ĐẠO HÀM VÀ VI PHÂN TRONG TỐI ƯU HÓA HỌC MÁY: CƠ SỞ TOÁN HỌC CỦA GRADIENT DESCENT VÀ BACKPROPAGATION

Lê Bích Phượng[1,*], Nguyễn Tiến Khởi[1]
[1]*Trường Đại học Mỏ - Địa chất*
[*]*Email: lebichphuong@humg.edu.vn*

TÓM TẮT

Trong học máy, tối ưu hóa là quá trình quan trọng để đạt được hiệu suất cao nhất cho các mô hình. Hai công cụ then chốt trong quá trình này là đạo hàm và vi phân, giúp tính toán gradient và điều hướng tối ưu. Bài báo này tập trung vào vai trò của đạo hàm và vi phân trong Gradient Descent và Backpropagation – hai kỹ thuật quan trọng trong tối ưu hóa học máy.

Từ khóa: *Đạo hàm, vi phân, gradient, tối ưu, học máy.*

1. ĐẶT VẤN ĐỀ

1.1. Giới thiệu

Học máy dựa trên việc huấn luyện mô hình dữ liệu để tối thiểu hàm thất thoát (loss function). Trong quá trình này, Gradient Descent và Backpropagation đóng vai trò trung tâm, Gradient Descent dùng gradient (đạo hàm của hàm thất thoát theo tham số) để xác định hướng tối ưu, trong khi Backpropagation tính gradient qua các lớp của mạng nơ – ron.

1.2. Đạo hàm và Vi phân

Đạo hàm: Được sử dụng để xác định tốc độ thay đổi tại một điểm. Trong tối ưu hóa, đạo hàm giúp tính hướng di chuyển nhanh nhất nhằm giảm hàm thất thoát.

Vi phân: Liên quan đến sự thay đổi nhỏ trong hàm số khi tham số thay đổi nhỏ. Vi phân hỗ trợ trong việc định độ lớn của thay đổi gradient.

2. PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Đạo hàm

Định nghĩa:

Đạo hàm của một hàm số $f(x)$ tại một điểm x_o được định nghĩa là tốc độ thay đổi của hàm số đó khi x thay đổi nhỏ quanh x_o. Nó đo lường sự thay đổi ngay lập tức (instantaneous rate of change) của hàm số tại điểm đó:

Công thức:

$$f'(x_0)=\lim_{\Delta x\to 0}\frac{f(x_0+\Delta x)-f(x_0)}{\Delta x} \quad (1)$$

Trong thực tế, đạo hàm biểu thị độ dốc (slope) của tiếp tuyến tại một điểm trên đồ thị của hàm số.

Vai trò trong tối ưu hóa:

Trong học máy, đạo hàm giúp xác định hướng di chuyển nhanh nhất để giảm giá trị của hàm thất thoát (loss function). Cụ thể là:

Nếu đạo hàm dương $\left(f'(x)>0\right)$, điều này cho thấy hàm số đang tăng và ta cần di chuyển ngược lại chiều của đạo hàm để giảm hàm thất thoát.

Nếu đạo hàm âm $\left(f'(x)<0\right)$, ta di chuyển cùng chiều đạo hàm.

Ví dụ cụ thể:

- Giả sử $f(x)=x^2$, đạo hàm của hàm này là $f'(x)=2x$.
- Tại $x=2$: $f'(x)=4>0$ (hàm đang tăng).
- Tại $x=-1$: $f'(x)=-2<0$ (hàm đang giảm).

2.2. Vi phân

Định nghĩa

Vi phân là sự thay đổi nhỏ trong giá trị của hàm số khi đầu vào thay đổi một lượng nhỏ. Nếu một hàm $f(x)$ khả vi tại x, vi phân của

$f(x)$ tại x_o được biểu diễn bằng:

$$df = f'(x_0).dx \quad (2)$$

Trong đó:

df :sự thay đổi nhỏ trong giá trị của hàm số.

dx: sự thay đổi nhỏ trong giá trị đầu vào.

