

# Chương 2

## Đại số tuyến tính

---

- 2.1 Đại lượng vô hướng, vector, ma trận và tensor
- 2.2 Nhân các ma trận và các vector
- 2.3 Ma trận đơn vị và ma trận nghịch đảo
- 2.4 Phụ thuộc tuyến tính và span
- 2.5 Chuẩn
- 2.6 Các ma trận và vector đặc biệt
- 2.7 Eigen-decomposition
- 2.8 Singular Value Decomposition
- 2.9 Ma trận giả nghịch đảo Moore-Penrose
- 2.10 Toán tử vết của ma trận
- 2.11 Định thức
- 2.12 Ví dụ: Principal components analysis - PCA

Đại số tuyến tính (*linear algebra*) là một nhánh của toán học được sử dụng rộng rãi trong các lĩnh vực khoa học kỹ thuật. Tuy nhiên, đại số tuyến tính chủ yếu thực hiện tính toán trên các giá trị liên tục hơn là rời rạc, nên nhiều nhà khoa học máy tính thường ít có kinh nghiệm với nó. Một nền tảng đại số tuyến tính vững chắc là yêu cầu bắt buộc để hiểu và làm việc với các thuật toán ML nói chung và DL nói riêng. Chính vì vậy, ta sẽ điểm qua một số khái niệm quan trọng của đại số tuyến tính trước khi bắt đầu làm việc với DL.

Nếu đã quen thuộc với đại số tuyến tính, bạn hoàn toàn có thể bỏ qua chương này. Nếu đã có chút kinh nghiệm với những khái niệm dưới đây nhưng cần ôn tập lại các công thức quan trọng, chúng tôi khuyên bạn nên đọc cuốn *The Matrix Cookbook* [Petersen and Pederson, 2006]. Nếu đây là lần đầu bạn tiếp cận với đại số tuyến tính, đừng lo lắng, chương này sẽ cung cấp cho bạn lượng kiến thức vừa đủ để đọc và hiểu phần còn lại của cuốn sách, nhưng chúng tôi vẫn khuyến cáo bạn nên tham khảo thêm các giáo trình chuyên cho đại số tuyến tính như [Shilov, 1997]. Chương này sẽ bỏ qua rất nhiều khái niệm trọng tâm của đại số tuyến tính - những phần không cần thiết khi tìm hiểu về DL.

## 2.1 Đại lượng vô hướng, vector, ma trận và tensor

Đại số tuyến tính nghiên cứu nhiều loại đối tượng toán học khác nhau:

**Đại lượng vô hướng (scalar):** Đại lượng vô hướng là một số đơn lẻ, tương phản với hầu hết các đối tượng khác trong đại số tuyến tính, thường là các mảng gồm nhiều số. Các đại lượng vô hướng sẽ được ghi dưới dạng chữ in nghiêng. Chúng tôi sẽ thường xuyên ký hiệu đại lượng vô hướng dưới dạng tên biến không ghi hoa. Khi giới thiệu đến đại lượng vô hướng nào, chúng tôi cũng sẽ chỉ định rõ kiểu của nó. Ví dụ, chúng tôi có thể nói *Giả sử  $s \in \mathbb{R}$  là hệ số góc của đường thẳng* khi định nghĩa một đại lượng vô hướng là số thực, hoặc *Giả sử  $n \in \mathbb{N}$  là số lượng các đơn vị* khi định nghĩa một đại lượng vô hướng là số tự nhiên.

**Vector:** Một vector là một mảng gồm nhiều số được bố trí có thứ tự. Ta có thể nhận diện từng phần tử của vector bằng số thứ tự (*index*) của nó trong mảng. Chúng tôi đặt tên các vector bằng các chữ cái ghi thường, in đậm, ví dụ như  $\mathbf{x}$ . Thành phần của vector sẽ được nhận diện bằng tên vector in nghiêng kèm theo chỉ số dưới. Phần tử đầu tiên của  $\mathbf{x}$  là  $x_1$ , phần tử thứ hai là  $x_2$ , và cứ tiếp tục như vậy. Ta cũng cần phải chỉ định rõ kiểu phần tử của vector. Nếu vector có  $n$  phần tử thuộc  $\mathbb{R}$ , thì vector thuộc tập hợp được tạo thành bởi tích Cartesian của  $\mathbb{R}$   $n$  lần, kí hiệu là  $\mathbb{R}^n$ . Khi ta cần chỉ rõ các phần tử của vector, ta ghi ở dạng cột, đặt trong dấu ngoặc vuông:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad (2.1)$$

Có thể hiểu các vector như là các điểm trong không gian, với mỗi phần tử thể hiện tọa độ tương ứng trên mỗi trục tọa độ khác nhau.

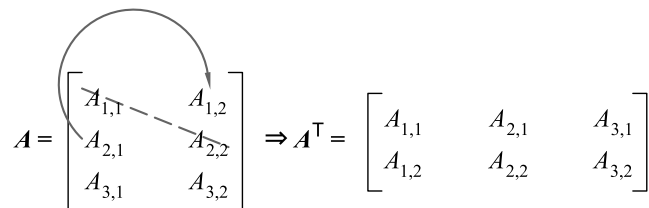
Đôi khi ta cần đánh chỉ số tất cả các phần tử của vector. Trong trường hợp này, ta định nghĩa một tập hợp chứa các chỉ số và viết tập hợp này ở chỉ số dưới. Ví dụ, để truy cập vào các phần tử  $x_1, x_3$  và  $x_6$ , ta định nghĩa tập hợp  $S = \{1, 3, 6\}$  và kí hiệu  $\{x_1, x_3, x_6\}$  là  $\mathbf{x}_S$ . Ta sử dụng dấu - để đánh chỉ số phần bù của một tập hợp. Ví dụ  $\mathbf{x}_{-1}$  là một vector chứa tất cả các phần tử của  $\mathbf{x}$  ngoài trừ phần tử

$x_1$ , và  $x_{-S}$  là một vector chứa tất cả các phần tử của  $x$  ngoại trừ các phần tử  $x_1, x_3$  và  $x_6$ .

**Ma trận:** Ma trận là một mảng 2D của các con số, do vậy mỗi phần tử sẽ được xác định bằng hai chỉ số thay vì một. Ta thường sử dụng tên biến in hoa và in đậm để kí hiệu ma trận, ví dụ như  $\mathbf{A}$ . Nếu một ma trận số thực  $\mathbf{A}$  có chiều cao  $m$  và có chiều dài  $n$ , thì ta nói  $\mathbf{A} \in \mathbb{R}^{m \times n}$ . Ta thường ký hiệu các phần tử của ma trận bằng tên (ma trận) dưới dạng in nghiêng và các chỉ số được tách biệt bằng dấu phẩy. Ví dụ,  $A_{1,1}$  là phần tử tận cùng bên trái của ma trận  $\mathbf{A}$  và  $A_{m,n}$  là phần tử tận cùng bên phải. Ta có thể xác định tất cả các phần tử trên cùng hàng  $i$  sử dụng ký hiệu ":" cho tọa độ cột. Ví dụ,  $\mathbf{A}_{i,:}$  kí hiệu tất cả các phần tử của  $\mathbf{A}$  nằm trên hàng  $i$ . Đây cũng chính là dòng thứ  $i$  của ma trận  $\mathbf{A}$ . Tương tự,  $\mathbf{A}_{:,i}$  là các phần tử trên cột  $i$  của  $\mathbf{A}$ . Khi ta cần chỉ rõ các phần tử, ta viết các phần tử của ma trận trong ngoặc vuông:

$$\begin{bmatrix} A_{1,1}, & A_{1,2} \\ A_{2,1}, & A_{2,2} \end{bmatrix} \quad (2.2)$$

**Tensor:** Trong một vài trường hợp, ta sẽ cần một mảng có số chiều nhiều hơn hai. Trong trường hợp tổng quát, một mảng các số được sắp xếp trên một lưới với một số chiều bất kì, ta gọi là một tensor. Ta kí hiệu tensor có tên "A" sử dụng phong chữ:  $\mathbf{A}$ . Ta kí hiệu phần tử của  $\mathbf{A}$  tại tọa độ  $(i, j, k)$  bằng  $A_{i,j,k}$ .



$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow \mathbf{A}^T = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

Hình 2.1: Ma trận chuyển vị

Một phép toán quan trọng với ma trận là phép chuyển vị (transpose). Chuyển vị của một ma trận là hình ảnh phản chiếu qua gương của ma trận đó thông qua *đường chéo chính*, bắt đầu từ góc tận cùng bên trái chạy xuống dưới và sang phải. Xem minh họa trong hình 2.1. Ta kí hiệu phép toán chuyển vị của ma trận  $\mathbf{A}$  là  $\mathbf{A}^T$ , và được định nghĩa như sau:

$$(\mathbf{A}^T)_{i,j} = A_{j,i} \quad (2.3)$$

Vector có thể hiểu như là ma trận chỉ có duy nhất một cột. Vì vậy, một vector chuyển vị sẽ trở thành một ma trận chỉ có duy nhất một dòng. Đôi khi ta định nghĩa một vector bằng cách biểu diễn các phần tử của nó như là một ma trận

dòng, sau đó sử dụng phép toán chuyển vị để chuyển về dạng chuẩn, dạng vector cột, ví dụ  $\mathbf{x} = [x_1, x_2, x_3]^T$ .

