Singular Value Decomposition

Vũ Hoàng Anh - B23DCVT037

Học viện Công nghệ Bưu Chính Viễn Thông

Tóm tắt nội dung

Hệ thống đề xuất (recommendation system) đóng vai trò quan trọng trong cuộc sống hiện đại, đặc biệt là trong ngành công nghiệp giải trí trực tuyến. Thuật toán đề xuất là cốt lõi của hệ thống này; thuật toán càng tốt thì các đề xuất càng chính xác và có trách nhiệm hơn. Trong lĩnh vực AI nói chung và học máy có giám sát (supervised learning) nói riêng, thuật toán **Singular Value Decomposition** (phân tích thành phần suy biến) được sử dụng để giảm chiều dữ liệu và phát hiện các yếu tố tiềm ẩn (latent factors) trong dữ liệu tương tác giữa người dùng và sản phẩm. Nhờ đó, hệ thống có thể đưa ra những đề xuất sản phẩm tương tự với độ chính xác cao hơn.

Keywords: SVD, latent factors, recommender system

1 Giới thiệu

Đại số tuyến tính là một lĩnh vực toán ứng dụng quan trọng trong khoa học máy tính nói chung và học máy nói riêng. Singular Value Decomposition cho phép phân rã (phân tích) ma trận bất kì (vuông hoặc không vuông) thành 3 ma trận cơ bản.

$$A = U\Sigma V^T$$

- U và V là các ma trận trực chuẩn (orthonormal matrix).
- Σ là ma trận đường chéo, chứa các trị riêng (eigenvalues).

SVD (Singular Value Decomposition) là một công cụ toán học mạnh mẽ với nhiều ứng dụng quan trọng trong phân tích dữ liệu và học máy. Hệ thống đề xuất (recommendation systems) được sử dụng rộng rãi để cá nhân hóa trải nghiệm của người dùng bằng cách dự đoán và gợi ý các sản phẩm, dịch vụ hoặc thông tin mà người dùng có thể quan tâm. Các hệ thống này thường phải xử lý và phân tích dữ liệu khổng lồ từ hành vi

người dùng, và SVD là một công cụ hữu ích trong việc cải thiện chất lượng của các dự đoán này.

Cụ thể, SVD có thể được sử dụng để giảm chiều dữ liệu và phát hiện các yếu tố ẩn (latent factors) trong ma trận đánh giá người dùng - sản phẩm. Bằng cách phân rã ma trận đánh giá thành các thành phần cơ bản, SVD giúp làm nổi bật các yếu tố chính ảnh hưởng đến sự lựa chọn của người dùng, từ đó cung cấp những gợi ý chính xác và phù hợp hơn.

Trong bài viết này, chúng ta sẽ đi sâu vào việc áp dụng SVD trong các mô hình hệ thống đề xuất. Chúng ta sẽ trình bày các ví dụ thực tiễn và phân tích hiệu quả của SVD trong các bài toán hệ thống đề xuất, cùng với những thách thức và giải pháp để tối ưu hóa việc áp dụng SVD trong môi trường thực tế.

2 Lý thuyết

Giả sử chúng ta có ma trận A= $\begin{pmatrix} 4 & 0 & 0 \\ 3 & 5 & 0 \\ 0 & 0 & 2 \end{pmatrix}$

Sau khi áp dụng SVD: A = UΣV^T

$$U = \begin{pmatrix} -0.45 & -0.89 & 0.0 \\ -0.89 & 0.45 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 6.23 & 0 & 0 \\ 0 & 3.16 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad V^T = \begin{pmatrix} -0.71 & -0.71 & 0.0 \\ -0.71 & 0.71 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$$

Chi tiết: eMath Calculator.
Đối với ma trận nghịch đảo không vuông: Moore Penrose Inverse.