Vai trò trong tối ưu hóa

Vi phân giúp cung cấp một ước lượng tuyến tính cho sự thay đổi của hàm số khi tham số thay đổi. Nó hỗ trợ trong việc định hướng độ lớn của gradient, tức là quyết định bước nhảy (step size) trong các thuật toán tối ưu hóa như Gradient Descent.

Ví dụ cụ thể

Với $f(x) = x^2$ và đạo hàm $f'(x) = 2x$:

Cho $x = 2$ và $dx = -0,1$:

$$df = f'(2).dx = 4.(-0,1) = -0,4$$

Điều này cho thấy giá trị của $f(x)$ sẽ giảm khoảng 0,4 nếu x giảm 0,1.

3. KẾT QUẢ VÀ THẢO LUẬN

3.1. Sự liên hệ giữa Đạo hàm và Vi phân

a) Đạo hàm cung cấp thông tin định hướng:

Đạo hàm cho biết ta nên tăng hay giảm tham số để giảm giá trị hàm thất thoát trong tối ưu hóa.

Ví dụ: Nếu $f'(x) > 0$, hàm số đang tăng. Để tối ưu hóa (giảm hàm thất thoát), ta cần di chuyển ngược lại chiều của đạo hàm

$$\left(x \to x - \eta.f'(x)\right). \quad (3)$$

b) Vi phân định lượng mức độ thay đổi:

Dựa trên đạo hàm, vi phân giúp ước lượng giá trị cụ thể của sự thay đổi trong hàm số, giúp điều chỉnh bước nhảy (step size) trong các thuật toán tối ưu hóa như Gradient Descent.

3.2. Ứng dụng trong học máy

Trong tối ưu hóa học máy:

- **Đạo hàm** được sử dụng để tính gradient, xác định hướng di chuyển tối ưu.
- **Vi phân** hỗ trợ trong việc kiểm soát độ lớn bước nhảy (step size), đảm bảo các tham số thay đổi một cách hiệu quả.

Ví dụ trong Gradient Descent:

Đạo hàm cung cấp hướng di chuyển:

$$x_{k+1} = x_k - \eta.f'(x_k) \quad (4)$$

Trong đó: $f'(x_k)$ là gradient tại bước k.

Vi phân ước lượng sự thay đổi giá trị hàm thất thoát:

$$\Delta f = f'(x_k).\Delta x \quad (5)$$

Điều này giúp kiểm soát sự thay đổi nhỏ trong các lần cập nhật, đảm bảo việc hội tụ ổn định. Đạo hàm xác định hướng và tốc độ thay đổi, cung cấp thông tin quan trọng để tìm ra hướng tối ưu. Vi phân định lượng mức độ thay đổi, hỗ trợ trong việc điều chỉnh bước nhảy phù hợp. Sự kết hợp của đạo hàm và vi phân là nền tảng cho các thuật toán tối ưu hóa như Gradient Descent và Backpropagation, góp phần tối ưu hóa hiệu quả mô hình học máy.

3.3. Gradient Descent: Tối ưu hóa dựa trên đạo hàm

Gradient Descent là một thuật toán tối ưu hóa cơ bản và hiệu quả, được sử dụng rộng rãi để tìm giá trị cực tiểu của một hàm thất thoát. Thuật toán dựa vào đạo hàm để xác định hướng di chuyển trong không gian tham số nhằm giảm giá trị hàm thất thoát nhanh nhất. Các biến thể phổ biến bao gồm: [1-2]

Gradient Descent toàn bộ (Batch Gradient Descent): sử dụng toàn bộ dữ liệu.

Mini-batch Gradient Descent: sử dụng tập con của dữ liệu.

Stochastic Gradient Descent (SGD): cập nhật tham số sau mỗi dữ liệu.

Gradient Descent khai thác quy tắc chuỗi (đạo hàm vi phân của hàm hợp) để tính gradient trong các hàm thất thoát phức tạp.

