

## 대학수학 경사하강법(gradient descent method) 교수·학습자료 개발

이 상 구 (성균관대학교, 교수)  
남 윤 (성균관대학교, 연구원)  
이 재 화 (성균관대학교, 연구원)<sup>†</sup>

본 논문에서는 인공지능 알고리즘에서 많이 사용되는 경사하강법(gradient descent method)을 대학수학 강좌에서 인공지능 활용사례로 사용할 수 있도록 연구한 교수·학습 기초자료를 소개한다. 특히 대학 미적분학 수준에서도 가르칠 수 있도록 자세한 개념 설명과 함께 복잡한 함수에 관해서도 쉽게 계산할 수 있도록 파이썬(Python) 기반의 SageMath 코드를 제공한다. 그리고 실제 인공지능 응용과 연계하여 선형회귀에서 발생하는 최소제곱문제를 경사하강법을 활용하여 풀이한 예시도 함께 소개한다. 본 연구는 대학 미적분학 뿐만 아니라 공학수학, 수치해석, 응용수학 등과 같은 고급 수학 과목을 지도하는 다양한 교수자들에게 도움이 될 수 있다.

### I. 서론

최근 인공지능(Artificial Intelligence) 기술이 발전하면서 우리의 생활 속으로 깊숙이 들어오며 따라, 인공지능에 대한 이해는 점차 기본 소양이 되고 있다. 특히 인공지능 속의 알고리즘은 수학적 사고를 바탕으로 설계되었으므로, 이를 이해하고 활용하기 위해서는 여러 수학 과목(선형대수학, 미적분학, 확률/통계, 이산수학, 수치해석학 등)의 지식이 동시에 요구된다. 이상구 외(2020a)의 최근 연구에서는 언급한 많은 수학 과목을 모두 수강하지 않고도 다양한 전공의 대학생 누구나 쉽게 인공지능에 접근할 수 있도록, 인공지능에 필수적인 기초수학 지식을 제공하는 한 학기(또는 두 학기) 분량의 강좌 운영사례를 제시하였다.

그러나 이렇게 인공지능에 필요한 수학을 제공하는 별도의 강좌를 개설하는 것 이외에, 기존의 대학수학 강좌(미적분학, 선형대수학, 확률/통계, 이산수학 등)에서도 각 과목과 관련된 인공지능 응용 예시를 소개해주는 것이 필요하다고 여겨진다. 이는 각 대학의 상황에 따라 별도의 강좌를 개설하는 것이 어려울 수도 있으며, 또 대학에서 학습하는 수학 과목이 현실에서도 잘 활용되고 있음을 보여주어, 학생들에게 구체적인 학습 동기를 부여할 수 있기 때문이다.

이에 본 연구진은 대학수학 강좌에서 활용할 수 있는 인공지능 관련 수학 교수·학습자료를 개발하기로 결정하였다. 본 논문에서는 그 중 인공지능 알고리즘에서 중요한 위치를 차지하는 경사하강법(gradient descent method)에 대하여 개발한 교수·학습자료를 소개한다. 경사하강법은 미분가능한 함수의 극솟값(local minimum)을 계산하는 대표적인 수치적 방법(numerical method)인데, 학습 수준에 따라 대학 미적분학 수업에서도 다룰 수 있는 내용과 이보다 높은 수준의 수학 수업에서 교육할 수 있는 내용으로 나누어 개발하였다.

영어로 gradient method 또는 steepest descent method라고도 불리는 경사하강법은 학부 수준의 수치해석

\* 접수일(2023년 8월 8일), 심사(수정)일(2023년 9월 11일), 게재 확정일(2023년 9월 26일)

\* MSC2000 분류 : 97U50, 97U60, 97U70

\* 주제어 : 대학수학, 극값, 임계점, 경사하강법, 인공지능, 최소제곱

<sup>†</sup> 교신저자 : jhlee2chn@skku.edu

\* 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2021R1F1A1046714).

(numerical analysis)이나 계산과학(computational science)과 같은 과목에서도 간간히 소개되는 주제이다. 그러나, 해당 과목들 자체가 수학과나 일부 공학 전공 학생들만 수강하는 경우가 많을뿐더러, 학부 수준에서는 경사하강법의 내용이나 교수 방법이 상당히 제한적이다. 예를 들어, Burden & Faires(2010)와 같이 경사하강법을 선형연립방정식의 반복적 수치해법(iterative method)이나 비선형방정식의 수치해법을 다룰 때 한하여 부분적으로 소개하는 식이다. 이런 경우에는 경사하강법의 본질인 최적화의 개념을 파악하기가 쉽지 않다. 또한 Chapra & Canale(2020)과 같이 경사하강법을 포함한 방대한 수치적 방법들이 나열된 교과서를 사용할 경우, 교수자의 관점에 따라 경사하강법이 제외될 수도 있다. 이런 경우에는 학생들이 대학 내내 상당한 수학 과목을 수강하고도 경사하강법의 전체적 내용을 전혀 접하지 못할 수 있다. 그러나 사실 경사하강법은 형식이 단순하여 선형회귀(linear regression), 로지스틱 회귀(logistic regression), 서포트 벡터 머신(support vector machine) 등의 인공지능 알고리즘에 많이 사용되기 때문에 이를 전반적으로 다루어야 할 필요가 있다.

게다가 경사하강법의 핵심적인 아이디어는 대학 미적분학 수준에서 충분히 이해할 수 있음에도 불구하고 교과 내용에는 거의 포함되지 못하였다. 일반적으로 미분을 응용한 최적화 문제에서는 함수의 극소점을 찾기 위해, 먼저 후보가 되는 임계점(stationary point)을 계산하고, 구한 임계점에서 함수가 극솟값을 갖는지를 이계도함수 또는 헤시안(Hessian)을 활용하여 판단한다. 이때 함수가 복잡한 경우에는 임계점을 구하기 위한 방정식 자체도 복잡하여 지필로 계산하기가 쉽지 않다. 그래서 대부분의 미적분학 교재에서는 지필로 계산할 수 있는 단순한 문제만 제시되고 있다. 그래서 지필로 계산하기 어려운 복잡한 함수들을 만날 때 학생들이 해결할 수 있도록 대학 미적분학 강좌에서 경사하강법이 소개될 필요가 있다고 생각한다.

본 논문에서는 먼저 경사하강법에 관하여 소개하고, 복잡한 함수에 관해서도 쉽게 계산할 수 있도록 파이썬(Python) 기반의 SageMath 코드를 제공한다. 이를 활용하면 미리 입력된 코드에서 함수와 조건들을 변경해가면서 유사한 문제도 모두 해결할 수 있다. 또한 선형회귀에서 나타나는 최소제곱 문제를 경사하강법으로 해결하는 예시를 소개한다. 본 연구는 대학 미적분학을 지도하는 다양한 전공의 교수자들에게 도움이 될 기초자료가 될 것이다.

## II. 연구의 배경

### 1. 이론적 배경

데이터를 기반으로 하는 인공지능 알고리즘은 주어진 데이터에 잘 들어맞도록 모형을 학습시키는 과정을 거친다. 이는 데이터를 모형에 입력하여 얻은 예측값과 관측값 사이의 오차를 나타내는 함수를 최소화하는 문제로 나타난다. 예를 들어, <표 II-1>은 기본적인 인공지능 알고리즘인 선형회귀에서 발생하는 최소제곱문제(least squares problem)를 보여준다.