Sau khi phân rã ma trận, chúng ta có thể phát hiện các cấu trúc ẩn và mối quan hệ trong dữ liệu. Đối với hệ thống đề nghị, phân rã ma trận giúp chúng ta xác định các đặc trưng ẩn thông qua các trị riêng trong ma trận Σ. Chúng ta có thể chọn **k** đặc trưng quan trọng nhất, tương ứng với **k** trị riêng lớn nhất. Để làm điều này, chúng ta giữ lại **k** cột từ ma trận U và **k** hàng từ ma trận V^T.

Việc chọn **k** đặc trưng này nhằm mục đích lọc nhiễu và giảm chiều dữ liệu, đồng thời giữ lại các thông tin quan trọng nhất. Sau khi chọn, chúng ta

có các ma trận con U_k, Σ_k, và V_k^T, là các phiên bản nén của các ma trận thành phần ban đầu. Ma trận nén A_k được tính bằng cách nhân các ma trận này:

$$A_k = U_k \Sigma_k V_k^T$$

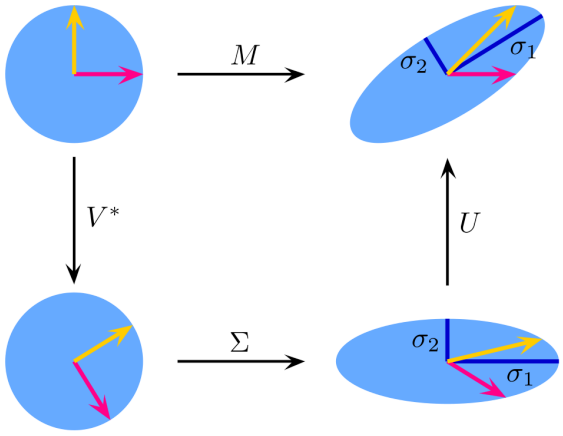
Ma trận A_k là phiên bản giảm chiều của ma trận ban đầu, giúp lọc nhiễu và làm rõ các cấu trúc quan trọng trong dữ liệu.

Lưu ý cách chọn **k**:

- k lớn: Giữ nhiều thông tin hơn, khả năng bị nhiễu cao hơn.
- k nhỏ: Nén tốt, có thể mất thông tin quan trọng.

Thông thường k nhỏ hơn kích thước của ma trận Σ.

Hầu hết các framework hay thư viện hiện nay hỗ trợ tự động chọn k tối ưu nên chúng ta không cần quá bận tâm về vấn đề này.



$$M = U \cdot \Sigma \cdot V^*$$

Hình 1: Hình dung SVD

Minh họa của phép phân rã giá trị riêng $U\Sigma V^*$ của một ma trận thực 2×2 M.
Trên: Tác động của M, được chỉ ra bởi ảnh hưởng của nó lên đĩa đơn vị D và hai vectơ đơn vị chuẩn e_1 và e_2.
Trái: Tác động của V^*, một phép xoay, lên D, e_1 và e_2.
Dưới: Tác động của Σ, một phép co dãn bởi các giá trị riêng σ_1 theo chiều ngang và σ_2 theo chiều dọc.

Phải: Tác động của U, chính là một phép xoay khác.

3 Áp dụng

3.1 Thuật toán

Giả sử ma trận đánh giá người dùng R là:

$$R = \begin{pmatrix} 4 & 0 & 3 & 5 & 0 & 4 \\ 5 & 5 & 0 & 0 & 4 & 5 \\ 3 & 2 & 0 & 4 & 1 & 1 \\ 0 & 3 & 5 & 0 & 2 & 2 \\ 2 & 0 & 0 & 3 & 4 & 0 \end{pmatrix}$$

