Matrix Formula – Công Thức Ma Trận:

Square Matrix – Ma Trận Vuông:

1. Cấp Của Ma Trận Vuông?

* Là số hàng/cột của nó

1. Đường Chéo Của Ma Trận?

* Đường chéo chính (Main Diagonal) của 1 ma trận là tập hợp các phần tử nằm trên đường chéo kẻ từ góc trái trên của ma trận chếch xuống góc 45 độ
* Đường chéo trên (Superdiagonal) của 1 ma trận là tập hợp các phần tử nằm trên đường chéo kẻ từ phần tử ở hàng 1 cột 2 chếch xuống góc 45 độ
* Đường chéo dưới (Subdiagonal) của 1 ma trận là tập hợp các phần tử nằm trên đường chéo kẻ từ phần tử ở hàng 2 cột 1 chếch xuống góc 45 độ
* Đường chéo trên Offset k của 1 ma trận là tập hợp các phần tử nằm trên đường chéo kẻ từ phần tử ở hàng 1 cột k + 1 chếch xuống góc 45 độ
* Đường chéo dưới Offset k của 1 ma trận là tập hợp các phần tử nằm trên đường chéo kẻ từ phần tử ở hàng k + 1 cột 1 chếch xuống góc 45 độ

1. Vết (Trace)?

* Cho ma trận vuông A, vết của A là tổng tất cả phần tử trên đường chéo chính của nó
* Ví dụ

* Ta có các định lý sau

Vết của 1 ma trận vuông bất kì trong trường số phức = tổng tất cả Eigen Value của nó, có nhân cho bội đại số tương ứng

1. Định Thức Con (Minor) Và Bù Đại Số (Cofactor) Của Ma Trận Vuông?

* Xét phần tử ở hàng i cột j, tính từ trên xuống, trái sang, bắt đầu từ 1 của ma trận vuông A
* Phần tử này sẽ tương ứng với định thức con có giá trị bằng định thức của cái ma trận sau khi lấy A bỏ đi hàng i cột j, kí hiệu là Mij
* Nếu i + j chẵn, thì bù đại số của phần tử = định thức con của nó, nếu lẻ, thì = âm định thức con của nó, kí hiệu là Aịj

1. Cách Tính Định Thức Bằng Mở Rộng Laplace?

* Cho ma trận vuông A, chọn 1 dòng hay cột bất kì, lấy mỗi phần tử trên đó nhân với bù đại số tương ứng rồi cộng tất cả lại được định thức của A

1. Ma Trận Bù Đại Số (Cofactor Matrix)?

* Cho ma trận vuông A, ma trận bù đại số của A được tạo ra bằng cách thay mỗi phần tử của A bằng bù đại số của nó, kí hiệu cof(A)
* Cho ma trận vuông A cấp n bất kì, ta có các định lý sau

Hạng của cof(A) = hạng của A khi A khả nghịch, = 1 khi hạng của A = n – 1, = 0 khi hạng của A = n – 2 trở xuống

* Ví dụ

1. Ma Trận Phụ Hợp (Adjugate Matrix, Adjoint Matrix)?

* Cho ma trận vuông A, ma trận phụ hợp của A là ma trận chuyển vị của ma trận bù đại số của A, kí hiệu PA
* Cho ma trận vuông A cấp n bất kì, I là ma trận Identity cấp n, ta có các định lý sau

* Ví dụ

1. Lũy Thừa Của Ma Trận Vuông?

* Cho A là ma trận vuông bất kì, khi này lũy thừa bậc n của A, kí hiệu An, nghĩa là lấy ma trận Identity nhân cho A n lần
* Cho A là ma trận vuông bất kì cấp n, I là ma trận Identity cấp n, ta có các định lý sau

A0 = I

Ta có thể tách A thành tổng của 2 ma trận đơn giản hơn rồi tính lũy thừa cho dễ, lưu ý khi khai triển phải giữ nguyên thứ tự của các ma trận, không được hoán vị

Animation Matrix – Ma Trận Hoạt Ảnh:

1. Ma Trận Biến Hình Thành Hình Đối Xứng Của Nó Qua Gốc Tọa Độ?

1. Ma Trận Quay Quanh Trục Cao, Hướng Ngược Chiều Kim Đồng Hồ Khi Nhìn Từ Trên Xuống?

* θ là góc quay

1. Ma Trận Quay Tổng Quát?

* Ban đầu, quay quanh trục Ox, hướng từ Oy quét sang Oz là chiều dương, góc quay = a
* Tiếp theo, quay quanh trục Oy, hướng từ Oz quét sang Ox là chiều dương, góc quay = b
* Cuối cùng, quay quanh trục Oz, hướng từ Ox quét sang Oy là chiều dương, góc quay = c
* Ví dụ, quay quanh trục Ox 25 độ, trục Oy 30 độ, trục Oz 80 độ

1. Ma Trận Scale, Quay Rồi Mới Tịnh Tiến Trong Không Gian 3D?

* Lưu ý phải đổi tọa độ điểm từ 3 chiều sang 4 chiều, 3 phần tử đầu giá trị giữ nguyên, phần tử thứ 4 giá trị luôn = 1
* Trong đó, ma trận Scale rồi quay là

* Vector tịnh tiến là

* Ví dụ dạng 4 chiều của tọa độ điểm

* Lưu ý các hành động Scale, quay và tịnh tiến không có tính giao hoán với nhau, riêng với quay thì 2 phép quay khác nhau cũng không giao hoán được với nhau
* Ví dụ, Scale theo trục Ox 2 lần, Oy 3 lần, Oz 4 lần, rồi quay quanh trục Ox 25 độ, trục Oy 30 độ, trục Oz 80 độ, rồi tịnh tiến dọc theo tia Ox 5 đơn vị, Oy 6 đơn vị, Oz 7 đơn vị
* Ma trận quay rồi Scale là

m

* Ghép với Vector tịnh tiến ta được ma trận cuối cùng

1. Ma Trận Shear?

* Shear theo trục hoành

* Shear theo trục tung

* Ma trận Identity cũng là ma trận Shear với a = 0

1. Tính Ma Trận Chiếu Vuông Góc 1 Không Gian Vector Vào 1 Không Gian Con Của Nó?

* A là ma trận kích thước m x n, m > n, các cột của A độc lập tuyến tính, có không gian cột là không gian con đang nói đến, khi đó ma trận chiếu vuông góc vào A là

* P là ma trận kích thước m x m, cho X là Vector bất kì trong không gian m chiều, thì PX = hình chiếu của X lên không gian cột của A
* Ví dụ
* Ta muốn tạo ra 1 ma trận P có tác dụng ánh xạ 1 điểm trong không gian 3D vào hình chiếu của nó trên 1 mặt phẳng qua gốc tọa độ, mặt phẳng này chính là 1 không gian con của không gian 3D, gọi A là ma trận có không gian cột là mặt phẳng này, giả sử

* Chứng minh công thức trên, đầu tiên P phải ánh xạ A vào chính nó

* Tiếp theo, Null Space của P phải vuông góc không gian cột của P, vâng, vì các Vector đơn vị bị ánh xạ lên A, nên không gian cột của P cũng chính là A, có như vậy thì nó mới chiếu vuông góc, mà ta lại có

* Như vậy P đối xứng, hay không gian hàng = không gian cột, mà Null Space vuông góc không gian hàng, nên nó cũng vuông khóc không gian cột
* Bây giờ xét khi m < n, các hàng của A độc lập tuyến tính, có không gian hàng là không gian con đang nói đến, khi đó ma trận chiếu vuông góc vào không gian hàng của A = ma trận chiếu vuông góc vào không gian cột của AT

* Nếu m = n thì ma trận chiếu vuông góc vào không gian cột = ma trận chiếu vuông góc vào không gian hàng = ma trận đơn vị

Linear Transformation – Biến Đổi Tuyến Tính:

1. Diện Tích Của 1 Hình Khi Bị Biến Đổi Bởi 1 Ma Trận?

* Diện tích của nó sẽ nhân k lần, k là định thức của ma trận

1. Định Thức Ma Trận Có Tỉ Lệ Với Độ Tăng Chiều Dài?

* Không, chiều dài của 1 đoạn thẳng bị biến đổi với ma trận không tỉ lệ với định thức của ma trận biến đổi nó

1. Biến Đổi Affine?

* Là biến đổi tuyến tính rồi tịnh tiến

1. Biến Đổi Cứng (Rigid Transformation, Euclidean Transformation)?

* Là 1 tổ hợp các phép quay, tịnh tiến, và Flip

1. Không Gian Hàng Và Không Gian Cột?

* Xét ma trận A có kích thước m x n
* Không gian hàng là Span của tất cả các hàng, kí hiệu C(AT)
* Không gian cột là Span của tất cả các cột, kí hiệu C(A)
* Số chiều của không gian hàng và không gian cột luôn = nhau và = hạng của ma trận = r(A)
* Thêm 1 cột hoặc 1 hàng mới vào A, thì hạng của A chỉ có thể tăng 1 hoặc giữ nguyên

1. Range Space?

* Range Space là không gian ảnh của 1 không gian V nào đó bị biến đổi bởi 1 ánh xạ Homomorphism f, còn gọi là ảnh của ánh xạ f
* Số chiều của Range Space gọi là hạng của ánh xạ f
* Kí hiệu

1. Null Space Và Left Null Space?

* Cho ánh xạ tuyến tính f từ 1 không gian Vector V vào chính nó, khi này Null Space của f = tập hợp tất cả các Vector trong V bị f ánh xạ vào Vector 0, nói cách khác, Null Space của f = nhân của f, kí hiệu ker(f) hoặc f–1(0)
* Số chiều của Null Space gọi là số vô hiệu của ánh xạ f (Nullity)
* Null Space của chuyển vị của ánh xạ tuyến tính f thì gọi là Left Null Space của f
* Null Space của 1 ma trận luôn vuông góc với không gian hàng của ma trận đó, tương tự Left Null Space luôn vuông góc không gian cột

1. Tính Số Chiều Của Null Space Và Left Null Space Của 1 Ma Trận?

* Số chiều của Null Space là

* r là hạng của ma trận
* n là số cột của ma trận
* Số chiều của Left Null Space là

* m là số hàng của ma trận

1. Right Eigen Value Và Right Eigen Vector?

* Cho ma trận vuông A cấp n với các phần tử thuộc trường số phức, khi áp dụng ma trận này để biến đổi không gian Vector phức, thì để ý thấy, có những Vector khác Vector 0 chỉ bị Scale chứ không bị quay, đây gọi là những Right Eigen Vector của A, hệ số Scale gọi là Right Eigen Value

* Bất cứ cặp giá trị λ và v nào thỏa mãn phương trình trên, sao cho v khác Vector 0, thì đều sẽ được gọi là Right Eigen Value và Right Eigen Vector của A
* Mỗi Right Eigen Value phân biệt sẽ có 2 thuộc tính ứng với nó, là bội đại số (Algebraic Multiplicity) và bội hình học (Geometric Multiplicity)
* Tổng bội đại số của tất cả Right Eigen Value phân biệt của A = n
* Bội hình học của 1 Right Eigen Value luôn ≤ bội đại số của nó
* Không gian con riêng bên phải (Right Eigen Space) của 1 Right Eigen Value của A là tập hợp tất cả Right Eigen Vector của A ứng với Right Eigen Value này, cùng với Vector 0

1. Left Eigen Value và Left Eigen Vector?

* Thông thường Right Eigen Value sẽ được gọi là Eigen Value còn Right Eigen Vector sẽ được gọi là Eigen Vector
* Bất cứ cặp giá trị λ và v nào thỏa mãn phương trình sau, sao cho v khác Vector 0, thì đều sẽ được gọi là Left Eigen Value và Left Eigen Vector của A

* Nghĩa là, Left Eigen Value và Left Eigen Vector của A chính là Right Eigen Value của Right Eigen Vector của A\*

1. Tính Eigen Value Và Eigen Vector?

* Cho A là 1 ma trận vuông cấp n với các phần tử thuộc trường số phức, I là ma trận Identity cấp n, đa thức đặc trưng của A là

* λ là Eigen Value của A khi và chỉ khi nó là nghiệm của phương trình đặc trưng sau

* Nếu λ là nghiệm bội n thì n chính là bội đại số của nó, vì dụ λ là nghiệm kép thì bội đại số của nó = 2
* Định lý Cayley Hamilton nói rằng A thỏa mãn chính phương trình đặc trưng của nó, nghĩa là ta luôn có

* 0 kí hiệu cho ma trận không
* Có thể dùng định lý này để tính lũy thừa của A
* Ví dụ tính A mũ 6, bằng A và A mũ 2

* Dùng mẹo tính phần dư, ta được

* Thế x thành A, ta được

* 1 Vector v là 1 Eigen Vector ứng với λ khi và chỉ khi nó thỏa mãn phương trình sau

* Nói cách khác, không gian con riêng của λ = Null Space của ma trận A – λI, số chiều của nó chính là bội hình học của λ
* Xét ma trận sau

* Có 1 sự thật là lấy mỗi Eigen Value của A cộng thêm λ, ta sẽ được Eigen Value tương ứng trong B, do đó người ta thường dùng B với λ nhỏ để xấp xỉ A trong trường hợp A không khả nghịch, giả sử các Eigen Value của A đều không âm, nhưng có thể có thằng = 0, và do đó khiến A không khả nghịch, khi này do ở B đã cộng thêm λ, khiến mọi Eigen Value của nó > 0, và do đó khả nghịch

1. Eigen Vector Tổng Quát?

* Cho A là 1 ma trận vuông cấp n với các phần tử thuộc trường số phức, I là ma trận Identity cấp n, λ là 1 Eigen Value của A, khi này 1 Vector v khác Vector 0 được gọi là Eigen Vector bậc k của A ứng với λ, k nguyên dương, nếu nó thỏa mãn 2 điều kiện sau

* Khi này, tập hợp tất cả các Eigen Vector bậc k của A ứng với λ kèm theo Vector 0 gọi là không gian con riêng bậc k của λ
* Dễ thấy nếu k = 1, thì v chính là Eigen Vector bình thường

1. Tính Định Thức Bằng Eigen Value?

* Cho 1 ma trận vuông A bất kì, lấy mỗi Eigen Value của nó mũ lên, với số mũ = bội đại số tương ứng, sau đó nhân tất cả Eigen Value đã mũ lại, ta được định thức của A
* Ta có các định lý sau

Một ma trận vuông bất kì trong trường số phức là khả nghịch khi và chỉ khi mọi Eigen Value của nó đều khác 0

