UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Applied Machine Learning at Scale 813

Recommender Systems

Thomas Marshall

21569967

April 16, 2022

Contents

1 Introduction

1.1 Recommender Systems

The primary goal of recommender systems is to predict the preferences of users. In a streaming context, this translates to serving the user with movie recommendations that best suit them. An online retailer might want to recommend items for the user to buy, in a bid to convince them to spend more money in the online store.

1.2 Motivation

Developing sophisticated recommendation engines can be instrumental in driving engagement on modern online platforms, as well as boosting expected revenue. Developing recommendation engines using machine learning typically trumps naive approaches.

1.3 Machine learning

Proper recommendation engines necessitate machine learning. There are several such approaches to developing recommendation engines; popular choices include feature-based, nearest neighbour-based methods as well as collaborative filtering. This document will focus on the collaborative filtering solution, because it is scalable and produces diverse sets of recommendations.

This document considers both cases where the user provides explicit and implicit feedback and presents the development of machine learning recommender systems that are suited for each type of information respectively.

1.4 The dataset

The MovieLens dataset will be used to develop these recommender systems. MovieLens was chosen because it is the MNIST of recommendation. MovieLens was also used by some teams during the Netflix prize competition. A short analysis of the MovieLens dataset follows in the subsequent section.

2 Theory

2.1 Data Analysis

Many interesting qualitites can be observed in the MovieLens dataset. The rating distribution in figure 1 shows that users prefer rating whole numbers over fractions. Already the data is speaking, and suggesting that online applications might be able to improve the quality of their user feedback by only offering integer ratings. This kind of idea would be a candidate for A/B testing.

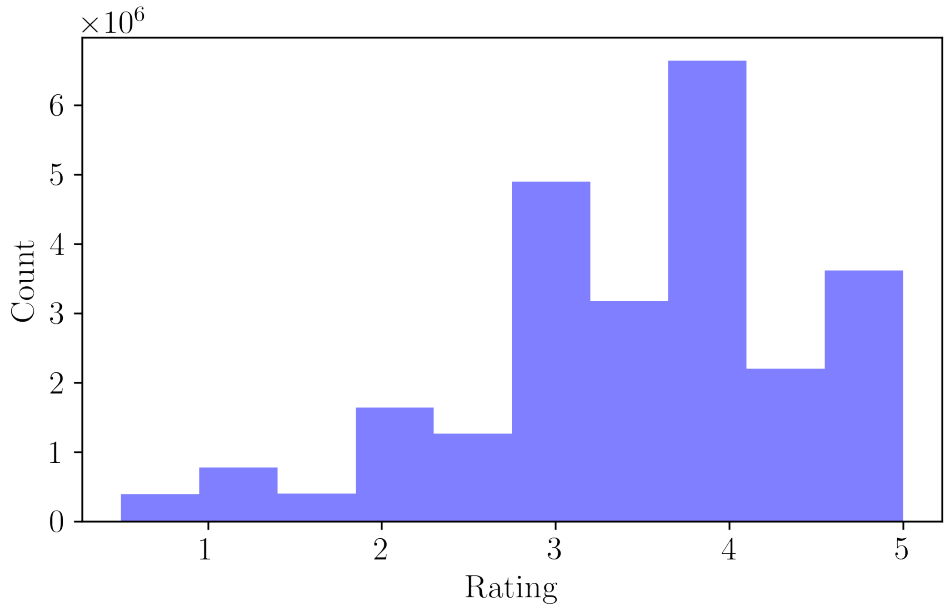

Figure 1: Distribution of movie ratings

Figures 2 and 3 show logarithmic plots of the users and movies ranked by the number of interactions they respectively had. This is significant, because the plots present a near linear relationship. This suggests that the majority of ratings are concentrated in the minority of users and vice versa. Hence there is a power law present in the data. This is also scale free, because it does not change with dataset size. The apparent linearity of the plots also suggests that the MovieLens data is relatively untampered with.

