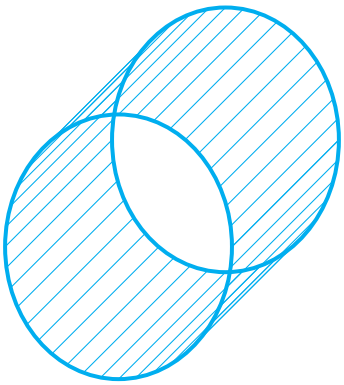
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG - HCM

Khoa Toán - Tin Học

BÁO CÁO CUỐI KỲ

XỬ LÝ ĐA CHIỀU



Khoa Toán - Tin học

Fac. of Math. & Computer Science

Giảng viên phụ trách: Ngô Minh Mẫn

TP.Hồ Chí Minh - Tháng 6, 2021

Nhóm sinh viên thực hiện

- Nguyễn Quốc Bảo - MSSV: 18110053
- Nguyễn Đức Vũ Duy - MSSV: 18110004
- Trần Tấn Phong - MSSV: 18110181

Mục lục

1 Autoencoder và PCA

1.1 Giới thiệu

Ta sử dụng dữ liệu Iris trên sklearn để so sánh giữa Autoencoder và PCA.Bộ dữ liệu Iris chứa bốn đặc điểm (chiều dài và chiều rộng của các lá đài và cánh hoa) của 150 mẫu của ba loài Iris (Iris setosa, Iris virginica và Iris versicolor).

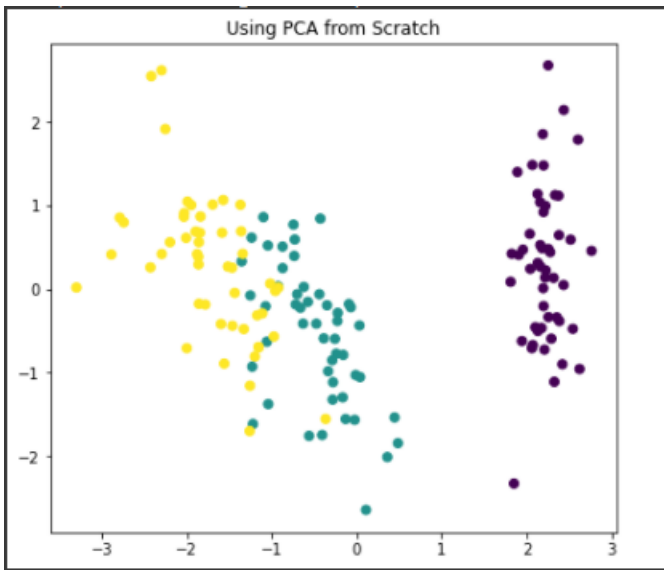
1.2 Thuật Toán PCA

Phép phân tích thành phần chính (Principal Components Analysis - PCA) là một thuật toán thống kê sử dụng phép biến đổi trực giao để biến đổi một tập hợp dữ liệu từ một không gian nhiều chiều sang một không gian mới ít chiều hơn (2 hoặc 3 chiều) nhằm tối ưu hóa việc thể hiện sự biến thiên của dữ liệu.

- Để thực hiện PCA, ta làm lần lượt các bước sau:

- Chuẩn hóa dữ liệu (d-chiều)
- Xây dựng ma trận hiệp phương sai từ dữ liệu đã được chuẩn hoá
- Tìm các vectơ riêng, trị riêng của ma trận hiệp phương sai
- Xây dựng tập trực chuẩn từ tập hợp các vecto trên
- Sắp xếp các trị riêng theo chiều giảm dần
- Chọn k vectơ riêng đầu tiên ứng với k trị riêng đầu tiên trong bộ trị riêng có thứ tự ở bước trên ($k < d$)
- Xây dựng ma trận chiếu $W \in M_{kxd}(R)$ từ các vectơ riêng trên
- Tìm hình chiếu Y của ma trận dữ liệu X trong không gian mới sinh bởi ma trận W, $Y = W^T X$

Bằng phương pháp PCA, ta thực trên tập dữ liệu Iris có 4 features và 150 samples, ta giảm số chiều của dữ liệu xuống còn 2 chiều nhưng vẫn giữ được thông tin của dữ liệu.