1. Bán Kính Phổ (Spectral Radius)?

* Cho ma trận vuông A trong trường số phức, khi này bán kính phổ của A là Module của Eigen Value có Module lớn nhất của nó, gọi các Eigen Value của A là λ1, λ2, …

1. EVD (Eigen Value Decomposition)?

* Cho 1 ma trận vuông A cấp n bất kì với các phần tử thuộc trường số phức, sao cho tất cả Eigen Value phân biệt của nó có bội đại số = bội hình học, thì khi này Jordan Decomposition trên A còn được gọi là EVD, nghĩa là, ta có thể tìm được 1 ma trận đường chéo Λ, còn gọi là ma trận phổ (Spectral Matrix) của A, chứa các phần tử là Eigen Value của A, và ma trận Q, được gọi là ma trận chuyển đổi Mod (Modal Matrix) của A, chứa các Eigen Vector độc lập tuyến tính của A, sao cho

* Những ma trận mà có thể phân tích được kiểu này gọi là ma trận khả chéo hóa (Diagonalizable Matrix), nếu Q là ma trận trực giao thì gọi là ma trận khả chéo hóa trực giao
* Để chứng minh công thức trên, ta tưởng tượng 1 phép biến đổi tuyến tính bởi ma trận A lên 1 hình bình hành như sau, xét trên trường số thực



* Dễ thấy 2 mũi tên màu đỏ là 2 Eigen Vector độc lập tuyến tính nào đó của A, gọi ma trận có 2 cột là 2 Vector này là Q, cũng dễ thấy Eigen Value của mũi tên bên trái = 1.5, của mũi tên bên phải = 2
* Ta sẽ phân tích A thành nhiều phép biến đổi, ban đầu cho Q-1 tác dụng vào hình bình hành, do Q-1Q = I, kết quả ta được hình vuông đơn vị



* Tiếp theo Scale hình vuông này theo chiều dọc 1.5 lần và chiều ngang 2 lần, dễ thấy đây chính là các Eigen Value



* Cuối cùng dùng ma trận Q để đưa hình chữ nhật về hình dạng hình bình hành ban đầu nhưng 2 cạnh bị Scale bởi Eigen Value



* Vậy, phép biến đổi A thực chất cũng chỉ là ta sử dụng phép đổi Q-1 rồi sau đó dùng Λ và cuối cùng dùng Q

1. Ma Trận Đồng Dạng (Matrix Similarity)?

* 2 ma trận vuông A và B gọi là đồng dạng nếu tồn tại ma trận P sao cho

* Điều này nghĩa là A và B có Eigen Value giống nhau, hay nói cách khác, bản chất của A và B là một, chỉ là chúng tác dụng lên cơ sở khác nhau

1. SVD (Singular Value Decomposition)?

* Mọi phép biến đổi tuyến tính đều có bản chất là quay và Scale, do đó, mọi ma trận đều có thể phân tích thành các ma trận quay và Scale
* Cho A là ma trận kích thước m x n bất kì trong trường số phức, ta luôn có thể phân tích A thành

* Sao cho Σ là ma trận đường chéo kích thước m x n, các phần tử trên đường chéo chính là số thực không âm, gọi là các Singular Value của A, Σ vừa là nhân tố Scale, vừa là nhân tố giảm chiều hoặc tăng chiều
* U là ma trận đơn nhất kích thước m x m, các cột của U gọi là các Left Singular Vector của A
* V là ma trận đơn nhất kích thước n x n, các cột của V gọi là các Right Singular Vector của A
* U và V đều là nhân tố quay và Flip
* SVD có thể được hiểu như sau, phép biến đổi tuyến tính A thực chất là ta chọn 1 cơ sở là các cột của 1 ma trận đơn nhất, và chỉ cần 1 lần Scale với hệ số Scale không âm theo các trục của cơ sở đó là tương đương với A, khi đó các hệ số Scale chính là các Singular Value
* Tưởng tượng ta có phép biến đổi tuyến tính là ma trận A tác dụng lên hình vuông đơn vị



* Bây giờ ta sẽ phân tích A thành nhiều phép biến đổi, ban đầu cho V-1 tác dụng vào hình vuông đơn vị, tương đương với việc quay và Flip hình vuông đó



* Tiếp theo Scale hình vuông này theo chiều dọc 2 lần và chiều ngang 3 lần, dễ thấy đây chính là các Singular Value



* Cuối cùng, dùng U để quay vào đúng vị trí



* Vậy, phép biến đổi A thực chất cũng chỉ là quay với Scale
* Cho A là ma trận kích thước bất kì trong trường số phức, ta có các định lý sau

A có thể có nhiều SVD khác nhau, nhưng tập hợp Singular Value là như nhau

Số Singular Value khác 0 của A = hạng của A

1. Tìm SVD Của 1 Ma Trận?

* Cho A là ma trận kích thước m x n bất kì trong trường số phức, ta muốn tìm 1 SVD của A

* Bước 1, để tìm V kích thước n x n, trước tiên tìm EVD của A\*A, đây luôn là ma trận Hermitian, kích thước n x n, nên ta luôn tìm được 1 bộ n Eigen Vector đơn vị đôi một vuông góc với nhau của nó, sắp các Eigen Vector này thành các cột, lưu ý sắp các Eigen Vector theo thứ tự giảm dần của Eigen Value từ trái sang phải, ta được V
* Bước 2, để xác định Σ kích thước m x n, đặt căn bậc 2 của từng Eigen Value ứng với từng Eigen Vector trong V vào đường chéo chính của Σ, đó là do A\*A luôn là ma trận bán xác định dương nên mọi Eigen Value của nó đều là số thực không âm, nên có thể lấy căn bậc 2
* Bước 3, tìm U kích thước m x m, tương tự như tìm V, trước tiên tìm EVD của AA\*, đây luôn là ma trận Hermitian, kích thước m x m, nên ta luôn tìm được 1 bộ m Eigen Vector đơn vị đôi một vuông góc với nhau của nó, sắp các Eigen Vector này thành các cột, và do là Eigen Value của các Eigen Vector này cũng chính = Eigen Value của các Eigen Vector trong V, nên bạn cần phải sắp các Eigen Vector vào U đúng thứ tự như trong V, hoặc dễ hơn, là vì ta đã có A, Σ và V, ta có thể tính U = AVΣ–1, lưu ý ở đây Σ–1 đạt được bằng cách chuyển vị Σ rồi lấy 1 chia cho từng phần tử
* Chứng minh cách làm trên

* Dễ thấy (\*) có dạng EVD
* Chứng minh tương tự cho AA\*
* Ví dụ

* Vậy

1. Tìm Nghịch Đảo Của Ma Trận Chữ Nhật?

* Cho ma trận A kích thước m x n trong trường số phức, nếu m > n và các cột trong A độc lập tuyến tính, thì sẽ tồn tại duy nhất 1 nghịch đảo của A, và nghịch đảo này sẽ là ma trận A–1 kích thước n x m thỏa mãn

* Chứng minh điều trên, đầu tiên, tích A–1A phải là ma trận Identity

* Ngược lại, tích AA–1 có thể không phải ma trận Identity, do đó, A–1 còn được gọi là nghịch đảo bên trái của A
* Nếu m < n, và các hàng của A độc lập tuyến tính, thì tồn tại duy nhất 1 nghịch đảo của A, và nghịch đảo này sẽ là ma trận A–1 kích thước n x m thỏa mãn

* Chứng minh điều trên, đầu tiên, tích AA–1 phải là ma trận Identity

* Ngược lại, tích A–1A có thể không phải ma trận Identity, do đó, A–1 còn được gọi là nghịch đảo bên phải của A
* Ví dụ

1. Ma Trận Ánh Xạ Tuyến Tính Ứng Với 1 Cặp Cơ Sở?

* Cho cơ sở chính tắc A, và 2 cơ sở khác là B và C, cho M là ma trận nào đó biến đổi A
* Cho D là 1 Vector ứng với A, DB là biểu diễn của D trong B, khi cho M tác dụng lên A, D sẽ ánh xạ tới điểm D’ = MD, biểu diễn của D’ trong C là D’C
* Bây giờ ta muốn tìm 1 ma trận, sao cho nó biến DB thành D’C, vâng, 1 ma trận N sao cho NDB = D’C, ma trận N gọi là ma trận của ánh xạ tuyến tính M ứng với cặp cơ sở B và C
* Để tìm N
* Bước 1, tính MB, ta tính được sau khi bị M tác dụng thì B sẽ thành cái gì
* Bước 2, tìm biểu diễn của MB ứng với cơ sở C, được N
* Công thức tổng quát

1. Xấp Xỉ Hạng Thấp (Low Rank Approximation)?

* Là việc mô phỏng gần đúng ma trận M kích thước m x n trong trường số phức bằng một ma trận N kích thước tương tự nhưng có hạng nhỏ hơn, sao cho Matrix Norm của M – N nhỏ nhất, Matrix Norm ở đây ta có thể chọn tùy ý
* Có thể sẽ có nhiều N khác nhau ứng với mỗi Matrix Norm khác nhau, sau đây ta sẽ chỉ tìm N sao cho Frobenius Norm của M – N nhỏ nhất, bằng cách tìm SVD của M, lưu ý ở đây thứ ta trả về là SVD đã được giảm hạng, không phải N, vì N có kích thước = M, thì trả về nó làm cái chó gì, trong khi đó tổng kích thước của các ma trận trong SVD nhỏ hơn nhiều
* Bước 1, tìm SVD của M, theo phương pháp truyền thống

* Bước 2, có 1 sự thật là, tích của ma trận A kích thước q x w và B kích thước

w x e có thể được tính như sau, lấy cột 1 của A, tương đương ma trận kích thước q x 1, nhân cho hàng 1 của B, tương đương ma trận kích thước 1 x e, ta được ma trận kích thước q x e, = kích thước của tích AB, như vậy, lần lượt ta nhân cột 2 của A với hàng 2 của B sẽ được 1 lớp nữa, rồi cột 3 của A với hàng 3 của B, được 1 lớp nữa, …, tổng các lớp này sẽ = tích AB, do đó ta có thể viết lại SVD của M như sau, gọi các cột trong U lần lượt là u1, u2, …, um, và các cột trong V lần lượt là v1, v2, …, vn, và các phần tử trên đường chéo chính của Σ lần lượt là a1, a2, …, ak, k ở đây = Min(m, n), dễ thấy các phần tử a giảm dần

* Do a giảm dần, nên các toán hạng ở công thức trên, càng ra xa thì càng giảm độ lớn, nghĩa là ta có thể vứt chúng đi mà không ảnh hưởng nhiều đến kết quả tổng thể, do đó, ta sẽ chỉ lấy các toán hạng từ 1 đến r, r là hạng của N mà ta muốn

* Lấy a1 tới ar trong biểu thức trên lắp vào ma trận Σ’ mới, các cột u1 đến ur lắp vào ma trận U’ mới, các cột v1 đến vr lắp vào ma trận V’ mới, ta sẽ được các ma trận cần tìm

1. PCA (Principle Component Analysis)?

* Ban đầu, ta có 1 ma trận D kích thước m x n, các cột trong nó chính là các điểm trong không gian m chiều
* Nhưng địt mẹ m quá lớn, ta không thể Plot các điểm đó, do đó ta sẽ cần giảm chiều dữ liệu của D xuống còn 1, 2 hoặc 3 để có thể Plot
* Để làm được điều này thực hiện các bước sau
* Bước 1, tịnh tiến bộ điểm về vị trí sao cho Mean của chúng = gốc tọa độ

* Bước 2, tính ma trận tự hiệp phương sai kích thước m x m của D’, ta có thể sử dụng công thức sau

* Bước 3, tìm EVD của ma trận C, sao cho các Eigen Vector có độ dài = 1, và các Eigen Value được sắp xếp theo thứ tự giảm dần từ trái sang phải

* Khi này, Eigen Vector thứ 1 trong Q sẽ có phương mà bộ điểm phân tán nhất theo phương này, tức Variance theo phương này là lớn nhất, Eigen Vector thứ 2 thì có điều kiện ràng buộc là vuông góc với Eigen Vector thứ 1, và phải có phương mà bộ điểm phân tán nhất có thể, …
* Bước 4, như vậy việc cần làm bây giờ chỉ là sử dụng 1 phép quay để đưa Eigen Vector thứ 1 trùng với trục Ox, Eigen Vector thứ 2 trùng với trục Oy, …, phép quay này chính là ma trận nghịch đảo của Q, mà do Q là ma trận trực giao nên nghịch đảo của nó = chuyển vị của nó, nên ta có thể dùng chuyển vị của Q để quay bộ điểm

* Bước 5, chỉ giữ lại các hàng đầu tiên, ví dụ hàng 1, hàng 2, hàng 3 trong D’’, bỏ các hàng còn lại, ta được D’’’ chính là bộ điểm đã được giảm chiều, Plot D’’’ vào hệ tọa độ 1D, 2D hoặc 3D

1. LDA (Linear Discriminant Analysis)?

* Ban đầu, ta có 1 ma trận D kích thước m x n, các cột trong nó chính là các điểm trong không gian m chiều, và mỗi điểm đều được gắn 1 nhãn, có tổng cộng k nhãn khác nhau tương ứng với k nhóm điểm
* Bây giờ ta muốn giảm chiều dữ liệu bằng cách dùng 1 ma trận W kích thước

h x m để quay và Scale bộ điểm tới vị trí sao cho tại đó khi ép nó thẳng xuống trục Ox, hoặc mặt phẳng Oxy, hoặc không gian Oxyz, …, tức là ta chỉ giữ lại các tọa độ đầu tiên, vứt đi các tọa độ còn lại, sao cho mỗi nhóm điểm cách xa nhau nhất có thể và độ phân tán của mỗi nhóm điểm là nhỏ nhất có thể, nghĩa là tối ưu sự phân biệt giữa các nhóm điểm

* h càng cao, thì độ phân cách càng tăng, nhưng đến một lúc nào đấy, độ phân cách sẽ không tăng nữa, và người ta đã chứng minh được h = k – 1 là tối đa, h cao hơn nữa sẽ không tăng độ phân cách
* Bước 1, tính ma trận Within Class Covariance SW là ma trận bán xác định dương cấp m
* Với mỗi nhóm điểm thứ i, gọi ma trận Di có các cột là các điểm của nhóm đó, tính ma trận tự hiệp phương sai của Di với gốc tọa độ là Mean của nó, lưu ý không chia cho số điểm trong nhóm

* Sau đó cộng ma trận C của tất cả các nhóm điểm lại, ta được SW
* Bước 2, tính ma trận Between Class Covariance SB là ma trận bán xác định dương cấp m
* Với mỗi nhóm điểm thứ i, tính ma trận sau, Ni là số điểm của nhóm, Di có các cột là tọa độ các điểm trong nhóm