Cách hoạt động của Gradient Descent

Gradient Descent dựa trên ý tưởng di chuyển tham số theo hướng ngược lại với gradient của hàm mất, vì gradient chỉ ra hướng tăng dần lớn nhất của hàm mất. Công thức cập

nhật tham số θ trong Gradient Descent là:

$$\theta_{t+1} = \theta_t - \eta.\nabla_\theta L(\theta_t) \quad [2] \qquad (6)$$

Trong đó:

θ_t: giá trị tham số tại bước t.

η: tốc độ học (learning rate), xác định bước nhảy trong mỗi lần cập nhật.

$\nabla_\theta L(\theta_t)$: gradient của hàm thất thoát L tại θ_t.

Các biến thể phổ biến của Gradient Descent

a) Gradient Descent toàn bộ (Batch Gradient Descent)

Đặc điểm: Sử dụng toàn bộ dữ liệu huấn luyện để tính gradient trong mỗi lần cập nhật. Công thức:

$$\nabla_\theta L(\theta) = \frac{1}{N}\sum_{i=1}^{m}\nabla_\theta L_i(\theta) \qquad (7)$$

Trong đó: N là số lượng mẫu trong tập dữ liệu, $L_i(\theta)$ là hàm mất của mẫu i.

Ưu điểm: Gradient được tính chính xác, do đó hướng di chuyển là tối ưu nhất.

Nhược điểm: Chậm nếu tập dữ liệu quá lớn, vì mỗi lần cập nhật đòi hỏi quét qua toàn bộ dữ liệu.

b) Mini – batch Gradient Descent

Đặc điểm: Sử dụng một tập con nhỏ (mini – batch) gồm m mẫu dữ liệu để tính gradient trong mỗi lần cập nhật. Công thức:

$$\nabla_\theta L(\theta) = \frac{1}{m}\sum_{j=1}^{m}\nabla_\theta L_j(\theta) \qquad (8)$$

Ưu điểm:

- Cân bằng giữa hiệu suất tính toán và sự chính xác của gradient.
- Tăng tốc độ học và tận dụng song song phần cứng.

Nhược điểm: Hướng di chuyển có thể bị nhiễu nhẹ do gradient được ước tính từ tập con dữ liệu.

c) Stochastic Gradient Descent (SGD)

Đặc điểm: Cập nhật tham số sau mỗi dữ liệu, thay vì sử dụng toàn bộ dữ liệu hoặc mini – batch. Công thức:

$$\nabla_\theta L(\theta) = \nabla_\theta L_i(\theta), \quad i \in \{1, 2, ..., N\} \qquad (9)$$

Ưu điểm:

- Cập nhật nhanh, phù hợp cho các tập dữ liệu lớn.
- Tránh rơi vào cực tiểu cục bộ nhờ tính ngẫu nhiên trong việc chọn mẫu.

Nhược điểm:

- Gradient dao động mạnh, làm chậm quá trình hội tụ.
- Đòi hỏi các kỹ thuật như giảm tốc độ học (learning rate decay) để đảm bảo hội tụ.

Gradient Descent và Quy tắc chuỗi

Gradient Descent áp dụng quy tắc chuỗi (chain rule) để tính gradient hiệu quả trong các hàm thất thoát phức tạp, đặc biệt trong mạng nơ-ron. Quy tắc chuỗi cho phép tính toán gradient của các tham số ở từng lớp thông qua việc lan truyền gradient từ đầu ra ngược về đầu vào.

