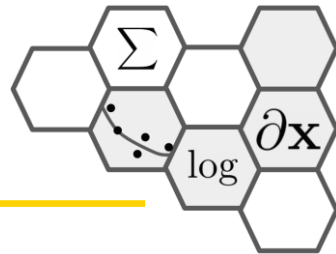


조건 명제: 일상 언어

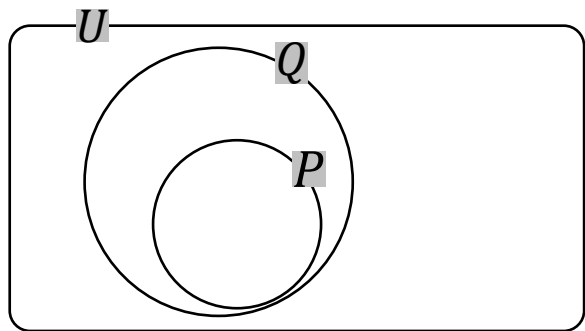


- “90점 이상 맞으면 아이폰 사 줄 게”
- p : 90점 이상 맞음
- q : 아이폰 사 줌
- 아빠가 나에게 저 명제를 이야기했다면
 - 90점이상 맞음 → 아이폰 사 줌: ~~아빠 왜 거짓말하고 그래?~~
 - 90점이상 맞음 → 아이폰 안 사 줌: ~~아빠 왜 거짓말하고 그래?~~
 - 90점이상 못 맞음 → 아이폰 사 줌: ~~아빠 왜 거짓말하고 그래?~~
 - 90점이상 못 맞음 → 아이폰 안 사 줌: ~~아빠 왜 거짓말하고 그래?~~

필요조건과 충분조건



- $p \rightarrow q$ 가 참일 때 $p \Rightarrow q$



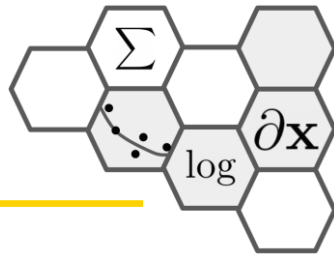
$$P \subset Q$$

q 이기 위한 충분조건
Necessary condition

$$p \Rightarrow q$$

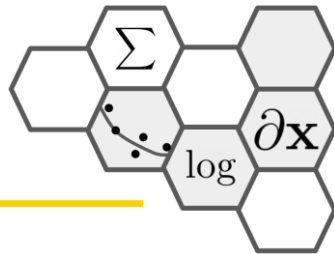
p 이기 위한 필요조건
Sufficient condition

필요조건과 충분조건



- 다음에서 p 는 q 이기 위한 무슨 조건? 필요조건!
 - $p: |x| \leq 4$, $q: -1 \leq x \leq 3$

👁️: 필요조건과 충분조건의 활용



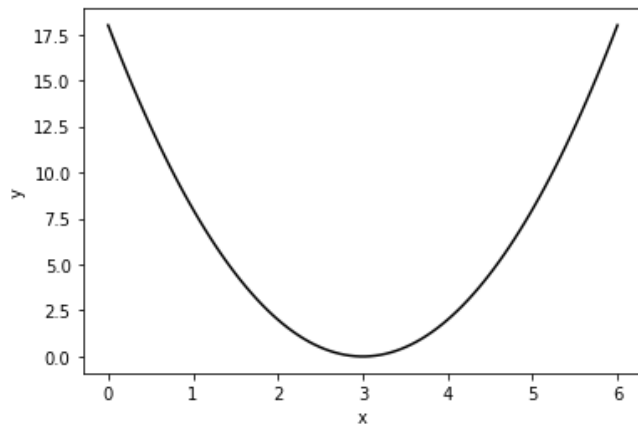
- 다음 함수를 최소로 하는 x 는?

