

Đạo hàm ma trận ứng dụng trong Học máy

GVC. TS. Lê Bích Phượng

Tháng 1, 2026

Tóm tắt nội dung

Báo cáo này cung cấp cái nhìn toàn diện về đạo hàm ma trận, một công cụ không thể thiếu trong tối ưu hóa các mô hình học sâu.

Mục lục

1	Mở đầu: Tại sao phải là ma trận?	3
2	Nền tảng cơ bản	3
2.1	Gradient và Jacobian	3
2.2	Đạo hàm theo ma trận	3
3	Phương pháp vi phân và Vết	3
3.1	Tính chất của Vết	3
3.1.1	Tính chất hoán vị (Cyclic Property)	4
3.1.2	Tính chất chuyển vị	4
3.2	Các quy tắc vi phân	4
3.2.1	Quy tắc tích	4
3.2.2	Quy tắc nghịch đảo	5
3.2.3	Quy tắc định thức	5
4	Kết nối giữa vi phân và gradient	5
4.1	Nguyên lý đồng nhất hệ số	6
4.2	Đạo hàm của Log-Determinant	6
4.3	Tính đối xứng	6
5	Kỹ thuật vector hóa và tích Kronecker trong đạo hàm ma trận bậc cao	7
5.1	Toán tử vector hóa	7
5.2	Tích Kronecker	7
5.3	Định lý cơ bản về sự kết hợp	8
5.4	Jacobian của ma trận theo ma trận	8
5.5	Ý nghĩa khoa học	8

6	Ứng dụng trong trí tuệ nhân tạo	8
6.1	Cơ chế lan truyền ngược	8
6.1.1	Cơ sở và nguyên lý: Tín hiệu Sai số	9
6.1.2	Thực hiện: Chứng minh bằng phương pháp vi phân và Vết .	9
6.1.3	Ứng dụng và Thực tế triển khai	10
6.2	Cơ chế Self-Attention	10
6.3	Tối ưu hóa bậc hai (K-FAC)	10
7	Bài tập thực hành	10
8	Kết luận	12

1 Mở đầu: Tại sao phải là ma trận?

Trong giải tích cổ điển, ta làm việc với các hàm số $f : \mathbb{R} \rightarrow \mathbb{R}$. Tuy nhiên, trong Machine Learning, các tham số (weights) thường tồn tại dưới dạng ma trận $W \in \mathbb{R}^{m \times n}$ và hàm mất mát (loss function) là một đại lượng vô hướng $L \in \mathbb{R}$. Việc tính đạo hàm theo từng thành phần $\frac{\partial L}{\partial W_{ij}}$ là quá chậm chạp và dễ sai sót. Đạo hàm ma trận cho phép ta tính toán song song và tối ưu trên GPU một cách hiệu quả.

2 Nền tảng cơ bản

2.1 Gradient và Jacobian

Cho $y = f(\mathbf{x})$ là một hàm vô hướng với $\mathbf{x} \in \mathbb{R}^n$. Gradient của y theo $\mathbf{x}$ là:

$$\nabla_{\mathbf{x}} y = \frac{\partial y}{\partial \mathbf{x}} = \left[\frac{\partial y}{\partial x_1} \quad \cdots \quad \frac{\partial y}{\partial x_n} \right]^T$$

Cho hàm vector $\mathbf{y} = \mathbf{f}(\mathbf{x})$ với $\mathbf{y} \in \mathbb{R}^m, \mathbf{x} \in \mathbb{R}^n$, ma trận Jacobian $J \in \mathbb{R}^{m \times n}$ được định nghĩa:

$$J = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

2.2 Đạo hàm theo ma trận

Nếu f là hàm vô hướng và X là ma trận $m \times n$:

$$\frac{\partial f}{\partial X} \in \mathbb{R}^{m \times n}, \quad \left(\frac{\partial f}{\partial X} \right)_{ij} = \frac{\partial f}{\partial X_{ij}}$$

3 Phương pháp vi phân và Vết

Đây là phương pháp mạnh mẽ nhất để giải quyết các bài toán phức tạp mà không cần xé lẻ ma trận.