<표 II-1> 최소제곱문제

$$\text{minimize } f(\mathbf{x}) = \frac{1}{2} \|\mathbf{Ax} - \mathbf{b}\|^2, \quad \mathbf{x} \in \mathbb{R}^n \quad (A \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m \text{ 일 때})$$

이렇게 제약조건(constraint)이 없이 주어진 함수를 최소화하기 위해서는 먼저 함수의 극솟값을 계산해야 한다. 이는 극솟값 중에 제일 작은 값이 최솟값(global minimum)이기 때문이다. 그러나 비볼록(nonconvex)인 다변수 함수의 경우, 극솟값이 모두 몇 개나 있는지 파악할 수가 없어서 어떤 알고리즘을 통하여 계산한 극솟값이 최솟값인지 파악하기가 쉽지 않다. 실제로 함수의 최솟값을 구하는 문제는 현재까지도 연구가 활발한 분야이다.

따라서 본 논문에서는 제약조건이 없이 함수의 극솟값을 구하는 것을 대상으로 한다.

미분가능한 함수의 극솟값을 구하는 방법은 <표 II-2>와 같이 나타낼 수 있다. 학생들이 쉽게 이해할 수 있도록 독립변수의 개수에 따라 1변수 함수, 2변수 함수,  $n$ 변수 함수의 경우로 나누어 기술하였지만, 본질은 모두 같다. 즉 극소점이 되는 후보는 페르마의 임계점 정리(Fermat's theorem on stationary points)를 만족해야 하며, 임계점이 극소점이 되는지 확인하기 위해서는 이계도함수 또는 헤시안을 조사해야 한다. 다변수 함수의 경우, 임계점에서의 헤시안의 고윳값(eigenvalue)이 모두 양수임을 보이거나 선행 주 소행렬식(leading principal minor)이 모두 양수임을 보이면, 그 임계점에서 함수가 극솟값을 갖게 된다. 1변수 함수의 경우, 이계도함수의 부호가 양수이면 극솟값으로 판정되는데, 이때는 헤시안이 바로 이계도함수를 성분으로 하는  $1 \times 1$  행렬이 되므로, 이계도함수의 부호가 바로 고윳값의 부호이자 선행 주 소행렬식의 부호가 되어 본질은 모두 동일하다(이상구 외, 2022).

<표 II-2> 미분가능한 함수의 극솟값을 구하는 방법(1변수, 2변수,  $n$ 변수 함수의 경우)

미분가능한 함수의 극솟값을 구하는 방법(1변수 함수의 경우)	
문제	minimize $f(x), \quad x \in \mathbb{R}$
[1단계]	$f'(a) = 0$ 인 점(임계점) $a$ 를 찾는다.
[2단계]	$f''(a) > 0$ 이면, $a$ 는 $f$ 의 극소점이다. 즉 $f(a)$ 는 $f$ 의 극솟값이다.
미분가능한 함수의 극솟값을 구하는 방법(2변수 함수의 경우)	
문제	minimize $f(x, y), \quad (x, y) \in \mathbb{R}^2$
[1단계]	$\nabla f(a, b) = (0, 0)$ 인 점(임계점) $(a, b)$ 를 찾는다.
[2단계]	헤시안을 $\nabla^2 f(a, b) = \begin{bmatrix} f_{xx}(a, b) & f_{xy}(a, b) \\ f_{xy}(a, b) & f_{yy}(a, b) \end{bmatrix}$ 이라 할 때, $f_{xx}(a, b) > 0$ 이고 $f_{xx}(a, b)f_{yy}(a, b) - \{f_{xy}(a, b)\}^2 > 0$ 이면, $(a, b)$ 는 $f$ 의 극소점이다. 즉 $f(a, b)$ 는 $f$ 의 극솟값이다. $\lambda_1, \lambda_2$ 를 $\nabla^2 f(a, b)$ 의 고윳값이라 할 때, $\lambda_1, \lambda_2 > 0$ 이면, $(a, b)$ 는 $f$ 의 극소점이다. 즉 $f(a, b)$ 는 $f$ 의 극솟값이다.
미분가능한 함수의 극솟값을 구하는 방법( $n$ 변수 함수의 경우)	
문제	minimize $f(\mathbf{x}) = f(x_1, \dots, x_n), \quad \mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$
[1단계]	$\nabla f(\mathbf{a}) = \mathbf{0}$ 인 점(임계점) $\mathbf{a}$ 를 찾는다.
[2단계]	헤시안 $\nabla^2 f(\mathbf{a})$ 에 대하여, $\Delta_1, \Delta_2, \dots, \Delta_n$ 을 $\nabla^2 f(\mathbf{a})$ 의 선행 주 소행렬식이라 할 때, 모든 $k = 1, 2, \dots, n$ 에 대하여 $\Delta_k > 0$ 이면, $\mathbf{a}$ 는 $f$ 의 극소점이다. 즉 $f(\mathbf{a})$ 는 $f$ 의 극솟값이다. $\lambda_1, \lambda_2, \dots, \lambda_n$ 을 $\nabla^2 f(\mathbf{a})$ 의 고윳값이라 할 때, 모든 $i = 1, 2, \dots, n$ 에 대하여 $\lambda_i > 0$ 이면, $\mathbf{a}$ 는 $f$ 의 극소점이다. 즉 $f(\mathbf{a})$ 는 $f$ 의 극솟값이다.

이와 같이  $f$ 의 임계점을 구하기 위해서는 방정식( $f'(x)=0$  또는  $\nabla f(\mathbf{x})=\mathbf{0}$ )을 풀어야 한다. 그러나  $f$ 가 복잡한 경우에는 이 방정식을 풀기도 쉽지 않다. 대학 미적분학 교재에서는 대개 지필로 계산할 수 있는 단순한 문제만 제시되고 있어서 현실에서 지필로 계산하기 어려운 복잡한 함수를 만날 때, 미적분학 지식을 어떻게 활용해야 하는지 어려움을 느낄 수 있다. 따라서 수치적으로 계산하는 방법도 함께 소개해줄 필요가 있다고 생각한다.

## 2. 연구 방법 및 절차

본 연구진은 미적분학과 같은 대학 기초수학 과목에서도 인공지능 응용예시와 같은 최신 동향을 소개할 필요가 있다고 판단하고, 대학 미적분학 강좌에 사용할 수 있도록 미분가능한 함수의 극소점을 계산하는 대표적인 수치적 방법인 경사하강법을 주제로 정하였다. 그리고 경사하강법과 관련된 논문과 교재 등을 검토하여 다음과 같은 방법으로 자료를 정리하였다.

- ① 먼저 다양한 전공의 학생들이 경사하강법을 이해할 수 있도록, 쉬운 개념과 간단한 예제로 시각화 자료를 구성하여 설명한다.
- ② 학생들이 수학에 관해 생각하고 논의할 시간을 충분히 갖도록 복잡한 계산에 대한 부담을 덜어준다. 특히 파이썬 기반의 SageMath 코드를 제시하여 함수와 조건들을 변경해가면서 유사한 문제도 쉽게 해결할 수 있도록 한다.
- ③ 선형회귀에서 나타나는 최소제곱문제를 경사하강법으로 해결한 예시를 설명한다.
- ④ 실제 인공지능 알고리즘에는 경사하강법을 변형한 확률적 경사하강법(Stochastic Gradient Descent, SGD)이 주로 사용되므로 경사하강법의 한계에 대해서도 함께 소개한다.
- ⑤ 필요시 개발된 자료의 타당성에 관하여 사전 및 사후 전문가 검토를 통해 자료를 보완/완성한다.

## III. 연구 결과 및 논의

이 장에서는 경사하강법에 대한 개념 소개 및 SageMath 코드, 최소제곱문제를 경사하강법으로 해결하는 예시에 대하여 설명한다.