$$f(x) = 2x^2 - 12x + 18$$

$$f'(x) = 0$$

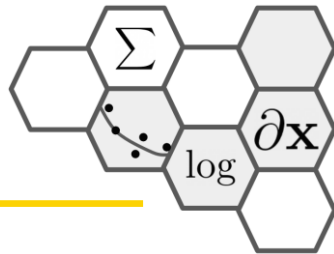
$$4x - 12 = 0$$

$$x = 3$$





필요조건과 충분조건의 활용



- 다음 함수를 최소로 하는 x 는?

$$f(x) = -x^5 + 16x^4 - 100x^3 + 304x^2 - 448x + 256$$

$$f'(x) = -5x^4 + 64x^3 - 300x^2 + 608x - 448$$

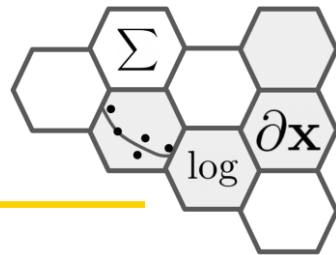
$$f'(x) = 0$$

$$x = 2, \quad \frac{14}{5}, \quad 4$$

$$f(x) = 0, \quad \frac{3456}{3125}, \quad 0$$



필요조건과 충분조건의 활용



- 다음 함수를 최소로 하는 x 는?

$$f(x) = -x^5 + 16x^4 - 100x^3 + 304x^2 - 448x + 256$$

$$f'(x) = -5x^4 + 64x^3 - 300x^2 + 608x - 448$$

$f'(x) = 0$ 1st order necessary cond.

$f''(x) > 0$ 2nd order sufficient cond.

$$x = 2, \quad \frac{14}{5}, \quad 4$$

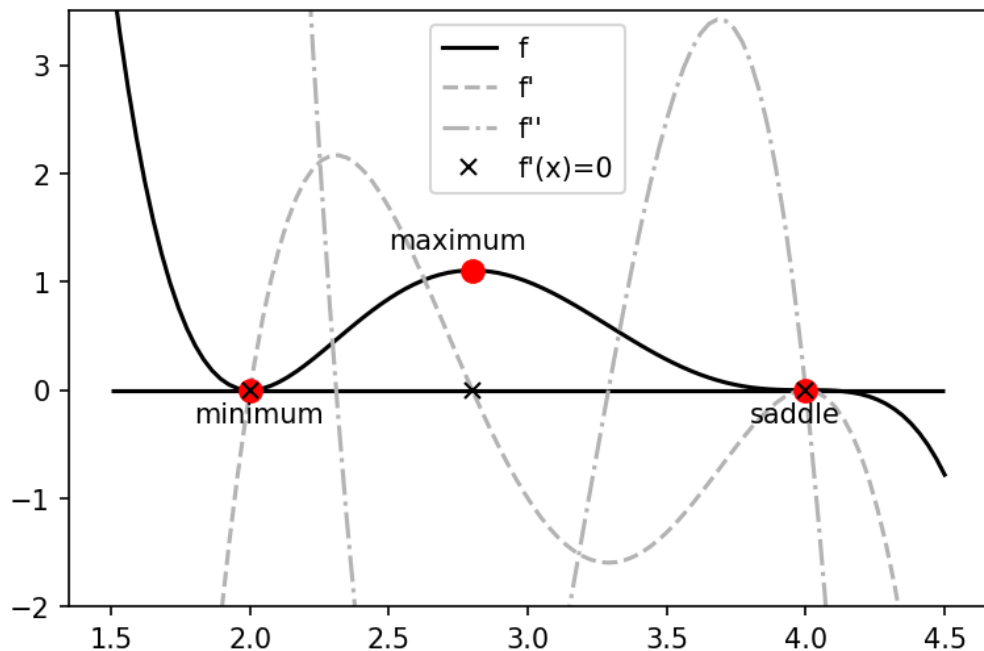
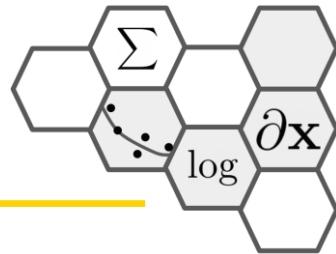
$$x = 2, \quad \frac{14}{5}, \quad 4$$