Một đại lượng vô hướng có thể được hiểu như là một ma trận có duy nhất một phần tử. Từ góc nhìn này, ta có thể thấy một đại lượng vô hướng cũng là chuyển vị của chính nó:  $a = a^T$ .

Ta có thể cộng các ma trận lại với nhau miễn là chúng có cùng hình dạng (shape), đơn giản bằng cách cộng các phần tử tương ứng:  $\mathbf{C} = \mathbf{A} + \mathbf{B}$  trong đó  $C_{i,j} = A_{i,j} + B_{i,j}$ .

Ta cũng có thể cộng hoặc nhân một ma trận với một đại lượng vô hướng (scalar), chỉ đơn giản bằng cách thực hiện phép toán đó với mỗi phần tử của ma trận:  $\mathbf{D} = a.\mathbf{B} + c$  trong đó  $D_{i,j} = a.B_{i,j} + c$ .

Trong trường hợp của DL, chúng tôi có sử dụng thêm một vài ký hiệu ít sử dụng theo thông lệ. Chúng tôi cho phép thực hiện cộng giữa một ma trận và một vector, tạo ra một ma trận khác:  $\mathbf{C} = \mathbf{A} + \mathbf{b}$ , trong đó  $C_{i,j} = A_{i,j} + b_j$ . Nói cách khác, vector  $\mathbf{b}$  được cộng vào từng dòng của ma trận  $\mathbf{A}$ . Sử dụng ký hiệu này, ta không cần phải tạo ra một ma trận mới, với mỗi cột là một bản sao của  $\mathbf{b}$ , trước khi thực hiện phép cộng. Kiểu sao chép vector  $\mathbf{b}$  ngầm định tới nhiều vị trí như thế này được gọi là phép *phát tán* (broadcasting).

## 2.2 Nhân các ma trận và các vector

Một trong những phép toán quan trọng nhất của ma trận là phép nhân 2 ma trận. *Tích ma trận* của  $\mathbf{A}$  và  $\mathbf{B}$  là ma trận  $\mathbf{C}$ . Để phép nhân xác định,  $\mathbf{A}$  phải có số cột bằng với số hàng của  $\mathbf{B}$ . Nếu  $\mathbf{A}$  có kích thước  $m \times n$  và  $\mathbf{B}$  có kích thước  $n \times p$  thì  $\mathbf{C}$  sẽ có kích thước  $m \times p$ . Ta có thể viết tích ma trận bằng cách đặt 2 hay nhiều ma trận ở gần nhau, ví dụ:

$$\mathbf{C} = \mathbf{AB} \quad (2.4)$$

Cách tính tích được định nghĩa như sau:

$$C_{i,j} = \sum_k A_{i,k} B_{k,j} \quad (2.5)$$

Lưu ý rằng tích của 2 ma trận không phải chỉ đơn giản là ma trận chứa tích của các phần tử đơn lẻ tương ứng với nhau. Thực tế là có cách tính như vậy, và nó

được gọi là *tích theo từng phần tử* (element-wise product), hay *tích Hadamard*, ký hiệu là  $\mathbf{A} \odot \mathbf{B}$

*Tích vô hướng* (dot product) giữa 2 vector  $\mathbf{x}$  và  $\mathbf{y}$  có cùng số chiều chính là tích ma trận  $\mathbf{x}^T \mathbf{y}$ . Ta có thể coi tích ma trận  $\mathbf{C} = \mathbf{AB}$  như là tính  $C_{i,j}$  bằng cách lấy tích vô hướng giữa hàng  $i$  của  $\mathbf{A}$  với cột  $j$  của  $\mathbf{B}$ .

Phép lấy tích ma trận có nhiều tính chất hữu dụng, giúp cho các phân tích toán học trên ma trận thuận tiện hơn. Ví dụ, tích ma trận có tính phân phối:

$$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC} \quad (2.6)$$

Nó cũng có tính kết hợp:

$$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C} \quad (2.7)$$

Phép tích ma trận *không có tính giao hoán* ( $\mathbf{AB} = \mathbf{BA}$  không phải luôn đúng), không giống như tích giữa các đại lượng vô hướng. Tuy nhiên, tích vô hướng giữa hai vector có tính giao hoán:

$$\mathbf{x}^T \mathbf{y} = \mathbf{y}^T \mathbf{x} \quad (2.8)$$

Chuyển vị của tích ma trận có dạng đơn giản:

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T \quad (2.9)$$

Điều này cho phép ta giải thích phương trình (2.8) là do kết quả của tích vô hướng là một đại lượng vô hướng và vì vậy nó bằng với chuyển vị của chính nó:

$$\mathbf{x}^T \mathbf{y} = (\mathbf{x}^T \mathbf{y})^T = \mathbf{y}^T \mathbf{x} \quad (2.10)$$

Vì trọng tâm của cuốn sách này không phải là đại số tuyến tính, nên chúng tôi không cố liệt kê đầy đủ các tính chất của tích ma trận ở đây, nhưng bạn đọc nên biết rằng tích ma trận còn nhiều tính chất khác nữa.

Tại thời điểm này, ta đã có đầy đủ ký hiệu đại số tuyến tính để viết được hệ các phương trình tuyến tính:

$$\mathbf{Ax} = \mathbf{b} \quad (2.11)$$

với  $\mathbf{A} \in \mathbb{R}^{m \times n}$  là ma trận đã biết trước,  $\mathbf{b} \in \mathbb{R}^m$  là vector đã biết trước, và  $\mathbf{x} \in \mathbb{R}^n$  là vector gồm các biến số chưa biết mà ta muốn đi tìm. Mỗi phần tử  $x_i$  của vector  $\mathbf{x}$  là một trong số các biến chưa biết này. Mỗi dòng của  $\mathbf{A}$  và mỗi phần tử của vector  $\mathbf{b}$  cho ta một ràng buộc. Ta có thể viết lại phương trình (2.11) như sau:

$$\mathbf{A}_{1,:}\mathbf{x} = b_1 \quad (2.12)$$

$$\mathbf{A}_{2,:}\mathbf{x} = b_2 \quad (2.13)$$

...

$$\mathbf{A}_{m,:}\mathbf{x} = b_m \quad (2.15)$$

hoặc có thể viết rõ hơn như sau:

$$A_{1,1}x_1 + A_{1,2}x_2 + \dots + A_{1,n}x_n = b_1 \quad (2.16)$$

$$A_{2,1}x_1 + A_{2,2}x_2 + \dots + A_{2,n}x_n = b_2 \quad (2.17)$$

...

$$A_{m,1}x_1 + A_{m,2}x_2 + \dots + A_{m,n}x_n = b_m \quad (2.18)$$

Ký hiệu phép nhân vector-ma trận cho ta một cách biểu diễn các phương trình ở dạng súc tích hơn nhiều.

## 2.3 Ma trận đơn vị và ma trận nghịch đảo

Đại số tuyến tính cung cấp một công cụ rất mạnh, tên là *ma trận nghịch đảo*, cho phép ta giải phương trình (2.11) với nhiều giá trị của  $\mathbf{A}$

Để mô tả ma trận nghịch đảo, đầu tiên, ta cần định nghĩa khái niệm ma trận đơn vị. *Ma trận đơn vị* (identity matrix) là ma trận không làm thay đổi bất kỳ vector nào khi nhân vector với ma trận đó. Ta ký hiệu ma trận đơn vị  $n$  chiều là  $\mathbf{I}_n$ . Cụ thể,  $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ , và:

$$\forall \mathbf{x} \in \mathbb{R}^n, \mathbf{I}_n \mathbf{x} = \mathbf{x} \quad (2.20)$$

Cấu trúc của ma trận đơn vị rất đơn giản. Tất cả các phần tử trên đường chéo chính có giá trị là 1, còn lại là 0.

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

Hình 2.2: Ví dụ về ma trận đơn vị: Đây là ma trận  $\mathbf{I}_3$

*Ma trận nghịch đảo* (matrix inverse) của  $\mathbf{A}$  được ký hiệu là  $\mathbf{A}^{-1}$ , và được định nghĩa như sau:

$$\mathbf{A}^{-1} \mathbf{A} = \mathbf{I}_n \quad (2.21)$$



**ND:** Ma trận khả nghịch phải là ma trận vuông

Giờ ta có thể giải phương trình (2.11) qua các bước sau:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad (2.22)$$

$$\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \quad (2.23)$$

$$\mathbf{I}_n\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \quad (2.24)$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \quad (2.25)$$

Dĩ nhiên, quá trình biến đổi này phụ thuộc vào khả năng tìm được  $\mathbf{A}^{-1}$ . Ta sẽ bàn về điều kiện tồn tại  $\mathbf{A}^{-1}$  trong phần sau.

