Markdown 수식 연습

LaTeX 수식

웹에서 LaTeX 수식을 표현하기 위해서 주로 사용하는 라이브러리는 MathJax와 KaTeX 입니다. 둘 다 웹에서 LaTeX 문법을 이용해 수학 수식을 렌더링하는 자바스크립트 라이브러리지만, 각각 특징이 다릅니다. MathJax는 기능이 풍부하고 다양한 LaTeX 수식을 거의 완벽하게 지원하는 반면, 렌더링 속도가 상대적으로 느린 편입니다. 또한 마우스 우클릭 메뉴 등 추가 기능과 폭넓은 브라우저 호환성을 갖추고 있어서 복잡한 수식이나 다양한 환경에서 안정적입니다.

KaTeX는 칸 아카데미에서 개발한 라이브러리로, MathJax보다 훨씬 가볍고 렌더링 속도가 빠릅니다. 하지만 MathJax에 비해 지원하는 LaTeX 문법의 범위가 다소 제한적입니다. KaTeX는 서버 사이드 렌더링도 지원하며 SVG 형태로 수식을 그려서 크기 조절과 화면 환경에 유연하게 대응할 수 있습니다.

- MathJax: 완전한 기능과 호환성, 다양한 환경 지원, 상대적으로 무겁고 느림
- KaTeX: 빠른 렌더링과 가벼움, 제한된 기능, 서버 사이드 렌더링 가능

GitHub에서는 수학 수식 표현에 주로 MathJax 라이브러리를 사용합니다. Markdown 문서 내에서 LaTeX 기반 수식을 표시할 때, GitHub는 MathJax를 통해 수식을 렌더링하여 보여주는 방식을 채택하고 있습니다. 이는 MathJax가 다양한 LaTeX 문법을 폭넓게 지원하고, 복잡한 수학 표현도 웹에서 안정적으로 렌더링할 수 있기 때문입니다.

알고리즘의 시간 복잡도 분석

알고리즘의 실행 시간 $T(n)$은 입력 데이터의 크기(n)와 형태에 따라 달라질 수 있습니다. 알고리즘 분석은 성능을 평가하기 위해 다음 세 가지 기준에 초점을 맞춥니다.

최악의 경우 (Worst Case) - 상한(O)

최악의 경우 분석은 동일한 크기의 모든 입력 중 가장 긴 실행 시간을 나타냅니다. 이는 알고리즘 실행 시간이 결코 초과하지 않을 최대 점근적 상한(Upper Bound)을 보장하므로, Big-O(O) 표기법을 사용하여 정의합니다.

$$T_{\text{worst}}(n) = O(f_{\text{worst}}(n))$$

$$O(g(n)) \iff \exists c > 0, n_0 \text{ s.t. } \forall n \geq n_0, T(n) \leq c \cdot g(n)$$

설명: 충분히 큰 n에 대해, $T(n)$은 $g(n)$의 상수배(c)보다 항상 작거나 같음.

```
O(g(n)) \iff \exists c > 0, n_0 \text{ s.t. } \forall n \ge n_0, T(n) \le c \cdot g(n)
```

최선의 경우 (Best Case) - 하한(Ω)

최선의 경우 분석은 알고리즘이 가질 수 있는 최소 실행 시간을 나타냅니다. 이는 실행 시간이 이보다 더 짧아질 수 없음을 의미하는 최소 점근적 하한(Lower Bound)을 나타내므로, Big-Omega(Ω) 표기법을 사용하여 정의합니다.

$$T_{\text{best}}(n) = \Omega(f_{\text{best}}(n))$$

$$\Omega(g(n)) \iff \exists c > 0, n_0 \text{ s.t. } \forall n \geq n_0, T(n) \geq c \cdot g(n)$$

설명: 충분히 큰 n에 대해, $T(n)$은 $g(n)$의 상수배(c)보다 항상 크거나 같음.

```
\Omega(g(n)) \iff \exists c > 0, n_0 \text{ s.t. } \forall n \ge n_0, T(n) \ge c \cdot g(n
```

평균적인 경우 (Average Case) - 정확한 경계(Θ)

평균적인 경우 분석은 모든 가능한 입력에 대한 기대 실행 시간을 나타냅니다. 평균적인 성능이 상한과 하한 사이에 정확히 위치하는 경우, 정확한 점근적 성장률(Tight Bound)을 의미하는 Big-Theta(Θ) 표기법을 사용하여 정의합니다.

$$T_{\text{avg}}(n) = \Theta(f_{\text{avg}}(n))$$

$$\Theta(g(n)) \iff T(n) = O(g(n)) \land T(n) = \Omega(g(n))$$

설명: $T(n)$이 $g(n)$의 상수배 사이(상한과 하한)에 정확하게 존재함.

```
\Theta(g(n)) \iff T(n) = O(g(n)) \land T(n) = \Omega(g(n))
```

극한 (Limit): lim

미분의 정의 (그래디언트 기초)

인공지능(머신러닝) 분야에서 모델을 학습시킬 때 사용하는 경사하강법(Gradient Descent)의 기초가 되는 미분 계수의 정의입니다.

