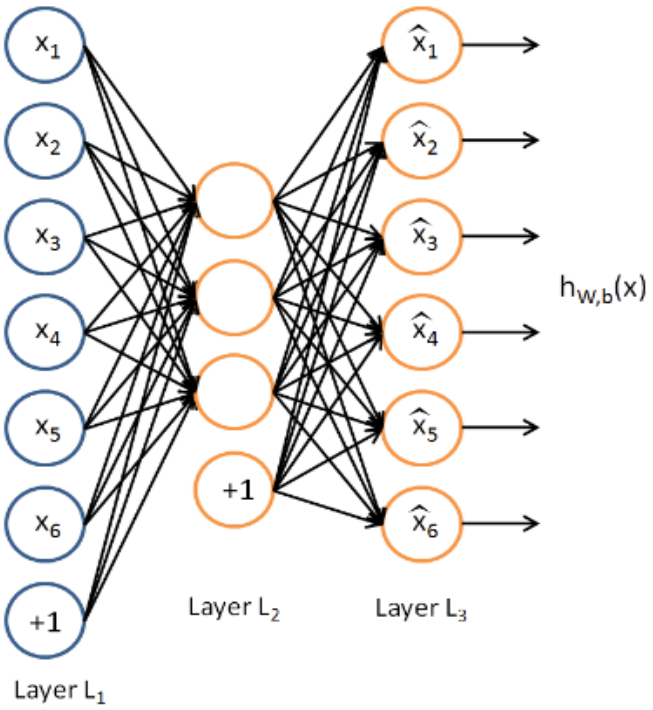
Báo cáo Xử lý đa chiều - Cuối kì

Nguyễn Minh Hoàng - 18110095
Nguyễn Phú Thành - 18110014
Vương Ngọc Hương Thảo - 18110223

Ngày 2 tháng 6 năm 2021

1 AutoEncoder

AutoEncoder là mạng neuron nhân tạo sử dụng trong bài toán unsupervised learning với target value chính là input mà ta đưa vào mạng này. Một mạng AutoEncoder thường có cấu trúc như sau:



Hình 1: Cấu trúc mạng neuron của AutoEncoder. Nguồn: [4]

Trong đó mạng neuron cố gắng tìm hàm h sao cho $h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x}) \approx \boldsymbol{x}$ (xấp xỉ theo nghĩa của một chuẩn nào đó). Lớp L_2 được gọi là lớp Encoder
Trong bài báo cáo này, hàm mất mát (loss function) của mạng neuron AutoEncoder là hàm có dạng:

$$J(\boldsymbol{W}, \boldsymbol{b}; \boldsymbol{x}) = \|\boldsymbol{x} - h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{x})\|^2$$

trong đó $\|\cdot\|$ là chuẩn
Nếu gọi trọng số giữa lớp L_1 và lớp L_2 là $\boldsymbol{W}_{\text{enc}}$, giữa lớp L_2 và lớp L_3 là $\boldsymbol{W}_{\text{dec}}$, hàm activation của lớp L_2 là $\sigma(\cdot)$ thì hàm loss của ta được viết lại thành:

$$J(\boldsymbol{W}, \boldsymbol{b}; \boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{W}_{\text{dec}}\sigma(\boldsymbol{W}_{\text{enc}}\boldsymbol{x} + \boldsymbol{b}_{\text{enc}}) - \boldsymbol{b}_{\text{dec}}\|^2 \tag{1}$$

2 Mối quan hệ giữa Kernel PCA và AutoEncoder

2.1 Kernel Principal Component Analysis (KPCA)

Cho tập dữ liệu $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m\}, \boldsymbol{x}_i \in \mathbb{R}^n$, đưa các điểm dữ liệu này lên một không gian mới (feature space) $\{\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \cdots, \phi(\boldsymbol{x}_m)\}$ với $\phi(\boldsymbol{x}_i) \in \mathbb{R}^N$

Giả sử các điểm dữ liệu trong không gian mới này có trung bình là $\boldsymbol{0}$, tức: $\frac{1}{m}\sum_{i=1}^{m}\phi(\boldsymbol{x}_i) = \boldsymbol{0}$

Khi đó ma trận hiệp phương sai (không hiệu chỉnh) của các điểm dữ liệu này là:

$$\boldsymbol{C} = \frac{1}{m}\sum_{i=1}^{m}\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^T$$