Khi biết  $\mathbf{A}^{-1}$  tồn tại, một vài thuật toán khác nhau có thể giúp ta có thể tính ra nó ở dạng biểu thức đóng (closed form). Về mặt lý thuyết, cùng một ma trận nghịch đảo có thể được sử dụng để giải phương trình nhiều lần với các giá trị khác nhau của  $\mathbf{b}$ .  $\mathbf{A}^{-1}$  là một công cụ lý thuyết hữu dụng, tuy vậy, thực tế ta không nên sử dụng trong các ứng dụng phần mềm. Bởi  $\mathbf{A}^{-1}$  chỉ có thể được biểu diễn với độ chính xác thập phân nhất định trên máy tính kỹ thuật số, các thuật toán sử dụng giá trị của  $\mathbf{b}$  để tính thường cho ra kết quả ước tính của  $\mathbf{x}$  tốt hơn.

**ND:** Trên thực tế, các số thập phân bị làm tròn ít nhiều khi biểu diễn trên máy tính. Có những hàm số cực kỳ "nhạy cảm" sẽ cho ra sai số lớn ở kết quả khi tham số đầu vào bị làm tròn nhỏ xíu như vậy. Trong giải tích số (numerical analysis) người ta gọi các hàm này có *hệ số điều hòa* (condition number) lớn. Chương 4 phần 4.2 ta sẽ nói rõ hơn một chút về vấn đề này. Để hiểu rõ hơn bạn đọc có thể tham khảo google với từ khóa "ill-conditioned matrix".

## 2.4 Phụ thuộc tuyến tính và span

Để  $\mathbf{A}^{-1}$  tồn tại, phương trình (2.11) phải có chính xác một nghiệm với mỗi giá trị của  $\mathbf{b}$ . Hệ phương trình cũng có thể không có nghiệm hoặc có vô số nghiệm ứng với một vài giá trị của  $\mathbf{b}$ . Tuy nhiên, nó không thể có nhiều hơn một nhưng ít hơn vô số nghiệm với một giá trị  $\mathbf{b}$  bởi vì nếu cả  $\mathbf{x}$  và  $\mathbf{y}$  đều là nghiệm, thì:

$$\mathbf{z} = \alpha\mathbf{x} + (1 - \alpha)\mathbf{y} \quad (2.26)$$

cũng là một nghiệm ứng với mọi số thực  $\alpha$ .

Để phân tích xem một phương trình có bao nhiêu nghiệm, hãy thử tưởng tượng mỗi cột của  $\mathbf{A}$  như là một hướng chuyển động cụ thể tính từ gốc tọa độ, và từ đó, xác định xem có bao nhiêu cách di chuyển đến  $\mathbf{b}$ . Dưới góc nhìn này, mỗi phần tử của  $\mathbf{x}$  xác định quãng đường mà ta cần di chuyển theo mỗi hướng, cụ thể:  $x_i$  chỉ ra độ dài cần di chuyển theo hướng của cột  $i$ :

$$\mathbf{Ax} = \sum_i x_i \mathbf{A}_{:,i} \quad (2.27)$$

Các phép tính loại này nói chung được gọi là *tổ hợp tuyến tính* (linear combination). Cụ thể, tổ hợp tuyến tính của một tập các vector  $\{\mathbf{v}^{(1)}, \dots, \mathbf{v}^{(n)}\}$  được tạo thành bằng cách nhân mỗi vector  $\mathbf{v}^{(i)}$  với một hệ số thực tương ứng và cộng dồn vào kết quả:

$$\sum_i c_i \mathbf{v}^{(i)} \quad (2.28)$$

*Span* của một tập các vector là tập tất cả các điểm thu được bằng cách lấy tổ hợp tuyến tính của các vector gốc.

Việc xác định xem  $\mathbf{Ax} = \mathbf{b}$  có bao nhiêu nghiệm tương đương với việc kiểm tra xem  $\mathbf{b}$  có nằm trong span của các cột của  $\mathbf{A}$  hay không. Loại span này được gọi là *không gian cột* (column space), hay *dải* (range) của  $\mathbf{A}$ .

Để hệ  $\mathbf{Ax} = \mathbf{b}$  có nghiệm với mọi giá trị của  $\mathbf{b} \in \mathbb{R}^m$ , không gian cột của  $\mathbf{A}$  phải chứa  $\mathbb{R}^m$ . Nếu có bất kỳ điểm nào trong  $\mathbb{R}^m$  nằm ngoài không gian cột, thì điểm đó có khả năng ứng với giá trị của  $\mathbf{b}$  làm cho hệ vô nghiệm. Điều kiện không gian cột của  $\mathbf{A}$  chiếm toàn bộ  $\mathbb{R}^m$  ngay lập tức kéo theo  $\mathbf{A}$  phải có ít nhất  $m$  cột, có nghĩa là  $n \geq m$ . Nếu không thì số chiều của không gian cột sẽ nhỏ hơn  $m$ . Ví dụ, xét ma trận kích cỡ  $3 \times 2$ . Đích đến  $\mathbf{b}$  là 3D, nhưng  $\mathbf{x}$  chỉ là 2D, nghĩa là việc thay đổi giá trị của  $\mathbf{x}$  tối đa chỉ cho phép ta đi lại trên một mặt phẳng 2D trong  $\mathbb{R}^3$ . Phương trình có nghiệm khi và chỉ khi  $\mathbf{b}$  nằm trên mặt phẳng đó.

Điều kiện  $n \geq m$  chỉ là một điều kiện cần để hệ có nghiệm với mỗi điểm  $\mathbf{b}$ . Nó không phải là điều kiện đủ, bởi có khả năng một vài cột bị dư thừa. Xét ma trận  $2 \times 2$  với 2 cột giống nhau. Ma trận này có cùng không gian cột với ma trận kích thước  $2 \times 1$  chỉ gồm một bản sao của hai cột giống nhau. Nói cách khác, không gian cột của ma trận này chỉ là một đường thẳng và không thể bao quát toàn bộ  $\mathbb{R}^2$ , kể cả khi có 2 cột.

Về mặt hình thức, kiểu dư thừa loại này được gọi là *phụ thuộc tuyến tính* (linear dependence). Một tập các vector là *độc lập tuyến tính* (linearly independent) nếu



không có vector nào trong tập là tổ hợp tuyến tính của các vector còn lại. Nếu ta cộng một vector mới với một tập vector cho trước mà kết quả lại bằng tổ hợp tuyến tính của các vector trong tập cho trước, thì vector mới đó không bổ sung thêm bất cứ điểm nào trong span của tập vector cho trước. Điều này có nghĩa là để không gian cột của ma trận bao quát toàn bộ  $\mathbb{R}^m$ , thì ma trận phải chứa tối thiểu một tập gồm  $m$  cột độc lập tuyến tính. Điều kiện này vừa là điều kiện cần và cũng là điều kiện đủ cho phương trình (2.11) có nghiệm với mọi giá trị của  $\mathbf{b}$ . Lưu ý là điều kiện này là để một tập có chính xác  $m$  cột độc lập tuyến tính, chứ không phải có tối thiểu  $m$  cột độc lập tuyến tính. Không có tập vector  $m$  chiều nào có thể có nhiều hơn  $m$  cột độc lập tuyến tính, nhưng một ma trận có nhiều hơn  $m$  cột có thể có nhiều hơn một tập (các cột độc lập tuyến tính) như thế.

Để ma trận khả nghịch, ta cần đảm bảo thêm rằng phương trình (2.11) có nhiều nhất một nghiệm với mỗi giá trị của  $\mathbf{b}$ . Có nghĩa là ta phải đảm bảo ma trận  $\mathbf{A}$  có tối đa  $m$  cột. Nếu không, ta sẽ có nhiều hơn một cách tham số hóa cho mỗi nghiệm.

Tất cả các điều kiện này cộng lại buộc ma trận phải vuông, nghĩa là, ta buộc  $m = n$  và tất cả cột phải độc lập tuyến tính. Một ma trận vuông với các cột phụ thuộc tuyến tính được gọi là một *ma trận suy biến* (singular matrix).

Nếu  $\mathbf{A}$  không vuông nhưng suy biến (singular), ta vẫn có thể giải phương trình, nhưng ta không thể sử dụng phương pháp ma trận nghịch đảo để tìm nghiệm được.

Đến đây, ta đã bàn về ma trận nghịch đảo được nhân ở bên trái ( $\mathbf{A}^{-1}$  nằm bên trái  $\mathbf{A}$  trong (2.21)). Có thể định nghĩa ma trận nghịch đảo với vị trí bên phải trong phép nhân:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I} \quad (2.29)$$

Đối với các ma trận vuông, nghịch đảo trái bằng nghịch đảo phải.