Ví dụ: Với một mạng nơ-ron nhiều lớp, hàm thất thoát L là hàm của đầu ra y, và đầu ra phụ thuộc vào các tham số của mạng:

$$L = f\left(g\left(h\left(x;\theta_1\right);\theta_2\right);\theta_3\right) \qquad (10)$$

Gradient của L theo θ_1 có thể được tính bằng quy tắc chuỗi:

$$\frac{\partial L}{\partial \theta_1} = \frac{\partial L}{\partial y}.\frac{\partial y}{\partial z}.\frac{\partial z}{\partial \theta_1} \qquad (11)$$

Tương tự, gradient của L theo θ_2 và θ_3:

$$\frac{\partial L}{\partial \theta_2} = \frac{\partial L}{\partial y}.\frac{\partial y}{\partial z}.\frac{\partial z}{\partial \theta_2}; \qquad (12)$$

$$\frac{\partial L}{\partial \theta_3} = \frac{\partial L}{\partial y}.\frac{\partial y}{\partial z}.\frac{\partial z}{\partial \theta_3} \qquad (13)$$

Gradient Descent là một công cụ mạnh mẽ trong tối ưu hóa, với các biến thể phù hợp cho các bài toán và quy mô dữ liệu khác nhau. Quy tắc chuỗi là nền tảng để tính gradient trong các hàm phức tạp, giúp thuật toán hoạt động hiệu

quả trên các mô hình học sâu.

3.4. Backpropagation: Lan truyền gradient qua các lớp

Backpropagation đóng vai trò trung tâm trong huấn luyện mạng nơ-ron. Quy tắc chuỗi được sử dụng để lan truyền gradient qua các lớp, tính gradient từ lớp sau ngược về lớp trước. Cách tính toán hiệu quả này giúp giảm thiểu khối lượng tính toán so với tính từng gradient riêng lẻ, đảm bảo việc cập nhật tham số diễn ra chính xác và hiệu quả [3].

Cách hoạt động của Backpropagation

Mạng nơ-ron bao gồm nhiều lớp, mỗi lớp thực hiện một phép biến đổi tuyến tính và phi tuyến tính. Để huấn luyện mạng, mục tiêu là tối thiểu hóa một hàm thất thoát L, thường phụ thuộc vào đầu ra y dự đoán của mạng và nhãn thực tế y_{true}. Backpropagation tính gradient của L với các tham số θ trong mạng thông qua 3 bước chính:

Lan truyền xuôi (Forward Propagation)

Đầu vào x được truyền qua từng lớp để tính đầu ra y.

Ví dụ: Với mạng gồm 3 lớp, các phép toán như sau:

$$z_1 = W_1 x + b_1, \qquad a_1 = f(z_1) \tag{14}$$

$$z_2 = W_2 a_1 + b_2, \qquad a_2 = f(z_2) \tag{15}$$

$$z_3 = W_3 a_2 + b_3, \qquad y = f(z_3) \tag{16}$$

Trong đó:

- W_i, b_i: trọng số và độ lệch (bias) của lớp i.
- $f(z)$: hàm kích hoạt (ReLU, sigmoid, softmax, v.v.).
- y: đầu ra của mạng.

Lan truyền ngược (Backward Propagation)

Dựa vào hàm mất mát L, tính gradient của L với các tham số của mạng.

Gradient của hàm mất mát tại lớp cuối cùng là:

$$\delta^{(3)} = \frac{\partial L}{\partial z_3} \tag{17}$$

Sau đó lan truyền gradient ngược qua các lớp bằng quy tắc chuỗi:

$$\delta^{(2)} = \delta^{(3)}.W_3^T.f'(z_2) \tag{18}$$

$$\delta^{(1)} = \delta^{(2)}.W_2^T.f'(z_1) \tag{19}$$

Cập nhật tham số:

Gradient của các tham số W_i và b_i được tính dựa trên:

$$\frac{\partial L}{\partial W_i} = a_{i-1}^T.\delta^{(i)}, \tag{20}$$

$$\frac{\partial L}{\partial b_i} = \delta^{(i)} \tag{21}$$

Cập nhật tham số

Sau khi tính gradient, tham số W_i và b_i được cập nhật dựa trên Gradient Descent:

$$W_i = W_i - \eta.\frac{\partial L}{\partial W_i}, \tag{22}$$

$$b_i = b_i - \eta.\frac{\partial L}{\partial b_i} \tag{23}$$

Trong đó: η là tốc độ học.