1.3 Autoencode để làm PCA

Autoencoders (AE) là mạng nơ-ron được đào tạo để học cách biểu diễn nén của dữ liệu đầu vào mà chúng được cung cấp. Như vậy ta bằng có thể thực hiện AE để nén dữ liệu giống với PCA.

AE có hai phần đối xứng tương ứng được gọi là bộ mã hóa (encode) và bộ giải mã (decode). Bộ mã hóa lấy các đầu vào ban đầu và nén chúng thành mã. Bộ giải mã cố gắng tạo lại các đầu vào ban đầu từ các mã này. AE được đào tạo bằng cách giảm thiểu lỗi tái tạo, tức là một số phép đo về sự khác biệt giữa các mẫu ban đầu và các mẫu được tái tạo.

Có hai điểm khác biệt chính giữa trình mã tự động và PCA:

- PCA là một phép biến đổi tuyến tính, trong khi bộ mã tự động, là một mạng nơ-ron, là một phép biến đổi phi tuyến tính.
- Các trục trong PCA được sắp xếp theo thứ tự sức mạnh đại diện của chúng, trong khi tiêu chuẩn AE, không có thứ tự như vậy

Phương pháp sau được đề xuất trong papper **A PCA-LIKE AUTOENCODER**.

Độ lớn của hiệp phương sai của mỗi thành phần không gian tiềm ẩn càng nhỏ càng tốt. Chúng ta có thể đạt được điều này bằng cách thêm một thuật ngữ bổ sung vào hàm loss của AE để phản ánh điều này. Giả sử thêm k thành phần vào không gian tiềm ẩn. Khi đó, ta có thể tính toán hiệp phương sai bằng cách sử dụng:

$$\sum_{i=1}^{k-1}\left[\frac{1}{M}\sum_{j=1}^{M}(z_i^{(j)}z_k^{(j)}) - \frac{1}{M^2}\sum_{j=1}^{M}\left(z_i^{(j)}\right)\sum_{j=1}^{M}\left(z_k^{(j)}\right)\right]$$

Trong đó, M là kích thước của bộ dữ liệu, $z^{(i)} = E(x^i)$.Tta tính được covariance term:

$$\mathcal{L}_{cov}(X) = \frac{1}{M}\sum_{i=1}^{k-1}\sum_{j=1}^{M} z_i^{(j)}z_k^{(j)}$$

Vì thế hàm loss tổng hợp là:

$$\mathcal{L}(X) = \frac{1}{M}\sum_{j=1}^{M}||x^{(i)} - D \circ E(x^{(i)})||_2^2 + \lambda\mathcal{L}_{cov}(X)$$

Trong đó, λ là tham số trọng số

2 Autoencoder và ICA

Trước khi đi vào Autoencoder, ta sẽ nhắc lại về ICA và FastICA. Model của ICA có dạng $x = As$ trong đó model của ICA thuộc dạng generative model. Model miêu tả một bộ dữ liệu quan sát được sẽ được tạo ra như thế nào từ một quá trình trộn lẫn các thành phân s_i. Các biến độc lập gọi là các biến latent, nghĩa là chúng không được quan sát. Model ICA chúng ta giả sử là s_i đều độc lập về mặt thống kê và các thành phần độc lập không được có phân phối chuẩn. Lí do việc các biến theo phân phối chuẩn là không sử dụng được là vì hàm mật độ của phân phối chuẩn có dạng đối xứng. Do đó, nó không chứa thông tin về hướng của các cột của ma trận trộn A. Do đó, A không thể được ước lượng được. Do đó, mục đích của ICA sẽ tập trung vào việc maximize tính không chuẩn của các tín hiệu. Phép đơ đầu tiên là sử dụng Kurtosis:

$$kurt(y) = E(y^4) - 3(E(y^2))^2$$

Tuy nhiên, điểm bất lợi của việc sử dụng Kurtosis là kurtosis rất nhạy cảm với điểm ngoại lai nên nó không phải là 1 phương pháp mạnh cho ICA. Lúc này, người ta tiếp tục nghĩ