### 1. 경사하강법 소개

경사하강법의 아이디어는 1847년 코시(Augustin Louis Cauchy, 1789-1857)가 발표한 논문(Cauchy, 1847)에서 연립방정식을 푸는 일반적인 방법으로 처음 제시되었다. 당시 코시는 천체의 궤도를 계산하기 위해 미분방정식이 아니라 천체의 움직임을 나타내는 대수방정식을 풀려고 했는데, 소거법 대신 함수의 최솟값을 구하는 문제로 변형하여 해결하는 방법을 제시하였다. 엄밀하지는 않았던 코시의 아이디어를 설명하면 다음과 같다. 먼저 항상 0보다 크거나 같은 연속인 함수  $u=f(x, y, z, \dots)$ 를 만든다.<sup>1)</sup> 그리고 특정한 점  $(x_0, y_0, z_0, \dots)$ 에서  $f$ 의 함수값을  $u_0$ 라 하자. 만일  $X=f_x(x_0, y_0, z_0, \dots)$ ,  $Y=f_y(x_0, y_0, z_0, \dots)$ ,  $Z=f_z(x_0, y_0, z_0, \dots)$ , ...라 하면  $f$ 는 근사적으로 다음과 같이 나타낼 수 있다.

1) 예를 들면, 연립방정식  $f_i(x_1, \dots, x_n)=0, i=1, \dots, m$  문제를 함수  $u=f(x_1, \dots, x_n)=\frac{1}{2}\sum_{i=1}^m (f_i(x_1, \dots, x_n))^2$ 를 최소화 하는 문제로 변형할 수 있다.

$$f(x_0 + \alpha, y_0 + \beta, z_0 + \gamma, \dots) = u_0 + X\alpha + Y\beta + Z\gamma + \dots$$

이제 작은 양수  $\theta > 0$ 를 취하여  $\alpha = -\theta X$ ,  $\beta = -\theta Y$ ,  $\gamma = -\theta Z$ , ...라 하면, 다음과 같이  $u_0$ 보다 작은 함수값  $u_1$ 을 얻을 수 있다.

$$u_1 = f(x_0 - \theta X, y_0 - \theta Y, z_0 - \theta Z, \dots) = u_0 - \theta(X^2 + Y^2 + Z^2 + \dots) < u_0$$

즉  $x_1 = x_0 - \theta X$ ,  $y_1 = y_0 - \theta Y$ ,  $z_1 = z_0 - \theta Z$ , ...라 하면,  $(x_0, y_0, z_0, \dots)$ 에서보다 함수값이 더 감소한 점  $(x_1, y_1, z_1, \dots)$ 을 얻게 된다. 여기서  $x_1 = x_0 - \theta X$ ,  $y_1 = y_0 - \theta Y$ ,  $z_1 = z_0 - \theta Z$ , ...은 이후에 자세히 기술할 경사하강법의 반복 단계(iteration)를 보여준다. 코시는 위의 방법을 반복적으로 진행하면  $f$ 의 최솟값에 수렴하게 될 것이라고 하였다(Lemaréchal, 2012).

**가. 1변수 함수의 경우**

다음과 같이 1변수 함수  $f$ 를 최소화하는 문제를 생각해보자. 이 내용은 (이상구 외, 2020b)를 참조하였다.

$$\text{minimize } f(x), \quad x \in \mathbb{R}$$

여기서  $f$ 는 목적함수(objective function), 비용함수(cost function), 또는 손실함수(loss function) 등으로 불린다. 본 논문에서는 인공지능 알고리즘 용어와의 일관성을 위해 손실함수라는 용어를 사용하도록 한다. 경사하강법은 반복법(iterative method)으로 초기 근사해  $x_1$ 에서 시작하여 특정한 반복 단계를 거쳐 이전보다 나은 근사해  $x_2, x_3, x_4, x_5, \dots$ 를 생성한다. 목표는  $k$ 번째 근사해  $x_k$  또는 극한값  $(x_k \rightarrow) x^*$ 에서 방정식  $f'(x) = 0$ 을 만족하도록 하는 것이다. 경사하강법의 기본 아이디어는 함수의 기울기(경사)를 구하여 기울기가 낮은 쪽으로 계속 이동시켜서 극값에 이를 때까지 반복시키는 것으로 기본적인 알고리즘은 <표 III-1>과 같다.

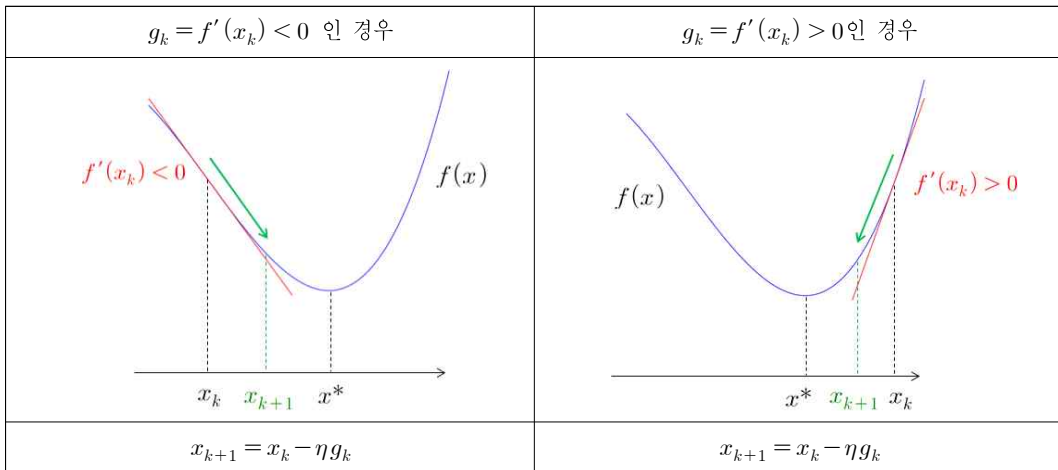
<표 III-1> 경사하강법 알고리즘(1변수 함수)

<b>[단계 1]</b>	초기 근사해 $x_1$ , 허용오차 $0 \leq \epsilon \ll 1$ , 학습률 $\eta > 0$ 를 설정하고, $k := 1$ 로 한다.
<b>[단계 2]</b>	$g_k = f'(x_k)$ 를 계산한다. 만일 $ g_k  \leq \epsilon$ 이면, 알고리즘을 멈추고 $x_k$ 를 반환한다.
<b>[단계 3]</b>	$x_{k+1} = x_k - \eta g_k$ , $k := k + 1$ 이라 두고, [단계 2]로 이동한다.

우리의 목표는 함수를 최소화하는 것이므로, 경사하강법의 반복단계  $x_{k+1} = x_k - \eta g_k$ 가 의미하는 것은 현재 위치  $x_k$ 에서의 함수값 보다 더 작은 함수값을 갖는  $x_{k+1}$ 을 찾는 것이다. 이를 위해  $x_k$ 에서  $-\eta g_k$ 만큼 이동한다. [그림 III-1]은  $g_k = f'(x_k)$ 의 부호에 따라 경사하강법의 움직임을 보여준다.

