Block 2

Recommender Systems II

Janis Keuper

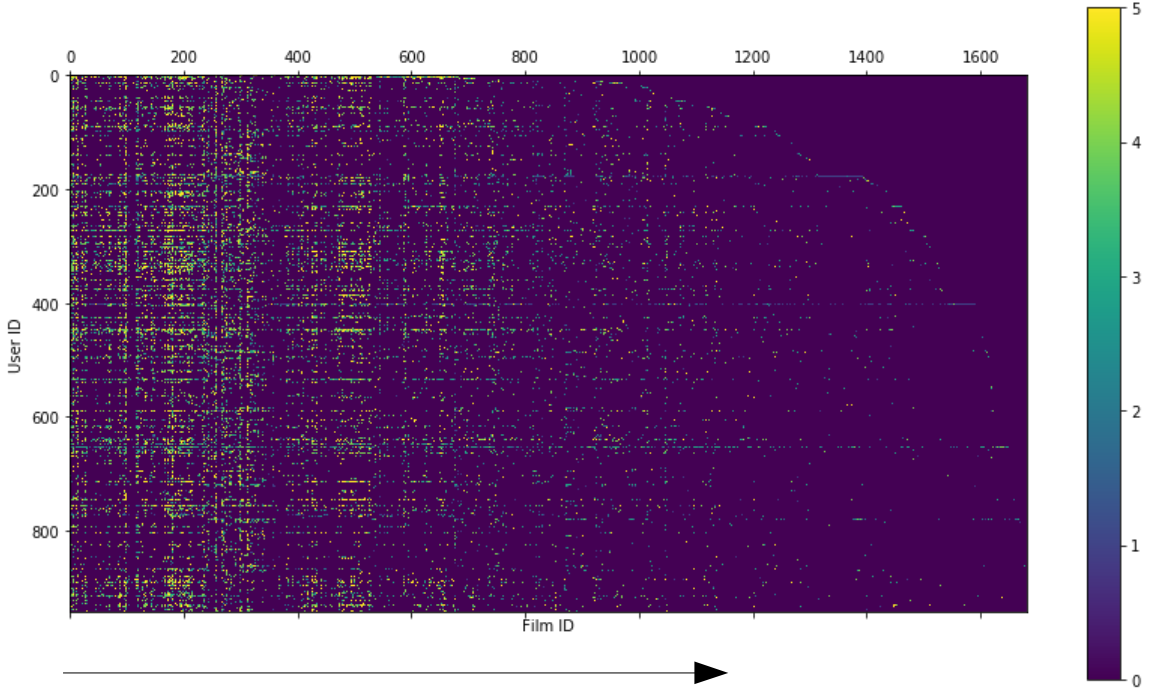
Basics of Recommender Systems

Complexity of Collaborative Filter Approach

build Matrix

R of size $n \times m$ with context measure r_{ij}, for $i \in 1 \dots n, j \in 1 \dots m$

