Multidimensional Scaling and Singular Value Decomposition

Introduction

In the wide, wide, wide world of data analysis, we often encounter high-dimensional data that's difficult to visualize and interpret. Two powerful techniques for dimensionality reduction and data visualization are Multidimensional Scaling (MDS) and Singular Value Decomposition (SVD). These methods help us uncover hidden patterns and relationships in complex datasets.

Why Use These Analyses?

1. **Dimensionality Reduction**: Both MDS and SVD can reduce high-dimensional data to a lower-dimensional space, making it easier to visualize and analyze.
2. **Pattern Recognition**: These techniques help identify underlying patterns and structures in the data that might not be apparent in the original high-dimensional space.
3. **Noise Reduction**: By focusing on the most important dimensions, these methods can help filter out noise in the data.
4. **Recommendation Systems**: In contexts like movie recommendations, these techniques can reveal latent factors that influence user preferences.

Example Dataset: Movie Ratings

Let's use a simple example of movie ratings to demonstrate these techniques. We'll use a matrix where rows represent viewers and columns represent movies.

```r
mat = matrix(c(
    1, 1, 1, 0, 0,
    3, 3, 3, 0, 0,
    4, 4, 4, 0, 0,
    5, 5, 5, 0, 0,
    0, 2, 0, 4, 4,
    0, 0, 0, 5, 5,
    0, 1, 0, 2, 2
), ncol=5, byrow=TRUE)
rownames(mat) = c('Mike', 'Cindy', 'Hyerin', 'Emily', 'Cassy', 'Juan', 'Max')
colnames(mat) = c('Star Wars', 'Alien', 'Blade Runner', 'Casablanca', 'Pretty Woman')

print(mat)
```

```
##        Star Wars Alien Blade Runner Casablanca Pretty Woman
## Mike           1     1            1          0            0
## Cindy          3     3            3          0            0
## Hyerin         4     4            4          0            0
## Emily            5     5            5          0          0
## Cassy            0     2            0          4          4
## Juan             0     0            0          5          5
## Max              0     1            0          2          2
```

The matrix above represents the ratings given by seven viewers to five movies. Each row corresponds to a viewer, and each column represents a movie, with ratings ranging from 0 (not seen) to 5 (liked a lot).

This matrix reveals distinct patterns in viewing preferences. The movies can be broadly categorized into two genres: science fiction (Star Wars, Alien, Blade Runner) and romantic classics (Casablanca, Pretty Woman). Viewers like Mike, Cindy, Hyerin, and Emily have only rated the sci-fi movies, with Emily giving perfect scores to all three, indicating a strong preference for this genre. Conversely, Juan has only watched and highly rated the romantic classics. Cassy and Max display more varied tastes, with ratings across both genres. This clear delineation suggests distinct viewer groups, such as sci-fi enthusiasts and romantic film lovers.

Our goal is to cluster these users into different types to understand their preferences better and recommend movies that align with their tastes. Similarly, clustering movies can help identify similar genres, further refining our recommendation capabilities. However, manually classifying users and movies becomes increasingly complex and time-consuming as the dataset grows. This is where techniques like MDS and SVD become invaluable. These methods can automatically identify patterns, cluster similar users and movies, and uncover hidden factors influencing viewer preferences. By employing MDS and SVD, we can create a scalable, data-driven approach to movie recommendations that goes beyond simple genre classifications and captures nuanced viewer preferences, offering a more efficient and sophisticated solution to the challenge of personalized movie recommendations.

Multidimensional Scaling (MDS)

MDS is a technique that takes a set of dissimilarities between items and returns a set of points such that the distances between the points are approximately equal to the dissimilarities.

MDS is used to visualize the level of similarity of individual cases of a dataset. It seeks to represent data points in a lower-dimensional space, while preserving the pairwise distances between them as much as possible. This is particularly useful in exploratory data analysis, where the goal is to uncover patterns or clusters within the data.