먼저  $k$ 번째 근사해  $x_k$ 에서 접선의 기울기  $g_k = f'(x_k) < 0$ 이면([그림 III-1] 왼쪽 그림)  $x_k$ 에서 오른쪽으로 이동할 때 함수가 감소하므로 극소점  $x^*$ 는  $x_k$ 의 오른쪽에 있다고 판단할 수 있다. 따라서  $x_k$ 에서  $-g_k (> 0)$  방향(오른쪽)으로 이동하여  $x_{k+1} (= x_k - \eta g_k)$ 을 생성한다. 마찬가지로  $g_k = f'(x_k) > 0$ 이면([그림 III-1] 오른쪽 그림)  $x_k$ 에서 왼쪽으로 이동할 때 함수가 감소하므로 극소점  $x^*$ 는  $x_k$ 의 왼쪽에 있다고 판단할 수 있다. 따라서  $x_k$ 에서  $-g_k (< 0)$  방향(왼쪽)으로 이동하여  $x_{k+1} (= x_k - \eta g_k)$ 을 생성한다. 즉 경사하강법은

$f(x_1) > f(x_2) > \dots$  가 만족되도록  $x_1, x_2, \dots$  을 찾는 것이라 볼 수 있다. 이때  $\eta$  는 학습률(learning rate)로  $-g_k$  방향으로 얼마만큼 이동해야 하는지 결정한다.<sup>2)</sup>



[그림 III-1] 경사하강법 원리

이제 다음과 같이 간단한 예제를 경사하강법으로 풀어보자.

**예제 1.** 함수  $f(x) = 2x^2 - 3x + 2$ 의 최솟값을 구하시오. (단,  $x_1 = 0, \eta = 0.1, \epsilon = 10^{-6}$ )

이 예제는 간단한 이차함수의 최솟값을 계산하는 것으로  $f'(x) = 4x - 3 = 0$ 에서 임계점은  $x = \frac{3}{4} = 0.75$ 이고, 특히  $f$ 는 아래로 볼록한 이차함수이므로  $x = \frac{3}{4}$ 에서 최솟값이  $f\left(\frac{3}{4}\right) = \frac{7}{8} = 0.875$ 임을 쉽게 알 수 있다. <표 III-2>의 SageMath 코드<sup>3)</sup>를 활용하면 함수의 그래프  $y = f(x)$ 와 점  $(x_k, f(x_k))$ 를 시각적으로 나타내어 경사하강법 알고리즘이 수렴하는 것을 쉽게 확인할 수 있다. 이 알고리즘을 통해 얻은 해를 소수점 아래 셋째 자리에서 반올림하면 0.75이다.

<sup>2)</sup> 알고리즘을 효과적으로 진행시키려면 적절한  $\eta$  값을 사용해야 한다. 예를 들어,  $\eta$ 가 너무 크면 함수값이 증가하여 수렴하지 않을 수도 있고,  $\eta$ 가 너무 작으면 수렴하는 속도가 느릴 수 있다. 학습률은 대개  $10^{-6}$ 에서 1사이의 범위에서 정하는 것으로 알려져 있으며, 초기 학습률은  $\eta = 0.1$  또는  $\eta = 0.01$ 이 주로 사용된다고 한다.  
<https://www.andreaperlato.com/theorypost/the-learning-rate/>

<sup>3)</sup> 제시된 SageMath 코드를 복사하여 웹사이트 <https://sagecell.sagemath.org/>에 붙여넣기 한 후 “Evaluate”를 클릭하면 바로 실행된다.

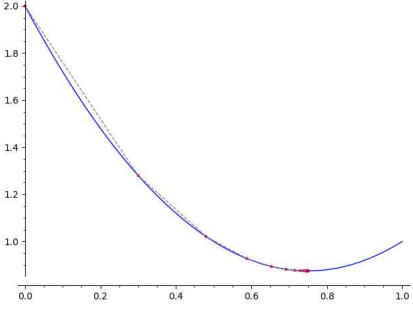
<표 III-2> 예제 1을 해결하는 경사하강법 코드(SageMath)와 결과

```

f(x) = 2*x^2 - 3*x + 2 # 함수 입력
df(x) = diff(f(x), x) # 도함수 계산
x0 = 0.0 # 초기 근사해
tol = 1e-6 # 허용 오차
eta = 0.1 # 학습률
r = [] # 그래프를 그리기 위한 용도
for k in range(300):
    g0 = df(x0)
    r.append((x0, f(x0))) # 점 (x_k, f(x_k)) 저장
    if abs(g0) <= tol:
        print("알고리즘 성공!")
        break
    x0 = x0 - eta*g0 # 경사하강법 반복단계

print("x* =", x0) # 해 출력
print("|g*| =", abs(g0)) # 도함수의 절댓값 출력
print("f(x*) =", f(x0)) # 극솟값 출력
print("반복 횟수 =", k + 1) # 반복 횟수
p1 = plot(f(x), (x, 0, 1)) # 함수 y = f(x)의 그래프
p2 = line2d(r, linestyle = '--', color = 'grey') + point(r, color = 'red') # (x_k, f(x_k)) 그리기
p1 + p2 # 동시에 그리기

```



---

알고리즘 성공! # 도함수의 절댓값이 허용 범위내에서 매우 작다는 의미  $g_k \approx 0$

```

x* = 0.749999834194560
|g*| = 6.63221759289456e-7
f(x*) = 0.875000000000055
반복 횟수 = 31

```

**나. 다변수 함수의 경우**

다변수 함수의 경우에도 <표 III-3>과 같이 경사하강법이 쉽게 확장된다.

<표 III-3> 경사하강법 알고리즘(다변수 함수)

[단계 1]	초기 근사해 $\mathbf{x}_1$ , 허용오차 $0 \leq \epsilon < 1$ , 학습률 $\eta > 0$ 를 설정하고, $k := 1$ 로 한다.
[단계 2]	$\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ 를 계산한다. 만일 $\ \mathbf{g}_k\  \leq \epsilon$ 이면, 알고리즘을 멈추고 $\mathbf{x}_k$ 를 반환한다.
[단계 3]	$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \mathbf{g}_k$ , $k := k+1$ 이라 두고, [단계 2]로 이동한다.

여기서 스칼라  $x$ 가 벡터  $\mathbf{x}$ 로, 절댓값  $|g|$ 가 벡터의 노름  $\|\mathbf{g}\|$ 로, 도함수  $f'(x)$ 가 그레이디언트 (gradient)  $\nabla f(\mathbf{x})$ 로 바뀌었지만, 기본적인 구성은 1변수 함수의 경우와 완전히 동일하다. 물론 이와같이 1변수 함수에 대한 경사하강법을 다변수 함수로 확장하여 이해하는 방식 이외에 그레이디언트의 특성을 가지고 경사하강법의 아이디어를 설명할 수도 있다. 대학 미적분학에서 살펴보듯이 미분가능한 함수의 방향도함수 (directional derivative)와 코시-슈바르츠 부등식으로부터  $\mathbf{d} = -\nabla f(\mathbf{x}_k)$ 가  $\mathbf{x}_k$ 상에서 함수  $f$ 가 가장 빠르게 감소하는 방향임을 알 수 있다.

$$D_{\mathbf{d}}f(\mathbf{x}_k) = \lim_{t \rightarrow 0} \frac{f(\mathbf{x}_k + t\mathbf{d}) - f(\mathbf{x}_k)}{t} = \mathbf{d}^T \nabla f(\mathbf{x}_k), \quad |\mathbf{d}^T \nabla f(\mathbf{x}_k)| \leq \|\mathbf{d}\| \|\nabla f(\mathbf{x}_k)\|$$

따라서 경사하강법은 현재의 위치  $\mathbf{x}_k$  상에서 함수가 가장 빠르게 감소하는  $\mathbf{g}_k = -\nabla f(\mathbf{x}_k)$  방향으로  $\eta$ 만큼 이동하여 새로운 위치  $\mathbf{x}_{k+1}$ 을 얻는다고 이해할 수 있다. 다음 예제를 살펴보자.

**예제 2.** minimize  $f(x, y) = (x-2)^4 + (x-2)^2 y^2 + (y+1)^2, \quad (x, y) \in \mathbb{R}^2$   
 (단,  $(x_1, y_1) = (1, 1), \eta = 0.4, \epsilon = 10^{-3}$ ) (Dennis & Schnabel, 1996).