Để thực hiện PCA trên tập dữ liệu này, ta cần tìm trị riêng, vectơ riêng của ma trận $\boldsymbol{C}$, tức tìm các $\lambda_j > 0, \boldsymbol{v}_j \in \mathbb{R}^N$ sao cho:

$$\boldsymbol{C}\boldsymbol{v}_j = \lambda_j \boldsymbol{v}_j, j = 1, 2, 3, \cdots, N$$

Đặt kernel $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ với:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$$

Thay $\boldsymbol{C}$ vào (2) ta được:

$$\frac{1}{m}\sum_{i=1}^{m}\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^T\boldsymbol{v}_j = \lambda_j\boldsymbol{v}_j, j = 1, 2, \cdots, N$$

Do đó, mỗi vectơ riêng $\boldsymbol{v}_j$ có thể được viết thành một tổ hợp tuyến tính của các $\phi(\boldsymbol{x}_i)$:

$$\boldsymbol{v}_j = \sum_{i=1}^{m} a_{ji}\phi(\boldsymbol{x}_i)$$

Tìm các vectơ riêng của $\boldsymbol{C}$ tương đương với việc tìm các hệ số a_{ji}
Thay biểu thức trên vào ta được:

$$\begin{aligned}
\frac{1}{m}\sum_{i=1}^{m}\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^T\sum_{l=1}^{m}a_{jl}\phi(\boldsymbol{x}_l) &= \lambda_j\sum_{l=1}^{m}a_{jl}\phi(\boldsymbol{x}_l) \\
\frac{1}{m}\sum_{i=1}^{m}\phi(\boldsymbol{x}_i)\sum_{l=1}^{m}a_{jl}\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_l) &= \lambda_j\sum_{l=1}^{m}a_{jl}\phi(\boldsymbol{x}_l) \\
\frac{1}{m}\sum_{i=1}^{m}\phi(\boldsymbol{x}_i)\sum_{l=1}^{m}a_{jl}K(\boldsymbol{x}_i, \boldsymbol{x}_l) &= \lambda_j\sum_{l=1}^{m}a_{jl}\phi(\boldsymbol{x}_l) \\
\frac{1}{m}\sum_{i=1}^{m}\phi(\boldsymbol{x}_k)^T\phi(\boldsymbol{x}_i)\sum_{l=1}^{m}a_{jl}K(\boldsymbol{x}_i, \boldsymbol{x}_l) &= \lambda_j\sum_{l=1}^{m}a_{jl}\phi(\boldsymbol{x}_k)^T\phi(\boldsymbol{x}_l) \\
\frac{1}{m}\sum_{i=1}^{m}K(\boldsymbol{x}_k, \boldsymbol{x}_i)\sum_{l=1}^{m}a_{jl}K(\boldsymbol{x}_i, \boldsymbol{x}_l) &= \lambda_j\sum_{l=1}^{m}a_{jl}K(\boldsymbol{x}_k, \boldsymbol{x}_l)
\end{aligned}$$

Bằng cách đặt $\boldsymbol{K}$ là ma trận với $\boldsymbol{K}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\boldsymbol{a}_j$ là vectơ gồm các hệ số a_{jl}, ta được:

$$\boldsymbol{K}^2\boldsymbol{a}_j = m\lambda_j\boldsymbol{K}\boldsymbol{a}_j$$

hay:

$$\boldsymbol{K}\boldsymbol{a}_j = m\lambda_j\boldsymbol{a}_j$$

Hơn nữa, điều kiện các vectơ riêng có chiều dài là 1 có thể được viết lại thành:

$$\begin{aligned} 1 &= \boldsymbol{v}_j^T\boldsymbol{v}_j \\ &= \sum_{k=1}^{m}\sum_{l=1}^{m} a_{jl}a_{jk}\phi(\boldsymbol{x}_l)^T\phi(\boldsymbol{x}_k) \\ &= \boldsymbol{a}_j^T\boldsymbol{K}\boldsymbol{a}_j \end{aligned}$$

Vì $\boldsymbol{a}_j$ thỏa $\boldsymbol{K}\boldsymbol{a}_j = m\lambda_j\boldsymbol{a}_j$ nên ta được:

$$m\lambda_j\boldsymbol{a}_j^T\boldsymbol{a}_j = 1 \ \forall j$$

Khi tìm được các vectơ riêng $\boldsymbol{a}_j$, hình chiếu của vectơ $\boldsymbol{x} \in \mathbb{R}^n$ lên các thành phần chính $\boldsymbol{v}_j$ là:

$$\begin{aligned} \phi(\boldsymbol{x})^T\boldsymbol{v}_j &= \sum_{i=1}^{m} a_{ji}\phi(\boldsymbol{x})^T\phi(\boldsymbol{x}_i) \\ &= \sum_{i=1}^{m} a_{ji}K(\boldsymbol{x}, \boldsymbol{x}_i) \end{aligned}$$