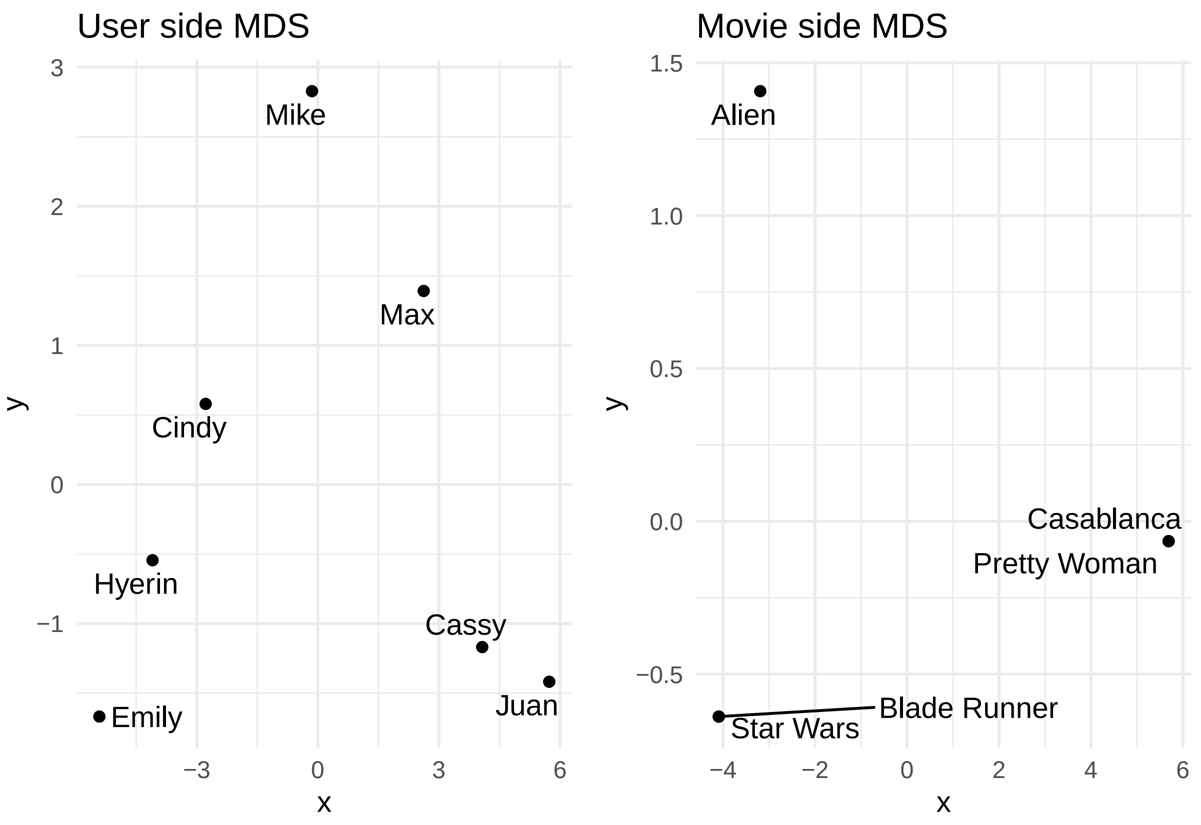
Mathematically, for a set of objects $i = 1, \ldots, n$, we start with a dissimilarity matrix $D = (d_{ij})$. The goal is to find a configuration of points $x_1, \ldots, x_n$ in a low-dimensional Euclidean space such that the distances between these points, $\|x_i - x_j\|$, approximate the original dissimilarities d_{ij}.

The objective function to minimize is typically of the form:

$$\text{Stress} = \sqrt{\frac{\sum_{i<j}(d_{ij} - \|x_i - x_j\|)^2}{\sum_{i<j} d_{ij}^2}}$$

```r
## User-side MDS
# Calculate MDS coordinates for users
mds_pos_user = cmdscale(dist(mat), eig=TRUE, k=2)
# Convert MDS results to a data frame
df_user = data.frame(mds_pos_user$points)
names(df_user) = c('x', 'y')
df_user$viewer = rownames(df_user)
rownames(df_user) = NULL

# Create a plot for user-side MDS
```

```r
viz_user_mds = ggplot(df_user, aes(x=x, y=y, label=viewer)) +
    geom_point() +
    geom_text_repel() +
    labs(title='User side MDS') +
    theme_minimal()

## Movie-side MDS
# Calculate MDS coordinates for movies (transpose matrix for movie perspective)
mds_pos_movie = cmdscale(dist(t(mat)), eig=TRUE, k=2)
# Convert MDS results to a data frame
df_movie = data.frame(mds_pos_movie$points)
names(df_movie) = c('x', 'y')
df_movie$movie = rownames(df_movie)
rownames(df_movie) = NULL

# Create a plot for movie-side MDS
viz_movie_mds = ggplot(df_movie, aes(x=x, y=y, label=movie)) +
    geom_point() +
    geom_text_repel() +
    labs(title='Movie side MDS') +
    theme_minimal()

# Combine user and movie MDS plots
viz_user_mds + viz_movie_mds
```



In these plots, we can see how users and movies are positioned relative to each other based on their similarities in the rating matrix. Users or movies that are closer together in these plots are more similar in terms of their rating patterns.