đến việc sử dụng Negentropy. Entropy của một biến ngẫu nhiên được coi là độ thông tin mà quan sát của biến đó thu được. Biến càng ngẫu nhiên, khó đoán định thì có entropy càng cao. Một biến Gaussian sẽ có entropy lớn nhất trong các biến ngẫu nhiên có cùng phương sai. Do đó, phép đo Negentropy được định nghĩa là:

$$J(y) = H(y_{gauss}) - H(y)$$

Tuy nhiên, bất lợi của Negentropy chính là việc tính toán khó khăn. Nên người ta tìm cách xấp xỉ Negentropy như sau:

$$J(y) = \frac{1}{12}Ey^3 + \frac{1}{48}kurt(y)^2$$

Phương pháp trên lại không mạnh vì vẫn vấp phải kurtosis. Vậy nên, người ta lại nảy ra phương pháp xấp xỉ mới.

$$J(y) \approx \sum_{i=1}^{p} k_i[E\{G_i(y)\} - E\{G_i(v)\}]^2$$

Trong đó, k_i là một hằng số dương, v là một biến Gaussian có trung bình không và phương sai bằng 1. y cũng được giả định là có trung bình bằng 0 và phương sai bằng 1. Trong trường hợp, chúng ta chỉ sử dụng hàm nonquadratic G, phép xấp xỉ trên trở thành:

$$J(y)\alpha[E\{G(y)\} - E\{G(v)\}]^2$$

Qua quá trình thực nghiệm, thì người ta chứng minh được 2 hàm sau đây rất hữu dụng:

$$G_1(u) = \frac{1}{a_1}logcosh(a_1u), G_2(u) = -exp(-u^2/2), 1 \leq a_1 \leq 2$$

Đây cũng sẽ liên quan hàm loss cho thuật toán FastICA mà tôi sẽ đề cập sau đây:
Tiền xử lí dữ liệu, trước khi thực hiện các thuật toán ICA thì ta cần làm 2 bước là centering và whitening. Bước centering là ta sẽ lấy dữ liệu trừ đi véctơ trung bình của nó để x có trung bình bằng 0. Ở bước whitening, ta sẽ biến đổi tuyến tính véctơ x quan sát được để thu được một véctơ mới x' mà có các thành phần của nó đều không tương quan với nhau và có phương sai bằng unity. Nói một cách khác, ma trận hiệp phương sai của x bằng ma trận đơn vị. Biến đổi whitening luôn thực hiện được. Một phương pháp phổ biến của Whitening là sử dụng phân rã trị riêng của ma trận hiệp phương sai

$E\{xx^T\} = EDE^T$ trong đó E là ma trận trực giao của $E\{xx^T\}$ và D là ma trận đường chéo của các trị riêng. Lúc này whitening được thực hiện như sau:

$$x' = ED^{1/2}E^T x$$

Phương pháp whitening giúp giảm số lượng các tham số cần phải ước lượng. Thay vì phải ước lượng n^2 tham số thì giờ chỉ cần ước lượng một ma trận trộn trực giao mới A' nào đó có $n(n-1)/2$ bậc tự do. Sau khi hoàn thành xong 2 bước tiền xử lí dữ liệu thì, ta sẽ đi tới các bước chính của FastICA.

for 1 *to number of components* c :

$w_p \equiv$ *random initialisation*

while w_p *not* $<$ *threshold* :

$w_p \equiv \frac{1}{n}(Xg(W^T X) - g'(W^T X)W)$

$w_p \equiv w_p - \sum_{j=1}^{p-1}(w_p^T w_j)w_j$

$w_p \equiv w_p / ||w_p||$

$W \equiv [w_1, \ldots, w_c]$

Các tính chất của FastICA:

- Độ hội tụ là bậc 3. Tức là model sẽ hội tụ rất nhanh
- Không có kích thước các bước chạy phải chọn tuỳ chỉnh nên dễ sử dụng
- Thuật toán tìm trực tiếp các thành phần độc lập của bất kỳ phân phối không chuẩn nào sử dungj hàm phi tuyến tính G
- Độ hiệu quả của thuật toán có thể được tối ưu bằng cách chọn hàm phi tuyến tính g
- Các thành phần độc lập có thể được ước lượng 1 1, nghĩa là tương ứng với việc đang làm cách tìm phép chiếu (projection pursuit)
- Hữu dụng trong việc khám phá dữ liệu và giảm tải tính toán của phương pháp

2.1 Autoencoder để làm ICA

Phương pháp sau được đề xuất trong paper Nonlinear Extensions of Reconstruction ICA - Apaar Sadhwani and Apoorv Gupta - CS229 Project Report, Fall 2011. Theo đó, ICA

thường dùng sẽ có dạng bài toán tối ưu sau:

$$Minimize_W \sum_{i=1}^{m} \sum_{j=1}^{k} g(W_j x^{(i)})$$

Với điều kiện là: $WW^T = I$.