Sau khi phân rã ma trận R bằng SVD, chúng ta có:

$$R = U\Sigma V^T$$

Trong đó:

$$U = \begin{pmatrix} 0.52 & 0.6 & 0.37 & 0.36 & 0.34 \\ 0.65 & -0.57 & -0.39 & 0.28 & 0.14 \\ 0.36 & 0.33 & -0.21 & -0.06 & -0.85 \\ 0.31 & -0.4 & 0.75 & -0.39 & -0.18 \\ 0.28 & 0.24 & -0.33 & -0.8 & 0.35 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 12.98 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 6.45 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 5.46 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 3.88 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 2.02 \end{pmatrix}$$

$$V^T = \begin{pmatrix} 0.54 & 0.38 & 0.24 & 0.38 & 0.36 & 0.49 \\ 0.16 & -0.52 & -0.03 & 0.78 & -0.28 & -0.14 \\ -0.32 & -0.02 & 0.89 & 0.01 & -0.29 & 0.15 \\ 0.27 & 0.03 & -0.23 & -0.22 & -0.75 & 0.51 \\ 0.1 & -0.76 & 0.07 & -0.32 & 0.37 & 0.42 \\ -0.71 & 0.09 & -0.31 & 0.33 & 0.11 & 0.53 \end{pmatrix}$$

Chọn $k = 3$ đặc trưng quan trọng nhất, chúng ta có các ma trận con:

$$U_k = \begin{pmatrix} 0.52 & 0.6 & 0.37 \\ 0.65 & -0.57 & -0.39 \\ 0.36 & 0.33 & -0.21 \\ 0.31 & -0.4 & 0.75 \\ 0.28 & 0.24 & -0.33 \end{pmatrix}$$

$$\Sigma_k = \begin{pmatrix} 12.98 & 0.0 & 0.0 \\ 0.0 & 6.45 & 0.0 \\ 0.0 & 0.0 & 5.46 \end{pmatrix}$$

$$V_k^T = \begin{pmatrix} 0.54 & 0.38 & 0.24 & 0.38 & 0.36 & 0.49 \\ 0.16 & -0.52 & -0.03 & 0.78 & -0.28 & -0.14 \\ -0.32 & -0.02 & 0.89 & 0.01 & -0.29 & 0.15 \end{pmatrix}$$

Ma trận nén A_k được tính bằng:

$$R_k = U_k \Sigma_k V_k^T$$

$$R_k = \begin{pmatrix} 3.62 & 0.51 & 3.3 & 5.6 & 0.76 & 3.07 \\ 4.65 & 5.16 & 0.24 & 0.32 & 4.68 & 4.33 \\ 3.23 & 0.69 & 0.04 & 3.42 & 1.42 & 1.82 \\ 0.45 & 2.79 & 4.69 & -0.44 & 0.98 & 2.95 \\ 2.79 & 0.61 & -0.78 & 2.57 & 1.4 & 1.29 \end{pmatrix}$$

Chúng ta có thể chuẩn hóa các phần tử trong ma trận R_k theo thang 0-5.

$$R_k = \begin{pmatrix} 4.0 & 1.0 & 3.0 & 5.0 & 1.0 & 3.0 \\ 5.0 & 5.0 & 0.0 & 0.0 & 5.0 & 4.0 \\ 3.0 & 1.0 & 0.0 & 3.0 & 1.0 & 2.0 \\ 0.0 & 3.0 & 5.0 & 0.0 & 1.0 & 3.0 \\ 3.0 & 1.0 & 0.0 & 3.0 & 1.0 & 1.0 \end{pmatrix}$$

Lưu ý: Kết quả trên chỉ là tương đối với dataset nhỏ như ví dụ, tham khảo: Collaborative Filtering.

3.2 Dự đoán đánh giá

Ma trận R_k cung cấp các dự đoán điểm số cho người dùng và sản phẩm. Ví dụ, để dự đoán điểm số của người dùng i cho sản phẩm j, ta tra cứu giá trị tại vị trí (i, j) trong ma trận R_k.

3.3 Đề xuất sản phẩm

Để gợi ý sản phẩm cho người dùng, thực hiện các bước sau:

1. **Dự Đoán Điểm Số**: Sử dụng ma trận A_k để dự đoán điểm số cho tất cả các sản phẩm chưa được đánh giá bởi người dùng.
2. **Xếp Hạng Sản Phẩm**: Sắp xếp các sản phẩm dựa trên điểm số dự đoán từ ma trận A_k và gợi ý những sản phẩm có điểm số cao nhất.