Singular Value Decomposition (SVD)

SVD is a matrix factorization method that decomposes a matrix into three other matrices. It is widely used in signal processing, statistics, and other fields to reduce the number of dimensions without losing significant information. SVD is particularly useful for compressing data, identifying the most influential components, and understanding the structure of data.

Mathematically, for a matrix $A \in \mathbb{R}^{m \times n}$, the SVD is given by:

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

where:

- $U \in \mathbb{R}^{m \times m}$ is an orthogonal matrix of left singular vectors
- $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix of singular values
- $V \in \mathbb{R}^{n \times n}$ is an orthogonal matrix of right singular vectors

In the context of our movie ratings matrix:

- U can be interpreted as a user-to-concept similarity matrix. Each column of U represents a latent concept, and each row represents how similar a user is to these concepts.
- V can be thought of as a movie-to-concept similarity matrix. Each column of V also represents a latent concept, and each row shows how similar a movie is to these concepts.
- Σ contains the singular values, which represent the strength or importance of each concept. Larger singular values indicate more important concepts.

This interpretation allows us to view the SVD as a way of uncovering hidden "concepts" or "features" in our data. These concepts might represent genres, themes, or other latent factors that influence movie preferences.

The singular values in Σ are typically arranged in descending order, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$.

For dimensionality reduction, we can approximate A by using only the first k singular values and corresponding singular vectors:

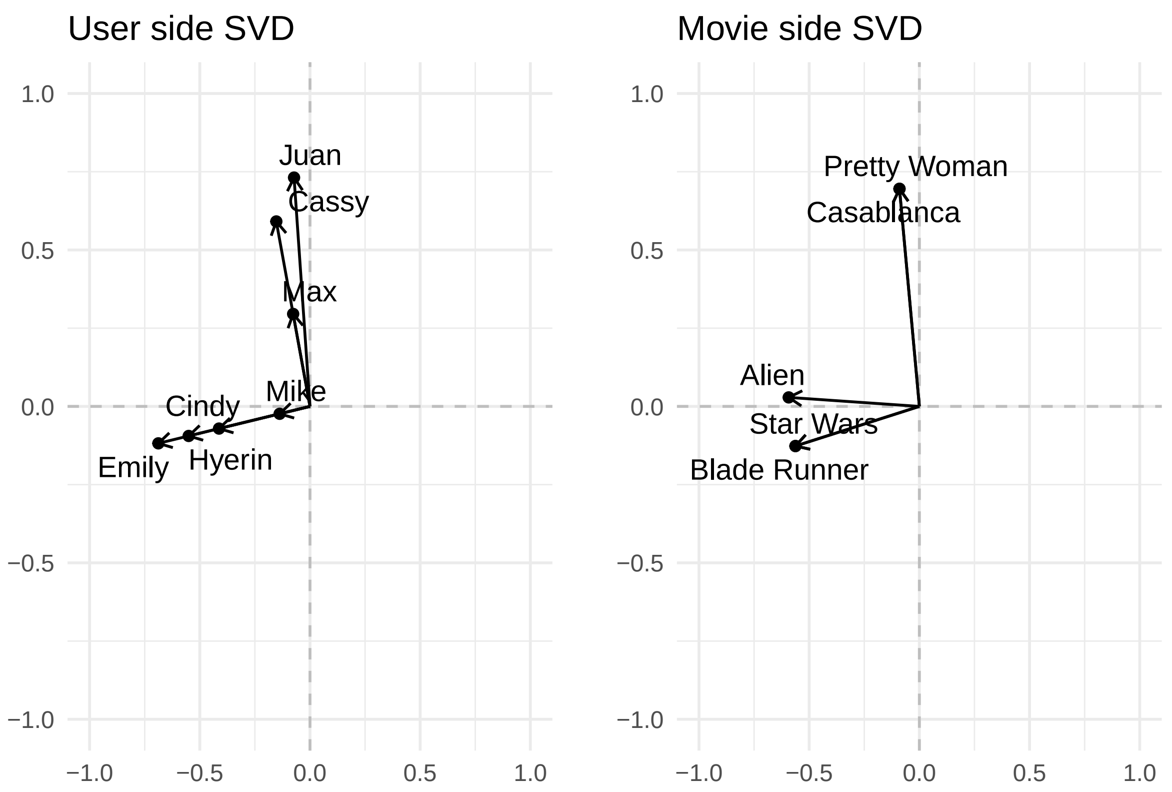
$$A \approx U_k \Sigma_k V_k^T$$

where $U_k \in \mathbb{R}^{m \times k}$, $\Sigma_k \in \mathbb{R}^{k \times k}$, and $V_k \in \mathbb{R}^{n \times k}$.