Lợi ích của Backpropagation

Tính toán hiệu quả

Backpropagation sử dụng **quy tắc chuỗi** (chain rule) trong toán học để tính toán gradient của hàm mất mát theo tất cả các tham số của mạng nơ-ron. Điều này giúp:

Tiết kiệm thời gian và tài nguyên: Thay vì tính toán gradient riêng lẻ cho từng tham số, Backpropagation tính toàn bộ gradient chỉ trong một lần lan truyền ngược qua mạng.

Tính toán nhanh: Với độ phức tạp O(n), thuật toán có thể huấn luyện các mạng nơ-ron lớn với hàng trăm triệu tham số một cách hiệu quả. Trong một mạng nơ-ron với nhiều lớp, Backpropagation giúp lan truyền lỗi từ lớp đầu ra về lớp đầu vào chỉ trong một lần, thay vì tính toán lại từng lớp.

- **Ứng dụng rộng rãi**

Backpropagation là phương pháp nền tảng trong huấn luyện mạng nơ-ron, được sử dụng rộng rãi trong nhiều lĩnh vực:

Nhận dạng hình ảnh (Image Recognition): Các mạng nơ-ron như CNN (Convolutional Neural Networks) dựa vào Backpropagation để học các đặc trưng phức tạp từ hình ảnh.

Xử lý ngôn ngữ tự nhiên (Natural Language Processing): Các mô hình như RNN, LSTM, và Transformer sử dụng Backpropagation để tối ưu hóa tham số trong việc dự đoán chuỗi hoặc xử lý văn bản.

Học tăng cường (Reinforcement Learning): Backpropagation hỗ trợ huấn luyện các mô hình học tăng cường sâu (Deep Reinforcement Learning) để giải quyết các bài toán như trò chơi hoặc điều khiển robot.

3.5. Ví dụ thực tế

a) Gradient Descent

Trong hồi quy tuyến tính: Tìm đường thẳng tối ưu để dự đoán giá trị y dựa trên đầu vào x. Mô hình hồi quy tuyến tính có dạng:

$$y_{pred} = \mathrm{w}.x + b \quad (24)$$

Hàm thất thoát sử dụng:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_{pred} - y_{true}\right)^2 \quad (25)$$

Ví dụ: Dữ liệu $x = [1,2,3], y_{true} = [2,4,6]$

Trọng số khởi tạo: $\mathrm{w} = 0.5, b = 0.5$

Học suất: $\eta = 0.1$

Quá trình cập nhật tham số:

Gradient Descent cập nhật w và b dựa trên gradient:

$$\frac{\partial MSE}{\partial \mathrm{w}} = \frac{2}{n}\sum_{i=1}^{n}\left(y_{pred} - y_{true}\right).x_i \quad (26)$$

$$\frac{\partial MSE}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}\left(y_{pred} - y_{true}\right) \quad (27)$$

Bước 1. Tính toán gradient tại $\mathrm{w} = 0.5, b = 0.5$

$y_{pred} = [1, 1.5, 2]$

Sai số: $y_{pred} - y_{true} = [-1, -2.5, -4]$

Gradient:

$$\frac{\partial MSE}{\partial \mathrm{w}} = \frac{2}{n}\sum_{i=1}^{n}\left(y_{pred} - y_{true}\right).x_i = \frac{2}{3}\left(-1\cdot 1 - 2.5\cdot 2 - 4\cdot 3\right) = -12$$

$$\frac{\partial MSE}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}\left(y_{pred} - y_{true}\right) = \frac{2}{3}\left(-1 - 2.5 - 4\right) = -5$$

Bước 2. Cập nhật vào w và b

$$\mathrm{w} = \mathrm{w} - \eta\cdot\frac{\partial MSE}{\partial \mathrm{w}} = 0.5 - 0.1\cdot(-12) = 1.7$$

$$b = b - \eta\cdot\frac{\partial MSE}{\partial b} = 0.5 - 0.1\cdot(-5) = 1$$

Lặp lại quá trình này cho đến khi sự thay đổi các giá trị của w và b giữa các bước lặp trở nên rất nhỏ hoặc hàm thất thoát giảm đến mức đủ thấp.

b) Backpropagation: là kỹ thuật chính để huấn luyện mạng nơ-ron sâu trong nhiều lĩnh vực:

Bài toán: Huấn luyện một mạng nơ-ron đơn giản với một đầu vào x, một lớp ẩn có 2 nơ - ron và một lớp đầu ra. Hàm thất thoát là hàm lỗi bình phương trung bình (MSE).