Trong trường hợp các điểm dữ liệu trong không gian mới có trung bình khác không, với mỗi điểm dữ liệu ta trừ nó với trung bình mẫu, tức:

$$\widetilde{\phi}(\boldsymbol{x}_j) = \phi(\boldsymbol{x}_j) - \frac{1}{m}\sum_{k=1}^{m}\phi(\boldsymbol{x}_k)$$

Ma trận Kernel tương ứng với các hệ số được cho bởi:

$$\tilde{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \widetilde{\phi}(\boldsymbol{x}_i)^T\widetilde{\phi}(\boldsymbol{x}_j)$$

Thực hiện các phép biến đổi ta được:

$$\widetilde{\boldsymbol{K}} = \boldsymbol{K} - 2\boldsymbol{1}_m\boldsymbol{K} + \boldsymbol{1}_m\boldsymbol{K}\boldsymbol{1}_m$$

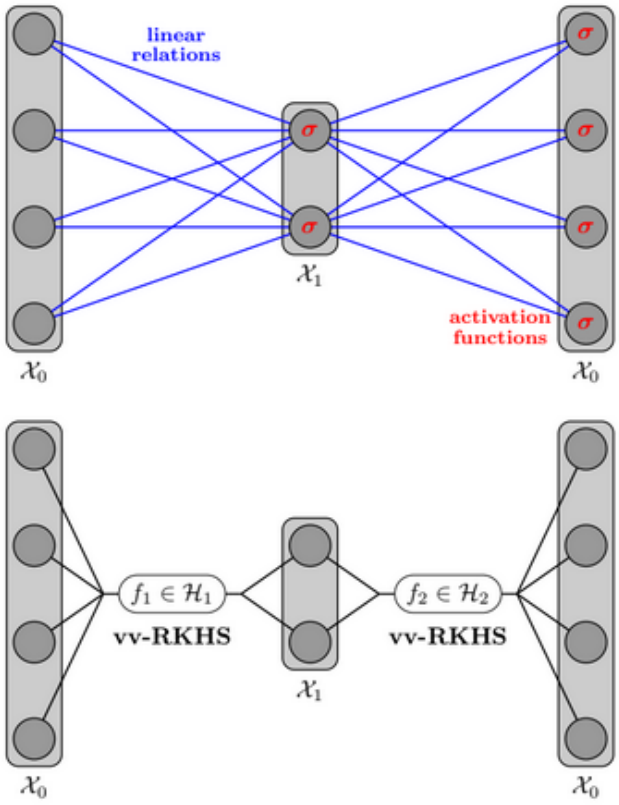
trong đó $\boldsymbol{1}_m$ là ma trận $m \times m$ mà mỗi phần tử đều là $\dfrac{1}{m}$

2.2 Mối quan hệ giữa Kernel PCA và AutoEncoder

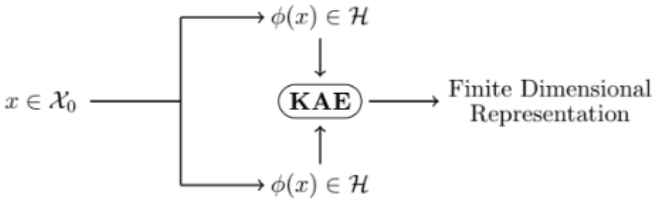
Ta đã biết AutoEncoder với hàm activation ở các lớp là linear thì AutoEncoder sẽ giải bài toán tương tự với bài toán PCA (mặc dù ma trận chiếu ta nhận được của AutoEncoder sẽ không trực giao như với PCA)

Như vậy, một cách đơn giản để thực hiện Kernel PCA qua AutoEncoder là ta đưa trực tiếp các điểm dữ liệu $\boldsymbol{x}_i$ sang $\phi(\boldsymbol{x}_i)$ và thực hiện AutoEncoder tuyến tính như trên với bộ dữ liệu $\{\phi(\boldsymbol{x}_i)\}$.