By choosing a smaller k, we can reduce the dimensionality of our data while retaining the most important concepts. This allows us to capture the essential structure of user preferences and movie characteristics in a lower-dimensional space, facilitating more efficient analysis and recommendation generation.

```r
# Perform Singular Value Decomposition (SVD)
svd_mod = svd(mat)

# Extract the first two components of U and V matrices
U = svd_mod$u[,1:2]
D = diag(svd_mod$d)[1:2,1:2]
V = svd_mod$v[,1:2]

# Prepare data frames for plotting
df_u = data.frame(U)
df_u$viewer = rownames(mat)
```

```r
df_v = data.frame(V)
df_v$movie = colnames(mat)

# Create a plot for user-side SVD
viz_user_svd = ggplot(
    df_u, aes(x=X1, y=X2, label=viewer)) +
    geom_hline(yintercept = 0, linetype = "dashed", color = "gray") +
    geom_vline(xintercept = 0, linetype = "dashed", color = "gray") +
    geom_segment(aes(x = 0, y = 0, xend = X1, yend = X2),
                 arrow = arrow(length = unit(0.2, "cm"))) +
    geom_point() +
    geom_text_repel() +
    labs(title='User side SVD', x='', y='') +
    theme_minimal() +
    xlim(-1, 1) +
    ylim(-1, 1)

# Create a plot for movie-side SVD
viz_movie_svd = ggplot(
    df_v, aes(x=X1, y=X2, label=movie)) +
    geom_hline(yintercept = 0, linetype = "dashed", color = "gray") +
    geom_vline(xintercept = 0, linetype = "dashed", color = "gray") +
    geom_segment(aes(x = 0, y = 0, xend = X1, yend = X2),
                 arrow = arrow(length = unit(0.2, "cm"))) +
    geom_point() +
    geom_text_repel() +
    labs(title='Movie side SVD', x='', y='') +
    theme_minimal() +
    xlim(-1, 1) +
    ylim(-1, 1)

# Combine user and movie SVD plots
viz_user_svd + viz_movie_svd
```