3.1 Tính chất của Vết

Vết của một ma trận vuông $A \in \mathbb{R}^{n \times n}$ được định nghĩa là tổng các phần tử trên đường chéo chính:

$$\text{Tr}(A) = \sum_{i=1}^n A_{ii}$$

3.1.1 Tính chất hoán vị (Cyclic Property)

Định lý: $\text{Tr}(AB) = \text{Tr}(BA)$ với $A \in \mathbb{R}^{m \times n}$ và $B \in \mathbb{R}^{n \times m}$.

Chứng minh: Xét phần tử trên đường chéo chính của tích AB :

$$(AB)_{ii} = \sum_{j=1}^n A_{ij} B_{ji}$$

Khi đó:

$$\text{Tr}(AB) = \sum_{i=1}^m (AB)_{ii} = \sum_{i=1}^m \sum_{j=1}^n A_{ij} B_{ji}$$

Tương tự, xét phần tử trên đường chéo chính của tích BA :

$$(BA)_{jj} = \sum_{i=1}^m B_{ji} A_{ij}$$

Khi đó:

$$\text{Tr}(BA) = \sum_{j=1}^n (BA)_{jj} = \sum_{j=1}^n \sum_{i=1}^m B_{ji} A_{ij}$$

Vì tổng hữu hạn có tính chất hoán vị, ta có:

$$\sum_{i=1}^m \sum_{j=1}^n A_{ij} B_{ji} = \sum_{j=1}^n \sum_{i=1}^m B_{ji} A_{ij}$$

Vậy $\text{Tr}(AB) = \text{Tr}(BA)$. $\square$

3.1.2 Tính chất chuyển vị

Định lý: $\text{Tr}(A) = \text{Tr}(A^T)$.

Chứng minh:

$$\text{Tr}(A^T) = \sum_{i=1}^n (A^T)_{ii} = \sum_{i=1}^n A_{ii} = \text{Tr}(A)$$

Điều này hiển nhiên vì các phần tử trên đường chéo chính không thay đổi vị trí khi lấy chuyển vị.

3.2 Các quy tắc vi phân

3.2.1 Quy tắc tích

Định lý: $d(AB) = (dA)B + A(dB)$.

Chứng minh: Sử dụng định nghĩa vi phân:

$$d(AB) = (A + dA)(B + dB) - AB$$

$$d(AB) = AB + (dA)B + A(dB) + (dA)(dB) - AB$$

Vì dA và dB là các đại lượng vô cùng bé, tích $(dA)(dB)$ là đại lượng bậc cao hơn (second-order term) và có thể bỏ qua. Do đó:

$$d(AB) = (dA)B + A(dB)$$

□

3.2.2 Quy tắc nghịch đảo

Định lý: $d(X^{-1}) = -X^{-1}(dX)X^{-1}$.

Chứng minh: Xét đẳng thức: $XX^{-1} = I$. Lấy vi phân hai vế:

$$d(XX^{-1}) = d(I) = 0$$

Áp dụng quy tắc tích vừa chứng minh:

$$(dX)X^{-1} + Xd(X^{-1}) = 0$$

Giải phương trình cho $d(X^{-1})$:

$$Xd(X^{-1}) = -(dX)X^{-1}$$

Nhân hai vế với X^{-1} về phía bên trái:

$$d(X^{-1}) = -X^{-1}(dX)X^{-1}$$

□

3.2.3 Quy tắc định thức

Định lý: $d|X| = |X|\text{Tr}(X^{-1}dX)$.