* Sau đó cộng ma trận R của tất cả các nhóm điểm lại, ta được SB
* Bước 3, đảm bảo cho SW khả nghịch, dễ thấy dù tất cả Eigen Value của SW đều không âm, nhưng vẫn sẽ có trường hợp = 0, do đó ta cần cộng vào SW một lượng λI, với λ là 1 số thực nhỏ và I là ma trận Identity, khi này SW sẽ luôn khả nghịch
* Bước 4, tìm Eigen Value lớn nhất của ma trận sau, gọi nó là λmax

* Bước 5, tìm 1 bộ Eigen Vector độc lập tuyến tính của M, sao cho chúng có cùng Eigen Value là λmax, như ta đã nói, số lượng Eigen Vector này = h và chắc chắn không vượt quá k – 1, có thể có nhiều bộ Eigen Vector khác nhau, nhưng độ phân cách các bộ điểm khi áp dụng chúng là như nhau
* Bước 6, đặt lần lượt các Eigen Vector tìm được ở bước 5 vào từng hàng trong W
* Bước 7, nhân W với D được D’, Plot D’ vào hệ trục tọa độ h chiều, thông thường là 1D, 2D hoặc 3D, quan sát kết quả ta sẽ thấy từng nhóm điểm đã được phân cách ra rõ ràng
* Chứng minh cách làm trên
* Gọi sB là độ phân cách giữa các nhóm điểm, hay Between Class Variance, ta định nghĩa sB là tổng bình phương khoảng cách từ Mean của các nhóm điểm tới Mean của toàn bộ dữ liệu, với trọng số là số điểm ở mỗi nhóm

* Gọi sW là tổng độ phân tán của mỗi nhóm điểm, hay Within Class Variance, ta định nghĩa sW là tổng bình phương khoảng cách của mỗi điểm tới Mean của nhóm nó thuộc về

* Mục tiêu của ta sẽ là tìm cực đại của hàm sau, tức là làm cho sB lớn nhất có thể và đồng thời làm sW nhỏ nhất có thể

* Như vậy ta chỉ cần lấy đạo hàm f(W) theo W và giải phương trình đạo hàm = 0 để tìm cực trị

* Dễ thấy, để (\*) có thể xảy ra, thì các cột của WT phải là các Eigen Vector của M, đồng thời chúng phải ứng với cùng 1 Eigen Value, khi này, f(W) sẽ = Eigen Value đó, và để f(W) lớn nhất, thì Eigen Value này phải lớn nhất, nói cách khác, các hàng của W phải là các Eigen Vector ứng với Eigen Value lớn nhất của M, dễ thấy, do số Eigen Vector của W không ảnh hưởng đến Eigen Value lớn nhất của M, và do đó không ảnh hưởng tới giá trị lớn nhất của f(W), ta có thể chọn 1, hoặc 2, … hoặc số lượng Eigen Vector tùy ý ứng với Eigen Value lớn nhất này để đặt vào W mà f(W) không đổi, gọi số lượng này = h
* Bây giờ ta sẽ chứng minh h không vượt quá k – 1, ta có

* Ở đây, dễ thấy PPT = mỗi cột trong P nhân cho chuyển vị của nó rồi cộng hết lại, do đó ta có thể định nghĩa cột thứ i trong P như sau, nó cũng sẽ tương ứng với nhóm điểm thứ i

* Dễ thấy P có k cột, xét cột thứ k là Pk là cột cuối cùng, ta có thể tính Pk dựa vào

k – 1 cột còn lại từ phương trình sau

* Nghĩa là, Pk là 1 tổ hợp tuyến tính của k – 1 cột còn lại, do đó hạng của P không thể vượt quá k – 1, mà SB = PPT, do đó hạng của SB cũng không thể vượt quá

k – 1, mà ta lại có

* Do đó hạng của M không thể vượt quá k – 1, mà do Eigen Vector của M phải nằm trong không gian Span bởi M, do đó M không thể có nhiều hơn k – 1 Eigen Vector độc lập tuyến tính, và do đó h không thể vượt quá k – 1

1. SVM (Support Vector Machine)?

* Ban đầu, ta có 1 ma trận D kích thước m x n, các cột trong nó chính là các điểm trong không gian m chiều, và mỗi điểm đều được gắn 1 trong 2 nhãn có giá trị là 1 hoặc –1, Vector Y n chiều gồm n nhãn ứng với n điểm
* Bài toán đặt ra là tìm phương trình của 1 Hyperplane m – 1 chiều ngăn cách 2 nhóm điểm này 1 cách hợp lí nhất, giả sử tồn tại mặt phân cách hoàn toàn giữa 2 nhóm
* Bước 1, tính ma trận V kích thước m x n = cách nhân mỗi cột, hay mỗi điểm của D với phần tử tương ứng trong Y
* Bước 2, tính ma trận K là ma trận bán xác định dương cấp n x n

* Bước 3, giải bài toán tối ưu thuộc kiểu quy hoặc bậc 2 sau
* Hàm cần tìm giá trị nhỏ nhất là

* λ là Vector n chiều ứng với n điểm trong D
* 1 kí hiệu cho Vector chỉ toàn 1, trong biểu thức trên thì nó có số chiều = n
* Các điều kiện là

* Bước 4, tính pháp tuyến W của Hyperplane cần tìm, nó là Vector m chiều
* Ban đầu lấy mỗi cột, hay mỗi điểm trong V nhân cho giá trị tương ứng trong λ, sau đó cộng tất cả các cột lại ta được Vector W
* Bước 5, tính Offset B của Hyperplane cần tìm, nó là giá trị vô hướng
* Gọi số phần tử khác 0 trong λ là N, ban đầu tính Vector R n chiều sau

* Tiếp theo tính B bằng cách lấy tổng tất cả phần tử trong R ứng với giá trị trong λ khác 0, tức là phần tử thứ i trong R được cộng vào tổng khi và chỉ khi phần tử thứ i trong λ khác 0, rồi cuối cùng lấy tổng này chia N được B, do λ luôn ≥ 0, nên ta có thể thay điều kiện thành λ > 0
* Bước 6, phương trình của Hyperplane là

* Bước 7, dự đoán nhãn của dữ liệu mới xi bằng cách xét dấu của

* Chứng minh cách làm trên
* Giả sử mặt dương của Hyperplane chứa các điểm ứng với nhãn 1, và mặt âm của nó chứa các điểm ứng với nhãn –1, khi này khoảng cách từ 1 điểm bất kì tới Hyperplane là, xi là tọa độ điểm đó, yi là nhãn của nó

* Thứ ta cần làm cho lớn nhất là khoảng cách từ điểm gần nhất tới Hyperplane, khoảng cách này tạm gọi là margin

* Như vậy ta cần tìm W và B sao cho margin là lớn nhất
* Nếu thế W thành kW và B thành kB với k là 1 số nào đó sao cho yi(WTxi + B) = 1, thì Hyperplane không thay đổi, nghĩa là margin không thay đổi, do đó ta có thể thế biến đổi bài toán thành
* Hàm cần tìm cực đại là

* Điều kiện

* Ta có thể biến đổi bài toán này thành
* Hàm cần tìm cực tiểu là

* Ta đạt được hàm này = cách nghịch đảo hàm cần tìm cực đại, do đó cực đại sẽ thành cực tiểu, ta bình lên rồi nhân 1 / 2 cho tí nữa dễ tính đạo hàm
* Điều kiện

* Điều kiện này tương đương với điều kiện ban đầu, đó là do khi ta cố gắng tối thiểu Module của W, thì khoảng cách yi(WTxi + B) với mọi i cũng cố gắng giảm xuống, mà do điều kiện ≥ 1 nên nó bị chặn dưới = 1, do đó sẽ luôn tồn tại điểm có yi(WTxi + B) = 1, mà do điểm này gần nhất rồi nên các điểm còn lại sẽ ≥ 1
* Biến đổi điều kiện này 1 lần nữa để ra dạng chuẩn của bài toán quy hoạch bậc 2

* Ta có thể giải ngay bài toán quy hoạch bậc 2 này tại đây, tuy nhiên dễ thấy ta phải giải hệ bất phương trình gồm n bất phương trình, và khi n là 1 triệu điểm thì giải cực lâu, do đó thay vì làm như trên ta sẽ tối ưu hàm đối ngẫu của nó
* Dễ thấy bài toán trên là lồi, ta chọn 1 Hyperplane ở rất rất xa chùm điểm dữ liệu, tức khi này yi(WTxi + B) sẽ rất lớn và do đó

* Điều trên nghĩa là tiêu chuẩn Slater đã được thỏa mãn, và do đó đối ngẫu mạnh xảy ra, ta sẽ tiếp tục xây dựng hàm Lagrange của bài toán

* Trong đó λ là Vector n chiều, λi là phần tử thứ i của nó
* Ta có hàm đối ngẫu là

* Do hàm này trích xuất giá trị nhỏ nhất, hay những cực tiểu của hàm Lagrange, nên ta có thể tìm biểu thức cụ thể của nó = cách tính đạo hàm = 0 theo W, B của hàm Lagrange

* Dễ thấy công thức trên là công thức mà ta đã dùng để tính W

* Thế (1) và (2) vào hàm đối ngẫu

* Dễ thấy công thức trên của g(λ) = –Obj(λ), do đó lúc trước ta tối thiểu Obj là để tối đa g, kèm theo điều kiện là (2) và λ ≥ 0 là miền xác định hàm đối ngẫu
* Do bài toán gốc là lồi và thỏa mãn tiêu chuẩn Slater, do đó nghiệm tối ưu của nó cũng thỏa mãn hệ điều kiện KKT, 1 trong các điều kiện đó là

* Do chỉ các điểm gần với Hyperplane nhất, thì 1 – yi(WTxi + B) mới = 0, còn các điểm còn lại thì đều < 0, do đó ở các điểm này, λi bắt buộc phải = 0, tức là λi chỉ có thể dương khi nó ứng với 1 trong các điểm gần Hyperplane nhất
* Xét các điểm mà λi tương ứng dương, tức là các điểm gần Hyperplane nhất

* Do yi là nhãn do đó chỉ có thể = 1 hoặc –1, do đó (3) có thể viết thành

* Tại đây, ta chỉ cần xét 1 điểm xi gần Hyperplane nhất là đã có thể tính được B, nhưng thay vào đó ta sẽ tính trung bình của tất cả các điểm gần Hyperplane nhất, điều này là để tránh sai số trong quá trình tính toán, N như ta đã nói, là số điểm ứng với λi dương, chính là số điểm gần Hyperplane nhất, gọi S là tập chứa Index của các điểm này, Y’ là Y nhưng chỉ chọn những phần tử có Index trong S, tương tự D’

* Công thức trên chính là công thức mà ta đã dùng để tính B

1. Soft SVM (Soft Support Vector Machine)?

* Ý tưởng như SVM, nhưng áp dụng cho dữ liệu mà không tồn tại mặt phân cách hoàn toàn 2 nhóm điểm, tức 2 nhóm điểm đan xen vào nhau
* Ban đầu, ta có 1 ma trận D kích thước m x n, các cột trong nó chính là các điểm trong không gian m chiều, và mỗi điểm đều được gắn 1 trong 2 nhãn có giá trị là 1 hoặc –1, Vector Y n chiều gồm n nhãn ứng với n điểm
* Bài toán đặt ra là tìm phương trình của 1 Hyperplane m – 1 chiều đặt giữa 2 nhóm điểm này sao cho hợp lí nhất
* Bước 1, tính ma trận V kích thước m x n = cách nhân mỗi cột, hay mỗi điểm của D với phần tử tương ứng trong Y
* Bước 2, tính ma trận K là ma trận bán xác định dương cấp n

* Bước 3, giải bài toán tối ưu thuộc kiểu quy hoặc bậc 2 sau
* Hàm cần tìm giá trị nhỏ nhất là

* λ là Vector n chiều ứng với n điểm trong D
* 1 kí hiệu cho Vector chỉ toàn 1, trong biểu thức trên thì nó có số chiều = n
* Các điều kiện là

* C là Vector n chiều với các phần tử có cùng 1 giá trị là 1 số dương do ta tự đặt
* Bước 4, tính pháp tuyến W của Hyperplane cần tìm, nó là Vector m chiều
* Ban đầu lấy mỗi cột, hay mỗi điểm trong V nhân cho giá trị tương ứng trong λ, sau đó cộng tất cả các cột lại ta được Vector W
* Bước 5, tính Offset B của Hyperplane cần tìm, nó là giá trị vô hướng
* Gọi số phần tử > 0 và < C trong λ là N, ban đầu tính Vector R n chiều sau

* Tiếp theo tính B bằng cách lấy tổng tất cả phần tử trong R ứng với giá trị trong λ vừa > 0 vừa < C, rồi cuối cùng lấy tổng này chia M được B
* Bước 6, phương trình của Hyperplane là

* Bước 7, dự đoán nhãn của dữ liệu mới xi bằng cách xét dấu của

* Chứng minh cách làm trên
* Giả sử ta đã tìm được Hyperplane cần tìm, gọi nó là H, khi này ta định nghĩa vùng không an toàn của H là tập hợp tất cả các điểm xi trong không gian sao cho |WTxi + B| ≤ 1, tức là lân cận Hyperplane, vùng không gian còn lại gọi là vùng an toàn
* Ta định nghĩa sự hinh sinh ứng với mỗi điểm xi trong D là si, nếu xi nằm trong vùng an toàn và đúng phía, thì độ hi sinh = 0, đúng phía tức là ví dụ 1 điểm có nhãn = 1, nằm về phía dương Hyperplane, thì gọi là đúng phía, còn nếu nhãn = 1 và nằm về phía âm Hyperplane thì gọi là ngược phía, nếu nằm trong vùng không an toàn hoặc nằm ngược phía thì nó được tính bằng công thức sau, yi là nhãn tương ứng

* Tức là các điểm nằm đúng phía và nằm trong vùng không an toàn sẽ có độ hi sinh nằm khoảng từ 0 tới 1, còn các điểm nằm ngược phía thì có độ hi sinh từ 1 trở lên
* Ở đây, ta muốn hạn chế số điểm nằm ngược phía, hay nằm trong vùng không an toàn nhất có thể, đồng thời cũng làm tối đa margin của Hyperplane, ta có thể đạt được tất cả điều này bằng cách tìm tối thiểu của hàm sau