$$f(x) = 0, \quad \frac{3456}{3125}, \quad 0$$

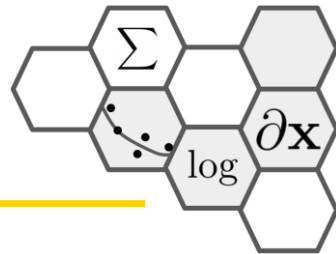
$$f''(x) = 16, \quad -\frac{144}{25}, \quad 0$$



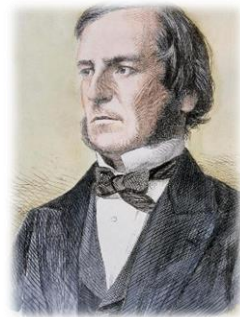
필요조건과 충분조건의 활용



Boolean Algebra

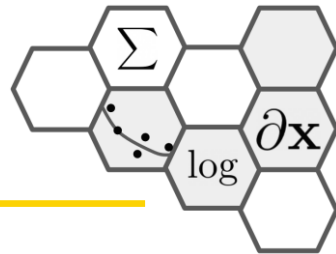


- 부울 대수 Boolean algebra: 집합 $\{0,1\}$ 과 NOT, AND, OR 연산으로 구성되는 식과 그에 대한 연산
- NOT 연산 complement
 - $\bar{0} = 1, \bar{1} = 0$
- OR 연산, 불린 합
 - $1 + 1 = 1, 1 + 0 = 1, 0 + 1 = 1, 0 + 0 = 0$
- AND 연산, 불린 곱
 - $1 \cdot 1 = 1, 1 \cdot 0 = 0, 0 \cdot 1 = 0, 0 \cdot 0 = 0$
- 0, 1은 각각 논리연산자에서 True, False에 대응
- Complement, 불린합, 불린곱은 각각 논리연산자 \neg, \vee, \wedge 에 대응