Chứng minh: Sử dụng công thức khai triển Laplace hoặc tính chất của giá trị riêng. Một cách tiếp cận hiện đại trong AI là sử dụng hàm Logarit: Xét $f(X) = \ln |X|$. Ta biết rằng $\ln |X| = \text{Tr}(\ln X)$. Lấy vi phân:

$$d(\ln |X|) = \text{Tr}(d(\ln X)) = \text{Tr}(X^{-1}dX)$$

Mặt khác, theo quy tắc chuỗi cho hàm số:

$$d(\ln |X|) = \frac{1}{|X|}d|X|$$

Cân bằng hai vế, ta được:

$$\frac{1}{|X|}d|X| = \text{Tr}(X^{-1}dX) \implies d|X| = |X|\text{Tr}(X^{-1}dX)$$

□

4 Kết nối giữa vi phân và gradient

Trong giải tích ma trận, thay vì tính toán rời rạc từng đạo hàm riêng, chúng ta sử dụng một cầu nối trung gian cực kỳ mạnh mẽ: **Vi phân toàn phần**.

4.1 Nguyên lý đồng nhất hệ số

Cho hàm vô hướng $f(X)$ với $X \in \mathbb{R}^{m \times n}$. Vi phân của f tại X theo hướng dX được định nghĩa thông qua tích vô hướng Frobenius:

$$df = \sum_{i=1}^m \sum_{j=1}^n \frac{\partial f}{\partial X_{ij}} dX_{ij} = \langle \nabla_X f, dX \rangle_F \quad (1)$$

Sử dụng tính chất của vết (Trace), tích vô hướng này được viết lại thành:

$$df = \text{Tr} \left(\left(\frac{\partial f}{\partial X} \right)^T dX \right) \quad (2)$$

Để tìm Gradient $G = \frac{\partial f}{\partial X}$, quy trình gồm 2 bước:

1. Biến đổi biểu thức df bằng các quy tắc vi phân sao cho xuất hiện dạng $\text{Tr}(A \cdot dX)$.
2. Đồng nhất hệ số: $G^T = A \implies G = A^T$.

4.2 Đạo hàm của Log-Determinant

Bài toán: Tìm Gradient của $f(X) = \ln |X|$ với X là ma trận vuông không suy biến.

Bước 1: Tính vi phân df Áp dụng quy tắc chuỗi cho hàm số $\ln(u)$:

$$df = d(\ln |X|) = \frac{1}{|X|} d|X|$$

Thay công thức Jacobi $d|X| = |X| \text{Tr}(X^{-1} dX)$ vào:

$$df = \frac{1}{|X|} \cdot |X| \text{Tr}(X^{-1} dX) = \text{Tr}(X^{-1} dX)$$

Bước 2: Đồng nhất với dạng chuẩn Dạng chuẩn yêu cầu: $df = \text{Tr}(G^T dX)$.
Đối chiếu hai biểu thức:

$$\text{Tr}(G^T dX) = \text{Tr}(X^{-1} dX)$$

Từ đó suy ra:

$$G^T = X^{-1} \implies G = (X^{-1})^T$$

Kết luận:

$$\frac{\partial \ln |X|}{\partial X} = (X^{-1})^T$$

4.3 Tính đối xứng

Trong thực tế (như các bài toán hiệp phương sai trong AI), ma trận X thường đối xứng ($X = X^T$). Khi đó:

1. Vì $X = X^T \implies X^{-1} = (X^{-1})^T$, nên kết quả rút gọn thành X^{-1} .

2. Nếu X được ràng buộc đối xứng *trước* khi lấy đạo hàm, một số tài liệu sử dụng định nghĩa đạo hàm trên đa tạp đối xứng sẽ cho kết quả:

$$\frac{\partial \ln |X|}{\partial X} = 2X^{-1} - \text{diag}(X^{-1})$$

Tuy nhiên, trong hầu hết các khung tính toán của Deep Learning (như PyTorch), ta coi X là các biến độc lập và chỉ áp dụng tính đối xứng *sau* khi tính xong Gradient. Do đó, kết quả X^{-1} vẫn là dạng phổ biến nhất.

5 Kỹ thuật vector hóa và tích Kronecker trong đạo hàm ma trận bậc cao

Khi tính đạo hàm của một ma trận theo một ma trận, cấu trúc Tensor bậc 4 gây khó khăn cho việc biểu diễn và tính toán. Giải pháp chuẩn mực trong toán học ứng dụng là "phẳng hóa" các ma trận thành các vector và sử dụng đại số Kronecker để duy trì các mối quan hệ tuyến tính.

