THE UNIVERSITY OF TEXAS AT AUSTIN
Department of Statistics and Data Sciences
College of Natural Sciences

Advanced Predictive Models for Complex Data

Lecture 2: SVD for matrix completion and denoising

Purnamrita Sarkar
Department of Statistics and Data Science
The University of Texas at Austin

https://psarkar.github.io/teaching

Matrix completion (with some assumption on structure)

	Problem 1	Problem 2	Problem 3	Problem 4
Student 1	?	3	1	1
Student 2	1	4	?	2
Student 3	3	?	1	4

- The above is a matrix with rows representing students and columns representing problems in a take home exam.
- An element is how many hours a student spent on a problem.
- How is it possible to fill in this matrix? Seems pretty difficult, right?
- But what if I tell you that there is more "structure".
- For example, say each student spent 10 hours in total.

Matrix completion (with some assumption on structure)

	Problem 1	Problem 2	Problem 3	Problem 4
Student 1	5	3	1	1
Student 2	1	4	3	2
Student 3	3	2	1	4

- Say each student spent 10 hours in total.
- Now its very easy to fill in all the entries!

Matrix factorization

- Lets make this problem a bit simpler.
- Say there were no missing entries in our user/product Y matrix.
- Also assume that
  - For the i^{th} user, there is a factor vector $u_i \in \mathbb{R}^k$
  - For the j^{th} product, there is a factor vector $v_j \in \mathbb{R}^k$.

Matrix factorization

- Lets make this problem a bit simpler.
- Say there were no missing entries in our user/product Y matrix.
- Also assume that
  - For the i^{th} user, there is a factor vector $u_i \in \mathbb{R}^k$
  - For the j^{th} product, there is a factor vector $v_j \in \mathbb{R}^k$.
- Our probabilistic model is as follows:

$$Y_{ij} = u_i^T v_j + \epsilon_{ij} \qquad \epsilon_{ij} \sim N(0, \sigma^2) \tag{1}$$

Matrix factorization

- Maximum likelihood solution is equivalent to solving the least squares solution.

$$\min_{U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{n \times k}} \|Y - UV^T\|_F^2 \tag{2}$$

- Note of course, that U and V are not unique in this representation, which can be dealt with by adding more conditions like columns of U are unit norm.

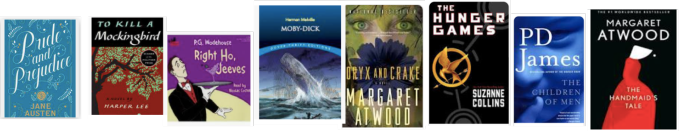
Matrix factorization

- The natural solution to this is to do a Singular Value Decomposition (SVD) of the input matrix, in which case, the solution to the above problem is given as follows.
- Let
  - U be a matrix of top k left singular vectors
  - W_k is the matrix with top k right singular vectors along its columns
  - Σ_k is the $k \times k$ diagonal matrix of top k singular values
  - Let $V = W_k \Sigma_k$
- Now set $\hat{Y} = UV^T$

What does this mean?

- Basically, we are saying that there are a “few factors” that characterize a user’s likes and dislikes.
- These could be, different genres and how an user likes one genre above another.
- Each movie also can be described using these genres.
- Now, the rating of a user on a movie is just a dot product of these two vectors.

Toy example



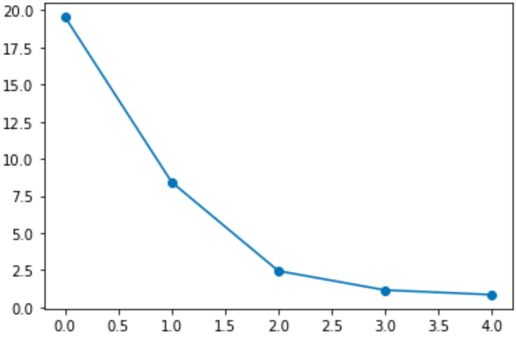
Alice	4	3	5	4	1	1	1	2
Bob	4	5	4	5	1	2	2	1
Meena	4	5	4	4	4	5	5	3
Asaf	1	1	1	1	4	4	4	5
Arthur	2	1	1	1	5	4	4	4

Figure 1: A user-book rating matrix

SVD on the toy example

```
In [49]: u, s, vt = np.linalg.svd(a)
```

```
In [53]: plot(np.arange(len(s)), s,'o-')

Out[53]: [<matplotlib.lines.Line2D at 0x134a33be0>]
```



SVD on the toy example

```
In [82]: a_reconstructed=u[:,0:2].dot(np.diag(s[0:2])).dot(vt[0:2,:])
```

```
In [88]: print('\n'.join([' '.join(['{:4}'.format(item) for item in row])
                for row in np.round(a_reconstructed,1)]))
```

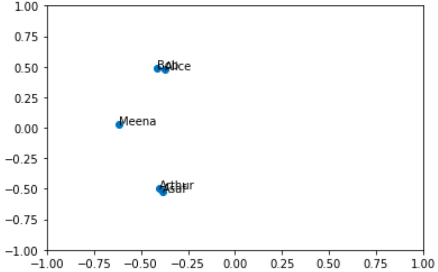
```
 3.7  3.7  4.2  4.2  0.9  1.6  1.6  1.0
 4.0  4.0  4.5  4.5  1.1  1.9  1.9  1.2
 4.3  4.2  4.3  4.3  4.1  4.5  4.5  3.9
 1.3  1.2  0.8  0.8  4.5  4.2  4.2  4.1
 1.5  1.4  1.0  1.0  4.5  4.2  4.2  4.2
```