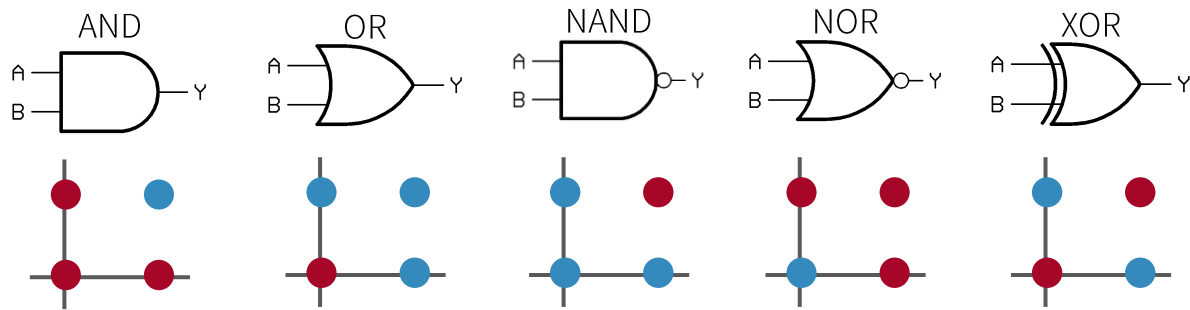


부울 George Boole (1815–1864)
Public domain

👁️: Logic Gates

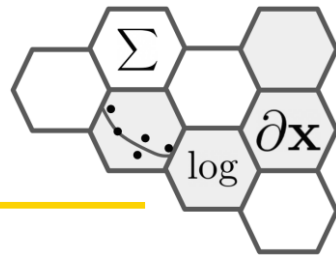


- 논리 게이트 logic gates: 부울 대수를 이용하여 회로를 설계할 때 회로의 기본 단위

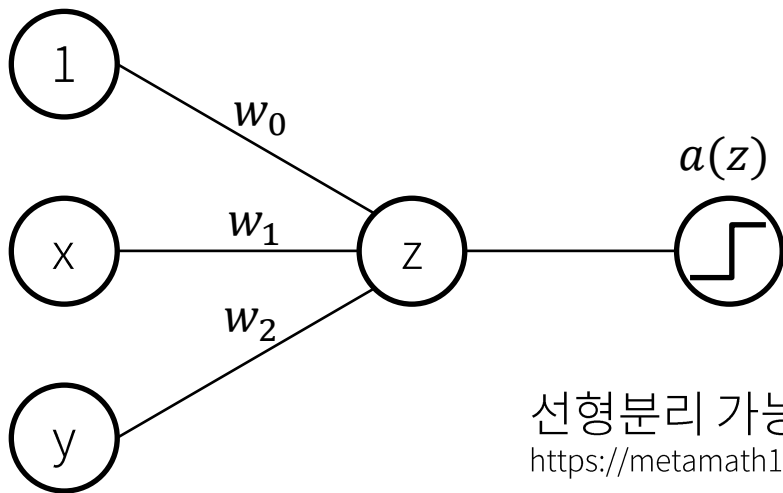


새년 CLAUDE ELWOOD
SHANNON (1916–2001)
CC BY-SA 2.0 DE

Perceptron



- 인공신경망의 한 종류로서, 1957년에 코넬 항공 연구소(Cornell Aeronautical Lab)의 프랑크 로젠 블라트 (Frank Rosenblatt)에 의해 고안되었다. 이것은 가장 간단한 형태의 피드포워드 (Feedforward) 네트워크 - 선형분류기- 로도 볼 수 있다. - 위키백과 (<https://ko.wikipedia.org/wiki/퍼셉트론>)

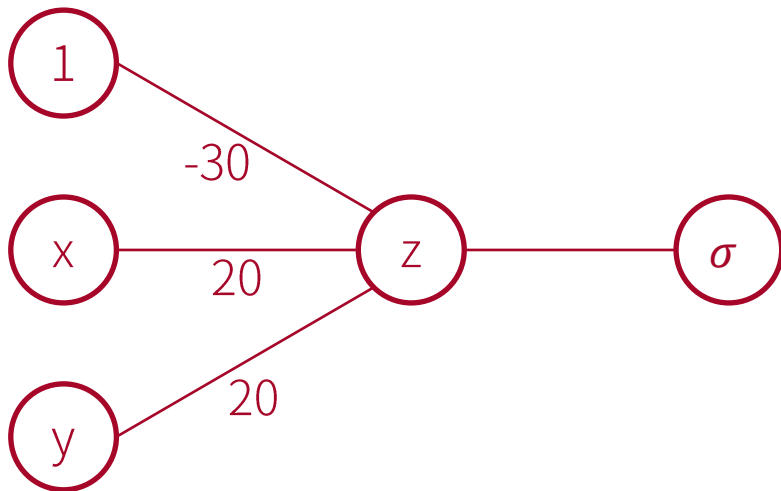
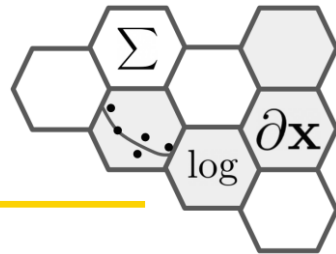
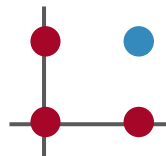