Trong khi đó, Bài toán RICA(Reconstruction ICA) mà mình sử dụng sẽ đi minimize hàm loss sau:

$$\frac{\lambda}{m} \sum_{j=1}^{m} ||W^T W x^{(i)}||_2^2 + \sum_{i=1}^{m} \sum_{j=1}^{k} g(W_j x^{(i)})$$

Trong đó, g là một hàm lồi phi tuyến tính. Ở đây là $g(.) = log(cosh(.))$. Hàm loss trên có thể viết lại thành:

$$J(W) = \frac{\lambda}{m} H(W) + G(W)$$

trong đó, $H(W) = (W^T W X - X)^T (W^T W X - X), G(W) = g(W_j X)$. Và các bước encoding và decoding đều sẽ là linear. Trong đó, encoding weights và decoding weights lần lượt là W, W^T. Ta coi model RICA này có tied-weights. Chi tiết code bằng pytorch sẽ có ở trong file colab.

Tôi sử dụng Adam trong thư viện optim của pytorch để tìm source signal ban đầu. Chi tiết về thuật toán Adam có thể được tìm thấy ở paper: ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION - Diederik P. Kingma* University of Amsterdam, OpenAI, dpkingma@openai.com, Jimmy Lei Ba -University of Toronto, jimmy@psi.utoronto.ca.

3 Autoencoder và Nonnegative Matrix Factorization

Non-negative matrix factorization (NMF) được cho là một phương pháp phân rã hữu hiệu cho dữ liệu nhiều chiều. Có 2 thuật toán cập nhật cho NMF đã được phân tích. Nó chỉ khác ở nhân tố cộng vào trong quy tắc update. Một thuật toán được dùng để minimize sai số bình phương nhỏ nhất truyến thống. Cách còn lại thì minimize hội tụ Kullback-Leibler. Ta sẽ coi bài toán NMF như sau:

Cho một ma trận không âm V và tìm một ma trận không âm W và H thoả mãn:

$$V \approx WH$$

NMF có thể được áp dụng để phân tích thống kê cho dữ liệu nhiều chiều theo cách sau. Cho trước một tập hợp các véctơ dữ liệu nhiều chiều (n chiều), véctơ được xếp theo cột của ma trận n x m V trong m là số examples trong dataset. Ma trận này sau đó được phân rã xấp xỉ một ma trận n x r W và r x m H. Thường r sẽ được chọn nhỏ hơn n hoặc m. Do đó, nó dẫn đến phiên bản nén của ma trận dữ liệu ban đầu.
Như vậy, ta sẽ cần phải định nghĩa hàm cost functions cho NMF. Một phương pháp hiệu quả đơn giản là khoảng cách Euclidean giữa A và B. Nhưng xét tới 2 phương pháp tiếp cận mà tôi đã đề cập ở trên thì NMF được cho sẽ là bài toán tối ưu như sau:

- Minimize $||V - WH||^2$ với W, H đều không âm
- Minimize $D(V||WH)$ với W, H đều không âm.

Mặc dù 2 hàm loss trên đều lồi trên hoặc chỉ trên W hoặc chỉ trên H, nó không lồi trên cả 2 biến. Do đó, nó là không thể để mong đợi một thuật toán giải quyết cả 2 bài toán trên ở điểm global minima. Tuy nhiên, có những kĩ thuạt từ tối ưu số có thể áp dụng để tìm được điểm local minima.
Quy tắc update cộng dồn sau được đề xuất ở paper : Algorithms for Non-negative Matrix Factorization (Daniel D.LEE, H. Sebastian Seung).
Định lý 1 Khoảng cách Euclidean là không tăng với quy tắc update sau:

$$H_{a\mu} < -H_{a\mu}\frac{(W^TV)_{a\mu}}{W^TWH)_{a\mu}}$$

$$W_{ia} < -W_{ia}\frac{(VH^T)_{ia}}{(WHH^T)_{ia}}$$

Định lý 2 Chỉ số $D(V||WH)$ là không tăng với quy tắc update sau:

$$H_{a\mu} < -H_{a\mu}\frac{\Sigma_i W_{ia}V_{i\mu}/(WH)_{i\mu}}{\Sigma_k W_{ka}}$$

$$W_{ia} < -W_{ia}\frac{\Sigma_\mu H_{a\mu}V_{i\mu}/(WH)_{i\mu}}{\Sigma_v H_{av}}$$