* Dễ thấy hàm này giống như trong SVM thường, chỉ có thêm 1 số hạng là tổng sự hi sinh nhân cho 1 hằng số C dương do ta tự đặt
* Nếu C cực lớn, thì hàm sẽ tập trung tối thiểu tổng sự hi sinh, và nếu tối thiểu đến mức không còn điểm nào hi sinh, thì ta có nghiệm gần với bài toán SVM thường
* Nếu C nhỏ, thì tổng sự hi sinh sẽ ảnh hưởng không lớn tới f(W, s), do đó hàm sẽ tập trung tối đa margin của Hyperplane thay vì tổng sự hi sinh, và làm cho tổng sự hi sinh lớn, cho phép các điểm được nằm ngược phía với Hyperplane
* Ta dễ dàng suy ra được các điều kiện ràng buộc sau

* Đến đây, ta có thể giải ngay bài toán quy hoạch bậc 2 này tại đây, đây là bài toán quy hoạch bậc 2 là vì thứ nhất, hàm tối ưu là tổng của hàm lồi + hàm tuyến tính cũng là lồi, nến hàm tối ưu cũng lồi, các điều thì dễ thấy chúng là các hàm Affine, do đó cũng lồi, thứ 2 là hàm tối ưu có dạng bậc 2, tuy nhiên, ta sẽ giải nó bằng hàm đối ngẫu vì số lượng ẩn số sẽ được giảm đi rất nhiều
* Ta luôn có thể chọn các độ hi sinh cho mỗi điểm là 1 con số cực lớn, đủ để

* Điều trên nghĩa là tiêu chuẩn Slater đã được thỏa mãn, và do đó đối ngẫu mạnh xảy ra, ta sẽ tiếp tục xây dựng hàm Lagrange của bài toán

* Trong đó s, λ, μ là các Vector n chiều
* Ta có hàm đối ngẫu là

* Do hàm này trích xuất giá trị nhỏ nhất, hay những cực tiểu của hàm Lagrange, nên ta có thể tìm biểu thức cụ thể của nó = cách tính đạo hàm = 0 theo W, B, s của hàm Lagrange

* Dễ thấy công thức trên là công thức mà ta đã dùng để tính W

* Thế (1), (2) và (3) vào hàm đối ngẫu, ta dễ dàng suy ra được kết quả sau, bằng cách sử dụng lại kết quả có được như ở SVM thường

* Ở đây, dữ kiện (3) cho ta thấy 0 ≤ μ ≤ C, do λ ≥ 0, điều này cũng có nghĩa là

0 ≤ λ ≤ C

* Như vậy ta có thể lập được bài toán đối ngẫu sau
* Tối đa hàm sau

* Với các điều kiện

* Dễ thấy bài toán này tương đương bài toán tối ưu ở bước 3
* Do bài toán gốc là lồi và thỏa mãn tiêu chuẩn Slater, do đó nghiệm tối ưu của nó cũng thỏa mãn hệ điều kiện KKT, 2 trong các điều kiện đó là

* Xét các điểm có giá trị λi tương ứng > 0 và < C, khi này từ (3), ta suy ra μi > 0 và < C, kết hợp với (5), ta suy ra si = 0, do λi khác 0, từ (4), ta suy ra

* Như vậy chỉ cần biết tọa độ 1 điểm có giá trị λi tương ứng > 0 và < C, ta có thể suy trực tiếp ra B, nhưng để tránh sai số, ta sẽ lấy trung bình tất cả điểm này
* N như ta đã nói, là số điểm ứng với λi vừa > 0 vừa < C, chính là số điểm nằm trên biên của margin của Hyperplane, gọi S là tập chứa Index của các điểm này, Y’ là Y nhưng chỉ chọn những phần tử có Index trong S, tương tự D’

* Công thức trên chính là công thức mà ta đã dùng để tính B

1. Kernel SVM (Kernel Vector Machine)?

* Ý tưởng như Soft SVM, nhưng áp dụng cho dữ liệu mà nó đã quá xáo trộn vào nhau, và cần làm cách nào đó để đưa các dữ liệu này lên chiều không gian lớn hơn để dễ phân tách
* Ban đầu, ta có 1 ma trận D kích thước m x n, các cột trong nó chính là các điểm trong không gian m chiều, và mỗi điểm đều được gắn 1 trong 2 nhãn có giá trị là 1 hoặc –1, Vector Y n chiều gồm n nhãn ứng với n điểm
* Bài toán đặt ra là tìm phương trình của 1 Hyperplane m – 1 chiều đặt giữa 2 nhóm điểm đã được đưa lên chiều không gian lớn hơn sao cho hợp lí nhất
* Cho hàm k(x, y) ánh xạ mỗi cặp điểm trong không gian m chiều tới 1 số thực, hàm này ta tự chọn
* Bước 1, tính ma trận K là ma trận bán xác định dương cấp n, với phần tử hàng i cột j = k(xi, xj)yiyj, với xi và xj lần lượt là điểm thứ i và j trong D, yi và yj lần lượt là nhãn của chúng
* Bước 2, tối thiểu hàm sau

* Với các điều kiện

* Dễ thấy bài toán trên y chang như ở Soft SVM, giải tương tự
* Bước 3, gọi tập S chứa các Index của các điểm ứng với giá trị λ > 0 và < C, số phần tử của S là N, tìm nhãn của dữ liệu xm mới bằng công thức sau

* Các hàm k(x, y) phải là nhân xác định dương, thì khi này K mới là ma trận bán xác định dương, khi đó hàm Obj(λ) mới lồi
* Các hàm k(x, y) thường dùng
* Linear

* Polynomial

* Ví dụ

* Radial Basis Function

* Ví dụ

* Sigmoid

Special Matrix – Ma Trận Đặc Biệt:

1. Ma Trận Rỗng (Empty Matrix)?

* Là ma trận vuông cấp 0, nghĩa là không có hàng và không có cột, nên không chứa phần tử nào hết
* Định thức của ma trận rỗng = 1

1. Ma Trận Không?

* Là ma trận sao cho toàn bộ phần tử = 0

1. Ma Trận Đường Chéo (Diagonal Matrix)?

* Là ma trận mà mọi phần tử không nằm trên đường chéo chính đều = 0
* Ví dụ

1. Ma Trận Thưa Thớt (Sparse Matrix)?

* Là ma trận mà bạn cảm thấy số phần tử = 0 hơi nhiều

1. Ma Trận Dày Đặc (Dense Matrix)?

* Là ma trận mà bạn cảm thấy số phần tử = 0 hơi ít

1. Độ Rộng Dải (Bandwidth)?

* Cho ma trận A bất kì, khi này độ rộng dải dưới (Lower Bandwidth) của A là số k nhỏ nhất để tất cả đường chéo dưới Offset n, với n từ k + 1 trở lên, đều chứa toàn 0, tương tự độ rộng dải trên (Upper Bandwidth) của A là số p nhỏ nhất để tất cả đường chéo trên Offset m, với m từ p + 1 trở lên, đều chứa toàn 0

1. Ma Trận Dải (Band Matrix)?

* Là ma trận mà bạn cảm thấy độ rộng dải trên và dưới hơi bé

1. Ma Trận Song Đường Chéo (Bidiagonal Matrix)?

* Là ma trận có độ rộng dải dưới = 1 và độ rộng dải trên = 0, hoặc ngược lại độ rộng dải dưới = 0 và độ rộng dải trên = 1

1. Ma Trận Tam Đường Chéo (Tridiagonal Matrix)?

* Là ma trận có độ rộng dải dưới = độ rộng dải trên = 1

1. Ma Trận Mở Rộng (Augmented Matrix)?

* Là ma trận nhận được khi ghép 2 ma trận cùng số dòng lại với nhau theo chiều ngang
* Ví dụ

* Ma trận mở rộng của A và B là

1. Ma Trận Suy Biến (Singular Matrix)?

* Là ma trận vuông có định thức = 0
* Tính chất

Không khả nghịch

Hệ phương trình thuần nhất có các hệ số của ma trận suy biến thì sẽ có vô số nghiệm

* Ví dụ

1. Ma Trận Không Suy Biến (Non Singular Matrix)?

* Là ma trận vuông có định thức khác 0
* Tính chất

Khả nghịch

Hệ phương trình thuần nhất có các hệ số của ma trận không suy biến thì sẽ có nghiệm duy nhất là x = y = z = … = 0

* Ví dụ

1. Ma Trận Lũy Linh (Nilpotent Matrix)?

* Là ma trận vuông sao cho tồn tại số nguyên dương K nhỏ nhất để lũy thừa bậc K của ma trận này = 0, khi này ta nói ma trận này là ma trận lũy linh bậc K
* Mọi ma trận lũy linh đều là ma trận suy biến, nhưng không phải ma trận suy biến nào cũng là ma trận lũy linh
* Mọi ma trận tam giác chặt đều là ma trận lũy linh
* Ví dụ

* Như vậy A là ma trận lũy linh bậc 2
* Các mệnh đề sau tương đương

M là ma trận lũy linh cấp n

Đa thức đặc trưng của M là xn

M có Eigen Value duy nhất là 0

Mk = 0, k ≤ n

1. Ma Trận Khiếm Khuyết (Defective Matrix)?

* Là ma trận vuông sao cho tồn tại 1 Eigen Value có bội hình học < bội đại số

1. Ma Trận Tam Giác (Triangular Matrix)?

* Ma trận tam giác trên (Upper Triangular Matrix) là ma trận vuông mà mọi phần tử ở phía dưới đường chéo chính đều = 0
* Ví dụ

* Nếu mọi phần tử ở phía trên đường chéo chính đều = 0 thì gọi là ma trận tam giác dưới (Lower Triangular Matrix)
* Tích 2 ma trận tam giác trên cũng là 1 ma trận tam giác trên, tương tự ma trận tam giác dưới
* Tất cả phần tử phân biệt trên đường chéo chính của ma trận tam giác bất kì đều là Eigen Value của nó, số lần xuất hiện = bội đại số
* Ma trận tam giác chặt là ma trận tam giác mà các phần tử trên đường chéo chính = 0

1. Ma Trận Sơ Cấp (Elementary Matrix)?

* Là ma trận E đạt được = cách áp dụng 1 lần phép khử Gauss bất kì trên ma trận Identity
* Ví dụ

* Thực hiện nhân hàng 2 với 3, ta được 1 ma trận sơ cấp

* Cho ma trận A kích thước m x n, khi này giả sử ta thực hiện 1 lần phép khử Gauss nào đó trên 1 hàng nào đó của A, thì kết quả tương đương với việc lấy 1 ma trận sơ cấp cấp m ứng với phép khử Gauss đó nhân với A
* Ví dụ

* Sử dụng phép khử Gauss thay hàng 2 = hàng 3 + 0.5 \* hàng 2, ta được

* Cũng lấy ma trận I 3 x 3, và thực hiện phép khử Gauss tương tự, thay hàng 2 = hàng 3 + 0.5 \* hàng 2, ta được ma trận sơ cấp tương ứng

* Lấy ma trận trên nhân với A

* Tương tự, nếu ta thực hiện 1 lần phép khử Gauss nào đó trên 1 cột nào đó của A, thì kết quả tương đương với việc lấy A nhân với 1 ma trận sơ cấp cấp n ứng với phép khử Gauss đó
* Ví dụ

* Sử dụng phép khử Gauss thay cột 1 = cột 3 + cột 2 + cột 1, ta được

* Cũng lấy ma trận I 3 x 3, và thực hiện phép khử Gauss tương tự, thay cột 1 = cột 3 + cột 2 + cột 1, ta được ma trận sơ cấp tương ứng

* Lấy A nhân ma trận trên

1. Ma Trận Hermitian (Hermitian Matrix, Self Adjoint Matrix)?

* Là ma trận vuông A trong trường số phức bằng chính chuyển vị liên hợp phức của nó

* Cho ma trận A kích thước bất kì trong trường số phức, ta có các định lý

AA\* luôn là ma trận Hermitian

Nếu A là ma trận vuông, thì A + A\* luôn là ma trận Hermitian

* Cho ma trận Hermitian A ctracấp n bất kì, ta có các định lý

Mọi Eigen Value của A đều là số thực

A không khiếm khuyết, do đó nó có đủ n Eigen Vector độc lập tuyến tính

Các phần tử trên đường chéo chính của A đều là số thực

Mọi cặp Eigen Vector độc lập tuyến tính và ứng 2 Eigen Value khác nhau của A đều vuông góc với nhau

Ta luôn có thể xây dựng 1 bộ n Eigen Vector đơn vị đôi một vuông góc với nhau từ A

Ma trận không là ma trận Hermitian duy nhất có mọi Eigen Value = 0

* Chứng minh
* Gọi Eigen Value của A là λ, 1 Eigen Vector của λ là v, ta luôn có

* Cho x và y là 2 Eigen Vector độc lập tuyến tính bất kì của A, a và b là Eigen Value tương ứng của x và y, a khác b, ta có

1. Ma Trận Đối Xứng (Symmetric Matrix)?

* Là ma trận vuông A trong trường số thực, bằng chính chuyển vị của nó
* Cho A là ma trận đối xứng bất kì, ta có các định lý sau

A là 1 ma trận Hermitian, nên có đầy đủ tính chất của nó

1. Ma Trận Đơn Nhất (Unitary Matrix)?

* Là ma trận vuông A trong trường số phức thỏa mãn

* Cho A là ma trận đơn nhất cấp n bất kì, B là ma trận vuông bất kì trong trường số phức, ta có các định lý

A không khiếm khuyết, nghĩa là nó có đủ n Eigen Vector độc lập tuyến tính trên trường số phức

Các Eigen Value của A, kể cả phức, luôn có Module = 1

Mọi cặp không gian con riêng của A ứng với 2 Eigen Value khác nhau đều vuông góc với nhau

B là ma trận đơn nhất khi và chỉ khi các cột của B đều là Vector có độ dài = 1 và vuông góc với nhau từng đôi một

B là ma trận đơn nhất khi và chỉ khi các hàng của B đều là Vector có độ dài = 1 và vuông góc với nhau từng đôi một

1. Ma Trận Lũy Đẳng (Idempotent Matrix, Projection Matrix)?

* Là ma trận vuông A trong trường số phức thỏa mãn

* Tưởng tượng A ép không gian V vào 1 không gian con của nó theo 1 hướng nào đó không nhất thiết phải vuông góc
* Cho A là ma trận lũy đẳng, ta có các tính chất sau

A không khiếm khuyết, nghĩa là nó có đủ n Eigen Vector độc lập tuyến tính trên trường số phức

Mỗi Eigen Value của A chỉ có thể là 0 hoặc 1

A không suy biến khi và chỉ khi A là ma trận Identity

Vết của A chính bằng hạng của nó

1. Ma Trận Chiếu Vuông Góc (Orthogonal Projection Matrix)?

* Là ma trận vuông A vừa là ma trận Hermitian vừa là ma trận lũy đẳng
* Tưởng tượng A ép không gian V vào 1 không gian con của nó theo hướng vuông góc với không gian con đó