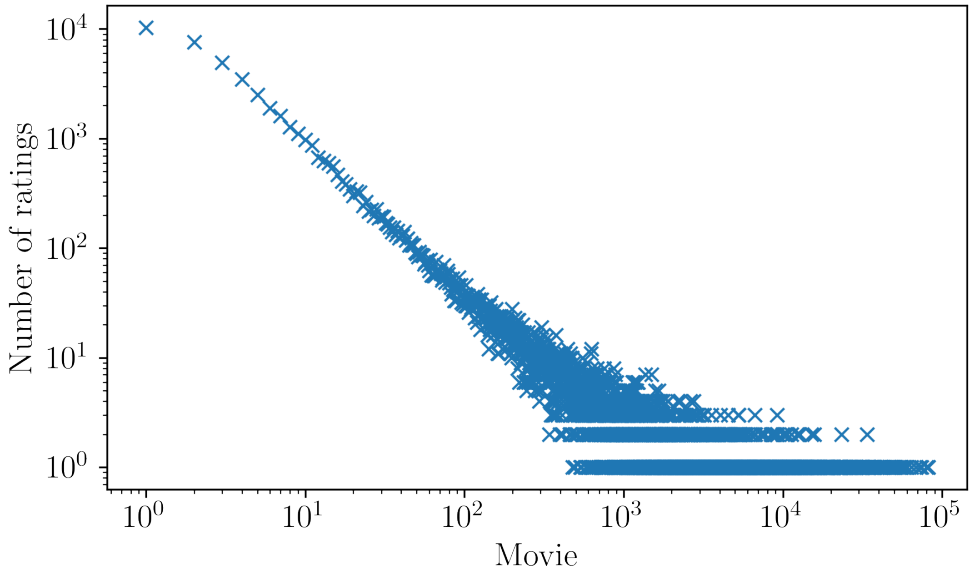

Figure 2: Ranked logarithmic plot of the number of ratings by each movie

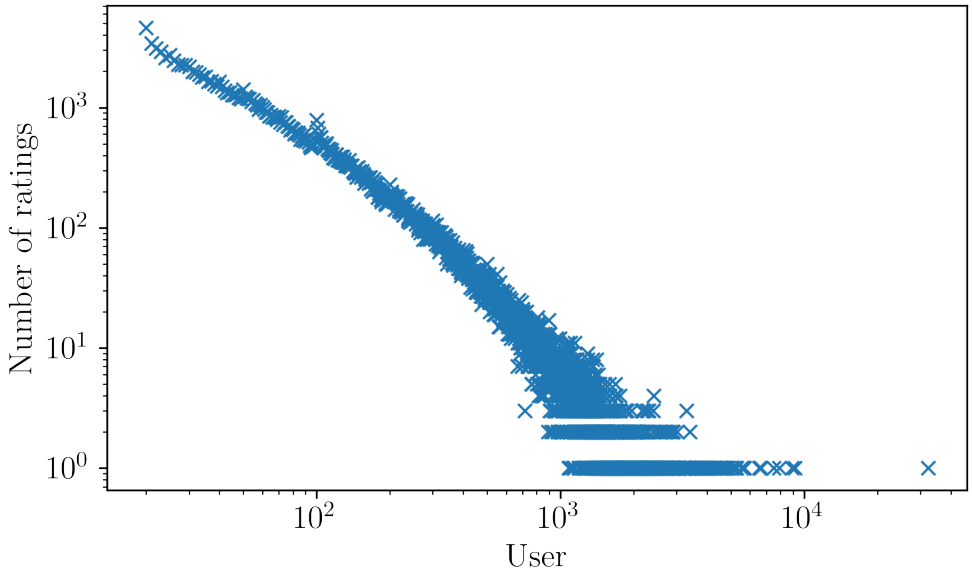

Figure 3: Ranked logarithmic plot of the number of ratings by each user

2.2 Matrix Factorisation

The situation in a recommendation setting can be described by a graph consisting of users and items. Some of the users and items are connected to one another; these edges represent interactions, for instance the user rated the movie. These interactions can be arranged into a large interaction matrix of size movies $\times$ users.

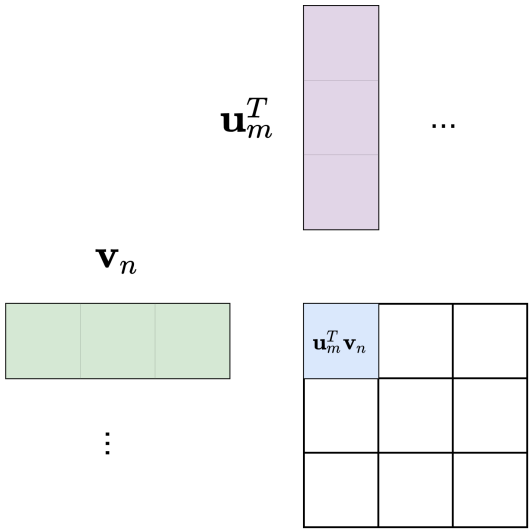

Figure 4: Matrix Factorisation

The power laws in the previous section indicate that the interaction matrix will be very sparse, seeing as a minority of users provided nearly all the ratings and a minority of movies received nearly all the ratings. If oracular knowledge of this matrix was available, recommendation would be a trivial task. Unfortunately, this is not the case, some other means must be pursued to estimate the missing values in the sparse interaction matrix.