$$a(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$$

선형분리 가능 데이터에 대해서 반드시 수렴

<https://metamath1.github.io/2017/05/12/perceptron-conv-theorem.html>

AND



x	y	result
0	0	0
0	1	0
1	0	0
1	1	1

$$z = 20 \times 0 + 20 \times 0 + 1 \times (-30) = -30$$

$$g = \sigma(-30) \approx 0$$

$$z = 20 \times 0 + 20 \times 1 + 1 \times (-30) = -10$$

$$g = \sigma(-10) \approx 0$$

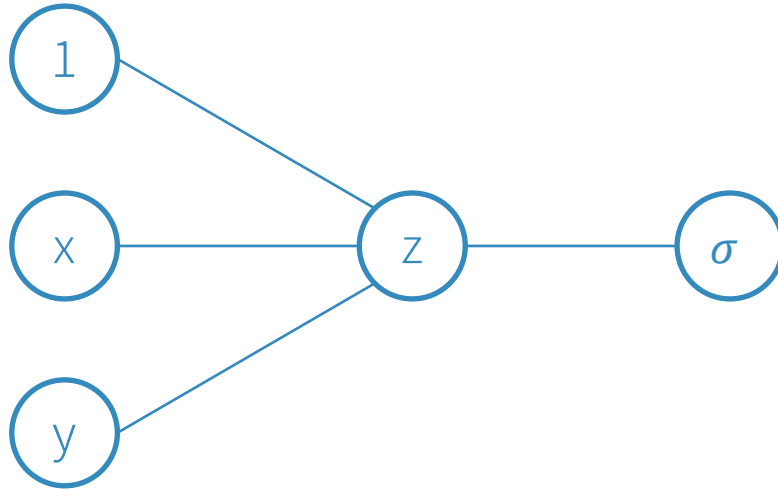
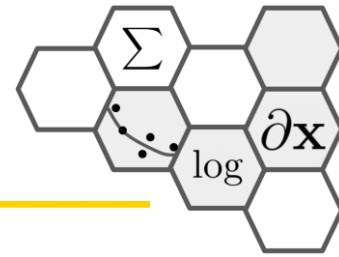
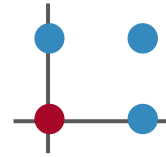
$$z = 20 \times 1 + 20 \times 0 + 1 \times (-30) = -10$$

$$g = \sigma(-10) \approx 0$$

$$z = 20 \times 1 + 20 \times 1 + 1 \times (-30) = 10$$

$$g = \sigma(10) \approx 1$$

OR



x	y	result
0	0	0
0	1	1
1	0	1
1	1	1

$$z = \quad \times 0 + \quad \times 0 + 1 \times \quad =$$

$$z = \quad \times 0 + \quad \times 1 + 1 \times \quad =$$

$$z = \quad \times 1 + \quad \times 0 + 1 \times \quad =$$

$$z = \quad \times 1 + \quad \times 1 + 1 \times \quad =$$

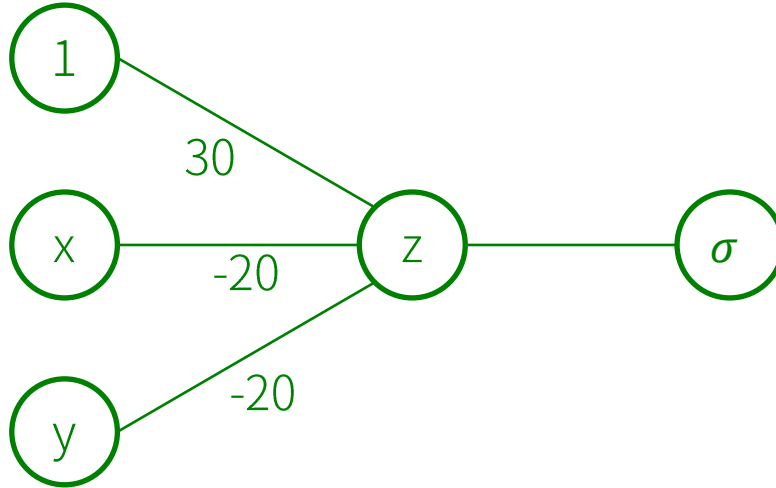
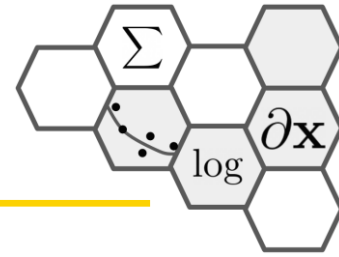
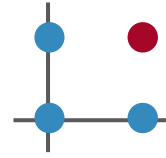
$$g = \sigma(\quad) \approx 0$$