User ID



Item ID

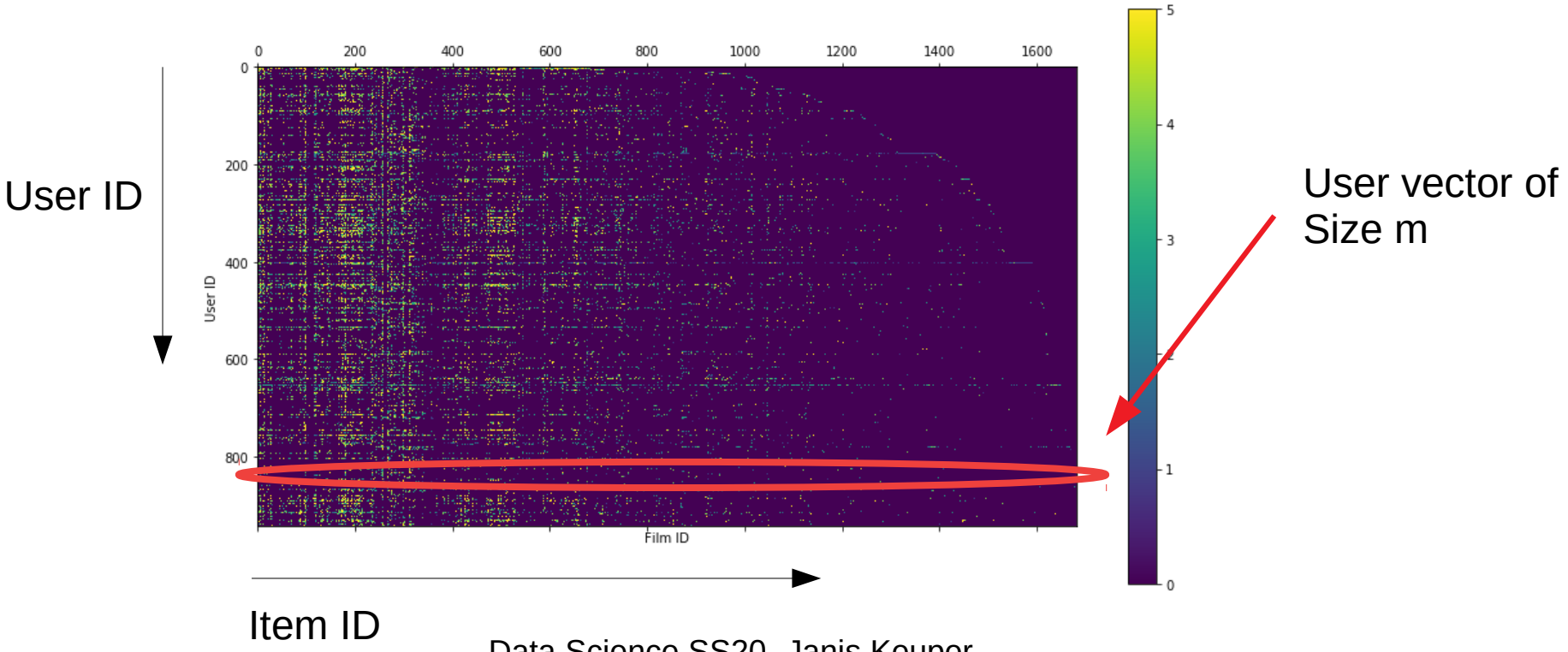
Basics of Recommender Systems

Hochschule Offenburg
offenburg.university

Number of Users n

build Matrix

R of size $n \times m$ with context measure r_{ij}, for $i \in 1 \ldots n, j \in 1 \ldots m$