Tuy nhiên cách làm này là tốn kém và không khả thi
Một cách khắc phục điều này là thông qua Kernel AutoEncoder (KAE) và K^{2}AE (Xem tại [1])



Hình 2: Cấu trúc của **AE** và **KAE** hai lớp. Nguồn ảnh: [1]



Hình 3: Mô hình **K^{2}AE**. Nguồn ảnh: [1]

Các kết quả của các bài báo ở [1,2,3] đã chỉ ra được rằng mô hình K^{2}AE sẽ cho ra kết quả gần giống kết quả cho ra ở Kernel PCA
Một cách cài đặt mô hình KAE có thể xem tại: https://github.com/plaforgue/kae

3 Mối quan hệ giữa ICA và AutoEncoder

3.1 Independent Component Analysis (ICA)

Mô hình ICA được định nghĩa như sau:

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}$$

Trong đó:

- $\boldsymbol{x}$ là vector chứa các hỗn hợp (hay dữ liệu ban đầu) $(x_1, x_2...x_n)$
- $\boldsymbol{A}$ là ma trận trộn (mixing matrix)
- $\boldsymbol{s}$ là các vector tương ứng với các nguồn độc lập nhau $s_1, s_2, ..s_n$

Mô hình ICA sẽ mô tả làm cách nào để dữ liệu quan sát $\boldsymbol{x}$ được định nghĩa bằng cách trộn các thành phần $\boldsymbol{s}$. Với dữ liệu có sẵn $\boldsymbol{x}$ ta không thể trực tiếp trích xuất từ nguồn $\boldsymbol{s}$ cũng như từ ma trận trộn (mixing matrix) $\boldsymbol{A}$. Do đó ta cần phải ước lượng cả $\boldsymbol{A}$ và $\boldsymbol{s}$ từ dữ liệu ban đầu $\boldsymbol{x}$. Chú ý rằng việc tìm $\boldsymbol{A}$ hay $\boldsymbol{s}$ sẽ giúp ta tìm ra cái còn lại bằng cách áp dụng tính chất ma trận.
Cho $\boldsymbol{W}$ là ma trận nghịch đảo của $\boldsymbol{A}$, khi đó mô hình có thể được ghi lại như sau :

$$\boldsymbol{s} = \boldsymbol{W}\boldsymbol{x}$$

Ta có thể hiểu theo như sau : Với các dữ liệu ban đầu $\boldsymbol{X}$, ta cần tìm một bộ các vectors (là các cột của ma trận $\boldsymbol{W}$) làm cho các thuộc tính của $\boldsymbol{s}$ thưa (sparse) khi đã là hệ cở sở trực chuẩn (orthonormal basis). Nói cách khác, thì ma trận $\boldsymbol{W}$ sẽ là ánh xạ dữ liệu $\boldsymbol{x}$ sang các thuộc tính $\boldsymbol{s}$. Ta có thể định nghĩa ý tưởng đó thành một bài toán tối ưu.

$$\min_{\boldsymbol{W}} ||\boldsymbol{W}\boldsymbol{x}||_1 \text{ sao cho } \boldsymbol{W}\boldsymbol{W}^{\boldsymbol{T}} = I$$

Tuy nhiên ràng buộc trực chuẩn trong điều kiện trên sẽ gây ra những bất lợi cho thuật toán này. Khi số thuộc tính (số dòng của ma trận $\boldsymbol{W}$) bằng hoặc lớn hơn số chiều của dữ liệu đầu vào $\boldsymbol{x}$ thì việc tìm bài toán tối ưu của thuật toán này sẽ khó khăn hơn và cần huấn luyện lâu hơn. Ta cần tìm ra giải pháp để đẩy nhanh quá trình này.
Diều đó dẫn đến một thuật toán cải tiến của ICA.

3.2 Reconstruction ICA (RICA)

RICA thay thế ràng buộc $\boldsymbol{W}\boldsymbol{W}^{\boldsymbol{T}} = \boldsymbol{I}$ trong ICA bằng cách nới lỏng penalty cho sai số khi khôi phục, giống với sparse coding và sparse autoencoders.
Bài toán tối ưu được định nghĩa như sau :

$$\min_{W} \lambda||\boldsymbol{W}\boldsymbol{x}||_1 + \frac{1}{2}||\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{x} - \boldsymbol{x}||_2^2$$

Thuật toán này có 3 lợi thế rõ ràng hơn khi sau sử dụng ICA :

- 1. Chi phí tính toán của bài toán tối ưu sẽ được giảm vì ràng buộc đã được loại bỏ. Ta có thể sử dụng các thuật toán như L-BFGS hay CG để giải bài toán này.
- 2. RICA cho phép trích xuất đầy đủ các thuộc tính, điều mà ICA không thể làm được do ràng buộc $\boldsymbol{W}\boldsymbol{W}^{\boldsymbol{T}} = I$.
- 3. RICA ít nhạy cảm hơn với whitening. Khi sử dụng ICA , dữ liệu cần phải whitened nhưng vấn đề này có chi phí tính toán rất lớn đặc biệt là khi số lượng thuộc tính càng nhiều, việc thường xảy ra đối với các tấm ảnh lớn. RICA có thể xử lý dữ liệu với xấp xỉ whitening hoặc không cần dùng tới chúng.