SVD on the toy example

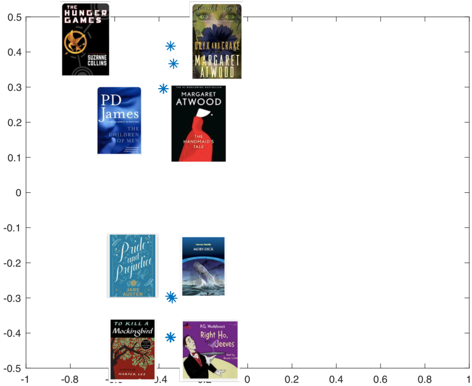
```
In [82]: a_reconstructed=u[:,0:2].dot(np.diag(s[0:2])).dot(vt[0:2,:])

In [88]: print('\n'.join([' '.join(['{:4}'.format(item) for item in row])
                for row in np.round(a_reconstructed,1)]))

 3.7  3.7  4.2  4.2  0.9  1.6  1.6  1.0
 4.0  4.0  4.5  4.5  1.1  1.9  1.9  1.2
 4.3  4.2  4.3  4.3  4.1  4.5  4.5  3.9
 1.3  1.2  0.8  0.8  4.5  4.2  4.2  4.1
 1.5  1.4  1.0  1.0  4.5  4.2  4.2  4.2

In [81]: print('\n'.join([''.join(['{:4}'.format(item) for item in row])
                for row in a]))

   4   3   5   4   1   1   1   2
   4   4   4   5   1   2   2   1
   4   5   4   4   4   5   5   3
   1   1   1   1   4   4   4   5
   2   1   1   1   5   4   4   4
```

Embedding of the users

In [12]:
```
import matplotlib.pyplot as plt
import numpy as np

names = ["Alice","Bob","Meena","Asaf","Arthur"]
n = [0 ,1, 2, 3, 4]

fig, ax = plt.subplots()
ax.scatter(u[:,0], u[:,1])
plt.xlim(-1, 1)
plt.ylim(-1, 1)
for i in n:
    ax.annotate(names[i], (u[i,0], u[i,1]))
```



Embedding of the books



- Classics with -ve y coordinate
- Dystopian novels with +ve y coordinate

Why SVD?

- Compression: SVD uses a low rank approximation to represent the original data.
  - For example, instead of storing this large possibly dense Y matrix, we could just store the U and V, thus going from $O(\text{users} \times \text{movies})$ to $O(K \times \max(\text{users}, \text{movies}))$.

Why SVD?

- Compression: SVD uses a low rank approximation to represent the original data.
  - For example, instead of storing this large possibly dense Y matrix, we could just store the U and V, thus going from $O(\text{users} \times \text{movies})$ to $O(K \times \max(\text{users}, \text{movies}))$.
- Matrix completion: while we showed an example will all entries available, SVD can be used to do matrix completion. How to fill the missing/unobserved entries?
  - By zeroes
  - Mean of the observed entries in the matrix
  - Mean of observed entries in the same column/row.

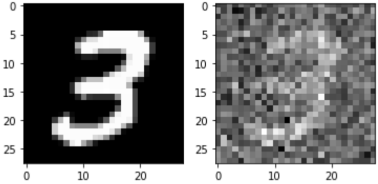
Why SVD?

- Denoising: Say there is a "ground truth" low rank signal matrix, and you only observe a noisy version of that. SVD has been used to denoise the noisy version. Lets try an example.

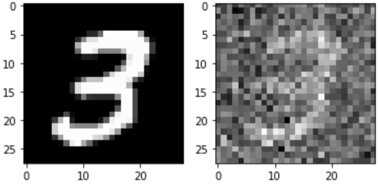
Why SVD?

- Denoising: Say there is a "ground truth" low rank signal matrix, and you only observe a noisy version of that. SVD has been used to denoise the noisy version. Lets try an example.
- In the MNIST handwritten digits dataset, each datapoint is a 28x28 image of a handwritten digit.
- We look at a matrix X each row of which represents the image for digit 3 (there are around 6000 of these in the training set).
- We add a whole bunch of noise to this.

Why SVD?

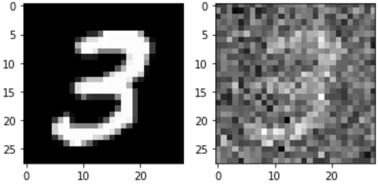


Why SVD?



- Now we do SVD on the data matrix, i.e. $X = USV^T$

Why SVD?



- Now we do SVD on the data matrix, i.e. $X = USV^T$

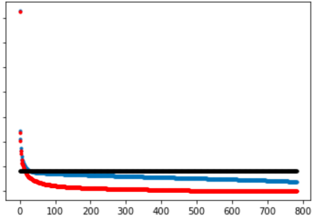


Figure 2: Plot of singular values. Red line is for signal. Blue for noisy signal and black for the added noise.

Reconstruction with top 5 singular vectors

- Picking the right number of vectors is crucial
- Too small–you lose some signal
- Too large–you start including noise

Reconstruction with top 5 singular vectors

- Picking the right number of vectors is crucial
- Too small–you lose some signal
- Too large–you start including noise

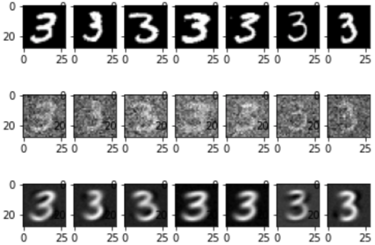


Figure 3: First row signal. Second row noisy signal. Third row reconstructed signal

Reconstruction with top 5 singular vectors

- So what exactly happened there?
- Somehow the top 5 singular vectors of a matrix picked up the signal
- In other words, we can use SVD to "pick out" the most interesting directions.
- Time to talk about PCA! This will be our next set of lectures.