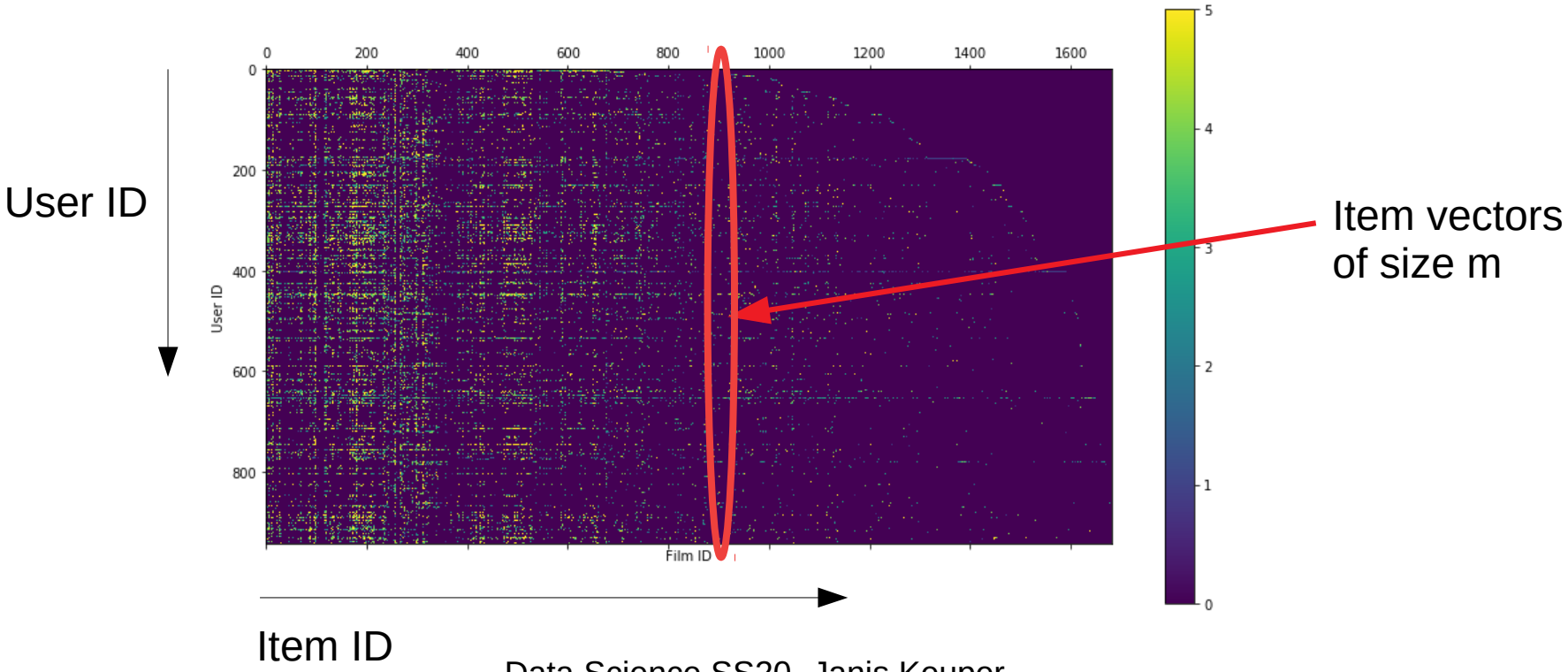
Basics of Recommender Systems

Hochschule Offenburg
offenburg.university

Number of items m

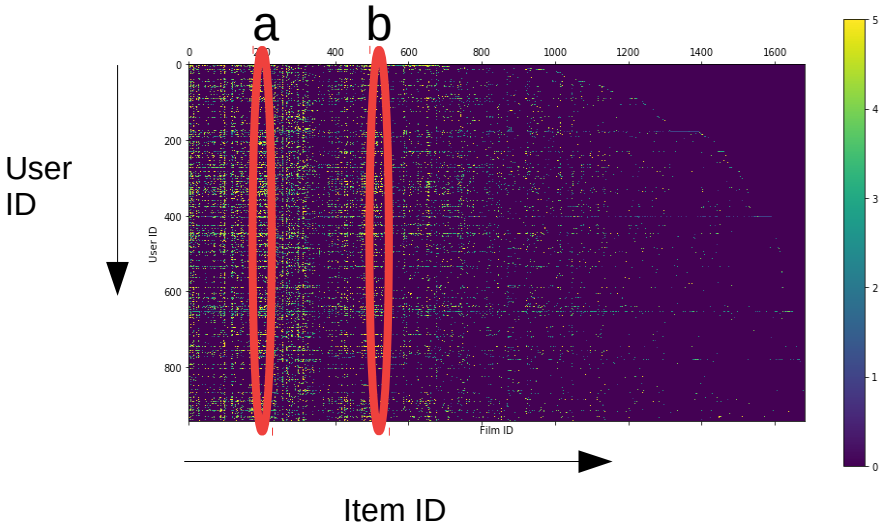
build Matrix

R of size $n \times m$ with context measure r_{ij}, for $i \in 1 \ldots n, j \in 1 \ldots m$