Matrix factorisation [1] is the idea that every entry in a matrix can be factorised into the inner product of two k-dimensional vectors. Every rating r in the dataset becomes the inner product of some k-dimensional user vector $\mathbf{u}_m$ and some k-dimensional item vector $\mathbf{v}_n$,

$$r_{mn} = \mathbf{u}_m^T \mathbf{v}_n \tag{1}$$

Using this formulation, it is easy to see that any missing user-movie interactions (ratings) can be filled in simply by taking the inner product of the particular user and item vectors.

2.3 Alternating Least Squares

The objective of the alternating least squares (ALS) algorithm is to consider maximum a posteriori estimation on each target variable considered, and then to alternate execution of the update rules. The full derivation can be found in the appendix of this document. The

result of this is the following update rule for the user vector (and biases)

$$\mathbf{z}_m = \left(\lambda \sum_{n \epsilon \Omega(m)} \mathbf{y}_n \mathbf{y}_n^T + \tau \mathbf{I}^* \right)^{-1} \left(\lambda \sum_{n \epsilon \Omega(m)} x_{mn} \mathbf{y}_n \right) \tag{2}$$

Where $n \epsilon \Omega(m)$ is the set of items that user m has interacted with. Similarly the update rule for the item vectors (and biases) is

$$\mathbf{z}_n = \left(\lambda \sum_{m \epsilon \Omega(n)} \mathbf{y}_m \mathbf{y}_m^T + \tau \mathbf{I}^* \right)^{-1} \left(\lambda \sum_{m \epsilon \Omega(n)} x_{mn} \mathbf{y}_m \right) \tag{3}$$

The ALS algorithm is executed by alternating these two aforementioned update equations for the users and the items.

λ is the variance of the prior for rating predictions, this has been chosen to have unit variance as the prior for predictions should be relatively wide. The regularisation parameter was chosen to be $\tau = \frac{3\sqrt{k}}{2.5}$, where k is the number of dimensions in the latent space. A full derivation for this choice can be found in the appendix of this document.

2.4 Bayesian personalized ranking

Bayesian personalized ranking (BPR) [2] is a method that can be used to train a collaborative filtering solution when the only feedback available from the users is implicit, for instance indicating whether or not the user enjoyed an interaction based on some metric. BPR presents a maximum posterior estimator for personalised ranking. The optimisation critereon for personalised ranking is given by

$$\text{BPR-OPT} = \sum_{(u,i,j)} \ln \sigma(\hat{x_{uij}}) - \lambda_\Theta ||\Theta||^2 \tag{4}$$

where $\hat{x_{uij}}$ is the difference between predicted feedback for user u and positive example i and negative example j respectively.

This BPR-OPT function can then be optimised via SGD. The update rule is given by

$$\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-x_{\hat{u}ij}}}{1 + e^{-x_{\hat{u}ij}}} \cdot \frac{\delta}{\delta \Theta} \hat{x_{uij}} + \lambda_\Theta \Theta \right) \tag{5}$$

Bootstrap sampling is used to update (u, i, j) triples instead of simply user-wise iteration to signifcantly speed up convergence.

2.5 Performance metrics

The root-mean-square error (RMSE) can be used to evaluate the performance of the ALS recommender. RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{1}{|R|} \sum_{(u,i) \epsilon R} (\hat{r_{ui}} - r_{uj})} \tag{6}$$

RMSE is not adequate for evaluating the performance of the implicit feedback model because the goals do not align. the Area Under the Curve (AUC) metric is more appropriate, since BPR is essentially classifying items as the user would like or the user would not like. This metric integrates the graph of true positives vs false positives for a classifier and produces a number between 0 and 1, where 1 is the best.

3 Implementation

The code for the algorithms described in this section can be found on my github, as linked in the appendix of this document.

3.1 Data pipeline

The MovieLens dataset is stored in comma separated value (CSV) files. The ratings file is read into a Pandas DataFrame. This is a large table of ratings that particular users gave to particular movies. There is one problem evident in the data, the movie IDs are not contiguous from zero up to the number of unique movies. This will complicate the implementation to come. A simple map and corresponding inverse map is created to solve this problem.