5.1 Toán tử vector hóa

Cho ma trận $X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n] \in \mathbb{R}^{m \times n}$, trong đó $\mathbf{x}_i \in \mathbb{R}^m$ là các cột của X . Toán tử $\text{vec}(X)$ chồng các cột của X lên nhau để tạo thành một vector cột duy nhất:

$$\text{vec}(X) = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \in \mathbb{R}^{mn \times 1} \quad (3)$$

5.2 Tích Kronecker

Cho ma trận $A \in \mathbb{R}^{m \times n}$ và $B \in \mathbb{R}^{p \times q}$. Tích Kronecker $A \otimes B$ là một ma trận khối kích thước $mp \times nq$ được định nghĩa bởi:

$$A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \dots & A_{1n}B \\ A_{21}B & A_{22}B & \dots & A_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}B & A_{m2}B & \dots & A_{mn}B \end{bmatrix} \quad (4)$$

Các tính chất then chốt:

1. $(A \otimes B)^T = A^T \otimes B^T$
2. $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ (Quy tắc phân phối tích)
3. $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$ (Nếu A, B khả nghịch)

5.3 Định lý cơ bản về sự kết hợp

Đây là công thức quan trọng nhất để chuyển đổi các phương trình ma trận sang không gian vector:

$$\text{vec}(AXB) = (B^T \otimes A)\text{vec}(X) \quad (5)$$

Chứng minh sơ lược: Xét phần tử thứ (i, j) của AXB và biểu diễn nó dưới dạng tích các vector hàng và cột, sau đó đối chiếu với cấu trúc khối của ma trận Kronecker.

5.4 Jacobian của ma trận theo ma trận

Xét hàm ánh xạ ma trận $F : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{p \times q}$. Jacobian của F theo X được định nghĩa là Jacobian của vector hóa $F(X)$ theo vector hóa X :

$$\frac{\partial F(X)}{\partial X} \triangleq \frac{\partial \text{vec}(F(X))}{\partial \text{vec}(X)} \in \mathbb{R}^{pq \times mn} \quad (6)$$

Trường hợp cụ thể: Biến đổi tuyến tính $Y = AXB$ Để tìm Jacobian $\frac{\partial \text{vec}(Y)}{\partial \text{vec}(X)}$, ta lấy vi phân:

$$dY = A(dX)B \quad (7)$$

Áp dụng toán tử vec và sử dụng công thức (5):

$$\text{vec}(dY) = \text{vec}(A(dX)B) = (B^T \otimes A)\text{vec}(dX) \quad (8)$$

Theo định nghĩa của Jacobian trong không gian vector $dy = Jdx$, ta có:

$$\frac{\partial \text{vec}(AXB)}{\partial \text{vec}(X)} = B^T \otimes A \quad (9)$$

5.5 Ý nghĩa khoa học

1. **Second-order Optimization (K-FAC):** Trong việc xấp xỉ ma trận Fisher Information cho mạng Neural, ta thường gặp các kỳ vọng có dạng $E[\mathbf{a}\mathbf{a}^T \otimes \mathbf{g}\mathbf{g}^T]$. Nhờ tính chất tích Kronecker, ta có thể tách biệt thông tin từ lớp trước ($\mathbf{a}$) và Gradient từ lớp sau ($\mathbf{g}$), giúp giảm độ phức tạp tính toán từ $O(N^2)$ xuống $O(N)$.

2. **Convolutional Neural Networks (CNN):** Phép toán Convolution có thể được biểu diễn dưới dạng nhân ma trận Toeplitz. Việc tính đạo hàm cho các Filter trong không gian vector hóa chính là ứng dụng trực tiếp của tích Kronecker.