## 2.5 Chuẩn

Đôi khi ta cần phải đo kích thước của một vector. Trong ML, ta thường đo kích thước của một vector thông qua một hàm gọi là *chuẩn* (norm). Một chuẩn bậc  $p$ , kí hiệu là  $L^p$ , có công thức tính như sau:

$$||\mathbf{x}||_p = \left( \sum_i |x_i|^p \right)^{\frac{1}{p}} \quad (2.30)$$

Với  $p \in \mathbb{R}, p \geq 1$ .

Các chuẩn, bao gồm cả chuẩn  $L^p$ , là các hàm ánh xạ từ tập các vector tới các giá trị không âm. Hay nói một cách trực quan thì chuẩn của vector  $\mathbf{x}$  sẽ đo lường khoảng cách từ điểm gốc tọa độ tới điểm  $\mathbf{x}$ . Nói một cách chi tiết hơn, một chuẩn có thể là bất kì hàm số  $f$  nào thỏa mãn hệ điều kiện:

- $f(\mathbf{x}) = 0 \Rightarrow \mathbf{x} = \mathbf{0}$
- $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$  (còn gọi là *bất đẳng thức tam giác*)
- $\forall \alpha \in \mathbb{R}, f(\alpha \mathbf{x}) = |\alpha| f(\mathbf{x})$

Chuẩn bậc hai  $L^2$  được biết đến như là khoảng cách *Euclidean* trong không gian đa chiều từ gốc tọa độ đến điểm  $\mathbf{x}$ . Vì chuẩn  $L^2$  được sử dụng rất thường xuyên trong ML, ta chỉ đơn giản kí hiệu là  $||\mathbf{x}||$ , trong đó chỉ số 2 bị lược bỏ. Ta thường đo kích thước của một vector bằng độ lớn của bình phương  $L^2$ , mà đại lượng này chính là  $\mathbf{x}^\top \mathbf{x}$ .

Thông thường thì bình phương của chuẩn  $L^2$  là đại lượng thuận tiện trong việc khai triển và tính toán hơn là chính chuẩn này. Chẳng hạn như đạo hàm của bình phương  $L^2$  với mỗi một phần tử của  $\mathbf{x}$  chỉ phụ thuộc vào phần tử tương ứng đó của  $\mathbf{x}$ , trong khi đạo hàm của chuẩn  $L^2$  phụ thuộc vào toàn bộ vector  $\mathbf{x}$ .

Trong nhiều tình huống, ta không dùng  $L^2$  vì nó tăng rất chậm khi  $\mathbf{x}$  ở quanh gốc tọa độ. Trong một vài ứng dụng của ML, ta cần phân biệt rõ ràng các phần tử đúng bằng 0 và các phần tử nhỏ nhưng khác 0. Trong những trường hợp như vậy, ta sử dụng một hàm có tính chất tăng với cùng một tốc độ theo mọi hướng, nhưng vẫn đơn giản về mặt toán học: chuẩn  $L^1$ . Chuẩn này có thể viết gọn như sau:

$$||\mathbf{x}||_1 = \sum_i |x_i| \quad (2.31)$$

Chuẩn  $L^1$  thường được sử dụng trong ML khi sự khác biệt giữa các phần tử khác 0 và bằng 0 là quan trọng. Mỗi lần khi  $\mathbf{x}$  di chuyển một khoảng  $\epsilon$  từ 0, chuẩn  $L^1$  tăng một lượng  $\epsilon$ .

Ngoài ra, đôi khi ta cũng có thể đo lường độ lớn của một vector bằng cách đếm số lượng các phần tử khác 0 của vector. Một số tác giả gọi hàm này chuẩn  $L^0$ ,

tuy nhiên cách gọi này không chính xác. Số các phần tử khác 0 không được gọi là chuẩn vì khi ta nhân một vector với  $\alpha$ , số lượng phần tử khác 0 không thay đổi.

**ND:** Một số nguồn gọi  $L^0$  là *giả chuẩn* (pseudo-norm).

Một chuẩn khác cũng thường xuất hiện trong ML đó là chuẩn  $L^\infty$ , hay còn được gọi là *chuẩn cực đại* (max norm). Chuẩn này có thể viết đơn giản là trị tuyệt đối của phần tử có độ lớn lớn nhất trong một vector.

$$\|\mathbf{x}\|_\infty = \max_i |x_i| \quad (2.32)$$

Đôi khi ta cũng muốn đo lường kích cỡ của một ma trận. Trong DL, cách thông dụng nhất là sử dụng *chuẩn Frobenius* (Frobenius norm):

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} A_{i,j}^2} \quad (2.33)$$

Chuẩn Frobenius tương tự như chuẩn  $L^2$  cho vector.

Tích có hướng của hai vector có thể biểu diễn thông qua tích của các norm như sau:

$$\mathbf{x}^\top \mathbf{y} = \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \cos \theta \quad (2.34)$$

ở đây  $\theta$  là góc giữa  $\mathbf{x}$  và  $\mathbf{y}$ .

## 2.6 Các ma trận và vector đặc biệt

Trong chương này, chúng ta cùng làm quen với một số dạng ma trận và vector cực kì hữu ích.

*Ma trận đường chéo* (diagonal matrix) là ma trận bao gồm hầu hết các phần tử là 0 và các phần tử khác 0 chỉ nằm trên đường chéo chính. Cụ thể,  $\mathbf{D}$  là ma trận đường chéo khi và chỉ khi  $D_{i,j} = 0$  với  $\forall i \neq j$ . Chúng ta đã làm quen với một ví dụ của ma trận đường chéo: ma trận đơn vị, khi mà tất cả các phần tử trên đường chéo bằng 1. Ta kí hiệu  $\text{diag}(\mathbf{v})$  là ma trận vuông có vector đường chéo chính là  $\mathbf{v}$ . Ma trận đường chéo là một ma trận được quan tâm đặc biệt bởi phép nhân với ma trận chéo cực kì hiệu quả. Để tính  $\text{diag}(\mathbf{v})\mathbf{x}$ , ta chỉ cần nhân mỗi phần tử  $x_i$  với  $v_i$ . Hay nói một cách khác  $\text{diag}(\mathbf{v})\mathbf{x} = \mathbf{v} \odot \mathbf{x}$ . Lấy ma trận nghịch đảo của ma trận đường chéo cũng rất dễ dàng. Ma trận đường chéo khả nghịch chỉ khi tất cả các phần tử trên đường chéo khác không và trong trường hợp này:

$$\text{diag}(\mathbf{v})^{-1} = \text{diag}\left(\frac{1}{v_1}, \dots, \frac{1}{v_n}\right)^T$$

Trong nhiều trường hợp, ta thiết kế các thuật toán ML với ma trận bất kì, nhưng ta có thể làm cho thuật toán đó hiệu quả hơn (có điều khả năng mô tả lại kém hơn) bằng cách giới hạn trong các ma trận đường chéo.

Một lưu ý đó là không phải tất cả các ma trận đường chéo đều bắt buộc phải vuông. Ta có thể xây dựng các ma trận đường chéo hình chữ nhật. Ma trận không vuông thì không có nghịch đảo, nhưng ta vẫn có thể thực hiện các phép nhân tốn rất ít chi phí tính toán. Đối với một ma trận đường chéo  $\mathbf{D}$  không vuông, tích  $\mathbf{D}\mathbf{x}$  sẽ là một vector thu được bằng cách nhân mỗi phần tử trên đường chéo với phần tử tương ứng trên vector  $\mathbf{x}$  và sau đó hoặc ta nối thêm những số 0 vào kết quả, nếu  $\mathbf{D}$  là ma trận gầy hoặc ta sẽ bỏ đi một số phần tử ở cuối cùng của vector kết quả, nếu  $\mathbf{D}$  là ma trận béo.

*Ma trận đối xứng* (symmetric matrix) là ma trận mà chuyển vị của nó bằng chính nó:

$$\mathbf{A} = \mathbf{A}^T \quad (2.35)$$

Ma trận đối xứng thường xuất hiện khi các đầu vào được tạo thành bằng những hàm có hai đối số mà giá trị của nó không phụ thuộc vào thứ tự các đối số. Chẳng hạn như nếu  $\mathbf{A}$  là ma trận khoảng cách với  $A_{i,j}$  là kí hiệu khoảng cách từ điểm  $i$  đến điểm  $j$  thì  $A_{i,j} = A_{j,i}$ .