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

```
\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}
```

알고리즘 점근적 분석 (Big-O의 기초)

알고리즘 수업에서 시간 복잡도를 분석할 때 사용합니다. 입력 크기 n이 무한대로 갈 때, 특정 알고리즘의 실행 시간 $T(n)$이 n^2과 어떤 관계가 있는지(2차 시간 복잡도인지) 확인할 때 사용하는 식의 형태입니다.

$$\lim_{n \to \infty} \frac{T(n)}{n^2} = C$$

```
\lim_{n \to \infty} \frac{T(n)}{n^2} = C
```

합 (Summation): $\sum$

뉴런의 가중치 합 (인공 신경망)

딥러닝에서 하나의 뉴런(퍼셉트론)이 입력값(x_i)들과 가중치(w_{ji})들을 곱해서 모두 더하고 편향(b_j)을 더하는 과정을 나타냅니다. 가장 기초적인 신경망 연산입니다.

$$z_j = \sum_{i=1}^{n} w_{ji} x_i + b_j$$

```
z_j = \sum_{i=1}^{n} w_{ji} x_i + b_j
```

중첩 루프의 실행 횟수 계산

자료구조나 알고리즘 수업에서 이중 for 문의 총 반복 횟수를 계산할 때 자주 보는 식입니다. 1부터 N까지의 정수 합 공식을 유도하는 과정이기도 합니다.

$$\sum_{i=1}^{N} \sum_{j=1}^{i} 1 = \frac{N(N+1)}{2}$$

```
\sum_{i=1}^{N} \sum_{j=1}^{i} 1 = \frac{N(N+1)}{2}
```

적분 (Integral): $\int$

확률 밀도 함수 (연속 확률 분포)

확률 및 통계, 머신러닝에서 연속 확률 변수의 확률 밀도 함수(PDF) 전체 면적의 합은 항상 1이라는 기본 공리입니다.

$$\int_{-\infty}^{\infty} p(x)\,dx = 1$$

```
\int_{-\infty}^{\infty} p(x) \, dx = 1
```

푸리에 변환 (신호 처리)

영상 처리나 음성 신호 처리 등의 분야에서 시간 영역(t)의 신호를 주파수 영역(ω)으로 변환할 때 사용하는 푸리에 변환 공식입니다.

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t}\,dt$$

```
F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} \, dt
```

분수 (Fraction): $\frac{\cdots}{\cdots}$

베이즈 정리 (Bayes' Theorem)

인공지능의 확률적 추론, 스팸 메일 필터링 등에 핵심적으로 사용되는 베이즈 정리입니다. 조건부 확률을 다룰 때 필수적인 식입니다.

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

```
P(A|B) = \dfrac{P(B|A)P(A)}{P(B)}
```

코사인 유사도 (Cosine Similarity)

정보 검색이나 자연어 처리에서 두 문서(또는 벡터)가 얼마나 유사한지 측정할 때 사용하는 공식입니다. ($\|\cdot\|$는 노름(norm)을 표현하는 기호입니다.)

$$\text{similarity} = \frac{\mathrm{A} \cdot \mathrm{B}}{\|\mathrm{A}\|\|\mathrm{B}\|}$$

```
\text{similarity} = \dfrac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}
```

지수 표현 (Exponentiation): $\wedge$

RSA 암호화/복호화

컴퓨터 보안, 암호학 수업에서 배우는 공개키 암호 알고리즘인 RSA의 핵심 연산식입니다. 거듭제곱과 모듈러 연산이 사용됩니다.

$$c \equiv m^e \pmod{n}, \quad m \equiv c^d \pmod{n}$$

```
c \equiv m^e \pmod{n}, \quad m \equiv c^d \pmod{n}
```

이진 트리의 노드 수

자료구조에서 높이가 h인 포화 이진 트리(full binary tree)가 가질 수 있는 최대 노드의 개수를 나타내는 식입니다.

$$N_{\text{total}} = 2^{h+1} - 1$$

```
N_{\text{total}} = 2^{h+1} - 1
```

벡터 (Vector): $\vec{}$

벡터의 선형 결합 (머신러닝 예측식)

선형 회귀(Linear Regression) 등에서 입력 데이터 행렬 X와 가중치 벡터 $\vec{w}$, 편향 벡터 $\vec{b}$를 이용해 예측값 벡터 $\vec{y}$를 계산하는 식입니다. (논문에서는 벡터를 $\vec{w}$ 대신 굵은 글씨 w로 쓰기도 합니다.)