Basics of Recommender Systems

Compute Distances between items



Items are encoded in column vectors

→ distance between items: vector distance

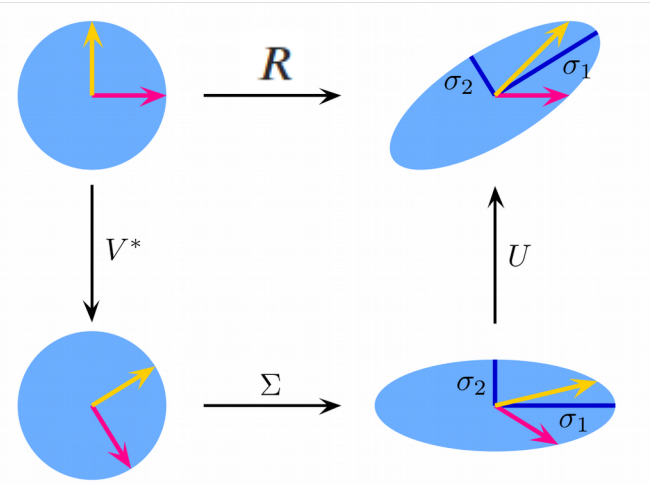
$$d_{cos}(\vec{a}, \vec{b}) := \frac{< \vec{a}, \vec{b} >}{|\vec{a}||\vec{b}|}$$

e.g. cosine distance

Problem I: R very large and very sparse (e.g. Amazon: 300 million items!)
Problem II: R is changing over time → need to recompute

Basics of Recommender Systems

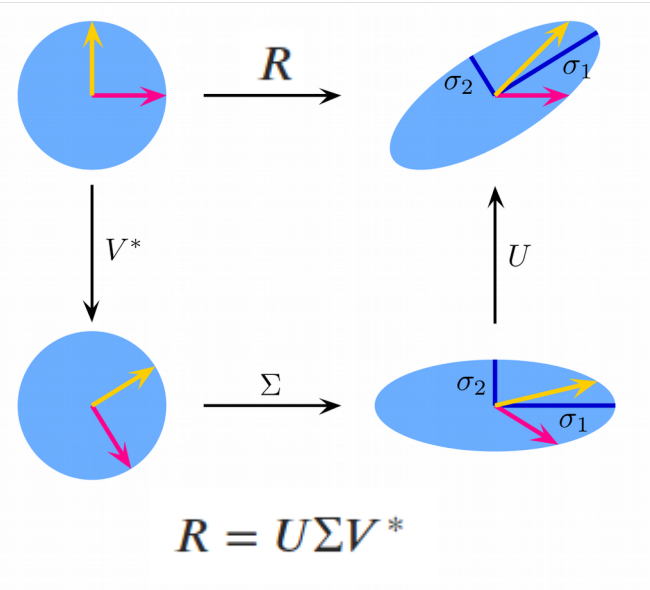
Use SVD to reduce size of R



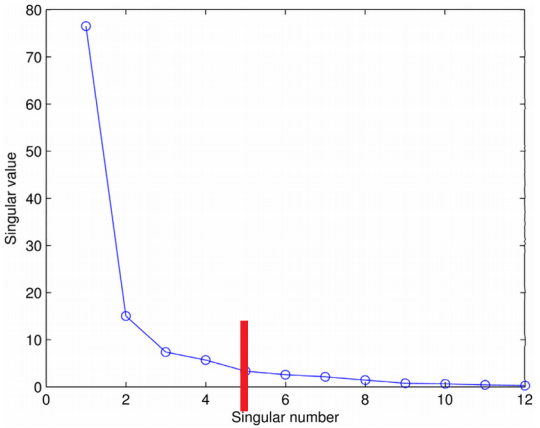
- U is an $m \times m$ unitary ($U^*U = I$) matrix,
- Σ is a diagonal $m \times n$ matrix with non-negative real numbers, the **singular values**, on the diagonal,
- V is an $n \times n$ unitary matrix, and V^* is the conjugate transpose of V.