3.3 Autoencoder trong bài toán ICA

ICA được cho là có liên kết với sparse autoencoders vì chúng cùng học các chi tiết cạnh của ảnh. Sự khác nhau giữa ICA và sparse autoencoders là việc sử dụng các ràng buộc trực chuẩn. Có thể thấy rằng khi dữ liệu $\{x^{(i)}\}_{i=1}^m$ có mean bằng 0, ta có thể xây dựng RICA từ ICA với ràng buộc trực chuẩn.
Các bổ đề sau đây sẽ cho thấy mối quan hệ giữa 2 thuật toán này :

- **Bổ đề 1:** Nếu dữ liệu ban đầu đã được whitened, thì chi phí của trực chuẩn sẽ tương tự như chi phí xây dựng RICA :

$$\lambda\|\boldsymbol{W^TW} - \boldsymbol{I}_n\|_F^2 = \frac{\lambda}{m}\sum_{i=1}^{m}\|\boldsymbol{W^TWx^{(i)}} - \boldsymbol{x^{(i)}}\|_2^2$$

Chứng minh:
Trước hết ta sẽ chỉ ra:

$$\|\boldsymbol{A}\|_F^2 = \text{tr}(\boldsymbol{A}^T\boldsymbol{A})$$

Thật vậy, với ma trận $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, ta có:

$$\begin{aligned}
\text{tr}(\boldsymbol{A}^T\boldsymbol{A}) &= \sum_{i=1}^{n}(\boldsymbol{A}^T\boldsymbol{A})_{ii} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m}(\boldsymbol{A}^T)_{ij}\boldsymbol{A}_{ji} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m}a_{ji}^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}a_{ij}^2 = \|\boldsymbol{A}\|_F^2
\end{aligned}$$

Nhận thấy khi $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ và $\boldsymbol{A}$ đối xứng thì $\|\boldsymbol{A}\|_F^2 = \text{tr}(A^2)$
Cho dữ liệu $\{\boldsymbol{x}^{(i)}\}_{i=1,\cdots,m}$ là dữ liệu đã được whiten, khi đó ma trận hiệp phương sai (không hiệu chỉnh) của $\{\boldsymbol{x}^{(i)}\}_{i=1,\cdots,m}$ là ma trận đơn vị, tức: $\frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)^T} = \boldsymbol{I}$

Ta có:

$$\begin{aligned}
\|\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)}\|_2^2 &= \|(\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I})\boldsymbol{x}^{(i)}\|_2^2 \\
&= \boldsymbol{x}^{(i)^T}(\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I})^2\boldsymbol{x}^{(i)} \\
&= \text{tr}\left[\boldsymbol{x}^{(i)^T}(\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I})^2\boldsymbol{x}^{(i)}\right] \\
&= \text{tr}\left[(\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I})^2\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)^T}\right] \;\; \forall i = 1, 2, \cdots, m
\end{aligned}$$

Khi đó:

$$\begin{aligned}
\frac{\lambda}{m}\sum_{i=1}^{m}\|\boldsymbol{W^TWx^{(i)}} - \boldsymbol{x^{(i)}}\|_2^2 &= \frac{\lambda}{m}\sum_{i=1}^{m}\text{tr}\left[(\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I})^2\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)^T}\right] \\
&= \lambda\text{tr}\left[(\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I})^2\frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)^T}\right] \\
&= \lambda\text{tr}\left[(\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I})^2\right] = \lambda\|\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I}\|_F^2
\end{aligned}$$

Dấu đẳng thức cuối cùng có được là do ma trận $\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I}$ là ma trận đối xứng.
Như vậy:

$$\lambda\|\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W} - \boldsymbol{I}_n\|_F^2 = \frac{\lambda}{m}\sum_{i=1}^{m}\|\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{x}^{(\boldsymbol{i})} - \boldsymbol{x}^{(\boldsymbol{i})}\|_2^2$$