이 예제는 2변수 함수  $f(x, y)$ 의 최솟값을 계산하는 것으로, 함수  $f$ 의 수식으로 판단해볼 때 점  $(x^*, y^*) = (2, -1)$ 에서  $f$ 가 최솟값을 가짐을 쉽게 알 수 있다. <표 III-4>의 SageMath 코드를 활용하면  $f$ 의 등고선(level curve) 그래프와  $(x_k, y_k)$ 를 시각적으로 나타내어 경사하강법 알고리즘이 수렴하는 것을 쉽게 확인할 수 있다. 이 알고리즘을 통해 얻은 해를 소수점 아래 셋째 자리에서 반올림하면  $(2.00, -1.00)$ 이다.

<표 III-4> 예제 2를 해결하는 경사하강법 코드(SageMath)와 결과

```

var('x, y') # 변수 선언
f(x, y) = (x - 2)^4 + (x - 2)^2*y^2 + (y + 1)^2 # 함수 입력
gradf = f.gradient() # 그레이디언트 계산
u0 = vector([1.0, 1.0]) # 초기 근사해
tol = 1e-6 # 허용오차
eta = 0.1 # 학습률
s = [] # 그래프를 그리기 위한 용도
for k in range(300):
    g0 = gradf(u0[0], u0[1])
    s.append(u0) # 점 (x_k, y_k) 저장
    if g0.norm() <= tol:
        print("알고리즘 성공!")
        break
    u0 = u0 - eta*g0 # 경사하강법 반복단계

print("x* =", u0) # 해 출력
print("||g*|| =", g0.norm()) # 그레이디언트의 노름
print("f(x*) =", f(u0[0], u0[1])) # 극솟값 출력
print("반복 횟수 =", k + 1) # 반복 횟수
p1 = contour_plot(f, (x, 0, 4), (y, -2, 2), contours = [0, 0.2,...,6], cmap = 'hsv', fill = False) # 함수 f의 등
고선
p2 = line2d(s, color = 'black') + point(s, color = 'red') # (x_k, y_k) 그리기
show(p1 + p2, aspect_ratio = 1) # 동시에 그리기

```

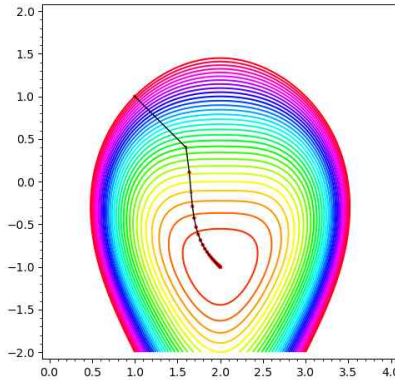
---

알고리즘 성공! # 그레이디언트의 노름이 허용 범위내에서 매우 작다는 의미  $\mathbf{g}_k \approx \mathbf{0}$

```

x* = (1.99999964647608, -0.999999671779490)
||g*|| = 9.64795555404069e-7
f(x*) = 2.32707780035600e-13
반복 횟수 = 72

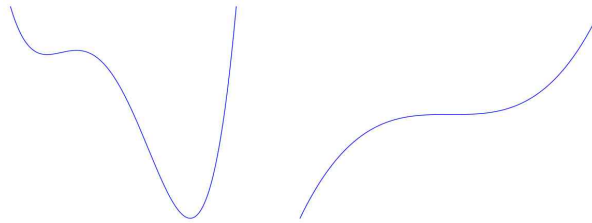
```





2. 경사하강법의 한계와 확률적 경사하강법

경사하강법은 물론 한계도 존재한다. 예를 들어, 1변수 함수와 같이 간단한 경우에도 함수가 비볼록이면([그림 III-2]), 경사하강법 적용시 시작점  $x_1$ 이 어디냐에 따라 서로 다른 극솟값 또는 (극대도 극소도 아닌) 임계점으로 수렴할 수도 있다. 즉 알고리즘을 통해 얻은 값이 임계점이라는 것 이외에는 보장하지 못한다. 그리고 다른 최적화 알고리즘(예. Newton's method, Quasi-Newton method)에 비해 이론적으로 보장하는 수렴 속도가 느리다. 또한 여기서 소개한 학습률  $\eta$ 는 모두 고정된 상수로 택하였는데, 문제에 따라서 적절한 학습률을 선택해야 한다. 이와 관련하여 다양한 방법이 있으나 구체적인 내용은 대학 미적분학 수준을 벗어나므로 최적화 관련 문헌을 참고하라(Bottou et al., 2018; Boyd & Vandenberghe, 2004; Fletcher, 1987; Nocedal & Wright, 2006; Sun & Yuan, 2010).



[그림 III-2] 비볼록인 함수의 그래프(예시)

실제 인공지능 알고리즘에서 최적화하려는 문제는 다음과 같이 특수한 형태를 가지고 있는데, 이러한 문제를 유한합 최적화(finite-sum minimization)라 한다. 여기서  $f_i$ 는 성분함수(component function)라 한다.

$$\text{minimize } F(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^m f_i(\mathbf{x}), \quad \mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$$

앞서 <표 II-1>에서 언급한 최소제곱문제도 이러한 유형에 속한다. 즉 행렬  $A \in \mathbb{R}^{m \times n}$ 의  $i$ 번째 행을  $A_{(i)} \in \mathbb{R}^n$ 라 하고  $\mathbf{b} = (b_1, \dots, b_m)$ 라 하면, 다음과 같은 형태로 나타낼 수 있다.<sup>4)</sup>

$$\text{minimize } \frac{1}{2m} \|A\mathbf{x} - \mathbf{b}\|^2 = \frac{1}{2m} \sum_{i=1}^m (A_{(i)}\mathbf{x} - b_i)^2 = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (A_{(i)}\mathbf{x} - b_i)^2, \quad \mathbf{x} \in \mathbb{R}^n$$

유한합 최적화 문제의 경우, 경사하강법을 적용하기 위해서는 그래디언트  $\nabla F(\mathbf{x}_t) = \frac{1}{m} \sum_{i=1}^m \nabla f_i(\mathbf{x}_t)$ 를 계산하여야 한다. 그러나  $m$ 이 매우 큰 경우,  $\nabla f_i(\mathbf{x}_t)$ 를 모두 계산하여 메모리에 저장하고 있는 것은 실용적이지 않다. 현실적인 데이터는 보통 중복된 데이터들을 포함하고 있어서 모든 데이터를 반복 단계마다 사용하는 것이 효율적이지 않을 수 있기 때문이다. 이때는, 앞서 소개한 경사하강법을 변형하여 만든 확률적 경사하강법

4) <표 II-1>에서는  $m$ 이 없이 최소제곱문제를 소개하였으나 두 문제의 본질은 같다.

(SGD)을 활용한다. 확률적 경사하강법에서는 각 반복 단계에서 임의로 하나의 성분함수를 선택하여 그 함수의 그래디언트  $\nabla f_{i_t}(\mathbf{x}_t)$ 로 전체 그래디언트  $\nabla F(\mathbf{x}_t)$ 를 대신한다. 이렇게 하면 경사하강법보다 적은 계산 비용으로 향상된 결과를 이끌어낼 수 있고, 특히 알고리즘의 시작 단계에서 매우 효과적일 수 있다. <표 III-5>는 확률적 경사하강법의 기본 알고리즘이다(이상구·이재화, 2019).