1. Ma Trận Chiếu Xiên (Oblique Projection Matrix)?

* Là bất kì ma trận lũy đẳng A nào không phải là ma trận Hermitian
* Tưởng tượng A ép không gian V vào 1 không gian con của nó theo chiều xiên góc

1. Ma Trận Hội Tụ (Convergent Matrix)?

* Là ma trận vuông A trong trường số phức thỏa mãn khi A lũy thừa đến vô cùng thì Module của mỗi phần tử trong A tiến dần tới 0, nghĩa là A dần tới ma trận không

* Cho A là ma trận vuông bất kì trong trường số phức, ta có các tính chất sau

A là ma trận hội tụ khi và chỉ khi bán kính phổ của nó < 1

1. Ma Trận Bán Hội Tụ (Semi Convergent Matrix)?

* Là ma trận vuông A trong trường số phức thỏa mãn khi A lũy thừa đến vô cùng thì mỗi phần tử của A tiến dần tới 1 giá trị phức hữu hạn nào đấy
* Cho A là ma trận vuông A bất kì trong trường số phức, ta có các tính chất sau

A là ma trận bán hội tụ khi và chỉ khi bán kính phổ của nó ≤ 1, và nếu tồn tại Eigen Value với Module = 1, thì nó phải = 1, đồng thời bội đại số của Eigen Value này phải = bội hình học của nó

Ma trận nào hội tụ thì cũng bán hội tụ

1. Ma Trận Trực Giao (Orthogonal Matrix)?

* Là ma trận vuông A trong trường số thực thỏa mãn

* Cho A là ma trận trực giao bất kì, ta có các định lý

A là ma trận đơn nhất, nên có đầy đủ tính chất của nó

Bản chất của A là ma trận quay có Flip

1. Ma Trận Hadamard (Hadamard Matrix)?

* Là ma trận vuông A thỏa mãn mỗi phần tử của nó phải là 1 hoặc –1, đồng thời các hàng phải vuông góc với nhau từng đôi một
* Cho A là ma trận Hadamard cấp n bất kì, I là ma trận Identity cấp n, ta có các tính chất sau

n phải = 1, 2, hoặc bội nguyên dương của 4

Mỗi cặp hàng bất kì của nó, số phần tử trùng nhau bằng đúng n / 2, trùng nhau nghĩa là cùng = 1 hoặc cùng = –1

Mỗi cặp cột bất kì của nó, số phần tử trùng nhau bằng đúng n / 2

1. Ma Trận Xác Định (Definite Matrix)

* Một ma trận A xác định thì chắc chắn nó là ma trận Hermitian, kích thước n x n, tuy nhiên không phải ma trận Hermitian nào cũng xác định
* Được gọi là bán xác định dương (Positive Semi Definite), nếu thỏa mãn

* Được gọi là toàn phương xác định dương (Positive Definite), nếu thỏa mãn

* Một ma trận toàn phương xác định dương thì cũng là bán xác định dương
* Được gọi là bán xác định âm (Negative Semi Definite), nếu thỏa mãn

* Được gọi là toàn phương xác định âm (Negative Definite), nếu thỏa mãn

* Một ma trận toàn phương xác định âm thì cũng là bán xác định âm
* Hàm f(x) = x\*Ax được gọi là 1 dạng bậc 2 (Quadratic Form), là 1 ánh xạ từ tọa độ x tới 1 số thực
* Ta có các định lý sau

A là ma trận toàn phương xác định dương khi và chỉ khi nó là ma trận Hermitian và mọi Eigen Value của nó > 0

A là ma trận bán xác định dương khi và chỉ khi nó là ma trận Hermitian và mọi Eigen Value của nó đều ≥ 0

A là ma trận toàn phương xác định âm khi và chỉ khi nó là ma trận Hermitian và mọi Eigen Value của nó đều < 0

A là ma trận bán xác định âm khi và chỉ khi nó là ma trận Hermitian và mọi Eigen Value của nó đều ≤ 0

A là ma trận Hermitian không xác định (Indefinite) khi và chỉ khi nó có cả Eigen Value âm và Eigen Value dương

Ma trận không là ma trận duy nhất vừa bán xác định âm vừa bán xác định dương

Nếu A là ma trận kích thước bất kì trong trường số phức, thì AA\* luôn là ma trận bán xác định dương

* Chứng minh
* Cho A là ma trận đường chéo cấp n bất kì trên trường số thực, các phần tử trên đường chéo chính của nó là λ1, λ2, …, λn, cho x = (x1, x2, …, xn) là 1 Vector bất kì, khi này ta có

* Rõ ràng, nếu tất cả λ đều dương, thì (1) sẽ dương với mọi Vector x khác Vector không, và do đó, A toàn phương xác định dương, nếu tất cả λ đều không âm, thì rõ ràng (1) không thể âm, và do đó, A bán xác định dương
* Tương tự, nếu tất cả λ đều âm, thì (1) sẽ âm với mọi Vector x khác Vector không, và do đó, A toàn phương xác định âm, nếu tất cả λ đều không dương, thì rõ ràng (1) không thể dương, và do đó, A bán xác định âm
* Như vậy ta đã chứng minh xong chiều thuận
* Không mất tính tổng quát, giả sử λ1 không dương, thì rõ ràng tồn tại 1 Vector x, sao cho x1 khác 0, còn lại 0 hết, để (1) không dương, và do đó, A không toàn phương xác định dương, nên, nếu A toàn phương xác định dương, thì tất cả λ phải dương,
* Tương tự dùng lập luận trên cho bán xác định dương, toàn phương xác định âm, bán xác định âm, ta chứng minh được chiều nghịch
* Mở rộng ra, cho A là ma trận Hermitian cấp n bất kì, do đó ta luôn có thể chọn 1 bộ Eigen Vector đơn vị của A sao cho chúng đôi một vuông góc với nhau, gọi ma trận chứa các Eigen Vector này là Q ma trận chứa Eigen Value tương ứng là Λ, rõ ràng Q là ma trận đơn nhất, ta có

* Dễ thấy một x khác biệt sẽ cho ra một y khác biệt do Q chỉ là ma trận quay rồi Flip, do đó, để với mọi x, x\*Ax dương, thì với mọi y, y\*Ay phải dương, nghĩa là, để A toàn phương xác định dương, thì Λ phải toàn phương xác định dương, và như ta đã chứng minh, để Λ toàn phương xác định dương, thì mọi phần tử trên đường chéo của nó phải dương, hay mọi Eigen Value của A phải dương
* Lặp lại suy luận trên cho bán xác định dương, toàn phương xác định âm, bán xác định âm, ta có điều phải chứng minh

1. Ma Trận Chuẩn (Normal Matrix)?

* Là ma trận vuông A trong trường số thực, nó được gọi là ma trận chuẩn khi và chỉ khi

* Ta có các định lý

Mọi ma trận Hermitian đều là ma trận chuẩn

Mọi ma trận đơn nhất đều là ma trận chuẩn

1. Ma Trận Hoán Vị (Permutation Matrix)?

* Là ma trận A nhận được khi hoán vị hàng hoặc cột của 1 ma trận Identity
* Dùng để hoán vị tọa độ của 1 điểm
* Ví dụ

* Vì ma trận hoán vị cũng là ma trận trực giao nên ta có

* Để xây dựng 1 ma trận hoán vị từ 1 phép hoán vị, thực hiện các bước sau
* Bước 1, biểu diễn phép hoán vị dưới dạng 1 hoán vị của dãy số 1, 2, 3, …, gọi hoán vị này là P
* Ví dụ, phép hoán vị tọa độ (a, b, c, d) thành (d, a, c, b) tương đương phép hoán vị 1234 thành P = 4132
* Bước 2, tạo ma trận không A cấp n, n là phần tử hoán vị
* Bước 3, lặp qua các phần tử của P từ trái sang, đồng thời lặp qua các hàng của A từ trên xuống, phần tử tương ứng của P có giá trị nào thì ô thứ đó trong hàng sẽ được đánh = 1
* Ví dụ P = 4132, thì trong hàng đầu tiên của A, phần tử thứ 4 sẽ được đánh 1, trong hàng thứ 2, phần tử thứ nhất được đánh 1, …

1. Ma Trận Tự Nghịch Đảo (Involutary Matrix)?

* Là ma trận vuông bằng chính nghịch đảo của nó
* Cho A là ma trận tự nghịch đảo bất kì cấp n, ta có các định lý

A không khiếm khuyết, nghĩa là nó có đủ n Eigen Vector

A chỉ có Eigen Value là 1 hoặc –1

Định thức của A chỉ có thể là 1 hoặc –1

An = A nếu n là số nguyên dương lẻ, hoặc = ma trận Identity nếu n là số nguyên dương chẵn

Matrix Form – Dạng Của Ma Trận:

1. Dạng (Form) Của Ma Trậnì?

* Cho ma trận A, dùng các phép biến đổi Gauss để tạo ra ma trận B, khi đó B là 1 dạng của A, nói cách hàng A và B tương đương hàng (Row Equivalent)

1. Tương Đương Hàng (Row Equivalent)?

* 2 ma trận gọi là tương đương hàng nếu có thể dùng các phép biến đổi Gauss để chuyển đổi qua lại giữa 2 ma trận
* Tập hợp tất cả các ma trận tương đương hàng tạo thành 1 lớp

1. Dạng Hàng Bậc Thang (Row Echelon Form) Và Dạng Hàng Bậc Thang Rút Gọn (Row Reduced Echelon Form, Row Canonical Form)?

* Xét dạng hàng bậc thang
* Từ trên xuống, xét mỗi hàng, phần tử đầu tiên của hàng khác 0 gọi là phần tử cơ sở, vị trí của số này gọi là vị trí cơ sở (Pivot Position), biến ứng với nó gọi là biến cơ sở (Leading Variable), và cột chứa nó là cột cơ sở (Pivot Column)
* Cột sơ sở ở hàng tiếp theo phải ở bên phải cột cơ sở của hàng hiện tại
* Các biến không phải là biến cơ sở của bất kì dòng nào gọi là biến tự do

(Free Variable)

* Nếu tất cả cột cơ sở đều là One Hot Vector thì là dạng hàng bậc thang rút gọn
* Mỗi ma trận sẽ có nhiều dạng hàng bậc thang nhưng chỉ có duy nhất 1 dạng hàng bậc thang rút gọn
* Ma trận hàng bậc thang rút gọn còn được gọi là ma trận chính tắc của tất cả ma trận tương đương hàng với nó
* Ví dụ

* Dạng hàng bậc thang rút gọn của M là

* Để ý thấy, nếu sắp xếp lại các cột của dạng hàng bậc thang rút gọn, ta luôn có dạng

* I là ma trận Identity
* F là ma trận có cùng số hàng với I
* 0 là ma trận 0
* Từ công thức trên, ta dễ dàng suy ra được ma trận Null Space, nói cách khác là ma trận mà các cột của nó đều là các nghiệm đặc biệt của hệ thuần nhất ứng với ma trận đang xét đã được sắp xếp cột

* I’ cũng là ma trận Identity, khác I
* Ví dụ

Complex Matrix – Ma Trận Phức:

1. Ma Trận Liên Hợp Phức (Conjugate Matrix)?

* Cho ma trận A bất kì trong trường số phức, lấy liên hợp các phần tử của A được ma trận liên hợp phức của A
* Ví dụ

1. Chuyển Vị Liên Hợp Phức (Conjugate Transpose, Hermitian Transpose)?

* Cho ma trận A bất kì trong trường số phức, lấy ma trận liên hợp phức của A rồi chuyển vị nó ta được ma trận chuyển vị liên hợp phức của A, kí hiệu A\*
* Ví dụ

1. Thương Số Rayleigh?

* Cho ma trận Hermitian A cấp n và Vector x, khi đó, thương số Rayleigh của A và x là

* Bản chất của thương số Rayleigh chính là trung bình có trọng số của tất cả Eigen Value
* Chứng minh
* Vì A là ma trận Hermitian nên có các Eigen Vector độc lập tuyến tính vuông góc với nhau từng đôi một, chọn tất cả các Eigen Vector độc lập tuyến tính với độ dài = 1 làm thành cột của ma trận chuyển đổi Mod của A, gọi là Q, rõ ràng Q là 1 ma trận trực giao, ta có

* Λ là ma trận đường chéo vuông chứa λ là Eigen Value của A

* Rõ ràng đây chính là trung bình có trọng số của tất cả Eigen Value, trọng số là bình phương tọa độ x ứng với Basis là Q
* Dễ thấy nếu x là Eigen Vector thì thương số Rayleigh sẽ là Eigen Value tương ứng
* Đồng thời, do trọng số là bình phương nên chắc chắn dương, do đó Min và Max của thương số Rayleigh lần lượt là Eigen Value nhỏ nhất và Eigen Value lớn nhất

Vector Space – Không Gian Vector:

1. Không Gian Vector?

* Là 1 tập hợp không rỗng V kết hợp với 2 toán tử là phép cộng giữa 2 phần tử trong V, kí hiệu là “+” và phép nhân giữa 1 phần tử trong F với 1 phần tử trong V để cho ra 1 phần tử trong V, kí hiệu là av, a thuộc F, v thuộc V, đồng thời thỏa mãn các tính chất sau, giả sử Identity của F ứng với toán tử cộng là e1, ứng với toán tử nhân là e2, của (V, +) là e3, phần tử đối xứng với e2 trong phép cộng kí hiệu là –e2, phần tử đối xứng với phần tử v nào đó trong V kí hiệu là –v

(V, +) phải là nhóm giao hoán

* Hệ quả

* Phần tử trong V gọi là Vector

1. Không Gian Tích Trong (Inner Product Space)?

* Là 1 không gian Vector V trên trường F là trường số thực R hoặc số phức C kèm theo 1 toán tử tích trong, toán tử này áp dụng lên 2 Vector trong V để cho ra 1 phần tử trong F, kí hiệu là 〈u,v〉, u và v là 2 Vector trong V, đồng thời với mọi Vector u, v, w trong V và với mọi phần tử a, b trong F, chúng phải thỏa mãn toàn bộ các tính chất sau, lưu ý 1 dấu gạch trên đầu là liên hợp phức, 2 dấu gạch trên đầu là lấy phần thực, Identity của của (V, +) là e

* Hệ quả

* Chiều dài của Vector v trong gian tích trong V được định nghĩa bằng biểu thức

* Khoảng cách giữa 2 Vector u và v trong không gian tích trong V được định nghĩa bằng biểu thức sau, phần tử đối xứng v trong V kí hiệu là –v