- **Bổ đề 2:** Ràng buộc các cột của ma trận $\boldsymbol{W}$ trực chuẩn tương đương với ràng buộc các dòng của ma trận $\boldsymbol{W}$ trực chuẩn, tức:

$$\lambda\|\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W} - \boldsymbol{I}_n\|_F^2 = \lambda\|\boldsymbol{W}\boldsymbol{W}^{\boldsymbol{T}} - \boldsymbol{I}_k\|_F^2 + c$$

với c là hằng số.
Chứng minh:
Cho ma trận $\boldsymbol{W} \in \mathbb{R}^{k\times n}$. Theo trên, vì $\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I}_n$ và $\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{I}_k$ là các ma trận đối xứng nên:

$$\begin{aligned}\|\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I}_n\|_F^2 &= \text{tr}\left[(\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I}_n)^2\right]\\ \|\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{I}_k\|_F^2 &= \text{tr}\left[(\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{I}_k)^2\right]\end{aligned}$$

Ta có:

$$\begin{aligned}\|\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I}_n\|_F^2 &= \text{tr}\left[(\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{I}_n)^2\right]\\ &= \text{tr}\left(\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{W} - 2\boldsymbol{W}^T\boldsymbol{W} + \boldsymbol{I}_n\right)\\ &= \text{tr}(\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{W}) - 2\text{tr}(\boldsymbol{W}^T\boldsymbol{W}) + n\\ &= \text{tr}(\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{W}^T) - 2\text{tr}(\boldsymbol{W}\boldsymbol{W}^T) + \text{tr}(\boldsymbol{I}_k) + (n-k)\\ &= \text{tr}(\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{W}^T - 2\boldsymbol{W}\boldsymbol{W}^T + \boldsymbol{I}_k) + (n-k)\\ &= \text{tr}\left[(\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{I}_k)^2\right] + (n-k) = \|\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{I}_k\|_F^2 + (n-k)\end{aligned}$$

Vậy:

$$\lambda\|\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W} - \boldsymbol{I}_n\|_F^2 = \lambda\|\boldsymbol{W}\boldsymbol{W}^{\boldsymbol{T}} - \boldsymbol{I}_k\|_F^2 + c$$

với $c = \lambda(n-k)$

Với 2 bổ đề trên, ta rút ra các kết luận sau:

- RICA sẽ tương tự như ICA với cách thể hiện đầy đủ hay không đầy đủ so với bộ dữ liệu ban đầu khi λ tiến tới vô cùng và dữ liệu được whitened. Công thức của RICA :

$$\min_{\boldsymbol{W}}\frac{\lambda}{m}\sum_{i=1}^{m}||\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{x}^{(\boldsymbol{i})} - \boldsymbol{x}^{(\boldsymbol{i})}||_2^2 + ||\boldsymbol{W}\boldsymbol{x}||_1$$

Sử dụng các bổ đề trên đẳng thức sẽ tương đương với :

$$\min_{\boldsymbol{W}}\lambda\|\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W} - \boldsymbol{I}\|_F^2 + \|\boldsymbol{W}\boldsymbol{x}\|_1 \quad \text{và}$$
$$\min_{\boldsymbol{W}}\lambda\|\boldsymbol{W}\boldsymbol{W}^{\boldsymbol{T}} - \boldsymbol{I}\|_F^2 + \|\boldsymbol{W}\boldsymbol{x}\|_1$$

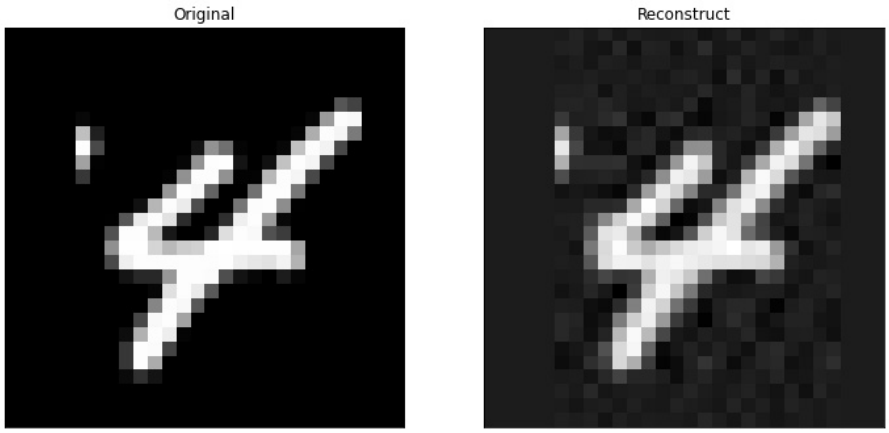
Vì thế, khi λ tiến tới vô cùng, bài toán tối ưu trở thành:

$$\min_{\boldsymbol{W}} \|\boldsymbol{W}\boldsymbol{x}\|_1 \text{ sao cho } \boldsymbol{W}\boldsymbol{W}^{\boldsymbol{T}} = I$$

và nó cũng chính là bài toán của ICA.