<표 III-5> 확률적 경사하강법의 기본 알고리즘

[단계 1]	초기 근사해 $\tilde{\mathbf{x}}_1$ , 허용오차 $0 \leq \epsilon \ll 1$ , 학습률 $\eta > 0$ 를 설정하고, $k := 1$ 로 한다.
[단계 2]	$\mathbf{x}_1 := \tilde{\mathbf{x}}_k$ 라 두고 다음을 $m$ 회 반복한다. ( $t = 1, 2, \dots, m$ ) $\{1, 2, \dots, m\}$ 에서 임의로 $i_t$ 를 선택한다. $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla f_{i_t}(\mathbf{x}_t)$ $\tilde{\mathbf{x}}_{k+1} := \mathbf{x}_{m+1}$ 로 한다.
[단계 3]	$\ \tilde{\mathbf{x}}_{k+1} - \tilde{\mathbf{x}}_k\  \leq \epsilon$ 이면 알고리즘을 멈추고 $\tilde{\mathbf{x}}_{k+1}$ 를 반환한다. 그렇지 않으면 $k := k+1$ 라 두고 [단계 2]로 이동한다.

본 논문에서는 경사하강법에 대한 이해를 목적으로 하기 때문에 SGD를 자세히 소개하지는 않는다. SGD에 관한 내용은 관련 문헌(Bottou et al., 2018; Wright & Recht, 2022)을 참고하라.

### 3. 인공지능 알고리즘에서 경사하강법의 활용 예시

경사하강법은 다양한 인공지능 알고리즘에 활용된다. 먼저 단순 선형회귀 모델의 계수를 최소제곱법으로 추정하는 예시를 소개한다. 다음 예제를 살펴보자.

**예제 3.** 주어진 다섯 개의 점 (1.2, 3.0), (2.0, 4.5), (3.1, 5.1), (4.6, 6.0), (7.2, 9.8)을 지나는 최소제곱 직선(least square line)  $y = a + bx$ 를 소수점 셋째 자리에서 반올림하여 계산하여라.

이 예제는 주어진 데이터에 대하여  $x$ 와  $y$ 의 관계를 가장 잘 보여주는 일차함수  $y = a + bx$ 를 찾으려는 것이다. 이와 같은 형태의 문제를 선형회귀라 한다. 가장 이상적인 상황은 모든 데이터  $(x_i, y_i)$ 에 대해서  $y_i = a + bx_i$ 가 만족되는  $y$ 절편  $a$ 와 기울기  $b$ 를 찾는 것이다. 그러나 실제로는 각 데이터  $(x_i, y_i)$ 에 대하여,  $\hat{y}_i = a + bx_i$ 라 할 때 오차  $(y_i - \hat{y}_i)^2$ 의 합  $E(a, b)$ 가 최소가 되는  $a, b$ 를 구한다. 따라서 다음과 같은 형태로 나타낼 수 있다(최용석, 2014; 최원, 2022).

$$\begin{aligned} \text{minimize } E(a, b) = & \frac{1}{2} \{ (a + 1.2b - 3.0)^2 + (a + 2.0b - 4.5)^2 \\ & + (a + 3.1b - 5.1)^2 + (a + 4.6b - 6.0)^2 + (a + 7.2b - 9.8)^2 \} \end{aligned}$$

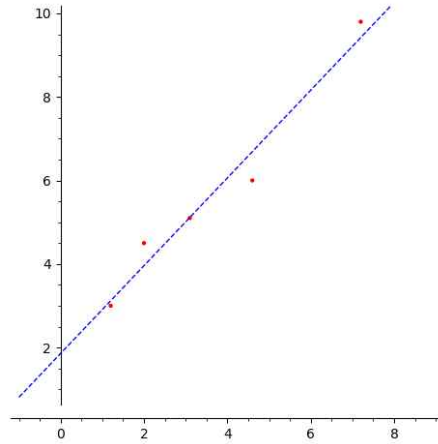
<표 III-4>의 SageMath 코드를 변형하여 시작점 (0, 0), 학습률  $\eta = 0.01$ , 허용오차  $\epsilon = 10^{-6}$ 으로 설정하고, 최대 반복 횟수를 2,000으로 늘려서 경사하강법을 진행하면,  $E(a, b)$ 가 최소가 되는  $a, b$ 는 각각  $a = 1.86$ ,  $b = 1.05$ 이다. 따라서 최소제곱직선은  $y = 1.86 + 1.05x$ 이다(<표 III-6>).

<표 III-6> 예제 3을 해결하는 경사하강법 코드(SageMath)와 결과 및 데이터와 최소제곱직선

```

var('a, b') # 변수 선언
E(a, b) = 0.5*((a + 1.2*b - 3.0)^2 + (a + 2.0*b - 4.5)^2 + (a + 3.1*b - 5.1)^2 + (a + 4.6*b - 6.0)^2 + (a + 7.2*b - 9.8)^2) # 함수 입력
gradE = E.gradient() # 그레이디언트 계산
u0 = vector([0.0, 0.0]) # 초기 근사해
tol = 1e-6 # 허용오차
eta = 0.01 # 학습률
for k in range(2000):
    g0 = gradE(u0[0], u0[1])
    if g0.norm() <= tol:
        print("알고리즘 성공!")
        break
    u0 = u0 - eta*g0 # 경사하강법 반복단계

print("(a*, b*) =", u0) # 해 출력
print("||g*|| =", g0.norm()) # 그레이디언트의 노름
print("E(a*, b*) =", E(u0[0], u0[1])) # 극솟값 출력
print("반복 횟수 =", k + 1) # 반복횟수
    
```



알고리즘 성공! # 그레이디언트의 노름이 허용 범위내에서 매우 작다는 의미  $\nabla E(a_k, b_k) \approx \mathbf{0}$   
 (a\*, b\*) = (1.86170908090266, 1.05477644336184)  
 ||g\*|| = 9.91504784356063e-7  
 E(a\*, b\*) = 0.462203036221992  
 반복 횟수 = 1176

다음은 앞서 언급한 경사하강법의 한계와 관련하여 손실함수 선택의 중요성을 로지스틱 회귀에 경사하강법을 적용하는 예시를 통해 설명한다. 로지스틱 회귀는 성공, 실패와 같은 이진 분류(binary classification) 문제에 사용되는 알고리즘으로, 사건이 발생할 확률을 추정한다. 일반적으로 분석해야 할 데이터의 종속 변수는 분류에 따라 0 또는 1로 나타낸다. 그리고 예측 결과는 확률이므로 0과 1 사이에 있다. 로지스틱 회귀 모형은 다음과 같은 시그모이드(sigmoid) 함수를 사용한다.

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

시그모이드 함수  $f(x)$ 는 다음과 같은 좋은 성질이 있다.

- ① 모든  $x$ 에 대하여  $0 < f(x) < 1$ 이다.  
 ②  $x > 0$ 일 때  $f(x) > 0.5$ ,  $x < 0$ 일 때  $f(x) < 0.5$ 이다.  
 ③ 시그모이드 함수의 미분 공식은 다음과 같다. 즉  $f(x)$ 를 그대로 활용할 수 있다.

$$f'(x) = -(1+e^{-x})^{-2}(-e^{-x}) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} = f(x)(1-f(x))$$

주어진 데이터가  $(x_i, y_i)$ ,  $i = 1, \dots, n$ 이고  $y_i$ 가 0 또는 1만 가질 때, 로지스틱 회귀는 다음과 같이 합숫값과  $y_i$  사이의 오차를 최소화하는  $a$ 와  $b$ 를 찾는 것이다.

$$y = f(a + bx) = \frac{1}{1+e^{-(a+bx)}}$$

그리고 오차를 수식으로 나타내면 다음과 같다.

$$E(a, b) = \sum_{i=1}^n [y_i - f(a + bx_i)]^2$$

그러나 로지스틱 회귀에서는 오차보다 다음과 같은 교차 엔트로피(cross entropy)를 최소화하는  $a$ 와  $b$ 를 찾는 것이 더 일반적이다.

$$CE(a, b) = - \sum_{i=1}^n [y_i \log f(a + bx_i) + (1 - y_i) \log(1 - f(a + bx_i))]$$

왜냐하면 오차를 최소화하는 문제에서는 손실함수가 비볼록이므로, 경사하강법의 초깃값을 설정하는 것이 더 어렵기 때문이다. 다음의 예제를 통해 이를 자세히 살펴보자.