Cấu trúc mạng:

- Đầu vào: $x = [1; 2]$
- Lớp ẩn:
  - Trọng số là $W_1 = \begin{bmatrix} 0,1 & 0,2 \\ 0,3 & 0,4 \end{bmatrix}$, độ lệch là $b_1 = [0,1 \quad 0,2]$
  - Hàm kích hoạt: ReLU.
- Lớp đầu ra
  - Trọng số $W_2 = [0,5 \quad 0,6]$, độ lệch $b_2 = 0,3.$
  - Hàm kích hoạt: Linear.
- Nhãn thực tế: $y_{true} = 1.$

(1) Lan truyền xuôi:

- Tính toán tại lớp ẩn:

$$z_1 = W_1.x + b_1 = \begin{bmatrix} 0,1 & 0,2 \\ 0,3 & 0,4 \end{bmatrix}.\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0,1 \\ 0,2 \end{bmatrix} = \begin{bmatrix} 0,6 \\ 1,3 \end{bmatrix}$$

$$a_1 = \mathrm{Re}LU(z_1) = \begin{bmatrix} 0,6 \\ 1,3 \end{bmatrix}$$

- Tính toán tại lớp đầu ra:

$$z_2 = W_2.a_1 + b_2 = \begin{bmatrix} 0,5 & 0,6 \end{bmatrix}.\begin{bmatrix} 0,6 \\ 1,3 \end{bmatrix} + 0,3 = 1,38$$

$$y = z_2 = 1,38$$

Lan truyền ngược:

- Tính gradient tại lớp đầu ra:

$$\frac{\partial L}{\partial z_2} = 2.(y - y_{true}) = 2.(1,38 - 1) = 0,76$$

- Gradient tại lớp ẩn:

$$\delta^{(1)} = \frac{\partial L}{\partial z_1} = W_2^T . \frac{\partial L}{\partial z_2} . f'(z_1)$$

Với $f'(z_1) = [1,1]$ (ReLU chỉ đạo hàm bằng 1 nếu $z > 0$):

$$\delta^{(1)} = [0,5 \quad 0,6]^T .0,76.[1 \quad 1] = [0,38 \quad 0,456]$$

(2) Cập nhật tham số:

Cập nhật W_2:

$$\frac{\partial L}{\partial W_2} = a_1 . \frac{\partial L}{\partial z_2} = [0,6 \quad 1,3].0,76$$

Tương tự tính cho b_1, W_1.

4. KẾT LUẬN VÀ KIẾN NGHỊ

4.1. Kết luận

Bài báo đã làm sáng tỏ vai trò quan trọng của đạo hàm và vi phân trong tối ưu hóa học máy, đặc biệt trong các thuật toán Gradient Descent và Backpropagation. Các khái niệm này không chỉ là nền tảng để xác định hướng và mức độ thay đổi trong quá trình huấn luyện mô hình, mà còn giúp đảm bảo quá trình hội tụ diễn ra hiệu quả và ổn định. Bằng cách kết hợp giữa lý thuyết toán học và ứng dụng thực tiễn, bài báo đã cung cấp một cái nhìn sâu sắc về cách tối ưu hóa có thể được cải thiện thông qua việc sử dụng hiệu quả các công cụ toán học.