Một *vector đơn vị* (unit vector) là vector có chuẩn bằng 1 (hay còn gọi là *chuẩn đơn vị*):

$$\|\mathbf{x}\|_2 = 1 \quad (2.36)$$

Hai vector  $\mathbf{x}$  và  $\mathbf{y}$  là *trực giao* (orthogonal) nếu  $\mathbf{x}^T \mathbf{y} = 0$ . Nếu cả hai vector đều có chuẩn khác 0, thì trực giao có nghĩa là góc giữa 2 vector này bằng 90 độ. Trong không gian  $n$  chiều  $\mathbb{R}^n$ , có tối đa  $n$  vector có chuẩn khác 0 đôi một trực giao với nhau. Nếu các vector này không chỉ trực giao mà còn có chuẩn đơn vị thì ta gọi là hệ cơ sở *trực chuẩn*.

Một *ma trận trực giao* là ma trận vuông thỏa mãn các dòng của nó đôi một trực chuẩn và các cột của nó cũng đôi một trực chuẩn:

$$\mathbf{A}^T \mathbf{A} = \mathbf{A} \mathbf{A}^T = \mathbf{I} \quad (2.37)$$

Từ đó ta suy ra

$$\mathbf{A}^{-1} = \mathbf{A}^T \quad (2.38)$$

Do đó, ma trận trực giao được quan tâm nhiều vì chi phí tính toán ma trận nghịch đảo là cực kì thấp. Chú ý rằng trong định nghĩa ma trận trực giao, các dòng của nó không đơn thuần là hệ trực giao mà nó còn là một hệ trực chuẩn. Không tồn tại thuật ngữ đặc biệt mô tả một ma trận có các dòng và các cột của nó là hệ trực giao nhưng không phải là một hệ trực chuẩn.

## 2.7 Eigen-decomposition

Rất nhiều đại lượng toán học có thể được hiểu tốt hơn bằng cách đem chia nhỏ thành các phần cấu tạo nên chúng, hoặc tìm các đặc tính chung nhất, không phụ thuộc vào cách ta biểu diễn chúng.

Chẳng hạn như, số nguyên có thể *phân tích* (decomposed) thành các số nguyên tố. Cách ta biểu diễn số 12 sẽ thay đổi dựa vào việc ta biểu diễn trên hệ thập phân hay nhị phân nhưng sẽ không làm thay đổi bản chất giá trị của số 12 vì dù biểu diễn theo cách nào thì ta luôn có  $12 = 2 \times 2 \times 3$ . Từ cách biểu diễn này ta có thể kết luận được những tính chất hữu ích chẳng hạn như 12 không chia hết cho 5 và bất kì một số nguyên nào nhân với 12 cũng sẽ chia hết cho 3.

Rất nhiều tính chất mà ta có thể khám phá về một số nguyên bằng việc phân tích chúng thành các số nguyên tố. Xuất phát từ ý tưởng với số nguyên, đối với ma trận, ta cũng có thể phân tích chúng để tìm ra những đặc tính mà ta không thể dễ dàng phát hiện ra nếu để ma trận ở dạng mảng các phần tử.

Một trong những phương pháp được biết đến rộng rãi để *phân tích ma trận* đó là eigen-decomposition (phân tích riêng), trong đó ta phân tích một ma trận thành một tập hợp các eigen-vector (vector riêng) và các eigen-value (trị riêng).

Một vector riêng của một ma trận vuông  $\mathbf{A}$  là một vector  $\mathbf{v}$  khác 0 thỏa mãn phép nhân ma trận đó với  $\mathbf{v}$  sẽ tỉ lệ với  $\mathbf{v}$ :

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \quad (2.39)$$

Hệ số tỉ lệ  $\lambda$  được gọi là trị riêng tương ứng với vector riêng (ở đây là  $\mathbf{v}$ ). (Ta cũng có thể định nghĩa *vector riêng trái* (left eigen-vector) thỏa mãn  $\mathbf{v}^T \mathbf{A} = \lambda \mathbf{v}^T$  nhưng thông thường ta chỉ quan tâm đến *vector riêng phải*.)

Nếu  $\mathbf{v}$  là một vector riêng của  $\mathbf{A}$  thì bất kì một vector nào tỉ lệ với  $\mathbf{v}$ , có dạng  $s\mathbf{v}$  với  $s \in \mathbb{R}, s \neq 0$ , cũng là một vector riêng với cùng một trị riêng. Vì lý do này, ta thường chỉ quan tâm tới vector riêng có chuẩn đơn vị.

Giả sử  $A$  có  $n$  vector riêng độc lập tuyến tính  $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(n)}$  với các trị riêng tương ứng  $\lambda_1, \lambda_2, \dots, \lambda_n$ . Ta có thể ghép tất cả các vector riêng thành một ma trận  $V$  sao cho mỗi cột là một vector riêng:  $V = [\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(n)}]$ . Tương tự như vậy các giá trị riêng được tập hợp thành một vector riêng  $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \dots, \lambda_n]^T$ . Khi đó eigen-decomposition của  $A$  được biểu diễn bởi công thức:

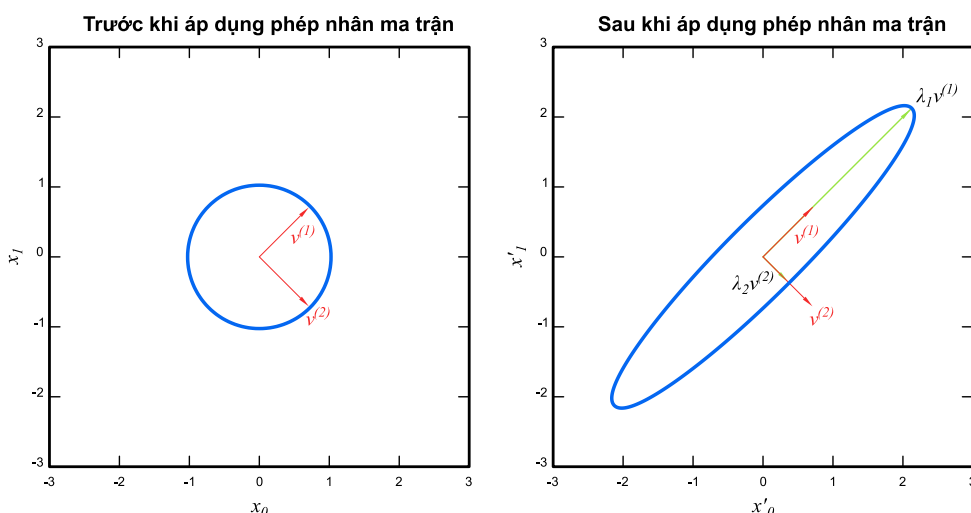
$$A = V \text{diag}(\boldsymbol{\lambda}) V^{-1} \quad (2.40)$$

Việc xây dựng một ma trận với trị riêng và vector riêng cụ thể cho phép ta co giãn không gian dữ liệu theo các hướng mong muốn. Tuy nhiên, ta thường phân tích ma trận thành những trị riêng và vector riêng để có thể phân tích một thuộc tính nào đó của ma trận sâu sắc hơn, cũng tương tự như việc phân tích một số nguyên thành các số nguyên tố có thể giúp ta hiểu hành vi của các số nguyên này hơn.

Không phải mọi ma trận đều có thể được phân tích thành các vector riêng và trị riêng. Trong một vài trường hợp phép phân tích tồn tại nhưng liên quan đến số phức nhiều hơn là các số thực. Rất may là trong cuốn sách này, chúng ta chỉ thường xuyên phân tích một lớp các ma trận mà phân tích có dạng đơn giản. Đặc biệt, mọi ma trận thực đối xứng đều có thể phân tích thành tích các ma trận thực chỉ sử dụng vector riêng và trị riêng:

$$A = Q \Lambda Q^T \quad (2.41)$$

Ở đây  $Q$  là một ma trận trực giao được tổng hợp từ các vector riêng của ma trận  $A$  và  $\Lambda$  (lambda) là ma trận đường chéo. Trị riêng  $\Lambda_{i,i}$  tương ứng với vector riêng trong cột  $i$  của  $Q$  và kí hiệu là  $Q_{:,i}$ . Bởi vì  $Q$  là một ma trận trực giao nên ta có thể xem  $A$  như một không gian tạo thành bởi các vector riêng  $\mathbf{v}^{(i)}$  bị co hoặc giãn với tỉ lệ  $\lambda_i$ . Xem ví dụ trong hình 2.3.





Hình 2.3: Một ví dụ về ảnh hưởng của vector riêng và trị riêng. Ở đây, ta có ma trận  $A$  với hai vector trực chuẩn:  $v^{(1)}$  tương ứng với trị riêng  $\lambda_1$  và  $v^{(2)}$  tương ứng với trị riêng  $\lambda_2$ . Hình bên trái ta biểu diễn tập hợp tất cả các vector đơn vị  $u \in \mathbb{R}^2$  bằng vòng tròn đơn vị. Hình bên phải, ta biểu diễn tập hợp các điểm  $Au$ . Quan sát cách  $A$  làm méo vòng tròn, ta thấy phép nhân ma trận đã có tác dụng co giãn không gian này theo phương  $v^{(i)}$  với tỉ lệ  $\lambda_i$ .