4 Cài đặt

Để thuận tiện áp dụng SVD cho mô hình recommender system, chúng ta sử dụng thư viện **scikit-surprise**.

4.1 Cài đặt thư viện

Thực hiện trên terminal: **pip install scikit-surpirse**

4.2 Áp dụng thuật toán

```python
import numpy as np
import pandas as pd
from surprise import Dataset, Reader, SVD
from surprise.model_selection import train_test_split
from surprise import accuracy
from surprise.model_selection import cross_validate

# Tạo một ma trận đánh giá ngẫu nhiên 7x6
ratings_dict = {
    'user_id': np.array([1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 7]),
    'item_id': np.array([1, 2, 4, 2, 3, 1, 2, 4, 5, 2, 3, 4, 5]),
    'rating': np.array([4, 3, 5, 4, 2, 3, 2, 4, 5, 3, 4, 5, 3])
}

df = pd.DataFrame(ratings_dict)

# Định nghĩa Reader và tải dữ liệu từ DataFrame
reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(df[['user_id', 'item_id', 'rating']], reader)

# Chia dữ liệu thành tập train và test
trainset, testset = train_test_split(data, test_size=0.25)

# Áp dụng SVD
algo = SVD()
algo.fit(trainset)

# Dự đoán trên tập test
predictions = algo.test(testset)

# Tính toán độ chính xác
accuracy.rmse(predictions)

# Dự đoán cho một người dùng và một sản phẩm cụ thể
user_id = 1  # ID của người dùng
item_id = 3  # ID của sản phẩm
predicted_rating = algo.predict(user_id, item_id).est
print(f'Dự đoán đánh giá cho người dùng {user_id} với sản phẩm {item_id}: {predicted_rating:.2f}')
```

5 Thảo luận

5.1 Ưu điểm

- **Giảm Dimensionality:** SVD giúp giảm số lượng yếu tố trong ma trận đặc trưng, từ đó giảm độ phức tạp tính toán và lưu trữ. Điều này giúp cải thiện hiệu suất của hệ thống gợi ý, đặc biệt khi làm việc với dữ liệu có kích thước lớn.
- **Khả năng Generalization:** Bằng cách giữ lại các giá trị đặc trưng chính, SVD có khả năng tổng quát tốt hơn và giảm hiện tượng over-fitting, giúp mô hình hoạt động ổn định hơn với dữ liệu chưa thấy.
- **Cải thiện chất lượng gợi ý:** SVD có thể giúp cải thiện chất lượng

các gợi ý bằng cách phát hiện các mối quan hệ tiềm ẩn giữa người dùng và sản phẩm mà có thể không rõ ràng trong ma trận dữ liệu gốc.
- **Xử lý dữ liệu thưa:** SVD là một phương pháp hiệu quả để xử lý ma trận thưa, mà trong đó có nhiều giá trị bị thiếu hoặc không có, thường gặp trong các hệ thống gợi ý.

5.2 Nhược điểm

- **Chi phí tính toán cao:** Mặc dù có thể giảm kích thước dữ liệu, việc tính toán SVD cho ma trận lớn vẫn có thể tiêu tốn nhiều tài nguyên tính toán và thời gian.
- **Khả năng giải thích kém:** Các yếu tố ẩn của SVD thường không dễ giải thích, điều này có thể làm khó khăn cho việc hiểu các kết quả gợi ý và cải thiện mô hình.
- **Chưa chắc đã tối ưu hóa cho tất cả các loại dữ liệu:** SVD có thể không hoạt động tốt với một số loại dữ liệu hoặc trong các tình huống cụ thể, chẳng hạn như khi có quá nhiều yếu tố hoặc khi dữ liệu có cấu trúc đặc biệt.

6 Tài liệu tham khảo

Ian Goodfellow, Yoshua Bengio, Aaron Courville. Singular Value Decomposition. Book
Hoang Phi Dung. Eigenvalues, Orthornomal Matrices. Documentation
Pham Dinh Khanh. PCA, SVD. Blog