Chi tiết chứng minh của hai định lý trên có thể tìm thấy ở paper trên.
Thư viện sklearn có NMF và họ sử dụng hàm objective như sau:

$$0.5*||X-WH||^2_{loss}+\alpha*l1_{ratio}*||vec(W)||_1+\alpha*l1_{ratio}*||vec(H)||_1+0.5*\alpha*(1-l1_{ratio})*||W||^2_{fro}+\alpha*l1_{ratio}*||$$

3.1 Autoencoder cho NMF

Phương pháp giải bài toán NMF bằng autoencoder được đề xuất trong paper A NEURAL NETWORK ALTERNATIVE TO NON-NEGATIVE AUDIO MODELS Paris Smaragdis],[Shrikant Venkataramani. Theo đó, ta sẽ bắt đầu với ma trận V ban đầu. Ta sẽ input một ma trận W cố định vào autoencoder làm tham số để optimize. Theo đó, ta sẽ thực hiện các bước encoder và decoder đều là linear với weight 2 bước lần lượt là $W^{\ddagger}, W$. Trong đó $W^{\ddagger}$ là ma trận giả nghịch đảo của ma trận W. Trong đại số tuyến tính, ma trận giả nghịch đảo $A^{\ddagger}$ của ma trận A là một tổng quát hóa của ma trận nghịch đảo. Ma trận giả nghịch đảo phải thoả mãn 4 tính chất sau:

- $AA^{\ddagger}A = A$
- $A^{\ddagger}AA^{\ddagger} = A^{\ddagger}$
- $(AA^{\ddagger})^* = AA^{\ddagger}$
- $(A^{\ddagger}A)^* = A^{\ddagger}A$

Kết quả của bước Encoder chính là ma trận H. Tuy nhiên để đảm bảo tính không âm của ma trận H thì ta sẽ áp dụng 1 hàm cho ra kết quả không âm. Có thể là relu, sigmoid, softplus hoặc thậm chí là tuyệt đối. Về mặt tính năng thì nếu không qua hàm cho ra kết quả không âm thì thuật toán đề xuất trên khá tương đương với NMF. Song mình không đảm bảo được tính không âm của W và H. Tuy nhiên, thuật toán đề xuất trên chỉ đảm bảo được H và ma trận predict $\hat{X}$ không âm. Không có gì đảm bảo là $W^{\ddagger}, W$ sẽ không âm. Tuy nhiên, nó không quá quan trọng vì ta chỉ cần output và latent state không âm. Thuật toán để optimize tôi chọn là Adam vì nó cho ra kết quả tốt hơn là Stochastic Gradient Descent. Loss function mà tôi chọn sẽ là loss của kullback-Leibler. Để tính loss Kullback-Leibler tôi sử dụng thư viện torchnmf có hàm loss. Theo đó, loss này chính là:

$$l(x, y) = \sum_{n=0}^{N-1} x_n log(\frac{x_n}{y_n}) - x_n + y_n$$

STT	Họ và tên	MSSV	Nhiệm vụ	Kiểm tra chéo	Đánh giá
1	Nguyễn Quốc Bảo	18110053	ICA-AE	Trần Tấn Phong	Hoàn thành
2	Nguyễn Đức Vũ Duy	18110004	NMF-AE	Vũ Thiện Nhân	Hoàn thành
3	Trần Tấn Phong	18110181	PCA-AE	Nguyễn Đức Vũ Duy	Hoàn thanh

Reference

(1) Paris Smaragdis, Shrikant Venkataramani, University of Illinois at Urbana-Champaign, "A Neural Network Alternative to Non-Negative Audio Models", arXiv:1609.03296v1 [cs.SD] 12 Sep 2016

(2) Saïd Ladjal, Alasdair Newson, Chi-Hieu Pham, "A PCA-LIKE AUTOENCODER", arXiv:1904.01277v1 [cs.CV] 2 Apr 2019

(3) Nonlinear Extensions of Reconstruction ICA- Apaar Sadhwani and Apoorv Gupta - CS229 Project Report, Fall 2011