6 Ứng dụng trong trí tuệ nhân tạo

6.1 Cơ chế lan truyền ngược

Lan truyền ngược thực chất là việc áp dụng Quy tắc chuỗi (Chain Rule) trong không gian ma trận để tính toán Gradient của hàm mất mát L đối với mọi tham số trong mạng.

6.1.1 Cơ sở và nguyên lý: Tín hiệu Sai số

Xét một lớp ẩn trong mạng Neural với phép biến đổi tuyến tính và kích hoạt phi tuyến:

$$\mathbf{z} = W\mathbf{x} + \mathbf{b} \quad (\text{Linear transformation}) \quad (10)$$

$$\mathbf{y} = \sigma(\mathbf{z}) \quad (\text{Non-linear activation}) \quad (11)$$

Trong đó $W \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$, và σ là hàm kích hoạt thực hiện theo từng phần tử (element-wise).

Nguyên lý: Để cập nhật W , ta cần biết mức độ đóng góp của từng trọng số vào sai số tổng thể L . Tín hiệu này được lan truyền từ đầu ra ngược về phía đầu vào thông qua biến trung gian $\mathbf{y}$ và $\mathbf{z}$.

6.1.2 Thực hiện: Chứng minh bằng phương pháp vi phân và Vết

Giả sử ta đã biết Gradient của L theo đầu ra của lớp là $\mathbf{g} = \nabla_{\mathbf{y}} L$. Ta cần tìm $\frac{\partial L}{\partial W}$.

Bước 1: Vi phân hóa hàm kích hoạt Vì $\mathbf{y} = \sigma(\mathbf{z})$ thực hiện theo từng phần tử, vi phân $d\mathbf{y}$ là:

$$d\mathbf{y} = \sigma'(\mathbf{z}) \odot d\mathbf{z} \quad (12)$$

Trong đó $\odot$ là tích Hadamard (element-wise product).

Bước 2: Vi phân hóa hàm tuyến tính Lấy vi phân $\mathbf{z}$ theo W (giữ $\mathbf{x}$ và $\mathbf{b}$ cố định):

$$d\mathbf{z} = (dW)\mathbf{x} \quad (13)$$

Bước 3: Kết hợp vào biểu thức Trace Vi phân của hàm mất mát L là:

$$dL = \text{Tr}(\mathbf{g}^T d\mathbf{y}) = \text{Tr}(\mathbf{g}^T (\sigma'(\mathbf{z}) \odot d\mathbf{z})) \quad (14)$$

Sử dụng tính chất $\mathbf{a}^T(\mathbf{b} \odot \mathbf{c}) = (\mathbf{a} \odot \mathbf{b})^T \mathbf{c}$, ta có:

$$dL = \text{Tr}((\mathbf{g} \odot \sigma'(\mathbf{z}))^T d\mathbf{z}) \quad (15)$$

Đặt $\boldsymbol{\delta} = \mathbf{g} \odot \sigma'(\mathbf{z})$ là *Tín hiệu sai số tại lớp ẩn* (Error Signal). Thay $d\mathbf{z} = (dW)\mathbf{x}$ vào:

$$dL = \text{Tr}(\boldsymbol{\delta}^T dW \mathbf{x}) \quad (16)$$

Sử dụng tính chất hoán vị của Trace $\text{Tr}(ABC) = \text{Tr}(CAB)$:

$$dL = \text{Tr}(\mathbf{x} \boldsymbol{\delta}^T dW) = \text{Tr}((\boldsymbol{\delta} \mathbf{x}^T)^T dW) \quad (17)$$

Bước 4: Đồng nhất hệ số Đối chiếu với dạng chuẩn $dL = \text{Tr}((\frac{\partial L}{\partial W})^T dW)$, ta thu được:

$$\frac{\partial L}{\partial W} = \boldsymbol{\delta} \mathbf{x}^T = (\nabla_{\mathbf{y}} L \odot \sigma'(W\mathbf{x} + \mathbf{b})) \mathbf{x}^T \quad (18)$$