Tuy mọi ma trận số thực đối xứng  $A$  đều đảm bảo có eigen-decomposition, nhưng giá trị này có thể không duy nhất. Nếu bất kì hai hoặc nhiều vector riêng có cùng một trị riêng thì bất kì tập vector trực giao nào nằm trong không gian sinh bởi các vector riêng đó cũng là tập các vector riêng với cùng một trị riêng, và ta có thể chọn một ma trận  $Q$  mới từ các vector riêng mới đó. Theo thông lệ, ta thường sắp xếp các phần tử của  $\Lambda$  theo thứ tự giảm dần. Cũng theo thông lệ này, phép phân tích riêng là duy nhất khi và chỉ khi các trị riêng là đôi một khác nhau.

Eigen-decomposition của một ma trận sẽ cho ta biết rất nhiều điều về ma trận. Ma trận là suy biến (singular) khi và chỉ khi tồn tại một trị riêng bất kì bằng 0. Eigen-decomposition của một ma trận thực đối xứng có thể cũng được sử dụng để tối ưu hóa biểu thức bậc hai có dạng  $f(x) = x^T Ax$  với ràng buộc  $\|x\|_2 = 1$ . Bất cứ khi nào  $x$  bằng với vector riêng của  $A$ ,  $f$  sẽ bằng với trị riêng tương ứng. Giá trị lớn nhất của  $f$  trong miền ràng buộc bằng với trị riêng lớn nhất và giá trị nhỏ nhất của  $f$  trong miền ràng buộc bằng với trị riêng nhỏ nhất.

Một ma trận có toàn bộ các trị riêng của nó là các số thực dương thì được gọi là *ma trận xác định dương* và một ma trận có các trị riêng của nó là số thực lớn hơn hoặc bằng 0 thì được gọi là *ma trận nửa xác định dương*. Tương tự như vậy cho các giá trị âm. Ma trận nửa xác định dương là các ma trận thú vị bởi  $\forall x, x^T Ax \geq 0$ . Ma trận xác định dương ngoài tính chất trên còn có thêm một tính chất nữa đó là  $x^T Ax = 0 \Rightarrow x = 0$ .

## 2.8 Singular Value Decomposition (SVD)

Trong phần 2.7, chúng ta đã tìm hiểu làm thế nào để phân tích một ma trận thành vector riêng và trị riêng. Phương pháp *phân tích giá trị suy biến* (singular

value decomposition - SVD) cung cấp một cách khác để phân tích một ma trận thành các *giá trị suy biến* và các *vector suy biến*. SVD còn cho phép ta tìm ra một số thông tin tương tự như phương pháp phân tích riêng; tuy nhiên, SVD có thể ứng dụng cho nhiều trường hợp hơn. Bất kì ma trận thực nào cũng có SVD, nhưng không chắc có eigen-decomposition. Ví dụ, nếu một ma trận không vuông, theo định nghĩa nó không có eigen-decomposition, và vì thế, ta phải sử dụng SVD.

Nhắc lại, eigen-decomposition liên quan đến phân tích ma trận  $\mathbf{A}$  để tìm ra ma trận các vector riêng  $\mathbf{V}$  và vector các trị riêng  $\lambda$ , do đó  $\mathbf{A}$  có thể viết lại thành:

$$\mathbf{A} = \mathbf{V} \text{diag}(\lambda) \mathbf{V}^{-1} \quad (2.42)$$

SVD cũng tương tự, ngoại trừ ta có thể viết ma trận  $\mathbf{A}$  thành 3 ma trận khác:

$$\mathbf{A} = \mathbf{U} \mathbf{D} \mathbf{V}^T \quad (2.43)$$

Trong đó,  $\mathbf{A}$  là ma trận kích thước  $m \times n$ .  $\mathbf{U}$  là ma trận kích thước  $m \times m$ ,  $\mathbf{D}$  là ma trận kích thước  $m \times n$  và  $\mathbf{V}$  là ma trận  $n \times n$ .

Mỗi ma trận được định nghĩa đều có một cấu trúc đặc biệt. Ma trận  $\mathbf{U}$  và  $\mathbf{V}$  là ma trận trực giao. Ma trận  $\mathbf{D}$  là ma trận đường chéo. Lưu ý, ở đây ma trận  $\mathbf{D}$  không nhất thiết phải là ma trận vuông.

Các phần tử thuộc đường chéo của ma trận  $\mathbf{D}$  được gọi là các *giá trị suy biến* của ma trận  $\mathbf{A}$ . Các cột của  $\mathbf{U}$  gọi là các vector suy biến trái. Các cột của  $\mathbf{V}$  gọi là các vector suy biến phải.

Chúng ta có thể giải thích SVD của  $\mathbf{A}$  từ eigen-decomposition của các hàm số trong  $\mathbf{A}$ . Vector suy biến trái của ma trận  $\mathbf{A}$  là vector riêng của  $\mathbf{A} \mathbf{A}^T$ . Vector suy biến phải của ma trận  $\mathbf{A}$  là vector riêng của  $\mathbf{A}^T \mathbf{A}$ . Giá trị suy biến khác không của ma trận  $\mathbf{A}$  là căn bậc hai của trị riêng của ma trận  $\mathbf{A}^T \mathbf{A}$ . Tính chất đó cũng đúng với ma trận  $\mathbf{A} \mathbf{A}^T$ .

Có lẽ tính năng hữu ích nhất của SVD là ta có thể sử dụng nó để phân nào tổng quát hóa phép lấy ma trận nghịch đảo áp dụng cho các ma trận không vuông, và ta sẽ thấy điều này trong phần tiếp theo.

## 2.9 Ma trận giả nghịch đảo Moore-Penrose

---

Ma trận nghịch đảo không được định nghĩa cho các ma trận không vuông. Giả sử ta muốn tính ma trận nghịch đảo trái  $B$  của ma trận  $A$  để ta có thể giải phương trình tuyến tính.

$$Ax = y \quad (2.44)$$

Nhân bên trái của cả hai vế với  $B$  ta được

$$x = By \quad (2.45)$$

Tùy vào đặc thù của từng bài toán, ta có thể hoặc không thể xây dựng một ánh xạ duy nhất từ  $A$  sang  $B$ .

Nếu ma trận  $A$  có số dòng nhiều hơn số cột, thì phương trình có thể vô nghiệm. Nếu  $A$  có số cột nhiều hơn số dòng, phương trình có thể có rất nhiều nghiệm.

Ma trận *giả nghịch đảo Moore-Penrose* (Moore-Penrose pseudoinverse) cho phép ta phần nào giải quyết những trường hợp này. Ma trận giả nghịch đảo của  $A$  được định nghĩa như sau:

$$A^+ = \lim_{\alpha \searrow 0} (A^T A + \alpha I)^{-1} A^T. \quad (2.46)$$

Tuy nhiên, trong các thuật toán thực tế, để tính ma trận giả nghịch đảo, người ta không dựa trên định nghĩa này, mà dựa theo công thức.

$$A^+ = VD^+U^T. \quad (2.47)$$

Trong đó  $U$ ,  $D$  và  $V$  là các thành phần trong SVD của ma trận  $A$  và ma trận giả nghịch đảo  $D^+$  của ma trận đường chéo  $D$  được tính bằng cách nghịch đảo các phần tử khác không, sau đó lấy chuyển vị của ma trận thu được.

Khi ma trận  $A$  có số cột nhiều hơn số dòng, việc giải phương trình tuyến tính sử dụng ma trận giả nghịch đảo sẽ cho ta một trong số rất nhiều nghiệm. Cụ thể, ta có nghiệm  $x = A^+ y$  có chuẩn Euclidean  $\|x\|_2$  nhỏ nhất trong số các nghiệm.

Khi ma trận  $A$  có số dòng nhiều hơn số cột, thì phương trình có thể không có nghiệm. Trong trường hợp này, sử dụng ma trận giả nghịch đảo sẽ cho ta vector  $x$  mà tích  $Ax$  gần với  $y$  nhất tính theo khoảng cách Euclidean  $\|Ax - y\|_2$ .