* 2 Vector u và v trong không gian tích trong V được gọi là vuông góc khi tích trong của chúng = 0
* Vector u vuông góc với tập hợp Vector A trong V chỉ khi u vuông góc với từng phần tử của A
* Tập hợp Vector M gọi là 1 họ trực giao của V khi mỗi cặp Vector trong M đều vuông góc với nhau
* M gọi là họ trực chuẩn khi tất cả Vector trong M đều có chiều dài = 1
* Tích vô hướng như ta đã biết ở cấp 2 gọi là tích trong chính tắc hay tích vô hướng chính tắc

1. Không Gian Vector Euclidean (Euclidean Vector Space)?

* Là không gian tích trong trên trường số thực R với số chiều hữu hạn
* Góc giữa 2 Vector u và v trong 1 không gian Vector Euclidean được định nghĩa bằng biểu thức sau

1. Không Gian Hilbert (Hilbert Space)?

* Là 1 không gian Vector vừa là không gian tích trong vừa là không gian Metric hoàn thiện, với Metric = khoảng cách giữa 2 Vector được định nghĩa bởi tích trong

1. Không Gian Affine (Affine Space)?

* Là 1 tập S kèm theo 1 không gian Vector V, sao cho thỏa mãn các điều kiện sau, 0 kí hiệu cho Identity của phép cộng trong V, phép cộng giữa 1 phần tử S với 1 phần tử trong V sẽ ra 1 phần tử trong S

Với mọi a thuộc S, hàm f(v) = a + v là 1 song ánh

* Cho dễ hình dung, tưởng tượng các phần tử trong S là các điểm, và các Vector trong V dùng để tịnh tiến các điểm này tới 1 điểm khác

1. Tổ Hợp Tuyến Tính (Linear Combination)?

* Cho không gian Vector V trên trường F, cho A = [v1, v2, …, vn] là 1 dãy các Vector nào đó trong V, các Vector có thể trùng nhau, cho B = [a1, a2, …, an] là 1 dãy các phần tử nào đó trong F, các phần tử có thể trùng nhau, khi này tổ hợp tuyến tính v của A với B là

* Nếu tất cả phần tử trong B đều = Identity của F ứng với phép cộng, thì tổ hợp tuyến tính trên gọi là tổ hợp tuyến tính tầm thường (Trivial Linear Combination)
* Còn nếu chỉ cần chứa 1 phần tử khác Identity, thì gọi là tổ hợp tuyến tính không tầm thường (Non Trivial Linear Combination)
* Nếu tồn tại B sao cho tổ hợp trên là không tầm thường và cho ra v = Identity của (V, +) thì ta nói A phụ thuộc tuyến tính (Linear Dependent)
* Ngược lại, ta nói A độc lập tuyến tính (Linear Independent)
* Nếu A là 1 tập hợp, tức là chứa các phần tử khác nhau, thì tất cả tổ hợp tuyến của A với B bất kì gọi là họ Vector A hay Span của A, kí hiệu là Span(A) hoặc L(A), nếu A rỗng thì Span(A) chỉ chứa phần tử Identity của (V, +), kí hiệu {0}
* Hạng của Span(A) = số tối đại các Vector độc lập tuyến tính của A, nghĩa là số chiều không gian nó Span
* A được gọi là tập sinh (Spanning Set) của V nếu mọi phần tử trong V = tổ hợp tuyến tính nào đó của A, tập sinh không nhất thiết phải độc lập tuyến tính
* Nếu A chứa m Vector, có thể phụ thuộc tuyến tính hoặc độc lập tuyến tính, B chứa nhiều hơn m Vector mà mỗi Vector trong B đều là tổ hợp tuyến tính của A, thì B phụ thuộc tuyến tính
* Cho ma trận A chứa m cột, các cột có thể phụ thuộc hoặc độc lập tuyến tính, nếu m cột của A là tập sinh của không gian V thì với mọi ma trận vuông B cấp m, tích AB cũng là tập sinh của V, và ngược lại, nếu tích AB là tập sinh của V, với B độc lập tuyến tính, thì A cũng là tập sinh của B
* Cho ma trận A chứa m cột, các cột có thể phụ thuộc hoặc độc lập tuyến tính, nếu m cột của A là tập sinh của không gian V, và cho ma trận B suy biến cấp m, thì tích AB có thể hoặc không là tập sinh của V, ngược lại, nếu tích AB là tập sinh của V, với B phụ thuộc tuyến tính, thì A chắc chắn là tập sinh của B, nhưng nếu AB không là tập sinh của V, với B phụ thuộc tuyến tính, thì A có thể hoặc không là tập sinh của V

1. Tổ Hợp Affine (Affine Combination)?

* Là tổ hợp tuyến tính mà trong đó tổng các hệ số = 1

1. Không Gian Đối Ngẫu Vector (Algebraic Dual Space)?

* Cho không gian Vector V trên trường F, khi này không gian Vector D có các phần tử Vector là các hàm ánh xạ tuyến tính từ V sang F, với phép cộng Vector = cộng 2 hàm và phép nhân 1 số với 1 Vector = nhân 1 số trong F với 1 hàm trong D, ví dụ f(x) = x + 3, g(x) = x + 5 thì (f + g)(x) = 2x + 8, và 3 \* f(x) = 3x + 9
* Dễ thấy không gian đối ngẫu có số chiều = số chiều của không gian gốc, ví dụ xét không gian 3D, tất cả các phần tử trong không gian đối ngẫu của không gian 3D sẽ là hàm có dạng ax + by + cz, với a, b, c là các hằng số, và x, y, z là 3 tọa độ của 1 Vector trong không gian 3D, dễ thấy ta có thể biểu diễn hàm này dưới dạng Vector (a, b, c), và rõ ràng nó có 3 chiều
* Các hàm trong không gian đối ngẫu còn gọi là Linear Form, Covector hoặc One Form

1. Không Gian Con (Subspace)?

* Là tập con của 1 không gian Vector V nào đó và đồng thời cũng là 1 không gian Vector
* Không gian con này được gọi là bình thường (Proper) nếu nó không phải là V và được gọi là tầm thường (Trivial) nếu nó = {0}
* Cho 2 không gian con nào đó là A và B
* Giao của 2 không gian con A ∩ B = không gian con khác bé hơn hoặc bằng 2 không gian con đầu
* Lấy tổng 2 phần tử bất kì tương ứng trong 2 không gian con, ta sẽ Span 1 không gian con khác lớn hơn hoặc bằng 2 không gian con đầu = A + B
* Số chiều Span bởi A + B = số chiều Span bởi A + số chiều Span bởi B – số chiều Span bởi A ∩ B
* Tập hợp tất cả Vector vuông góc với không gian con A trong V gọi là bù vuông góc của A trong V, ví dụ đường thẳng vuông góc mặt phẳng

* Số chiều của A + số chiều của bù vuông góc của A = số chiều của V
* Bù vuông góc cũng là 1 không gian con trong V
* Cho 1 Vector v bất kì trong V, cho không gian con A và bù vuông góc của A là B, khi này tồn tại duy nhất cặp Vector f, g, f thuộc A, g thuộc B, sao cho f + g = v, khi này f được gọi là hình chiếu vuông góc của v xuống A, kí hiệu f = prA(v) , và g được gọi là hình chiếu vuông góc của v xuống B, kí hiệu prB(v)
* Khoảng cách từ giữa v và A = chiều dài của g, tương tự, khoảng cách giữa v và B = chiều dài của f
* Ví dụ, A là 1 đường thẳng qua gốc tọa độ trong không gian 3D, B là mặt phẳng vuông góc A, cũng qua gốc tọa độ, cho 1 điểm M đâu đó, khi này dễ thấy M = Vector hình chiếu của M lên A + Vector hình chiếu của M lên B

1. Không Gian Con Bất Biến (Invariant Subspace)?

* Cho ánh xạ f từ không gian Vector V tới chính nó, khi này 1 không gian con bất biến của V ứng với f là 1 không gian con S của V, sao cho mọi phần tử trong nó sau khi bị f áp dụng, trở thành 1 phần tử cũng trong S
* V và không gian con tầm thường của V đều gọi là không gian con bất biến tầm thường của V ứng với f

1. Cơ Sở (Basis)?

* B được gọi là 1 cơ sở của không gian Vector V nếu nó là tập sinh của V và độc lập tuyến tính
* 1 không gian Vector có thể có nhiều Basis, và tất cả các Basis này đều có cùng số phần tử, gọi là số chiều của không gian Vector (Dimension) = dim(V)
* Những cơ sở có cùng phần tử nhưng khác thứ tự thì khác nhau
* B gọi là hữu hạn nếu số chiều nó Span là hữu hạn

1. Cơ Sở Chính Tắc (Standard Basis)?

* Còn gọi là cơ sở sắp thứ tự, là cơ sở chứa toàn One Hot Vector
* Ví dụ
* Cơ sở chính tắc của không gian tọa độ thực

1. Biểu Diễn Của 1 Vector Ứng Với 1 Cơ Sở?

* Cho Vector v, cơ sở B, biểu diễn của v ứng với B là 1 Vector mà các phần tử của nó là tọa độ của v ứng với B
* Kí hiệu

1. Cách Chuyển Hệ Tọa Độ?

* Cho Vector v trong không gian tọa độ thực, v đang có tọa độ ứng với cơ sở chính tắc, ta muốn biểu diễn tọa độ của v theo cơ sở khác thì dùng công thức

* B là ma trận có các cột là cơ sở ta muốn biểu diễn tọa độ theo, gọi là ma trận chuyển cơ sở từ cơ sở chính tắc tới cơ sở B
* Ví dụ

* Cho cơ sở E và cơ sở F, ma trận chuyển cơ sở từ E sang F hay từ F về E là

1. Cách Tạo Cơ Sở Vuông Góc Bằng Quá Trình Gram Schmidt?

* Ban đầu ta có hệ Vector v1, v2, …, vn, và ta muốn tạo ra hệ Vector khác sao cho mỗi Vector trong đó vuông góc với nhau từng đôi một, hệ này là u1, u2, …, un
* Đầu tiên u1 = v1, tiếp theo, u2 sẽ là cái Vector đồng phẳng với u1 và v2, đồng thời vuông góc u1, u2 đạt được = cách trừ v2 đi 1 lượng u1 sao cho nó vừa đủ vuông góc với u1

* Tiếp tục u3 sẽ đạt được = cách trừ v3 đi 1 lượng u2 và u1 sao cho nó vừa đủ vuông góc với cả u2 và u1

* Tổng quát hóa, un sẽ được tính như sau

* Hệ Vector u tạo thành 1 hệ vuông góc, đồng thời có Span = Span của hệ Vector cũ v, trong quá trình này, nếu un tính được = 0, thì un phụ thuộc tuyến tính vào những Vector trước đó, nên quá trình Gram Schmidt còn được dùng để Check xem hệ Vector có độc lập tuyến tính không
* Sau khi có được hệ Vector u, ta sẽ chuẩn hóa nó để độ dài mỗi Vector = 1, gọi ma trận Q có các cột là Vector u đã được chuẩn hóa, khi đó, do Span của Q = Span của hệ Vector v, nên ma trận chiếu vuông góc vào Q cũng chính là ma trận chiếu vuông góc vào Span của hệ Vector v, dùng công thức ma trận chiếu vuông góc, ta có ma trận chiếu là

* Như vậy, quá trình Gram Schmidt còn được dùng để thiết lập ma trận chiếu vuông góc vào hệ Vector ban đầu

1. Một Số Không Gian Vector?

* Không gian Vector tọa độ thực n chiều, mọi không gian Vector đều có thể quy về không gian này bằng cách chuyển cơ sở

* Không gian Vector với Vector là các đa thức bậc n theo biến x trên trường số thực

* Không gian Vector với Vector là các ma trận m x n với m khác n trên trường F

* Ví dụ các ma trận có kích thước 3 x 2 trên trường số thực

* Trường hợp ma trận vuông cấp n kí hiệu như sau

Norm:

1. Semi Norm?

* Cho 1 không gian Vector V trên 1 trường con F của trường số phức, khi này hàm f ánh xạ V vào tập số thực được gọi là 1 Semi Norm trên V khi và chỉ khi nó thỏa mãn những điều kiện sau, |s| nghĩa là trị tuyệt đối của s

* Hệ quả

1. Norm?

* Là Semi Norm f trong không gian Vector V nhưng có thêm điều kiện sau, 0 in đậm là Vector không

* Để ý thấy tích vô hướng tới chính nó trong không gian thực là 1 Norm
* Thay vì kí hiệu f(x), ta kí hiệu Norm như sau

1. Absolute Value Norm?

* Cho x là Vector phức 1 chiều, tức là x vô hướng

1. P Norm (Lp Norm)?

* Cho x là 1 Vector phức n chiều

1. Taxicab Norm (Manhattan Norm, L1 Norm)?

* Là P Norm với p = 1
* Cho x là Vector phức n chiều

1. Infinity Norm (Maximum Norm, L∞ Norm, Uniform Norm, Supremum Norm)?

* Là P Norm với p = vô cùng
* Cho x là Vector phức n chiều, khi này Infinity Norm của nó = Module của phần tử có Module lớn nhất trong nó

1. Euclidean Norm (Quadratic Norm, L2 Norm, Norm 2, 2 Norm, Square Norm)?

* Là P Norm với p = 2
* Cho x là Vector phức n chiều

* Cho A là ma trận kích thước m x n, ta có các công thức sau

1. Zero Norm (L0 Norm)?

* Đây không phải là 1 Norm, nó đếm số phần tử khác 0 trong 1 Vector x phức n chiều

1. Matrix Norm?

* Có 3 loại Matrix Norm, 1 là coi ma trận như 1 Vector, các Norm kiểu này gọi là Entry Wise Norm, 2 là coi ma trận như dãy các Vector, các Norm kiểu này gọi là Matrix Norm tạo ra từ Vector Norm, 3 là chỉ xét tới các Singular Value của ma trận, và xếp chúng vào 1 Vector rồi tính bằng Vector Norm, các Norm kiểu này gọi là Schatten Norm
* Xét kiểu 2, giả sử Vector Norm là f, áp dụng lên không gian Vector V, thì Matrix Norm tạo ra từ f cũng sẽ chỉ được áp dụng lên các ma trận có cột là Vector trong V, cho A là ma trận đang nói đến, x là Vector trong V, ta có Matrix Norm của A ứng với f là

1. Lp,q Norm?

* Là 1 Entry Wise Norm
* Cho ma trận A kích thước m x n trong trường số phức, aij là phần tử hàng i cột j