6.1.3 Ứng dụng và Thực tế triển khai

- **Cập nhật Trọng số:** Trong thực tế, W được cập nhật theo quy tắc Gradient Descent: $W \leftarrow W - \eta \frac{\partial L}{\partial W}$, trong đó η là tốc độ học (learning rate).
- **Tính toán Hiệu quả (Outer Product):** Công thức $\delta \mathbf{x}^T$ chính là một *tích ngoài* (outer product) giữa vector sai số và vector đầu vào. Phép toán này cực kỳ tối ưu trên GPU vì nó có thể tính toán song song cho cả một Batch dữ liệu.
- **Trong các Framework (PyTorch/TensorFlow):** Các thư viện này không tính đạo hàm tượng trưng (symbolic) mà lưu trữ *Computational Graph*. Khi gọi `loss.backward()`, hệ thống thực thi chính xác chuỗi phép nhân ma trận (18) để lan truyền tín hiệu sai số.
- **Mở rộng:** Công thức này là nền tảng cho các mạng MLP phức tạp, và khi thay đổi phép nhân ma trận bằng phép tích chập (Convolution), ta thu được Gradient cho các lớp của mạng CNN.

6.2 Cơ chế Self-Attention

Trong Transformer, đạo hàm của lớp Attention $Y = \text{Softmax}(QK^T)V$ đòi hỏi việc xử lý Jacobian của hàm Softmax:

$$\frac{\partial \text{Softmax}(\mathbf{z})}{\partial \mathbf{z}} = \text{diag}(\mathbf{a}) - \mathbf{a}\mathbf{a}^T, \quad \text{với } \mathbf{a} = \text{Softmax}(\mathbf{z})$$

6.3 Tối ưu hóa bậc hai (K-FAC)

K-FAC sử dụng cấu trúc Kronecker để xấp xỉ ma trận Fisher Information:

$$F \approx A \otimes G \implies F^{-1} \approx A^{-1} \otimes G^{-1}$$

Điều này cho phép thực hiện tối ưu hóa bậc hai trên các mạng Neural khổng lồ.

7 Bài tập thực hành

Bài tập 1: Đạo hàm tuyến tính đơn giản

Đề bài: Cho $f(X) = \text{Tr}(AX)$, tìm $\nabla_X f$.

Giải: Lấy vi phân: $df = \text{Tr}(AdX)$.

Đối chiếu với dạng chuẩn $df = \text{Tr}((\nabla_X f)^T dX)$, ta có $(\nabla_X f)^T = A$.

Vậy $\nabla_X f = A^T$.

Bài tập 2: Dạng toàn phương (Quadratic Form)

Đề bài: Cho $f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x}$, tìm $\nabla_{\mathbf{x}} f$.

Giải: $df = d(\mathbf{x}^T A \mathbf{x}) = (d\mathbf{x})^T A \mathbf{x} + \mathbf{x}^T A d\mathbf{x}$.

Vì df là số vô hướng, $(d\mathbf{x})^T A \mathbf{x} = \text{Tr}((d\mathbf{x})^T A \mathbf{x}) = \text{Tr}(\mathbf{x}^T A^T d\mathbf{x})$.

Vậy $df = \text{Tr}((\mathbf{x}^T A^T + \mathbf{x}^T A) d\mathbf{x}) = \text{Tr}((A + A^T) \mathbf{x})^T d\mathbf{x}$.

Kết luận: $\nabla_{\mathbf{x}} f = (A + A^T) \mathbf{x}$.

Bài tập 3: Chuẩn Frobenius

Đề bài: Cho $f(X) = \|X\|_F^2$, tìm $\nabla_X f$.

Giải: $f(X) = \text{Tr}(X^T X)$.

$df = \text{Tr}(dX^T X + X^T dX) = \text{Tr}(X^T dX + X^T dX) = \text{Tr}(2X^T dX)$.

$\nabla_X f = (2X^T)^T = 2X$.

Bài tập 4: Đạo hàm của hàm hợp (Chain Rule)

Đề bài: Cho $L = f(Y)$ và $Y = XW + B$. Tìm $\nabla_W L$ theo $\nabla_Y L$.