- RICA cũng tương tự như sparse autoencoder khi chúng ta đặt hàm kích hoạt $\sigma(x)$ là hàm tuyến tính hay hàm đồng nhất (identity function), ta sử dụng chuẩn L_1 cho hàm kích hoạt và gán các biến bias là 0.
- RICA sẽ tương tự với công thức của sparse coding nếu ta bỏ qua chuẩn cực đại và đặt $\boldsymbol{x}^{(j)} := \boldsymbol{W}\boldsymbol{x}^{(j)}$

Trong phần Demo Code, chuẩn L_1 được xấp xĩ bởi hàm $f(x) = \sqrt{x^2 + \epsilon}$ với $\epsilon = 0.01$
Demo Code: https://colab.research.google.com/drive/1wUj74NVpd9rjVsAgfOECYqc4rISgvObD?usp=sharing



Hình 4: Kết quả khôi phục ảnh gốc với RICA

4 Mối quan hệ giữa Nonnegative Matrix Factorization (NMF) với AutoEncoder

4.1 Nonnegative Matrix Factorization

Cho ma trận dữ liệu $\boldsymbol{X}$ với số chiều là $d \times n$ (d số chiều của mỗi điểm dữ liệu và n là số điểm dữ liệu) và các điểm dữ liệu là các số thực không âm
Bài toán NMF là phân rã ma trận $\boldsymbol{X}$ thành hai ma trận $\boldsymbol{W}$ và $\boldsymbol{H}$, với các hệ số của hai ma trận này là không âm, sao cho:

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$$

xấp xỉ theo nghĩa của chuẩn Frobenius tức ta cần:

$$J(\boldsymbol{W}, \boldsymbol{H}; \boldsymbol{X}) = \|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2 \text{ càng nhỏ càng tốt} \tag{2}$$

Ma trận $\boldsymbol{W}$ là ma trận $d \times p$ trong đó các cột tạo thành 1 cơ sở cho không gian của latent variables.
Ma trận $\boldsymbol{H}$ là ma trận $p \times n$ là ma trận của các điểm dữ liệu trong không gian ít chiều hơn ($p < d$)

4.2 AutoEncoder trong bài toán NMF

Từ hai hàm loss (1) và (2), ta thấy nếu ta đưa từng cột $\boldsymbol{X}_j$ của ma trận $d \times n$ $\boldsymbol{X}$ vào lớp input L_1 thì khi đó $\boldsymbol{W}_{dec}$ đóng vai trò là ma trận $\boldsymbol{W}$ và $\sigma(\boldsymbol{W}_{enc}\boldsymbol{X}_j)$ đóng vai trò là một cột của ma trận $\boldsymbol{H}$
Như vậy, để đảm bảo yêu cầu của bài toán NMF, ta đưa ra 3 điều kiện sau:

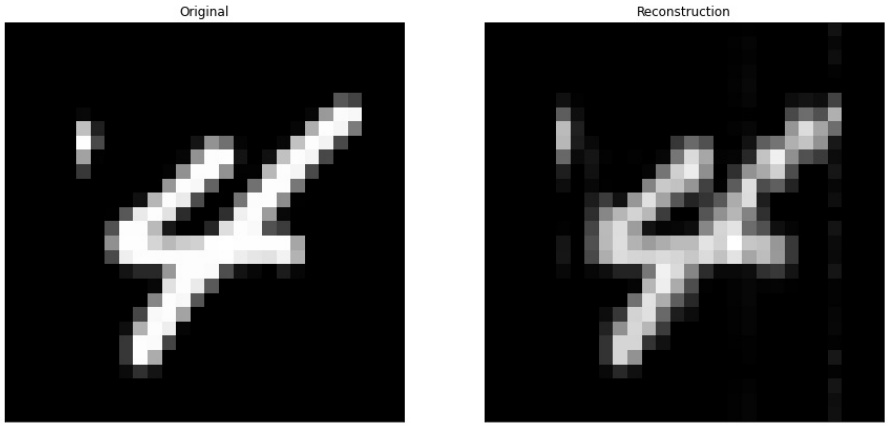
- Hàm kích hoạt $\sigma(\cdot)$ phải là hàm kích hoạt có giá trị output không âm, chẳng hạn hàm sigmoid, ReLU, $\cdots$ Điều kiện này là để sau khi huấn luyện mạng neuron thì các cột của ma trận $\boldsymbol{H} = \sigma(\boldsymbol{W}_{\text{enc}}(\boldsymbol{X})$ có các phần tử không âm
- Mạng **AE** không có trọng số bias để đảm bảo bài toán tìm $\boldsymbol{W}, \boldsymbol{H}$ sao cho $\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$ (nếu có bias thì bài toán trở thành $\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H} + \boldsymbol{B}$)
- Việc khởi tạo và cập nhật trọng số $\boldsymbol{W}_{\text{dec}}$ phải đảm bảo rằng các hệ số của $\boldsymbol{W}_{\text{dec}}$ không âm. Việc này để đảm bảo $\boldsymbol{W} = \boldsymbol{W}_{dec}$ là ma trận có các hệ số không âm sau khi huấn luyện mạng neuron.
  Có nhiều cách để đảm bảo việc này, hai cách thường dùng là hoặc sử dụng Gradient Descent thông thường và gán các hệ số âm sau cập nhật bằng 0 [1] [2] hoặc sử dụng cách cập nhật: [3]