1. Frobenius Norm (Hilbert Schmidt Norm)?

* Là Lp,q Norm khi p = q = 2
* Cho ma trận A kích thước m x n trong trường số phức, aij là phần tử hàng i cột j, s là các Singular Value của A

1. Max Norm?

* Là Lp,q Norm khi p = q = vô cùng
* Cho ma trận A kích thước m x n trong trường số phức, aij là phần tử hàng i cột j, khi này Max Norm của nó = Module của phần tử có Module lớn nhất trong nó

1. Spectral Norm (Matrix Norm Tạo Ra Từ Euclidean Norm, Norm 2)?

* Cho ma trận A kích thước m x n trong trường số phức, s là các Singular Value của A

* Ta có các định lý sau

1. Matrix Norm Tạo Ra Từ P Norm?

* Cho A là ma trận kích thước m x n trong trường số phức, x là Vector phức n chiều

* Ta có các công thức ứng với p khác nhau sau

1. Schatten P Norm?

* Là Schatten Norm ứng với P Norm
* Cho ma trận A bất kì trong trường số phức, s là các Singular Value của A

1. Nuclear Norm (Trace Norm)?

* Là Schatten P Norm với p = 1
* Cho ma trận A bất kì trong trường số phức, s là các Singular Value của A

Block Matrix – Ma Trận Khối:

1. Ma Trận Khối?

* Là 1 ma trận A mà mỗi phần tử của nó là 1 ma trận, gọi là khối, sao cho tất cả khối trong cùng 1 hàng của A thì có cùng số hàng, và tất cả khối trong cùng 1 cột của A thì có cùng số cột
* Ví dụ

* Trong đó

* Khai triển A

1. Nhân 2 Ma Trận Khối?

* Tương đương nhân ma trận thông thường, với phần tử là các khối, lưu ý thứ tự nhân các khối và các khối phải tương thích
* Nhân ma trận khối A cho ma trận khối B rồi khai triển, tương đương nhân A khai triển cho B khai triển
* Ví dụ

1. Ma Trận Khối Đường Chéo (Block Diagonal Matrix)?

* Là ma trận khối sao cho thỏa mãn các điều kiện sau

Là ma trận vuông

Các phần tử trên đường chéo chính là ma trận vuông, các phần tử còn lại đều là ma trận không

* Ví dụ

* Với

* Khai triển A

1. Ma Trận Jordan?

* Là ma trận khối đường chéo sao cho các phần tử ma trận trên đường chéo chính, còn gọi là các khối Jordan, thỏa mãn các điều kiện sau

Toàn bộ phần tử trên đường chéo chính của nó có giá trị bằng nhau, giá trị này gọi là Eigen Value của khối Jordan

Toàn bộ phần tử trên đường chéo trên của nó có giá trị = 1

* Ví dụ

* Với

* Khai triển A

1. Tính Mũ Của Ma Trận Jordan?

* Cho ma trận Jordan J, khi này, Jn tương đương với việc ta lấy từng khối Jordan trong J mũ lên, với số mũ là n
* Mũ n của 1 khối Jordan cấp m được tính như sau, λ là Eigen Value của khối này
* Bước 1, Reset khối Jordan về ma trận không
* Bước 2, thay các phần tử trên đường chéo trên Offset k thành biểu thức sau, k chạy từ 0 tới m – 1 hoặc chạy từ 0 tới n nếu m – 1 > n

* Ví dụ 1

* Ví dụ 2

1. Jordan Decomposition?

* Cho 1 ma trận vuông A cấp n bất kì với các phần tử thuộc trường số phức, khi này ta luôn có thể tìm ra ma trận P và ma trận J, J còn được gọi là dạng Jordan bình thường (Jordan Normal Form, Jordan Canonical Form) của A, sao cho

* Bước 1, tìm các tất cả Eigen Value phân biệt của A, kí hiệu λ1, λ2, …, λm, cùng với các bội đại số tương ứng, kí hiệu a1, a2, …, am
* Bước 2, kẻ bảng sau

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | λ1 | λ2 | λ3 | … | λm |
| … |  |  |  |  |  |
| Bậc 3 |  |  |  |  |  |
| Bậc 2 |  |  |  |  |  |
| Bậc 1 |  |  |  |  |  |

* Ở ô thuộc bậc k, cột λi, ta điền D dấu chấm, căn lề trái, với D là số chiều của không gian con riêng bậc k của λ
* Điền các ô từ bậc 1 lên trên, dừng lại khi số chấm trong cột λi = ai
* Bước 3, lặp qua tất cả các dấu chấm ở hàng bậc 1, ứng với mỗi dấu chấm, tạo 1 khối Jordan cấp w, với w là số chấm trên đường thẳng đứng đi qua chấm hiện tại, Eigen Value của nó = cột λi chứa dấu chấm, từ các khối Jordan này, tạo một ma trận Jordan tương ứng, khai triển nó được J
* Bước 4, tạo tập S rỗng, lặp qua tất cả các dấu chấm ở hàng bậc 1, với mỗi dấu chấm, ta kẻ 1 đường thẳng đứng đi qua dấu chấm đó, sau đó tạo tập R rỗng, rồi lặp qua tất cả dấu chấm trên đường thẳng này từ trên xuống, với dấu chấm đầu tiên, giả sử nó bậc k, cột λi, chèn 1 Vector khác 0 thuộc không gian con riêng bậc k của λi vào R, sao cho Vector này tạo với các Vector ứng với các dấu chấm bên trái trong cùng ô 1 họ Vector độc lập tuyến tính, gọi là Vector v, với các dấu chấm tiếp theo trên đường hiện tại, lấy ma trận A – λiI nhân với Vector được chèn vào R trước đó, rồi chèn Vector kết quả vào bên trái R, sau khi lặp qua hết dấu chấm trên đường, chèn R vào bên phải S
* Bước 5, các Vector trong S tạo thành ma trận P
* Ví dụ 1

* Bước 1
* Đa thức đặc trưng của A là

* Dễ thấy phương trình đặc trưng của A chỉ có duy nhất 1 nghiệm λ1 = 1, với bội đại số a1 = 5
* Bước 2
* Tìm số chiều của của không gian con riêng bậc 1 của λ1

* Null Space của A1 có số chiều = 2, điền 2 chấm vào ô bậc 1 cột λ1
* Tìm số chiều của không gian con riêng bậc 2 của λ1

* Null Space của A2 có số chiều = 4, điền 2 chấm vào ô bậc 2 cột λ1, để tổng số chấm = 4
* Tìm số chiều của không gian con riêng bậc 3 của λ1

* Null Space của A3 có số chiều = 5, điền 1 chấm vào ô bậc 3 cột λ1, để tổng số chấm = 5, ta dừng lại ở đây



* Bước 3, ma trận J tạo ra từ bảng trên là

* Bước 4
* Với chấm trên cùng ở đường màu đỏ bên trái, Vector v3 phải thuộc Null Space của (A – λ1I)3 nhưng không thuộc Null Space của (A – λ1I)2, từ cơ sở đã tính của các Null Space này, ta dễ dàng chọn

* Với chấm tiếp theo ngay bên dưới ở bậc 2, Vector v2 của nó là

* Với chấm tiếp theo ngay bên dưới ở bậc 1, Vector v1 của nó là

* Với chấm trên cùng ở đường màu đỏ bên phải, Vector u2 của nó phải thuộc Null Space của (A – λ1I)2 nhưng không thuộc Null Space của A – λ1I, đồng thời phải độc lập tuyến tính với v2, từ cơ sở đã tính của các Null Space này, ta dễ dàng chọn

* Với chấm tiếp theo ngay bên dưới ở bậc 1, Vector u1 của nó là

* Như vậy ta có ma trận P là

* Ví dụ 2

* Bước 1
* Đa thức đặc trưng của A là

* Dễ thấy phương trình đặc trưng của A chỉ có duy nhất 2 nghiệm λ1 = 2, với bội đại số a1 = 3, và λ2 = 1, với bội đại số a2 = 2
* Bước 2
* Tìm số chiều của của không gian con riêng bậc 1 của λ1

* Null Space của A1 có số chiều = 2, điền 2 chấm vào ô bậc 1 cột λ1
* Tìm số chiều của không gian con riêng bậc 2 của λ1

* Null Space của A2 có số chiều = 3, điền 1 chấm vào ô bậc 2 cột λ1, để tổng số chấm = 3, dừng lại ở ở đây, chuyển sang λ2
* Tìm số chiều của của không gian con riêng bậc 1 của λ2

* Null Space của A3 có số chiều = 2, điền 2 chấm vào ô bậc 1 cột λ2, dừng lại ở đây



* Bước 3, ma trận J tạo ra từ bảng trên là

* Bước 4
* Với chấm trên cùng ở đường màu đỏ thứ nhất từ trái sang, Vector v2 phải thuộc Null Space của (A – λ1I)2 nhưng không thuộc Null Space của A – λ1I, từ cơ sở đã tính của các Null Space này, ta dễ dàng chọn

* Với chấm tiếp theo ngay bên dưới ở bậc 1, Vector v1 của nó là

* Với chấm ở đường màu đỏ thứ 2, Vector u1 của nó phải thuộc Null Space của

A – λ1I, đồng thời phải độc lập tuyến tính với v1, từ cơ sở đã tính của các Null Space này, ta dễ dàng chọn

* Với chấm ở đường màu đỏ thứ 3, Vector w1 của nó phải thuộc Null Space của

A – λ2I, từ cơ sở đã tính của các Null Space này, ta có thể chọn

* Với chấm ở đường màu đỏ thứ 4, Vector t1 của nó phải thuộc Null Space của

A – λ2I, đồng thời phải độc lập tuyến tính với w1, từ cơ sở đã tính của các Null Space này, ta có thể chọn

* Như vậy ta có ma trận P là

Product – Tích:

1. Tích Hadamard (Hadamard Product)?

* Là phép nhân từng phần tử của ma trận này với phần tử tương ứng của ma trận kia
* Ví dụ

1. Tích Kronecker (Kronecker Product)?

* Tích Kronecker giữa Tensor A và Tensor B là phép thay thế mỗi phần tử C của A bằng 1 Tensor có giá trị = CB
* Ví dụ

1. Tích Ngoài (Outer Product) Của 2 Tensor?

* Giả sử Tensor A có Shape là (7, 8, 9), Tensor B có Shape là (10, 11, 12, 13, 15), gọi C là tích ngoài của A với B, ta có

* a, b, c, e, f, g, h, i là các Index
* Ví dụ

1. Tích Boolean (Boolean Product)?

* Cho A là ma trận chỉ có giá trị 0 hoặc 1, B cũng là ma trận chỉ có giá trị 0 hoặc 1, khi này tích Boolean của A với B = tích ma trận của A với B bị giới hạn trên, nghĩa là nếu có giá trị nào > 1 thì nó sẽ trở thành 1
* Ví dụ

1. Tensor Nén (Tensor Contraction)?

* Ví dụ Tensor A có Shape là (7, 8, 9), Tensor B có Shape là (10, 11, 12, 8, 15), gọi C là Tensor Nén của A với B, chiều nén tương ứng là chiều thứ 2 của A và thứ 4 của B, ta có

* a, c, e, f, g, i là các Index

1. Tích Nêm (Wedge Product) Bản Chất Là Gì?

* Tích nêm của 2 Vector u và v sẽ tạo 1 Vector ở chiều không gian khác gọi là không gian ngoài bình phương (Exterior Square) mà khi lấy Module của nó sẽ cho ra diện tích hình bình hành tạo bởi 2 Vector ấy, kí hiệu u ∧ v
* Vì diện tích của hình bình hành tạo bởi 2 Vector trùng nhau = 0 nên

* Hơn nữa, kéo dài 1 Vector bao nhiêu lần thì diện tích cũng tăng bấy nhiêu lần

* Tưởng tượng u, v, w tạo thành hình hộp chữ nhật để chứng minh tính chất phân phối của tích nêm với trường hợp hình hộp chữ nhật, rồi dùng ma trận biến đổi thành hình hộp bình hành bất kì để áp dụng cho u, v, w bất kì, ta được tính chất phân phối tổng quát

* Từ 2 tính chất trên, ta có

* Tính chất kết hợp của tích nêm

* Tích nêm của n Vector trong không gian m chiều sẽ có số phần tử là tổ hợp chập n của m, và Basis có dạng tích nêm của n Vector đơn vị

1. Tích Có Hướng Có Phải Tích Nêm?

* Không, mặc dù giá trị các phần tử giống nhau, nhưng tích có hướng tạo Vector có Basis là i, j, k, …, còn tích nêm tạo Vector có Basis là i ∧ j, i ∧ k, j ∧ k, …

1. Ví Dụ Tính Tích Nêm?

* Tìm diện tích hình bình hành tạo ra từ 2 Vector trong không gian 4 chiều sau, gọi các Vector đơn vị của không gian này là i, j, k, l

* Vector trong chiều không gian ngoài bình phương có Module là diện tích của hình bình hành này là

* Diện tích của hình bình hành này là

1. Mẹo Tính Tích Nêm?

* Giả sử ta muốn tính A = x1 ∧ x2 ∧ x3 ∧ … ∧ xn, đây đều là những Vector trong không gian m chiều, n không vượt quá m, xếp các Vector này thành ma trận sau

* Để tính hệ số của 1 Basis bất kì của A, ta xác định Basis đó là tích nêm của những Vector đơn vị nào, lưu ý các Vector đơn vị này sắp xếp theo thứ tự tăng dần, sau đó giữ lại các cột tương ứng với những Vector đơn vị này, loại bỏ các cột còn lại, rồi tính định thức của ma trận nhận được, ta được hệ số

Gaussian Elimination – Phép Khử Gauss:

1. Phép Khử Gauss Trên 1 Ma Trận?

* Là 1 trong 2 thao tác sau
* Thế 1 hàng/cột = tổ hợp tuyến tính của tất cả các hàng/cột, trong đó hệ số của hàng/cột được thế phải khác 0
* Tráo các hàng/cột

1. Phép Khử Gauss Làm Biến Đổi Và Không Làm Biến Đổi Những Tính Chất Nào Của Ma Trận?

* Xét trường hợp chỉ khử hàng
* Trong suốt quá trình khử, không gian hàng và Null Space của ma trận không thay đổi, nhưng không gian cột và Left Null Space có thể thay đổi
* Đồng thời, các cột trong ma trận ban đầu có vị trí là vị trí các cột cơ sở ở dạng hàng bậc thang rút gọn cuối cùng sẽ tạo thành cơ sở của không gian cột của ma trận ban đầu
* Xét trường hợp chỉ khử cột
* Y chang khử hàng, đổi chữ “hàng” thành “cột”, “Null Space” thành “Left Null Space” và ngược lại
* Nếu sử dụng cả khử hàng và khử cột cùng lúc thì có thể làm biến đổi mọi tính chất của ma trận