Giải: $dL = \text{Tr}((\nabla_Y L)^T dY)$. Với $dY = X dW$ (giữ X, B cố định).

$dL = \text{Tr}((\nabla_Y L)^T X dW) = \text{Tr}(X^T \nabla_Y L)^T dW$. (Lưu ý: $\text{Tr}(AB) = \text{Tr}(B^T A^T)$).

Vậy $\nabla_W L = X^T \nabla_Y L$.

Bài tập 5: Đạo hàm của Log-Determinant

Đề bài: Tìm $\nabla_X \ln |X|$ với X đối xứng và xác định dương.

Giải: $df = d(\ln |X|) = \frac{1}{|X|} d|X| = \frac{1}{|X|} |X| \text{Tr}(X^{-1} dX) = \text{Tr}(X^{-1} dX)$.

$\nabla_X f = (X^{-1})^T$. Vì X đối xứng nên X^{-1} đối xứng.

Vậy $\nabla_X f = X^{-1}$.

Bài tập 6: Đạo hàm Ma trận nghịch đảo

Đề bài: Cho $f(X) = \text{Tr}(AX^{-1})$, tìm $\nabla_X f$.

Giải: $df = \text{Tr}(A d(X^{-1})) = \text{Tr}(A(-X^{-1} dX X^{-1}))$.

Sử dụng tính hoán vị: $df = \text{Tr}(-X^{-1} A X^{-1} dX)$.

Vậy $\nabla_X f = (-X^{-1} A X^{-1})^T = -X^{-T} A^T X^{-T}$.

Bài tập 7: Hồi quy tuyến tính (Giải hệ phương trình chuẩn)

Đề bài: Tìm $\mathbf{w}$ tối thiểu $L(\mathbf{w}) = \frac{1}{2} \|X\mathbf{w} - \mathbf{y}\|_2^2$.

Giải: $\nabla_{\mathbf{w}} L = X^T (X\mathbf{w} - \mathbf{y})$.

Đặt $\nabla_{\mathbf{w}} L = 0 \implies X^T X \mathbf{w} = X^T \mathbf{y}$.

Vậy $\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$.

Bài tập 8: Jacobian của hàm Softmax

Đề bài: Tìm $\frac{\partial a_i}{\partial z_j}$ với $a_i = \frac{e^{z_i}}{\sum e^{z_k}}$.

Giải: Nếu $i = j$: $\frac{\partial a_i}{\partial z_i} = a_i(1 - a_i)$.

Nếu $i \neq j$: $\frac{\partial a_i}{\partial z_j} = -a_i a_j$.

Dạng ma trận: $J = \text{diag}(\mathbf{a}) - \mathbf{a}\mathbf{a}^T$.

Bài tập 9: Matrix Factorization (Trong Hệ thống gợi ý)

Đề bài: Tìm $\nabla_U \|R - UV^T\|_F^2$.

Giải: Đặt $E = R - UV^T$. $L = \text{Tr}(E^T E)$.

$dL = -2\text{Tr}(E^T dUV^T) = -2\text{Tr}(V^T E^T dU) = \text{Tr}((-2EV)^T dU)$.

Vậy $\nabla_U L = -2(R - UV^T)V$.

Bài tập 10: Tích Hadamard (Hadamard Product)

Đề bài: Cho $f(X) = \text{Tr}(A(B \odot X))$, tìm $\nabla_X f$.

Giải: Sử dụng tính chất $\text{Tr}(A(B \odot C)) = \text{Tr}((A \odot B^T)C)$.

$df = \text{Tr}((A \odot B^T)dX)$.

Vậy $\nabla_X f = (A \odot B^T)^T = A^T \odot B$.

8 Kết luận

Hiểu về đạo hàm ma trận không chỉ giúp chúng ta hiểu cách các thư viện như PyTorch hay TensorFlow hoạt động mà còn mở ra khả năng thiết kế các kiến trúc mạng và thuật toán tối ưu hóa mới.