**예제 4.** 실패를 0, 성공을 1로 표현하고, 각 독립변수의 값마다 성공과 실패를 나타내는 데이터가  $(-1, 0)$ ,  $(-0.6, 0)$ ,  $(-0.1, 1)$ ,  $(0.3, 1)$ ,  $(0.8, 1)$ ,  $(1.2, 1)$ ,  $(1.6, 0)$ ,  $(2.1, 0)$ ,  $(2.5, 1)$ ,  $(3, 1)$ 과 같이 열 개의 점으로 주어진다. 로지스틱 회귀 모델을 이용하여 오차와 교차 엔트로피를 계산하라.

[그림 III-3]은 예제 4에서 오차를 손실함수로 하는 2변수 함수  $E(a, b)$ 를 그린 것이다. 오차  $E(a, b)$ 는 볼록한 부분이 있지만 그라디언트  $\nabla E(a, b)$ 가  $\mathbf{0}$ 에 가까운 영역도 매우 넓어서 경사하강법이 잘 작동하지 않을 수도 있음을 직관적으로 알 수 있다. 반면 교차 엔트로피는 [그림 III-4]에서 보듯이 대부분의 영역에서 가파른 기울기를 가지는 볼록함수임을 알 수 있다. 이를 수학적으로 간단히 설명하기 위해 모수가 단 한 개인 경우를 생각해보자. 즉,  $f(x) = \frac{1}{1+e^{-bx}}$ 인 모형의 교차 엔트로피는 다음과 같다.

$$CE(b) = -y \log f(bx) - (1 - y) \log(1 - f(bx))$$

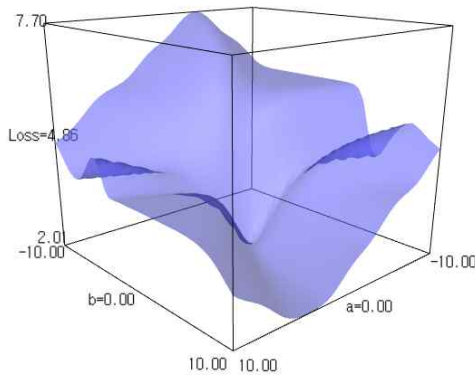
이제 함수  $CE$ 를 정리하면 다음과 같이 간단히 할 수 있다.

$$CE(b) = -y \log \left( \frac{1}{1+e^{-bx}} \right) - (1 - y) \log \left( 1 - \frac{1}{1+e^{-bx}} \right) = \log(1+e^{bx}) - bxy$$

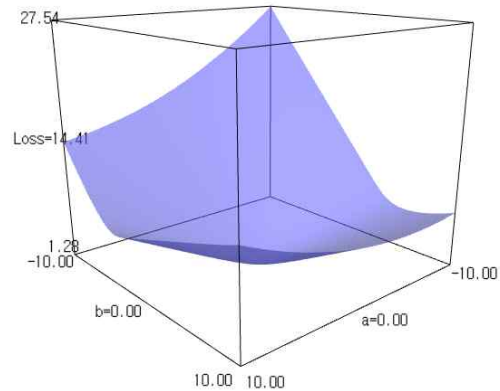
따라서  $CE$ 의 이계도함수는 다음과 같다.

$$\frac{d^2 CE}{db^2} = \frac{x^2 e^{-bx}}{(1 + e^{-bx})^2}$$

그러므로 모든  $x$ 에 대해  $\frac{d^2 CE}{db^2} \geq 0$ 이고, 이는  $CE$ 가 볼록함수임을 의미한다. 모수가 여러 개이거나 관찰 데이터의 차원이 높고 개수가 많더라도, 유사한 방식으로 교차 엔트로피가 항상 볼록함수임을 보일 수 있다. 따라서 교차 엔트로피에 경사하강법을 적용할 경우, 어떠한 초깃값을 설정하든 수렴하는 조건을 만족함을 의미한다.



[그림 III-3] 예제 4의 손실함수로 오차를 사용한 경우의 그래프



[그림 III-4] 예제 4의 손실함수로 교차 엔트로피를 사용한 경우의 그래프

실제 예제 4에 제시된 데이터에 대한 오차와 교차 엔트로피에 경사하강법을 적용하면, 오차는 시작점에 따라 수렴하지 않는 경우가 있었으나, 교차 엔트로피는 잘 작동하였다. 예를 들어, 시작점  $(10, 0)$ , 학습률  $\eta = 0.1$ , 허용오차  $\epsilon = 10^{-6}$ 로 설정하고, 최대 반복 횟수를 3,000으로 늘려서 경사하강법을 진행하면, 오차의 경우 3,000번 반복 후에 대략  $(9.88, -0.07)$ 을 해로 제시하였으나, 교차 엔트로피의 경우 652번 반복 후 허용오차 범위 내에서 대략  $(-0.04, 0.50)$ 를 해로 제시하였다. 또한 오차는 시작점에 따라 다른 점을 해로 제시하였고 허용범위 내에도 들지 못하였으나, 교차 엔트로피는 시작점을 변경하여도 허용범위 내에서 동일한 결과를 제시하였다. 따라서 로지스틱 회귀 문제에서는 교차 엔트로피가 경사하강법을 적용하기에 더 적절함을 확인할 수 있다.

참고로 인공신경망(artificial neural network) 모형에서 로지스틱 회귀는 매우 중요한 역할을 하는데, 로지스틱 회귀의 종속변수를 하나의 출력 노드로 보고 이러한 노드를 여러 개를 생성하는 경우를 단층 퍼셉트론(Single Layer Perceptrons, SLP)이라 하며, 여기에 은닉층을 포함하면 다층 퍼셉트론(Multi Layer Perceptrons, MLP)을 구성하여 심층학습이 가능하다. 그렇기 때문에 로지스틱 회귀에서 오차를 적용할 경우, 그레이디언트가 0에 가까운 영역이 많아서 학습 속도가 느려지는 경사하강법의 본질적 한계를 극복하지 못하여 더 이상 인공신경망을 발전시킬 수 없게 된다. 오차를 일반화한 평균 제곱오차(mean square error)는 1980년대와 1990년대 인공지능 연구에서 유행했으나 점점 교차 엔트로피가 그 역할을 대체하기 시작하였다. 현대 인공신경망에서는 오차를 최소화하는 대신 교차 엔트로피를 손실함수로 사용하여 최소화함으로써 인공지능 알고리즘 성능을 비약적

으로 개선할 수 있었다(Goodfellow et al., 2016).

지금까지 대학 미적분학 강좌에서 활용 가능한 인공지능 응용 사례로 경사하강법에 관하여 소개하였다. 학습자는 이를 통해 자신이 학습하는 수학 과목이 현실에서도 잘 활용되고 있음을 느끼며 학습 동기를 구체화할 수 있을 것이다. 또한 인공지능에 관심이 있는 학습자에게는 경사하강법이 추후 관련 알고리즘을 이해하는데 탄탄한 기초가 될 수 있다. 또한 제시한 SageMath 코드는 복잡한 함수에 대해서 수치적으로 문제를 해결하는데 적용 가능하다.