1. Dùng Phép Khử Gauss Để Tìm Nghịch Đảo Của Ma Trận?

* Giả sử A là ma trận vuông cấp m, I là ma trận Identity cấp m, để tìm nghịch đảo của A, ta thực hiện phép khử Gauss theo hàng trên ma trận mở rộng (A | I) để được dạng hàng bậc thang rút gọn F, nhìn vào F, nếu nó có hạng < m, thì kết luận A suy biến, nếu có hạng bằng m, thì F có dạng (I | B), khi đó B chính là nghịch đảo của A
* Ví dụ

* Chứng minh
* Vì khi dùng phép khử Gauss theo hàng, Null Space của (A | I) không thay đổi, hay ma trận Null Space của (A | I) y chang của (I | B), dễ thấy ma trận này là

* Như đã nói ở trên, vì đây cũng là ma trận Null Space của (A | I), nên

* Vậy B chính là nghịch đảo của A
* Trong toàn bộ quá trình biến đổi để tìm nghịch đảo

Số phép nhân, chia được thực hiện = m3

Số phép cộng, trừ được thực hiện = m3 – 2m2 + m

1. Dùng Phép Khử Gauss Để Tính Định Thức?

* Cho ma trận vuông A, cho a là 1 số bất kì, lấy 1 hàng trong A + a lần 1 hàng khác, định thức của A sẽ không thay đổi, đồng thời tất cả Eigen Value của A cũng không thay đổi, giống như hình hộp bình hành tạo ra từ 3 Vector, bạn tịnh tiến đầu của 1 Vector nào đó theo 1 Vector khác thì rõ ràng thể tích của hình hộp bình hành không đổi
* Sử dụng phép biến đổi trên, chỉ cần ta đưa A về dạng ma trận tam giác trên, rồi lấy tích tất cả phần tử trên đường chéo chính của ma trận đó thì được định thức của A

Nếu nhân 1 hàng/cột với k thì định thức sẽ tăng k lần

Nếu hoán vị 2 hàng/cột, định thức đổi dấu

1. Trình Bày Phép Khử Gauss Theo Hàng Để Giải Hệ Phương Trình?

* Ví dụ
* Nhân hàng 1 với 4

* Cộng –1 lần hàng 1 vào hàng 2 và –2 lần hàng 1 vào hàng 3

* Tráo hàng 2 với hàng 3

* Giải từ dưới lên

1. Biểu Diễn Nghiệm Của Hệ Phương Trình Vô Số Nghiệm?

* Ví dụ

1. Giải Hệ Phương Trình Dưới Dạng Ma Trận?

* Dùng ma trận mở rộng rồi thực hiện phép khử Gauss theo hàng để đưa ma trận về dạng hàng bậc thang
* Ví dụ
* Hệ phương trình ban đầu

* Biến đổi hệ phương trình dưới dạng ma trận mở rộng

1. Biểu Diễn Tập Nghiệm Vô Hạn Dưới Dạng Ma Trận Và Vector?

* Ví dụ

* Dạng ma trận của tập nghiệm

* Dạng Vector của tập nghiệm

* Dễ thấy tập nghiệm này Span 1 mặt phẳng trong không gian 4 chiều

1. Hệ Thuần Nhất?

* Là hệ phương trình mà tất cả hệ số tự do trong tất cả phương trình đều = 0
* Ví dụ

* Xét hệ thuần nhất dạng tổng quát

* A là 1 ma trận hình chữ nhật
* x là 1 Vector dọc, là ẩn cần tìm
* Giả sử có vô số nghiệm, khi đó tập hợp tất cả các nghiệm thỏa mãn phương trình trên sao cho chỉ có duy nhất 1 biến tự do có giá trị = 1, còn các biến tự do còn lại có giá trị = 0, gọi là các nghiệm đặc biệt (Special Solution)
* Tổ hợp tuyến tính của tất cả các nghiệm đặc biệt tạo thành Null Space hay tập nghiệm của phương trình này
* Hệ thuần nhất luôn có nghiệm = tất cả ẩn = 0, nghiệm này gọi là nghiệm tầm thường (Trivial Solution)

1. Mặt Phẳng Đa Chiều (Hyperplane)?

* Trong không gian n chiều, cho k Vector không đồng phẳng, k <= n, chọn 1 điểm cố định rồi từ điểm này Span k Vector kia, được mặt phẳng k chiều
* Ví dụ
* Đường thẳng là mặt phẳng 1 chiều

* Mặt phẳng 2 chiều trong không gian 4 chiều

1. Giải Pháp Bình Phương Tối Thiểu (Least Squares Solution) Cho Phương Trình Tuyến Tính Vô Nghiệm?

* Cho A là 1 ma trận chữ nhật bất kì, B là 1 Vector nào đó, tìm Vector X để

* Giả sử phương trình này vô nghiệm, vậy thì làm sao ta tìm ra X để AX gần với B nhất, để ý thấy, khi X chịu tác dụng của A, nó sẽ bị ánh xạ tới 1 điểm đâu đó trong không gian cột của A, gọi điểm này là X’, như vậy bài toán đưa về tìm X để khoảng cách X’B ngắn nhất, vâng X’ chính là hình chiếu của B lên không gian cột của A, kết hợp công thức ma trận chiếu vuông góc P, ta được phương trình

* A-1 là chính là nghịch đảo bên trái của A
* Phương pháp này áp dụng cho cả trường hợp B và X là ma trận
* Trong trường hợp có thêm Bias Y, lưu ý Y ở đây chỉ gồm 1 hàng, và khi cộng sẽ được Broadcast

* Thì ta có thể đưa về dạng thuần nhất bằng cách thêm 1 cột chứa toàn 1 vào bên phải A và thêm 1 dòng dưới cùng của X, dòng này chính là Y
* Ví dụ 1, tìm giải pháp bình phương tối thiểu cho phương trình

* Đừng ngu mà vội tìm giải pháp bình phương tối thiểu cho bài này, bởi vì phương trình trên có vô số nghiệm, mà yêu cầu của ta phải là vô nghiệm1
* Ví dụ 2, tìm giải pháp bình phương tối thiểu cho phương trình

* Đưa nó về dạng thuần nhất

* Khi này, thử thế X và Y vào

1. Hệ Không Tương Thích (Inconsistent System) Và Hệ Tương Thích (Consistent System)?

* Hệ không tương thích là hệ phương trình vô nghiệm
* Hệ tương thích thì có 1 nghiệm hoặc vô số nghiệm

1. Xác Định Nhanh Số Nghiệm Của Hệ Phương Trình Sử Dụng Định Lý Kronecker Capelli (Rouche Capelli)?

* Cho A là ma trận hệ số của hệ phương trình, B là ma trận cột hệ số tự do, nếu có m phương trình với n ẩn thì A có kích thước m x n, B có kích thước m x 1
* Hệ có nghiệm khi và chỉ khi hạng của A là r(A) = hạng của ma trận bổ sung là r(A|B), khi này hệ có nghiệm duy nhất khi r(A) = n, và có vô số nghiệm khi

r(A) < n

* Hệ vô nghiệm khi và chỉ khi r(A) < r(A|B)

1. Quy Tắc Cramer?

* Cho A là ma trận vuông cấp n không suy biến, X và B là 2 ma trận n x m bất kì sao cho thỏa mãn AX = B
* Khi này, gọi C là ma trận con được tạo thành từ các hàng thứ i1, i2, …, ik và các cột thứ j1, j2, …, jk của X
* Ví dụ

* Dễ thấy C được tạo thành từ các hàng 2, 4, 6 và các cột 2, 4, 5 của X
* Gọi D là ma trận nhận được bằng cách lấy A rồi thay các cột thứ i1, i2, …, ik trong nó thành các cột thứ j1, j2, …, jk tương ứng trong B, tiếp tục ở ví dụ trên, ta sẽ thay cột 2, 4, 6 của A bằng các cột 2, 4, 5 của B
* Khi này, ta có đẳng thức

Model – Mô Hình:

1. Chuỗi Markov (Markov Chain)?

* Là 1 mô hình mô tả 1 chuỗi các sự kiện, mà tỉ lệ sự kiện tiếp theo xảy ra chỉ phụ thuộc vào trạng thái hiện tại, số trạng thái là hữu hạn
* Ví dụ



* Như trong hình, giả sử ta đang đứng ở clouds, thì đéo cần biết quá khứ ta đã làm gì, ta luôn có 30% tỉ lệ chuyển thành sun, 30% chuyển thành rain và 40% đéo biến đổi, hay khi đứng tại sun, đéo cần biết trước đó ta đứng ở đâu, ta luôn có 10% tỉ lệ chuyển thành rain ở bước tiếp theo, 50% tỉ lệ chuyển thành clouds và 40% tỉ lệ đéo biến đổi
* Mỗi bước chuyển trạng thái sẽ cách nhau 1 khoảng thời gian cố định

1. Ma Trận Ngẫu Nhiên (Stochastic Matrix)?

* Còn gọi là ma trận Markov
* Giả sử ta gieo vào vị trí các trạng thái trong chuỗi Markov một số lượng vật thể nhất định, thì bài toán đặt ra là khi thời gian chạy đến 1 lúc nào đó, số lượng vật thể ứng với mỗi trạng thái là bao nhiêu
* Cho X là Vector, mỗi phần tử của nó tương ứng là số lượng vật thể của trạng thái 1, 2, 3, …, lúc mới gieo vào, M là ma trận Markov ứng với chuỗi Markov này, Si là trạng thái thứ i
* Hàng đầu tiên của M có các giá trị lần lượt là tỉ lệ số vật thể ở S1 giữ nguyên trạng thái, tỉ lệ số vật thể ở S2 chuyển sang S1, tỉ lệ số vật thể ở S3 chuyển sang S1, …
* Hàng thứ 2 tương tự có các giá trị lần lượt là tỉ lệ số vật thể ở S1 chuyển sang S2, tỉ lệ số vật thể ở S2 giữ nguyên trạng thái, tỉ lệ số vật thể ở S3 chuyển sang S2, …
* …
* Cứ như vậy, 1 cột của M sẽ có tổng luôn = 1, do chứa các giá trị là tỉ lệ vật thể từ cột đó tới toàn bộ các cột
* Sau n bước, số vật thể ở mỗi trạng thái là

* Khi n dần tới vô tận, thì Mn dần hội tụ, và ta được trạng thái cân bằng

1. Mô Hình Leslie?

* Là mô hình cho cơ cấu dân số của 1 quần thể theo độ tuổi
* Giả sử quần thể được chia làm các lớp tuổi bằng nhau lần lượt là lớp tuổi 1, lớp tuổi 2, …, ví dụ từ 0 đến 5 tuổi là lớp tuổi 1, từ 5 đến 10 tuổi là lớp tuổi 2, …
* Gọi si và fi lần lượt là tỉ lệ sống sót và tỉ lệ sinh của các cá thể thuộc lớp tuổi i, khi sinh con thì con sẽ thuộc lớp tuổi 1, giả sử có n lớp tuổi, khoảng cách giữa các lớp là k năm, lớp tuổi cuối sau k nữa năm sẽ chết
* Ma trận Leslie ứng với quần thể này sẽ có kích thước n x n, hàng đầu tiên sẽ chứa các phần tử lần lượt là f1, f2, …, fn, khu vực (n – 1) x (n – 1) ở dưới hàng 1, tay trái, có dạng 1 ma trận đường chéo, với các phần tử trên đường chéo là s1, s2, …, sn – 1, phần còn lại chứa toàn 0, ví dụ

* Cho Vector X chứa các phần tử lần lượt là số cá thể ở lớp tuổi 1, 2, …, n thời điểm hiện tại, thì sau m \* k năm nữa, số cá thể của từng lớp tuổi là

* Nếu lớp tuổi cuối vẫn đéo chết và có tỉ lệ sống tiếp là sn, thì trong L, phần tử cuối cùng sẽ chuyển từ 0 thành sn

1. Mô hình Input Output?

* Giả sử ta có các sản phẩm x1, x2, …, xn, mỗi sản phẩm này có công thức chế tạo như sau

x1 = a11x1 + a21x2 + a31x3 + … + an1xn (1)

x2 = a12x1 + a22x2 + a32x3 + … + an2xn (2)

…

xn = a1nx1 + a2nx2 + a3nx3 + … + annxn (n)

* Trong đó aij = số nguyên liệu xi cần thiết để tạo ra 1 sản phẩm xj
* Giả sử chúng ta muỗn sản xuất ra b1 sản phẩm x1, b2 sản phẩm x2, …, bn sản phẩm xn, khi này số nguyên liệu cần thiết sẽ = b1 \* (1) + b2 \* (2) + … + bn \* (n), ta sẽ sử dụng ma trận sau để tính toán dễ hơn, gọi là ma trận đầu vào

* Dễ thấy cột 1 của A ứng với công thức cho x1, cột 2 cho x2, …
* Cho Vector B là Vector chỉ số lượng sản phẩm muốn tạo ra, hay Vector tổng số lượng sản phẩm tạo ra, hay mức sản lượng

* Khi này Vector chỉ số lượng nguyên liệu cần để sản xuất ra lượng sản phẩm B là

* Nghĩa là để sản xuất ra b1 sản phẩm x1, b2 sản phẩm x2, …, bn sản phẩm xn, thì cần c1 sản phẩm x1, c2 sản phẩm x2, …, cn sản phẩm xn
* aịj nhân cho bj gọi là số sản phẩm ngành i đã cung cấp cho ngành j, tổng số sản phẩm của tất cả các ngành đã cung cấp cho ngành j gọi là đầu vào của ngành j
* (bi – đầu vào ngành i) / đầu vào ngành i = tỉ suất lợi nhuận của ngành i
* Lưu ý ở đây ta phải trừ lượng sản phẩm sản xuất được cho lượng nguyên liệu thì mới được lượng sản phẩm lời, ví dụ dùng 3 cục cứt để tạo ra 7 cục cứt, khi này ta đã tạo ra thêm được 4 cục cứt, đây gọi là nhu cầu cuối cùng, hay cầu cuối, hay lượng sản phẩm dùng để tiêu dùng và xuất khẩu, hay ngắn gọn là lượng sản phẩm, là thứ có thể bán kiếm tiền, thay vì bán hết
* Nhu cầu cuối cùng do đó được tính bằng

* Đảo ngược lại bài toán, cho trước nhu cầu cuối cùng và công thức tạo ra sản phẩm, hỏi tổng số lượng sản phẩm sản xuất được, còn gọi là đầu ra, ta có