  $$w \leftarrow w \cdot e^{-\eta \frac{\partial J}{\partial w}}$$

  trong đó η là tốc độ học, $\dfrac{\partial J}{\partial w}$ là đạo hàm riêng của J theo trọng số w
  Phần Demo code gồm hai phần, sử dụng hai cách cập nhật như trên

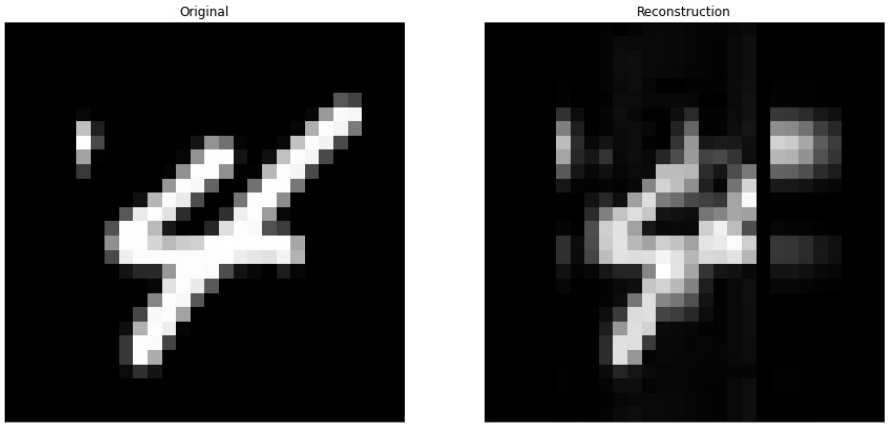
Demo Code 1 sử dụng cách cập nhật đầu tiên: https://colab.research.google.com/drive/1RM9DfReVaWFql3YDhJpApVGZZf3cOYQW?usp=sharing

[1] D.D.Lee and H.S.Seung, "Unsupervised learning by convex and conic coding"
[2] D.D.Lee and H.S.Seung, "Learning the parts of objects by nonnegative matrix factorization"
[3] A.E.Khatib, S.Huang, A.Ghodsi and F.Karray, "Nonegative Matrix Factorization Using Autoencoders and Exponentiated Gradient Descent"



Hình 5: Kết quả khôi phục ảnh gốc

Demo Code 2 sử dụng cách cập nhật thứ hai: https://colab.research.google.com/drive/14YeG_tBjZfs68tQKFXFBnCEZe7MMvz-e?usp=sharing



Hình 6: Kết quả khôi phục ảnh gốc

Tài liệu

[1] Pierre Laforge, Stephan Clémencon, Florence d'Alché-Buc, *Autoencoding any Data through Kernel Autonencoders*

[2] Scholkopf, B., Smola, A., Muller, K.-R. (1997). Kernel principal component analysis. In *International conference on artifical neural networks, 583-588.* Springer

[3] Scholkopf, B., Smola, A., Muller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem.*Neural computation*

[4] M.B.Segarra, L.I.Munoz. (2017). *Study of Reconstruction ICA for feature extraction in images and signals*

[5] D.D.Lee, H.S.Seung, *Unsupervised learning by convex and conic coding*

[6] D.D.Lee and H.S.Seung, *Learning the parts of objects by nonnegative matrix factorization*

[7] A.E.Khatib, S.Huang, A.Ghodsi and F.Karray. *Nonegative Matrix Factorization Using Autoencoders and Exponentiated Gradient Descent*