Accessing ratings of movies by users using Pandas DataFrames has $O(N)$ asymptotic complexity, because the DataFrame must be searched to find the desired quanity. Linear access time is very slow, and the implementation would benefit greatly from reducing this. The solution is to index this information in a more efficient data structure such that $O(1)$ access time is attained. This can be accomplished by

1. Storing the rating records in a Numpy array.
2. Keeping the index range for the records of a particular user in a separate Numpy array, indexed by the user ID.
3. Keeping the movie range for the records of a particular user in a separate Numpy array, indexed by the movie ID.

This system allows for $O(1)$ access to the required information.

3.2 Implicit dataset

Since no implicit user feedback dataset is available, the MovieLens dataset will be converted into such a dataset, simply by setting all the ratings larger than 2.5 to 1 and the rest to 0. This is not an ideal solution, since each user has their own biases in terms of what ratings they consider to be "good" and "not good".

3.3 Numerical stability

In the implementation of ALS it is required to calculate the inverse matrix, however this operation can be numerically unstable if the matrix in question turns out to have a zero determinant. In the case of ALS, the matrix being inverted is gauranteed to always be positive semidefinite (PSD) because it is formed from the matrix product of some vector with itself. The solution is to backsolve the system making use of the Cholesky decomposition,

$$\begin{aligned} \mathbf{Az} &= \mathbf{c} \\ \mathbf{LL}^T\mathbf{z} &= \mathbf{c} \\ \therefore \mathbf{z} &= \mathbf{L}^T\backslash(\mathbf{L}\backslash\mathbf{c}) \end{aligned} \tag{7}$$

3.4 Parallelisation

No parallelisation was implemented. However, ALS is trivially paralellisable, because it is required to iterate over every user/movie, and every iteration of this is independent from every other iteration. The iterations in this loop can be farmed out to a pool of worker threads for substantial processing speed gains.

For BPR a similar strategy may be pursued, except more care must be taken. It is required that samples being processed in parallel do not pertain to the same user, observed item or unobserved item. If this criteon is met, BPR is also easily paralellisable. In both cases this work can also be distributed on a network of computing nodes.

3.5 Alternating Least Squares

The following pseudocode documents the ALS algorithm as it has been implemented in Python code, followed by the RMSE over epochs when the algorithm was executed. A latent dimension size of $k = 20$ was used.

Algorithm 1 Alternating Least Squares

Input: $\mathbf{z}_m, \mathbf{z}_n, \lambda, \tau, N$
Output: $\mathbf{z}_m, \mathbf{z}_n$, all latent vectors and biases of the users and the items

```
k ← 0
while k < N do                                   ▷ Execute N epochs
    m ← 0
    while m < # users do                         ▷ Update every user vector
        z_m ← (λ Σ_n y_n y_n^T + τ I*)^{-1} (λ Σ_n x_mn y_n)
        m ← m + 1
    end while

    n ← 0
    while n < # item do                          ▷ Update every item vector
        z_n ← (λ Σ_m y_m y_m^T + τ I*)^{-1} (λ Σ_m x_mn y_m)
        n ← n + 1
    end while

    k ← k + 1
end while
return z_m, z_n, all latent vectors
```

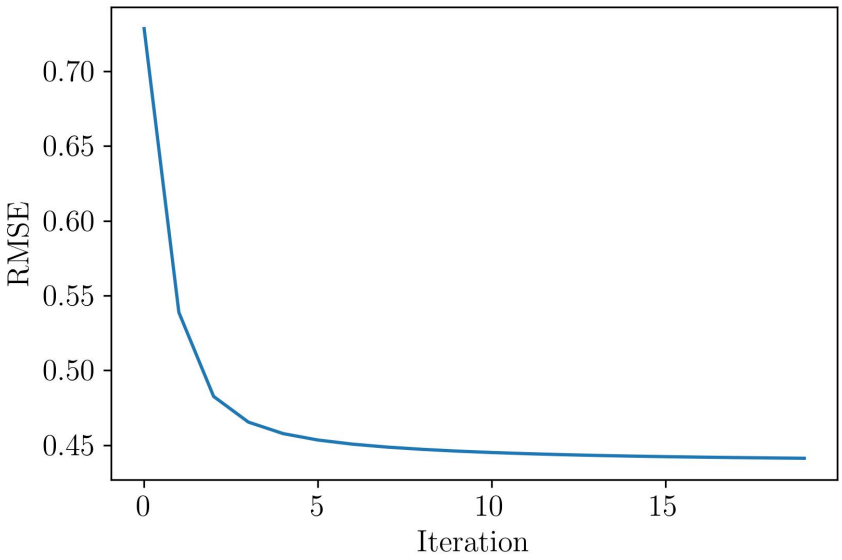

Figure 5: RMSE over training epochs for ALS

Adding new users or items

New users or items can be added by introducing a new initialised user or item vector into the system. To incorporate this information into the system, ALS must be executed only pertaining to interactions involving this new vector. For best results, the new vector should have more rather than less ratings to start with.