$$g = \sigma(\quad) \approx 1$$

$$g = \sigma(\quad) \approx 1$$

$$g = \sigma(\quad) \approx 1$$

NAND



x	y	result
0	0	1
0	1	1
1	0	1
1	1	0

$$z = -20 \times 0 + (-20) \times 0 + 1 \times 30 = 30$$

$$g = \sigma(30) \approx 1$$

$$z = -20 \times 0 + (-20) \times 1 + 1 \times 30 = 10$$

$$g = \sigma(10) \approx 1$$

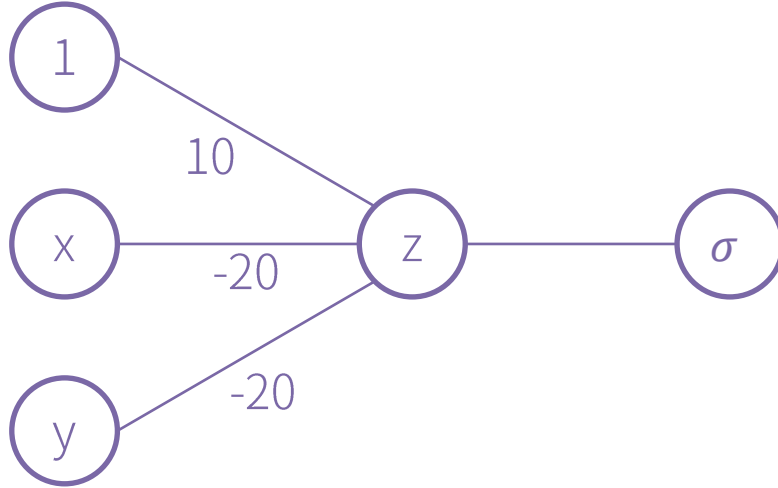
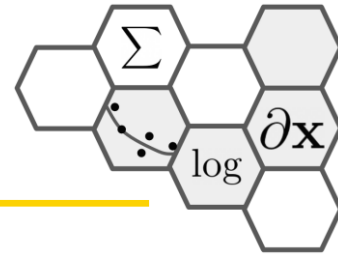
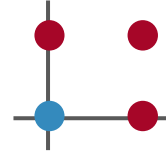
$$z = -20 \times 1 + (-20) \times 0 + 1 \times 30 = 10$$

$$g = \sigma(10) \approx 1$$

$$z = -20 \times 1 + (-20) \times 1 + 1 \times 30 = -10$$

$$g = \sigma(-10) \approx 0$$

NOR



x	y	result
0	0	1
0	1	0
1	0	0
1	1	0

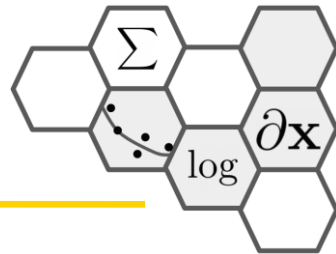
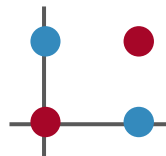
$$z = -20 \times 0 + (-20) \times 0 + 1 \times 10 = 10 \quad g = \sigma(10) \approx 1$$

$$z = -20 \times 0 + (-20) \times 1 + 1 \times 10 = -10 \quad g = \sigma(-10) \approx 0$$