## 2.10 Toán tử vết của ma trận

*Toán tử vết* (trace operator) là phép toán trả về tổng tất cả phần tử nằm trên đường chéo chính của một ma trận.

$$\text{Tr}(\mathbf{A}) = \sum_i A_{i,i}. \quad (2.48)$$

Toán tử vết rất hữu dụng vì một số lý do. Một số phép toán trở nên khó mô tả nếu không sử dụng kí hiệu tổng, nhưng có thể được mô tả bằng các phép nhân ma trận và toán tử vết. Ví dụ, toán tử vết cho ta cách khác để viết của chuẩn Frobenius của ma trận:

$$\|\mathbf{A}\|_F = \sqrt{\text{Tr}(\mathbf{A}\mathbf{A}^\top)} \quad (2.49)$$

Biểu diễn một biểu thức dưới dạng toán tử vết cho phép ta sử dụng các hằng đẳng thức của vết. Ví dụ, toán tử vết không thay đổi khi ta lấy chuyển vị:

$$\text{Tr}(\mathbf{A}) = \text{Tr}(\mathbf{A}^\top) \quad (2.50)$$

Vết của tích của nhiều ma trận cũng không thay đổi khi di chuyển nhân tử cuối cùng lên vị trí đầu tiên, và dĩ nhiên là trong điều kiện kích thước của các ma trận tương ứng cho phép thực hiện được phép nhân theo thứ tự mới:

$$\text{Tr}(\mathbf{ABC}) = \text{Tr}(\mathbf{CAB}) = \text{Tr}(\mathbf{BCA}) \quad (2.51)$$

hoặc tổng quát hơn, ta có:

$$\text{Tr}\left(\prod_{i=1}^n \mathbf{F}^{(i)}\right) = \text{Tr}\left(\mathbf{F}^{(n)} \prod_{i=1}^{n-1} \mathbf{F}^{(i)}\right). \quad (2.52)$$

Toán tử vết áp dụng cho hoán vị vòng quanh sẽ không thay đổi ngay cả nếu các ma trận tích thu được có kích thước khác nhau. Ví dụ, cho  $\mathbf{A} \in \mathbb{R}^{m \times n}$  và  $\mathbf{B} \in \mathbb{R}^{n \times m}$ , ta có

$$\text{Tr}(\mathbf{AB}) = \text{Tr}(\mathbf{BA}) \quad (2.53)$$

mặc dù  $\mathbf{AB} \in \mathbb{R}^{m \times m}$  và  $\mathbf{BA} \in \mathbb{R}^{n \times n}$ .

Một tích chất hữu dụng khác mà bạn nên nhớ là một scalar có giá trị bằng vết của chính nó:  $a = \text{Tr}(a)$ .

## 2.11 Định thức

Định thức của một ma trận vuông, ký hiệu là  $\det(\mathbf{A})$ , là một hàm ánh xạ ma trận với một giá trị vô hướng thực. Định thức bằng tích của tất cả các giá trị riêng của ma trận. Giá trị tuyệt đối của định thức có thể được coi là thước đo số lượng phép nhân của ma trận trong không gian. Nếu định thức của một ma trận bằng

0, ma trận này gọi là ma trận suy biến. Nếu định thức bằng 1, ma trận được gọi là ma trận đơn modula.

## 2.12 Ví dụ: Principal components analysis - PCA

*Phân tích thành phần chính* (Principal components analysis - PCA) là một thuật toán đơn giản của ML, có thể được suy ra từ những kiến thức đại số rất cơ bản.

Giả sử ta có một tập hợp  $m$  điểm  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  trong  $\mathbb{R}^n$  và ta muốn áp dụng thuật toán *nén có tổn hao* (lossy compression) cho tập hợp này. Nén có tổn hao có nghĩa là lưu trữ những điểm này theo cách tốn ít bộ nhớ hơn, nhưng tính chính xác có thể bị mất đi. Chúng ta muốn tính chính xác ít bị giảm nhất có thể.

Một cách để mã hóa những điểm này là biểu diễn chúng dưới dạng có ít số chiều hơn. Với mỗi điểm  $\mathbf{x}^{(i)} \in \mathbb{R}^n$  ta sẽ tìm một vector mã  $\mathbf{c}^{(i)} \in \mathbb{R}^l$ . Nếu  $l$  nhỏ hơn  $n$ , lưu trữ mã của các điểm cần ít bộ nhớ hơn là lưu trữ dữ liệu gốc. Ta muốn tìm một hàm mã hóa dữ liệu đầu vào  $f$ , sao cho  $f(\mathbf{x}) = \mathbf{c}$ , và hàm giải mã dữ liệu đầu ra  $g$ , sao cho  $\mathbf{x} \approx g(f(\mathbf{x}))$ .

PCA được xác định thông qua cách ta chọn hàm giải mã. Cụ thể, để hàm giải mã đơn giản, ta chọn sử dụng phép nhân ma trận để ánh xạ vector mã trở lại  $\mathbb{R}^n$ . Đặt  $g(\mathbf{c}) = \mathbf{D}\mathbf{c}$ , trong đó  $\mathbf{D} \in \mathbb{R}^{n \times l}$  là ma trận giải mã.

Tìm bộ mã tối ưu cho bộ giải mã này có thể là một bài toán khó. Để bài toán mã hóa trở nên dễ hơn, PCA buộc các cột của  $\mathbf{D}$  phải trực giao với nhau. (Lưu ý rằng  $\mathbf{D}$  về mặt kĩ thuật không phải là "ma trận trực giao" trừ khi  $l = n$ .)

Với cách ta mô tả bài toán ở trên, ta có thể thu được nhiều lời giải khác nhau, bởi vì ta có thể tăng  $\mathbf{D}_{:,i}$  nếu giảm tất cả các điểm  $\mathbf{c}_i$  với cùng một tỉ lệ. Để lời giải là duy nhất, ta áp đặt thêm điều kiện tất cả các cột của  $\mathbf{D}$  phải có chuẩn đơn vị.

Để biến ý tưởng đơn giản này thành thuật toán, ta cần tìm ra cách sinh điểm mã tối ưu  $\mathbf{c}^*$  cho mỗi điểm đầu vào  $\mathbf{x}$ . Một cách ta có thể dùng là tìm điểm mã sao cho khoảng cách giữa điểm đầu vào  $\mathbf{x}$  và điểm giải mã của nó  $g(\mathbf{c}^*)$  là cực tiểu. Ta có thể tính khoảng cách này bằng cách sử dụng chuẩn. Trong PCA, ta sử dụng chuẩn  $L^2$ :

$$\mathbf{c}^* = \arg \min_{\mathbf{c}} \|\mathbf{x} - g(\mathbf{c})\|_2 \quad (2.54)$$

Ta có thể chuyển sang chuẩn  $L^2$  bình phương thay vì sử dụng chuẩn  $L^2$  bình thường vì cả hai đều có chung điểm mã cực tiểu  $\mathbf{c}^*$ . Điều đó là vì chuẩn  $L^2$  không âm và bình phương là hàm đơn điệu tăng với các đối số không âm.

$$\mathbf{c}^* = \arg \min_{\mathbf{c}} \|\mathbf{x} - g(\mathbf{c})\|_2^2 \quad (2.55)$$

Hàm tối ưu có thể được đơn giản hóa thành:

$$(\mathbf{x} - g(\mathbf{c}))^T (\mathbf{x} - g(\mathbf{c})) \quad (2.56)$$

(theo định nghĩa về chuẩn  $L^2$ , phương trình (2.30))

$$= \mathbf{x}^T \mathbf{x} - \mathbf{x}^T g(\mathbf{c}) - g(\mathbf{c})^T \mathbf{x} + g(\mathbf{c})^T g(\mathbf{c}) \quad (2.57)$$

$$= \mathbf{x}^T \mathbf{x} - 2\mathbf{x}^T g(\mathbf{c}) + g(\mathbf{c})^T g(\mathbf{c}) \quad (2.58)$$

(vì  $g(\mathbf{c})^T \mathbf{x}$  là một đại lượng vô hướng, nó bằng chuyển vị của chính nó)

Giờ ta có thể biến đổi hàm cần tối thiểu hóa một lần nữa. Ta có thể bỏ qua thành phần đầu tiên vì nó không phụ thuộc vào  $\mathbf{c}$ :

$$\mathbf{c}^* = \arg \min_{\mathbf{c}} -2\mathbf{x}^T g(\mathbf{c}) + g(\mathbf{c})^T g(\mathbf{c}) \quad (2.59)$$

Để tiếp tục, ta thay định nghĩa của  $g(\mathbf{c})$  vào phương trình (2.59):

$$\mathbf{c}^* = \arg \min_{\mathbf{c}} -2\mathbf{x}^T \mathbf{D}\mathbf{c} + \mathbf{c}^T \mathbf{D}^T \mathbf{D}\mathbf{c} \quad (2.60)$$

$$= \arg \min_{\mathbf{c}} -2\mathbf{x}^T \mathbf{D}\mathbf{c} + \mathbf{c}^T \mathbf{I}_l \mathbf{c} \quad (2.61)$$

theo tính chất trực giao và điều kiện chuẩn đơn vị của ma trận  $\mathbf{D}$ , ta có

$$= \arg \min_{\mathbf{c}} -2\mathbf{x}^T \mathbf{D}\mathbf{c} + \mathbf{c}^T \mathbf{c} \quad (2.62)$$