Basics of Recommender Systems

Use SVD to reduce size of R



Reduce by selecting only the h largest Singular Values



Example plot of typical singular Values, here h=5

Basics of Recommender Systems

Approximate R by SVD

$$\tilde{R} = \tilde{U}\tilde{\Sigma}\tilde{V}^*$$

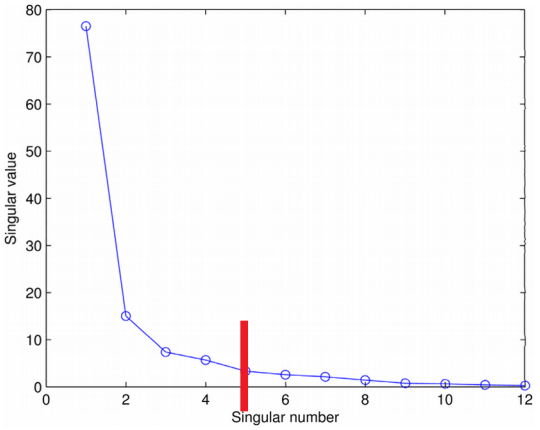
```
In [33]: #now reconstruct with loss, using only th first 2 of 4 singular values
         np.dot(U[:,:2]*S[:2],V[:2,:])

Out[33]: array([[1., 0., 0., 0.],
                [0., 0., 0., 0.],
                [0., 3., 0., 0.],
                [0., 0., 0., 0.],
                [2., 0., 0., 0.]])
```

→ shorten dimensions of m and n by constant factor

→ reduce both to quite small sizes (e.g. 500) without loosing a lot of accuracy!

Reduce by selecting only the h largest Singular Values



Example plot of typical singular Values, here h=5

Basics of Recommender Systems

Hochschule Offenburg
offenburg.university

Approximate R by SVD

$$\tilde{R} = \tilde{U}\tilde{\Sigma}\tilde{V}^*$$

```
In [33]: #now reconstruct with loss, using only th first 2 of 4 singular values
         np.dot(U[:,:2]*S[:2],V[:2,:])

Out[33]: array([[1., 0., 0., 0.],
                [0., 0., 0., 0.],
                [0., 3., 0., 0.],
                [0., 0., 0., 0.],
                [2., 0., 0., 0.]])
```

→ shorten dimensions of m and n by constant factor

→ reduce both to quite small sizes (e.g. 500) without loosing a lot of accuracy!

Problems with standard SVD:

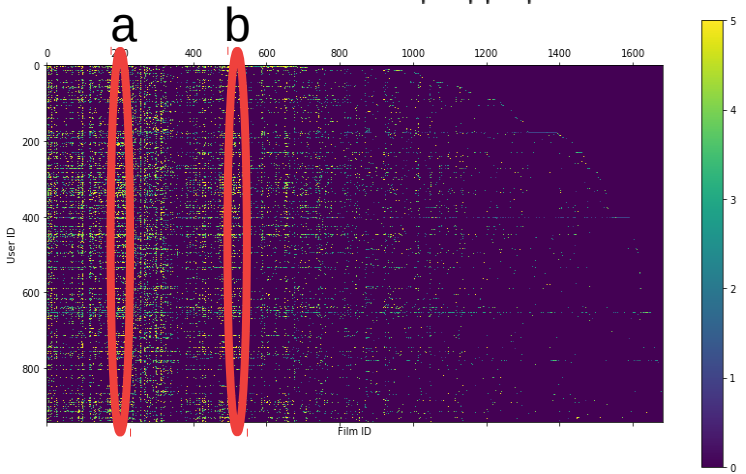
- Need full Matrix R to compute it‘s SVG
- SVG is quite expensive to compute
- R is changing and we would have to re-compute SVG every time…

Basics of Recommender Systems

Approximate R by SVD

$$\tilde{R} = \tilde{U}\tilde{\Sigma}\tilde{V}^*$$

$$d_{cos}(\vec{a}, \vec{b}) := \frac{< \vec{a}, \vec{b} >}{|\vec{a}||\vec{b}|}$$



Solution:

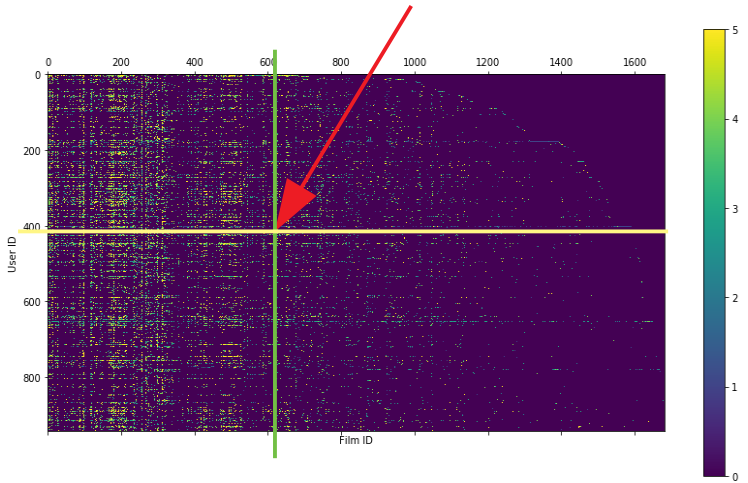
- We do not need the full R Matrix at any time
  → need only item vectors to compute distance
- Use a vector by vector SVD (next slide)
- Store and update vectors independently, only recompute for those items/users were changes happen
- Compute distance for every query, based on SVD approximated vectors a and b

Basics of Recommender Systems

Standard Approach for Collaborative Filters: rSVD

Use regularized Singular Value Decomposition (rSVG) to handle huge recommendation matrix

$$expected\ rating = \hat{r}_{ui} = q_i^T p_u$$



Approximate entry in rating matrix by vector factorization

Netflix Prize and SVD

Stephen Gower

April 18th 2014

Abstract

Singular Value Decompositions (SVD) have become very popular in the field of Collaborative Filtering. The winning entry for the famed Netflix Prize had a number of SVD models including SVD++ blended with Restricted Boltzmann Machines. Using these methods they achieved a 10 percent increase in accuracy over Netflix's existing algorithm.

In this paper I explore the different facets of a successful recommender model. I also will explore a few of the more prominent SVD based models such as Iterative SVD, SVD++ and Regularized SVD. This paper is designed for a person with basic knowledge of decompositions and linear algebra and attempts to explain the workings of these algorithms in a way that most can understand.

Introduction

On October 2nd, 2006 Netflix began their contest to find a more accurate movie recommendation system to replace their current system, Cinematch. They promised a prize of one million dollars to anyone who could improve over the Cinematch system by at least 10% Root Mean Squared error (RSME). After three years of the contest, in 2009 the grand prize was awarded to team "Bellkor's Pragmatic Chaos", much of this paper draws from papers written by one of the members of the team, Yehuda Koren.

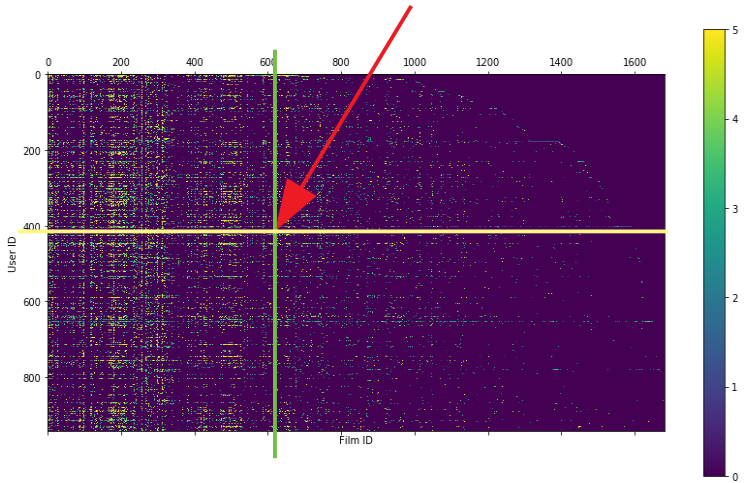
But before one can examine the algorithm developed to win the prize, first it must be known what must be addressed. Recommender systems are important to services such as Amazon, Netflix and other such online providers that aim to attract users. For Amazon, they want to present their users with products they want so they are more likely to spend their money through their site. On the part of Netflix, their motivation is to help users find movies and shows that fit their tastes because they are more likely to maintain a subscription.