Qua các ví dụ cụ thể, từ Gradient Descent cơ bản đến ứng dụng Backpropagation trong mạng nơ-ron sâu, bài báo khẳng định rằng việc hiểu rõ và áp dụng đúng các nguyên lý đạo hàm và vi phân là yếu tố cốt lõi trong việc phát triển các mô hình học máy hiện đại.

4.2. Kiến nghị

Tích hợp giáo dục: Đề xuất đưa các khái niệm toán học nền tảng như đạo hàm, vi phân và quy tắc chuỗi vào chương trình giảng dạy học máy ở các cấp độ khác nhau để nâng cao nhận thức và khả năng ứng dụng của người học.

Ứng dụng thực tiễn: Thúc đẩy việc áp dụng các thuật toán tối ưu hóa vào các lĩnh vực cụ thể như thị giác máy tính, xử lý ngôn ngữ tự nhiên và học tăng cường để giải quyết các vấn đề thực tế.

5. LỜI CẢM ƠN

Nghiên cứu này được tài trợ bởi trường Đại học Mỏ - Địa chất, trong đề tài mã số T25-20.

TÀI LIỆU THAM KHẢO

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
2. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
3. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature.
4. Nguyễn Trường Thanh và Mai Viết thuận (chủ biên) (2019). Giáo trình giải tích 1. Nhà xuất bản Đại học Quốc gia Hà Nội.
5. Nguyễn Đình Trí (chủ biên), Tạ Văn Đĩnh, Nguyễn Hồ Quỳnh (2019). Toán cao cấp (tập 2) Phép tính giải tích một biến số. Nhà xuất bản Giáo dục.
6. Thomas (2009). Calculus. Pearson.

7. Eric Maththes (2015). Python crash course.
8. A Modern Approach (2019). Artificial Intelligence. Pearrson.

Thông tin của tác giả:

TS. Lê Bích Phượng
Giảng viên chính, Bộ môn Toán, Khoa Khoa học cơ bản, Trường Đại học Mỏ - Địa chất.
Điện thoại: +(84).988.782.112 - Email: lebichphuong@humg.edu.vn

Nguyễn Tiến Khởi
Khoa Công nghệ thông tin, Trường Đại học Mỏ - Địa chất.
Điện thoại: +(84).387.067.695 - Email: 2321050038@student.humg.edu.vn

DERIVATIVES AND DIFFERENTIALS IN MACHINE LEARNING OPTIMIZATION: THE MATHEMATICAL FOUNDATION OF GRADIENT DESCENT AND BACKPROPAGATION

Information about authors:

Le Bich Phuong, Ph.D, Department of Mathematics, Faculty of Basic Sciences, Hanoi University of Mining and Geology.
Phone: +(84) 988 782 112 – Email: lebichphuong@humg.edu.vn

Nguyen Tien Khoi, Faculty of Information Technology, Hanoi University of Mining and Geology.
Phone: +(84).387.067.695 - Email: 2321050038@student.humg.edu.vn

ABSTRACT:

In machine learning, optimization is a critical process for achieving the highest performance of models. Two key tools in this process are derivatives and differentials, which facilitate gradient computation and optimization navigation. This paper focuses on the role of derivatives and differentials in Gradient Descent and Backpropagation—two essential techniques in machine learning optimization.

Keywords: derivative, differential, gradient, optimization, machine learning

REFERENCES

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
2. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
3. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature.
4. Nguyen Truong Thanh and Mai Viet Thuan (eds.) (2019). Calculus 1. Hanoi National University Publishing House.
5. Nguyen Dinh Tri (eds.), Ta Van Dinh, Nguyen Ho Quynh (2019). Advanced Mathematics (volume 2) Single-variable calculus. Education Publishing House.
6. Thomas (2009). Calculus. Pearson.
7. Eric Maththes (2015). Python Crash Course. No Starch Press.
8. A Modern Approach (2019). Artificial Intelligence. Pearrson.

Ngày nhận bài: 14/01/2025;
Ngày gửi phản biện: 21/01/2025;
Ngày nhận phản biện: 21/01/2025;
Ngày chấp nhận đăng: 21/02/2025.