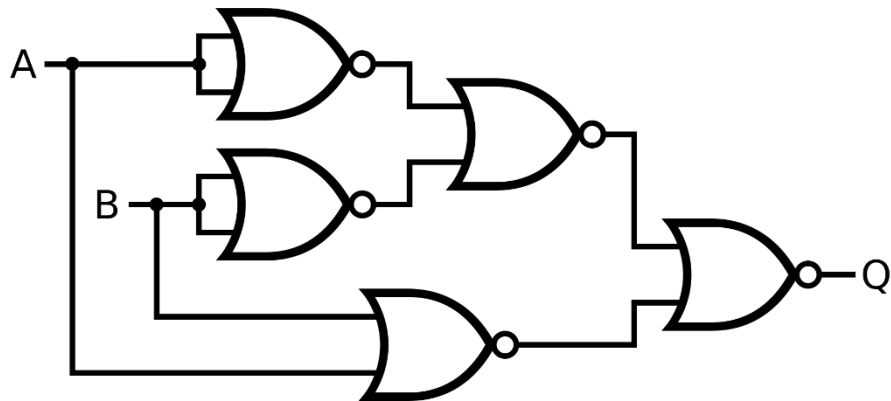
$$z = -20 \times 1 + (-20) \times 0 + 1 \times 10 = -10 \quad g = \sigma(-10) \approx 0$$

$$z = -20 \times 1 + (-20) \times 1 + 1 \times 10 = -30 \quad g = \sigma(-30) \approx 0$$

XOR

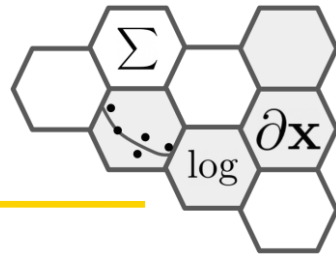
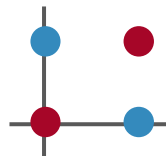


NOR 조합

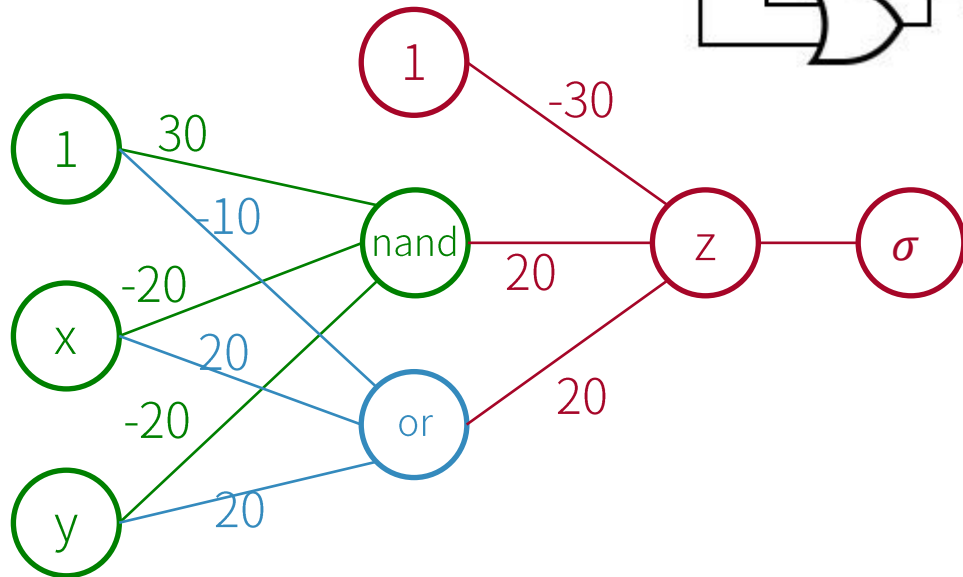
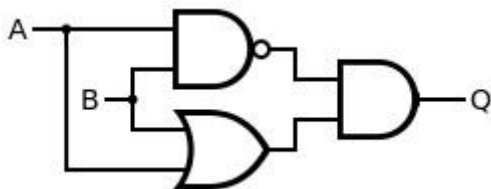


x	y	result
0	0	0
0	1	1
1	0	1
1	1	0

XOR



MLP



x	y	NAND	OR	result
0	0	1	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	1	0