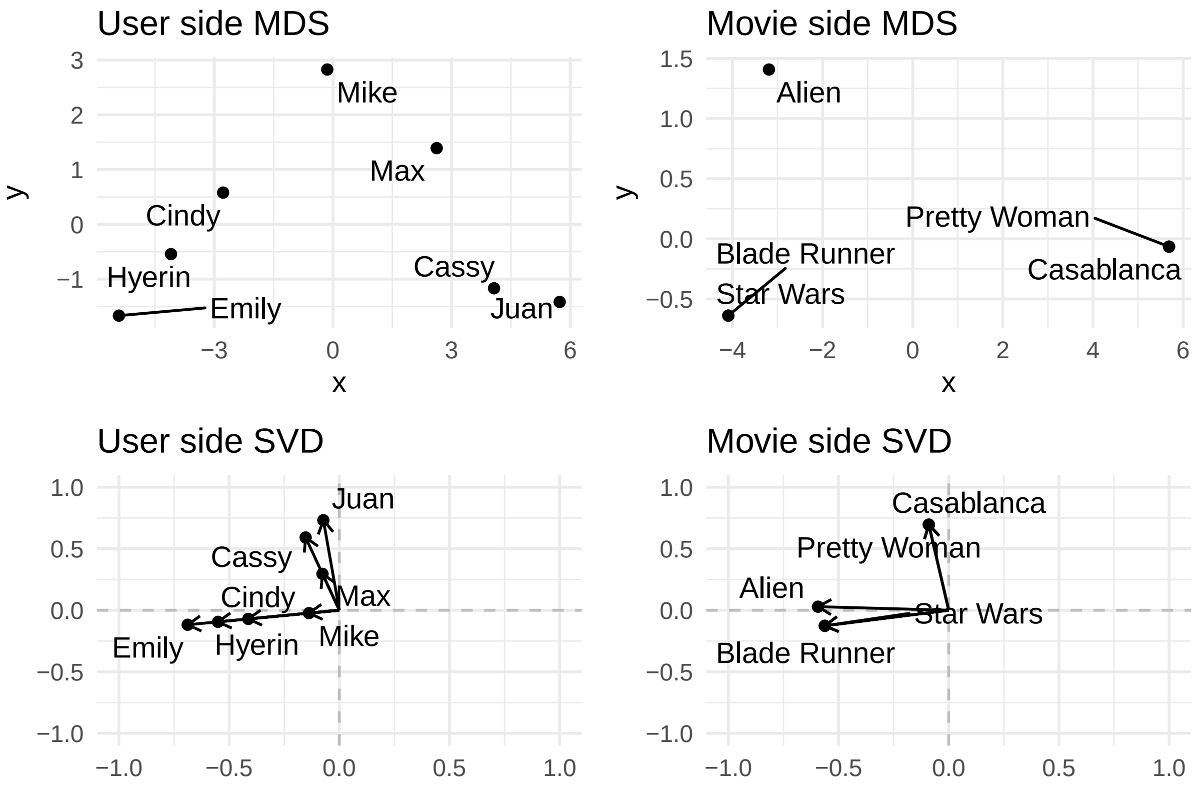


In these SVD plots, we can interpret the axes as latent factors. The position of each user or movie in this space represents how strongly they relate to these latent factors. For example, we might interpret the x-axis as representing a preference for action movies vs. romantic movies, and the y-axis as representing a preference for classic vs. modern films.

Implications for Our Movie Ratings Example

Lets show everything from the MDS (top row) and SVD (bottom row).

```
(viz_user_mds + viz_movie_mds) / (viz_user_svd + viz_movie_svd)
```



In the SVD plot, the vectors for users and movies show their relationship to the latent factors. For example, we can see that "Star Wars", "Alien", and "Blade Runner" have similar vector directions, suggesting they share common latent factors (perhaps they are all action or sci-fi movies).

In the MDS plot, the proximity of points indicates similarity. We can see similar groupings as in the SVD plot, but without the directional information. For instance, "Casablanca" and "Pretty Woman" are close to each other, suggesting similarity.

Conclusion

While both Singular Value Decomposition (SVD) and Multidimensional Scaling (MDS) are powerful techniques for dimensionality reduction, SVD is often preferred in machine learning due to its greater flexibility and broader applicability.

1. Data Representation and Versatility: SVD decomposes a data matrix into three other matrices — U, Σ, and V — each of which provides valuable insights into the structure of the data. The matrix U contains the left singular vectors, representing the relationship between data points and latent factors. Σ is a diagonal matrix of singular values, indicating the importance of each latent factor, while V contains the right singular vectors, representing the relationship between features and latent factors. This decomposition is highly versatile, allowing SVD to be applied to a wide range of data types, including image processing, text analysis, and collaborative filtering. In contrast, MDS focuses primarily on preserving pairwise distances between data points in a lower-dimensional space, which limits its use to specific types of dissimilarity data.

2. Mathematical Foundation and Computation: SVD has a strong mathematical foundation and is rooted in linear algebra, making it a fundamental tool in numerical methods. It can be efficiently computed even for large matrices, and many optimized algorithms are available for performing SVD, which is crucial in big data applications. Furthermore, the results from SVD, including the singular values, can provide direct insights into the rank and structure of the data, allowing for easy dimensionality reduction by truncating small singular values. MDS, however, often requires iterative optimization, which can be computationally intensive, especially when dealing with large datasets.

3. Application in Machine Learning: In machine learning, SVD is widely used for feature extraction, noise reduction, and matrix factorization in collaborative filtering systems, such as recommendation engines. For instance, in recommender systems, SVD can decompose user-item interaction matrices to uncover latent factors that explain user preferences and item characteristics. This factorization enables the prediction of missing entries (e.g., predicting user ratings for unseen items), a capability crucial for recommendation systems. Additionally, SVD is foundational in techniques like Principal Component Analysis (PCA), which is extensively used for feature reduction in various machine learning algorithms.

4. Handling Sparsity and Noise: SVD can effectively handle sparse and noisy data, making it suitable for applications like text mining, where term-document matrices are often sparse and high-dimensional. By focusing on the most significant singular values and corresponding vectors, SVD can reduce the dimensionality of the data while preserving its essential structure, filtering out noise and irrelevant information.

5. Flexibility and Extensions: SVD is not only limited to dimensionality reduction but also serves as a building block for numerous advanced techniques in machine learning. For example, Truncated SVD and Latent Semantic Analysis (LSA) are extensions that leverage the core principles of SVD for specific tasks. Moreover, the ability of SVD to provide orthogonal matrices allows for clear interpretations of the data transformations, which is often a requirement in machine learning tasks where interpretability is key.

Conclusion: In summary, SVD's robustness, computational efficiency, and ability to uncover latent structures in data make it a more versatile and widely used tool in machine learning than MDS. While MDS has its strengths in visualizing similarity structures, SVD's flexibility in handling diverse data types, its strong mathematical foundation, and its broad application in feature extraction, dimensionality reduction, and data compression underscore its central role in modern data analysis and machine learning.