Basics of Recommender Systems

Standard Approach for Collaborative Filters: rSVD

Use regularized Singular Value Decomposition (rSVG) to handle huge recommendation matrix

$$expected\ rating = \hat{r}_{ui} = q_i^T p_u$$



Approximate entry in rating matrix by vector factorization

Again, dimension of q and p are much smaller than the numbes of users and items...

Netflix Prize and SVD

Stephen Gower

April 18th 2014

Abstract

Singular Value Decompositions (SVD) have become very popular in the field of Collaborative Filtering. The winning entry for the famed Netflix Prize had a number of SVD models including SVD++ blended with Restricted Boltzmann Machines. Using these methods they achieved a 10 percent increase in accuracy over Netflix's existing algorithm.

In this paper I explore the different facets of a successful recommender model. I also will explore a few of the more prominent SVD based models such as Iterative SVD, SVD++ and Regularized SVD. This paper is designed for a person with basic knowledge of decompositions and linear algebra and attempts to explain the workings of these algorithms in a way that most can understand.

Introduction

On October 2nd, 2006 Netflix began their contest to find a more accurate movie recommendation system to replace their current system, Cinematch. They promised a prize of one million dollars to anyone who could improve over the Cinematch system by at least 10% Root Mean Squared error (RSME). After three years of the contest, in 2009 the grand prize was awarded to team "Bellkor's Pragmatic Chaos", much of this paper draws from papers written by one of the members of the team, Yehuda Koren.

But before one can examine the algorithm developed to win the prize, first it must be known what must be addressed. Recommender systems are important to services such as Amazon, Netflix and other such online providers that aim to attract users. For Amazon, they want to present their users with products they want so they are more likely to spend their money through their site. On the part of Netflix, their motivation is to help users find movies and shows that fit their tastes because they are more likely to maintain a subscription.

Basics of Recommender Systems

Standard Approach for Collaborative Filters: rSVD

Use regularized Singular Value Decomposition (rSVG) to handle huge recommendation matrix

$$expected\ rating = \hat{r}_{ui} = q_i^T p_u$$

Approximate entry in rating matrix by vector factorization

$$minimum(p,q) \sum_{(u,i)\epsilon K} \left(r_{ui} - q_i^T . p_u\right)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

In practice: regularized optimization via
Stochastic Gradient Descent

Netflix Prize and SVD

Stephen Gower

April 18th 2014

Abstract

Singular Value Decompositions (SVD) have become very popular in the field of Collaborative Filtering. The winning entry for the famed Netflix Prize had a number of SVD models including SVD++ blended with Restricted Boltzmann Machines. Using these methods they achieved a 10 percent increase in accuracy over Netflix's existing algorithm.

In this paper I explore the different facets of a successful recommender model. I also will explore a few of the more prominent SVD based models such as Iterative SVD, SVD++ and Regularized SVD. This paper is designed for a person with basic knowledge of decompositions and linear algebra and attempts to explain the workings of these algorithms in a way that most can understand.

Introduction

On October 2nd, 2006 Netflix began their contest to find a more accurate movie recommendation system to replace their current system, Cinematch. They promised a prize of one million dollars to anyone who could improve over the Cinematch system by at least 10% Root Mean Squared error (RSME). After three years of the contest, in 2009 the grand prize was awarded to team "Bellkor's Pragmatic Chaos", much of this paper draws from papers written by one of the members of the team, Yehuda Koren.

But before one can examine the algorithm developed to win the prize, first it must be known what must be addressed. Recommender systems are important to services such as Amazon, Netflix and other such online providers that aim to attract users. For Amazon, they want to present their users with products they want so they are more likely to spend their money through their site. On the part of Netflix, their motivation is to help users find movies and shows that fit their tastes because they are more likely to maintain a subscription.

Literature

[0] free icons taken from https://www.flaticon.com

[1] www.amazon.de

[2] www.netflix.de

[3] www.spotify.com

[4] www.zalando.de

[5] https://blogs.gartner.com/martin-kihn/how-to-build-a-recommender-system-in-python/

[6] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.895.3477&rep=rep1&type=pdf

[7] https://www.designmantic.com/blog/movie-moods-in-typography/

[8] https://arxiv.org/abs/1205.3193