Ta có thể giải bài toán tối ưu này bằng giải tích vector (xem mục 4.3 nếu bạn chưa biết làm thế nào):

$$\nabla_{\mathbf{c}} (-2\mathbf{x}^T \mathbf{D}\mathbf{c} + \mathbf{c}^T \mathbf{c}) = \mathbf{0} \quad (2.63)$$

$$-2\mathbf{D}^T \mathbf{x} + 2\mathbf{c} = \mathbf{0} \quad (2.64)$$

$$\mathbf{c} = \mathbf{D}^T \mathbf{x} \quad (2.65)$$

Các công thức trên làm cho thuật toán hiệu quả hơn: ta có thể mã hóa  $\mathbf{x}$  một cách tối ưu mà chỉ cần dùng các phép toán vector-ma trận. Để mã hóa một vector, ta có thể dùng hàm mã hóa:

$$f(\mathbf{x}) = \mathbf{D}^T \mathbf{x} \quad (2.66)$$

Tiếp tục sử dụng nhân ma trận, ta còn có thể định nghĩa quá trình khôi phục PCA :



$$r(\mathbf{x}) = g(f(\mathbf{x})) = \mathbf{D}\mathbf{D}^T \mathbf{x} \quad (2.67)$$

Tiếp theo, ta cần chọn ma trận mã hóa  $\mathbf{D}$ . Để làm vậy, ta quay lại ý tưởng tối thiểu hóa hàm khoảng cách  $L^2$  giữa đầu vào và kết quả khôi phục. Vì ta sẽ sử dụng cùng một ma trận  $\mathbf{D}$  để giải mã tất cả các điểm, ta không thể xét từng điểm riêng biệt. Thay vào đó, ta cần cực thiểu hóa chuẩn Frobenius của ma trận lỗi. Ma trận này được tính trên tất cả các điểm và tất cả các chiều:

$$\mathbf{D}^* = \arg \min_{\mathbf{D}} \sqrt{\sum_{i,j} (x_j^{(i)} - r(\mathbf{x}^{(i)})_j)^2} \text{ v i i u k i n } \mathbf{D}\mathbf{D}^T = \mathbf{I}_l \quad (2.68)$$

Để thu được thuật toán tìm  $\mathbf{D}^*$ , ta bắt đầu bằng việc xét trường hợp  $l = 1$ . Trong trường hợp này,  $\mathbf{D}$  chỉ là một vector  $\mathbf{d}$ . Thay phương trình (2.67) vào phương trình (2.68) và đơn giản  $\mathbf{D}$  thành  $\mathbf{d}$ , bài toán trở thành:

$$\mathbf{d}^* = \arg \min_{\mathbf{d}} \sum_i \|\mathbf{x}^{(i)} - \mathbf{d}\mathbf{d}^T \mathbf{x}^{(i)}\|_2^2 \text{ v i i u k i n } \|\mathbf{d}\|_2 = 1 \quad (2.69)$$

Công thức trên là các trực tiếp nhất để thay vào nhưng không phải cách hay nhất để viết phương trình. Nó đặt đại lượng vô hướng  $\mathbf{d}^T \mathbf{x}^{(i)}$  ngay bên phải vector  $\mathbf{d}$ . Các đại lượng vô hướng thường được đặt bên trái các vector mà nó thực hiện phép toán. Vì vậy, ta thường viết công thức như sau:

$$\mathbf{d}^* = \arg \min_{\mathbf{d}} \sum_i \|\mathbf{x}^{(i)} - \mathbf{d}^T \mathbf{x}^{(i)} \mathbf{d}\|_2^2 \text{ v i i u k i n } \|\mathbf{d}\|_2 = 1 \quad (2.70)$$

Do số thực là chuyển vị của chính nó:

$$\mathbf{d}^* = \arg \min_{\mathbf{d}} \sum_i \|\mathbf{x}^{(i)} - \mathbf{x}^{(i)T} \mathbf{d}\mathbf{d}\|_2^2 \text{ v i i u k i n } \|\mathbf{d}\|_2 = 1 \quad (2.71)$$

Bạn nên làm quen với cách sắp xếp lại thứ tự này.

Bây giờ, ta nên biểu diễn lại bài toán dưới dạng một ma trận các điểm đầu vào, thay vì tổng của các vector của các điểm đơn lẻ. Điều này sẽ giúp ta viết lại bài toán gọn hơn. Đặt  $\mathbf{X} \in \mathbb{R}^{m \times n}$  là ma trận được xác định bằng cách xếp chồng tất cả các vector của từng điểm, sao cho  $\mathbf{X}_{i,:} = \mathbf{x}^{(i)T}$ . Ta có thể viết lại bài toán như sau:

$$\mathbf{d}^* = \arg \min_{\mathbf{d}} \|\mathbf{X} - \mathbf{X}\mathbf{d}\mathbf{d}^T\|_F^2 \text{ v i i u k i n } \mathbf{d}^T \mathbf{d} = 1 \quad (2.72)$$

Tạm thời chưa xét tới các điều kiện, ta có thể đơn giản hóa phần chuẩn Frobenius như sau:

$$\arg \min_{\mathbf{d}} \|\mathbf{X} - \mathbf{X}\mathbf{d}\mathbf{d}^T\|_F^2 \quad (2.73)$$

$$= \arg \min_d \text{Tr}((\mathbf{X} - \mathbf{X} \mathbf{d} \mathbf{d}^T)^T (\mathbf{X} - \mathbf{X} \mathbf{d} \mathbf{d}^T)) \quad (2.74)$$

theo phương trình (2.49)

$$= \arg \min_d \text{Tr}(\mathbf{X}^T \mathbf{X} - \mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T - \mathbf{d} \mathbf{d}^T \mathbf{X}^T \mathbf{X} + \mathbf{d} \mathbf{d}^T \mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) \quad (2.75)$$

$$= \arg \min_d \text{Tr}(\mathbf{X}^T \mathbf{X}) - \text{Tr}(\mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) - \text{Tr}(\mathbf{d} \mathbf{d}^T \mathbf{X}^T \mathbf{X}) + \text{Tr}(\mathbf{d} \mathbf{d}^T \mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) \quad (2.76)$$

$$= \arg \min_d - \text{Tr}(\mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) - \text{Tr}(\mathbf{d} \mathbf{d}^T \mathbf{X}^T \mathbf{X}) + \text{Tr}(\mathbf{d} \mathbf{d}^T \mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) \quad (2.77)$$

bởi vì các thành phần không liên quan tới  $\mathbf{d}$  không ảnh hưởng tới  $\arg \min$

$$= \arg \min_d - 2\text{Tr}(\mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) + \text{Tr}(\mathbf{d} \mathbf{d}^T \mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) \quad (2.78)$$

bởi vì ta có thể hoán vị thứ tự ma trận trong một vết ma trận, phương trình (2.52)

$$= \arg \min_d - 2\text{Tr}(\mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) + \text{Tr}(\mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T \mathbf{d} \mathbf{d}^T) \quad (2.79)$$

Bây giờ, ta quay trở lại sử dụng các ràng buộc:

$$= \arg \min_d - 2\text{Tr}(\mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) + \text{Tr}(\mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T \mathbf{d} \mathbf{d}^T) \text{ với điều kiện } \mathbf{d}^T \mathbf{d} = 1 \quad (2.80)$$

$$= \arg \min_d - 2\text{Tr}(\mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) + \text{Tr}(\mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) \text{ với điều kiện } \mathbf{d}^T \mathbf{d} = 1 \quad (2.81)$$

do điều kiện  $\mathbf{d}^T \mathbf{d} = 1$

$$= \arg \min_d - \text{Tr}(\mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) \text{ với điều kiện } \mathbf{d}^T \mathbf{d} = 1 \quad (2.82)$$

$$= \arg \max_d \text{Tr}(\mathbf{X}^T \mathbf{X} \mathbf{d} \mathbf{d}^T) \text{ với điều kiện } \mathbf{d}^T \mathbf{d} = 1 \quad (2.83)$$

$$= \arg \max_d \text{Tr}(\mathbf{d}^T \mathbf{X}^T \mathbf{X} \mathbf{d}) \text{ với điều kiện } \mathbf{d}^T \mathbf{d} = 1 \quad (2.84)$$

Bài toán tối ưu này có thể giải sử dụng phân tích riêng. Cụ thể, giá trị tối ưu  $\mathbf{d}$  chính là vector riêng của của ma trận  $\mathbf{X}^T \mathbf{X}$  ứng với trị riêng cực đại.

Suy luận này chỉ gắn với trường hợp  $l = 1$  và chỉ khôi phục thành phần chính (principal component) đầu tiên. Tổng quát hơn, khi ta muốn khôi phục một hệ cơ sở của các thành phần chính, ma trận  $\mathbf{D}$  chính là  $l$  vector riêng ứng với  $l$  trị riêng lớn nhất. Ta có thể chứng minh điều này bằng quy nạp. Chứng minh này coi như bài tập cho bạn.

Đại số tuyến tính là một trong những mảng cơ bản nhất của toán học, cần thiết để hiểu về DL. Chương sau sẽ giới thiệu xác suất, một mảng khác rất phổ biến trong ML.