#### IV. 결론 및 제언

본 논문에서는 대학 미적분학 강좌에서 활용 가능한 인공지능 관련 응용 예시로, 인공지능 알고리즘에서 많이 활용되는 경사하강법에 관하여 소개하였다. 경사하강법에 대한 개념 소개 및 시각화 자료와 SageMath 코드, 선형회귀에서 발생하는 최소제곱문제를 경사하강법으로 해결한 예시를 함께 제시하였다. 이 과정은 미적분학에서 도함수(또는 그레이디언트)가 왜 중요한지를 보여주는 좋은 사례가 될 수 있다. 특히 다변수 함수의 그레이디언트가 인공지능을 구동하는데 어떻게 활용되는지를 잘 보여준다.

본 논문에서 제시하는 교수·학습자료를 활용하여 수업을 계획할 때 다음 사항을 고려하기를 제언한다.

- ① 학습자가 추후 인공지능 알고리즘을 공부하는 데 도움이 되도록 수업을 구성해야 한다. 예를 들어, 실제 인공지능 문제를 해결할 때, 경사하강법을 변형한 확률적 경사하강법(SGD)이 주로 사용되는데, 이는 많은 양의 데이터로부터 구성된 인공지능 문제에서 기인한다. 따라서 인공지능 알고리즘의 특징 및 경사하강법의 근본적 한계와 이를 극복하기 위한 조치 등도 간단히 소개된다면, 학습자가 경사하강법의 본질을 파악한 후 SGD도 유연하게 받아들일 수 있게 될 것이다.
- ② 인공지능을 위한 수학교육에서는 모델링의 관점에서 접근할 필요가 있다. 단순히 최적화 문제의 풀이를 위해 경사하강법을 배우는 것 이외에 주어진 문제를 최적화의 관점으로 접근할 수 있는 사고력을 길러 주는 것이 중요하다. 무엇을 손실함수로 생각할 것인가? 문제의 근본적인 제약조건은 무엇인가? 학생들이 질문하고 고민하는 과정을 가지도록 하면, 학생들이 실제 문제에 따라 변형된 인공지능 알고리즘의 핵심을 파악하고 이해하는데 도움이 될 것이다.

경사하강법은 미분가능한 함수를 최소화하는 기본적인 방법이지만 현재에도 인공지능 등 다양한 분야에서 지속적으로 활용되고 있다. 따라서 대학 미적분학 강의에서 사용 가능한 코드와 함께 인공지능 응용 예시를 소개한다면, 학생들이 미적분학을 왜 배우는지 또한 대학에서 학습하는 수학 과목이 현실에서도 잘 활용되고 있음을 보여주어, 학생들에게 구체적인 학습 동기를 부여할 수 있다. 또한 지필로 계산할 수 없는 실제 문제에 대해서도 활용 가능한 알고리즘을 제시하였으므로, 이를 바탕으로 실제적인 대안을 찾을 수 있게 된다. 본 연구는 대학 미적분학을 지도하는 다양한 전공의 교수자들에게 기초자료로 도움이 될 것이며, 수업 교안 및 관련 교재를 집필하는데 활용될 수 있다. 특히 인공지능경망과 관련된 기초 지식인 로지스틱 회귀 모형에서 단순한 예제와 그림을 통해 어떠한 손실함수가 더 효율적인지 이해할 수 있는 자료를 제시하였다. 이로써 복잡한 컴퓨터 공학적 용어를 사용하지 않고도 비교적 쉽고 직관적으로 설명할 수 있었다. 본 연구가 학생들이 경사하강법과 같은 인공지능 알고리즘을 이해하는데 많은 도움이 되기를 기대한다.

## 참 고 문 헌

- 이상구·이재화 (2019). [빅북] 인공지능을 위한 기초수학. 교보문고.
- Lee, S.-G., & Lee, J.H. (2019). [BigBook] *Basic mathematics for artificial intelligence*. Kyobo Book Centre. <http://matrix.skku.ac.kr/math4ai/Math4AI.pdf>
- 이상구·이재화·유주연·함윤미 (2022). *다변수 미적분학 & 코딩*. 경문사.
- Lee, S.-G., Lee, J.H., Yoo, J.Y. & Ham, Y. (2022). *Multivariable calculus & coding*. Kyung Moon Sa. <https://buk.io/@kc7895>
- 이상구·이재화·함윤미 (2020a). 인공지능(Artificial Intelligence)과 대학수학교육. *수학교육 논문집*, 34(1), 1-15.
- Lee, S.-G., Lee, J.H. & Ham, Y. (2020a). Artificial intelligence and college mathematics education. *Communications of Mathematical Education*, **34(1)**, 1-15.
- 이상구·이재화·함윤미·박경은 (2020b). *인공지능을 위한 기초수학 입문*. 경문사.
- Lee, S.-G., Lee, J.H., Ham, Y. & Park, K.-E. (2020b). *Introductory mathematics for artificial intelligence*. Kyung Moon Sa. <http://matrix.skku.ac.kr/math4ai-intro/>
- 최용석 (2014). [빅북] *R과 함께하는 통계학의 이해*. 교보문고.
- Choi, Y.-S. (2014). [BigBook] *Understanding statistics with R*. Kyobo Book Centre.
- 최원 (2022). *(알기 쉬운 통계 원리) 기초통계학 (제3판)*. 경문사
- Choi, W. (2022). *Elementary statistics (3rd edition)*. Kyung Moon Sa.
- Bottou, L., Curtis, F.E. & Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, **60(2)**, 223-311.
- Boyd, S. & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.
- Burden, R.L. & Faires, J.D. (2010). *Numerical Analysis (9th edition)*. Cengage Learning.
- Cauchy, A. (1847). Méthode générale pour la résolution des systèmes d'équations simultanées. *C. R. Acad. Sci. Paris*, **25**, 536-538.
- Chapra, S. & Canale, R. (2020). *Numerical methods for engineers (8th edition)*, McGraw-Hill Education.
- Dennis, J.E. Jr. & Schnabel, R.B. (1996). *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM.
- Fletcher, R. (1987). *Practical methods of optimization (2nd edition)*. John Wiley and Sons.
- Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep learning*. MIT Press. <http://www.deeplearningbook.org>
- Lemaréchal, C. (2012). Cauchy and the gradient method. *Documenta Mathematica*, Extra Vol. Optimization Stories, 251-254.
- Nocedal J. & Wright, S. (2006). *Numerical optimization (2nd edition)*. Springer.
- Sun, W. & Yuan, Y.-X. (2010). *Optimization theory and methods: Nonlinear programming*. Springer.
- Wright, S.J. & Recht, B. (2022). *Optimization for data analysis*. Cambridge University Press.

## A Study on the Development of Teaching-Learning Materials for Gradient Descent Method in College AI Mathematics Classes

**Lee, Sang-Gu**

Department of Mathematics, Sungkyunkwan University

E-mail : sglee@skku.edu

**Nam, Yun**

Institute of Basic Science, Sungkyunkwan University

E-mail : yunnam@skku.edu

**Lee, Jae Hwa<sup>†</sup>**

Research Institute of Basic Sciences, Sungkyunkwan University

E-mail : jhlee2chn@skku.edu

In this paper, we present our new teaching and learning materials on gradient descent method, which is widely used in artificial intelligence, available for college mathematics. These materials provide a good explanation of gradient descent method at the level of college calculus, and the presented SageMath code can help students to solve minimization problems easily. And we introduce how to solve least squares problem using gradient descent method. This study can be helpful to instructors who teach various college-level mathematics subjects such as calculus, engineering mathematics, numerical analysis, and applied mathematics.

---

\* 2000 Mathematics Subject Classification : 97U50, 97U60, 97U70

\* Key words : college mathematics, extreme value, stationary point, gradient descent method, artificial intelligence, least squares

\* This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2021R1F1A1046714).

<sup>†</sup> Corresponding author