$$\vec{y} = X\vec{w} + \vec{b}$$

```
\vec{y} = X\vec{w} + \vec{b}
```

그래픽스 조명 계산 (내적)

컴퓨터 그래픽스에서 물체 표면의 밝기(확산 반사)를 계산할 때, 빛의 방향 벡터 $\vec{L}$과 표면의 법선 벡터 $\vec{N}$의 내적(dot product)을 사용합니다.

$$I = k_d(\vec{L} \cdot \vec{N})$$

```
I = k_d (\vec{L} \cdot \vec{N})
```

행렬 (Matrix): [...]

2D 회전 변환 행렬

컴퓨터 그래픽스나 로보틱스에서 물체를 2차원 평면에서 θ만큼 회전시킬 때 곱해주는 행렬입니다.

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

```
R_\theta = \begin{bmatrix}
\cos\theta & -\sin\theta \\
\sin\theta & \cos\theta
\end{bmatrix}
```

그래프 인접 행렬 (Adjacency Matrix)

자료구조나 이산수학에서 그래프의 연결 상태를 컴퓨터에 저장하기 위해 사용하는 인접 행렬의 예시입니다. 0과 1로 연결 유무를 표현합니다.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

```
A = \begin{bmatrix}
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0
\end{bmatrix}
```

논리 연산자 (Logical operators)

드모르간의 법칙 (De Morgan's Laws)

이산수학, 논리회로 설계에서 가장 기본이 되는 법칙 중 하나입니다. 논리식의 부정을 다룰 때 사용합니다. ($\iff$ 는 양방향 함의 기호입니다.)

$$\neg(P \land Q) \iff (\neg P) \lor (\neg Q)$$

```
\neg(P \land Q) \iff (\neg P) \lor (\neg Q)
```

데이터베이스 쿼리 조건 (집합 표현)

관계형 데이터베이스 이론에서 튜플 관계 해석(Tuple Relational Calculus) 등을 이용해 쿼리 조건을 수학적으로 명시할 때 이러한 논리 연산자가 사용됩니다.

$$\{t \mid (t \in R) \land (t.\text{age} \geq 20) \lor (t.\text{status} = \text{'student'})\}$$

```
\{ t \mid (t \in R) \land (t.\text{age} \ge 20) \lor (t.\text{status} = \text{'student'})\}
```

집합 및 조건 표현 (Cases)

재귀 알고리즘 (피보나치 수열)

알고리즘 도입부에서 재귀(Recursion)를 설명할 때 가장 많이 쓰이는 피보나치 수열의 정의입니다. 기저 조건(base case)과 재귀 단계(recursive step)를 나눌 때 사용합니다.

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

```
F(n) = \begin{cases}
0 & \text{if } n = 0 \\
1 & \text{if } n = 1 \\
F(n-1) + F(n-2) & \text{if } n > 1
\end{cases}
```

활성화 함수 ReLU (딥러닝)

딥러닝에서 가장 많이 사용되는 활성화 함수 중 하나인 ReLU(Rectified Linear Unit) 함수의 정의입니다. 입력이 양수면 그대로, 음수면 0을 출력하는 구간별 함수를 표현합니다.

$$\text{ReLU}(x) = \begin{cases} x & \text{for } x > 0 \\ 0 & \text{for } x \le 0 \end{cases}$$

```
\text{ReLU}(x) = \begin{cases}
x & \text{for } x > 0 \\
0 & \text{for } x \le 0
\end{cases}
```

대수적 관계 기호 (Relations)

알고리즘 시간 복잡도 (Big-Omega 표기법)

알고리즘 분석에서 함수의 하한(lower bound)을 나타내는 Big-Omega 표기법의 엄밀한 정의입니다. 부등호(≥)와 존재 기호(∃), 모든 기호(∀) 등이 복합적으로 사용된 고급 예제입니다.

$$f(n) \in \Omega(g(n)) \iff \exists c > 0, n_0 \text{ s.t. } \forall n \ge n_0, f(n) \ge c \cdot g(n)$$

```
f(n) \in \Omega(g(n)) \iff \exists c > 0, n_0 \text{ s.t. } \forall n \ge n_0, f(n) \ge c
```

P-NP 문제 (계산 이론)

컴퓨터 과학의 최대 난제인 P vs NP 문제를 언급할 때 사용됩니다. 같지 않음(≠)을 표현하는 간단하지만 중요한 수식입니다.

$$\text{If } P \neq NP, \text{ then many interesting problems are hard to solve.}$$

```
\text{If } P \neq NP, \text{ then many interesting problems are hard to solve.}
```