Incuding movie features

There are several ways to include additional movie features into the ALS algorithm. Additional movie features could include information like the genres of the film. Some embedding of this information would need to be generated.

One way to do this could be to encode every possible genre in the dataset with a one-hot vector, then concatenating the the one-hot vectors to form a feature vector for each movie. This concatenated vector will then remain fixed. It is equivalent to specifying a fixed prior over a part of the latent space. However, this approach is in danger of blowing up the latent vector dimensions.

An interesting way to extend this formulation might be to consider training a vector quantised variational autoencoder to find a finite set of relatively low dimensional vectors that can represent each kind of movie. This assumes that there is a finite number of kinds of movies; this is a reasonable assumption. Using this method will endow the system with smaller and more representative feature vectors to kickstart the content-based part of the item latent representation with.

3.6 Bayesian Personalised Ranking

The following pseudocode documents the BPR algorithm as it has been implemented in Python code, followed by the AUC over epochs when the algorithm was executed. A latent dimension size of $k = 20$ was used.

Algorithm 2 Bayesian Personalised Ranking

Input: $\mathbf{u}_m, \mathbf{v}_n, \lambda, \alpha, N$
Output: $\mathbf{u}_m, \mathbf{v}_n$, all latent vectors of the users and the items

$k \leftarrow 0$
while $k < N$ **do** ▷ Execute N gradient updates
  $u \sim U(0, \#\text{ users})$
  $i \leftarrow$ uniformly sampled from the items of u
  $j \leftarrow$ uniformly sampled from the items u has not seen
  $\hat{x_{uij}} \leftarrow \mathbf{u}_u\mathbf{v}_i - \mathbf{u}_u\mathbf{v}_j$
  $\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x_{uij}}}}{1+e^{-\hat{x_{uij}}}} \cdot \frac{\delta}{\delta\Theta}\hat{x_{uij}} + \lambda_\Theta\Theta \right)$ ▷ Update all three vectors $\mathbf{u}_u, \mathbf{v}_i, \mathbf{v}_j$
  $k \leftarrow k + 1$
end while
return $\mathbf{u}_m, \mathbf{v}_n$, all latent vectors

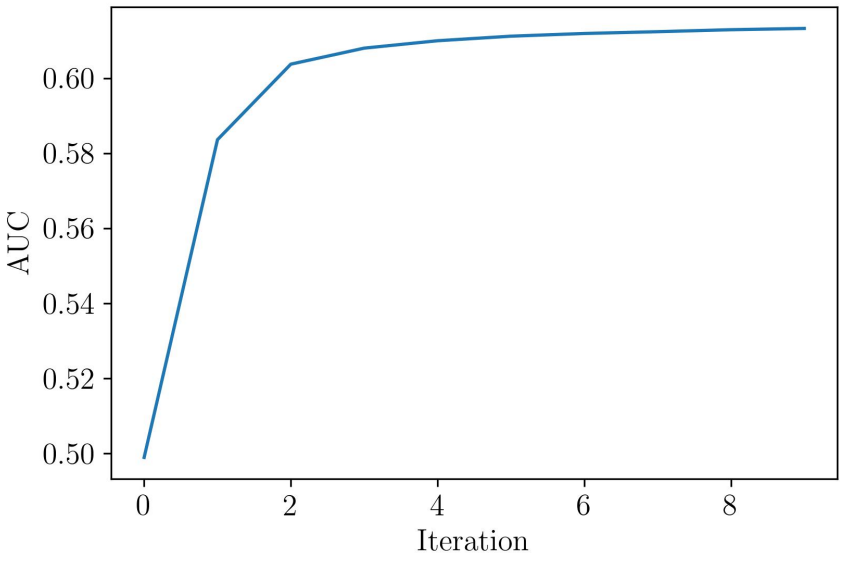

Figure 6: AUC over training epochs for BPR

4 Results

This section shows the top 20 movies for a randomly selected user, as well as that user's top 20 predicted movies.

4.1 Alternating Least Squares

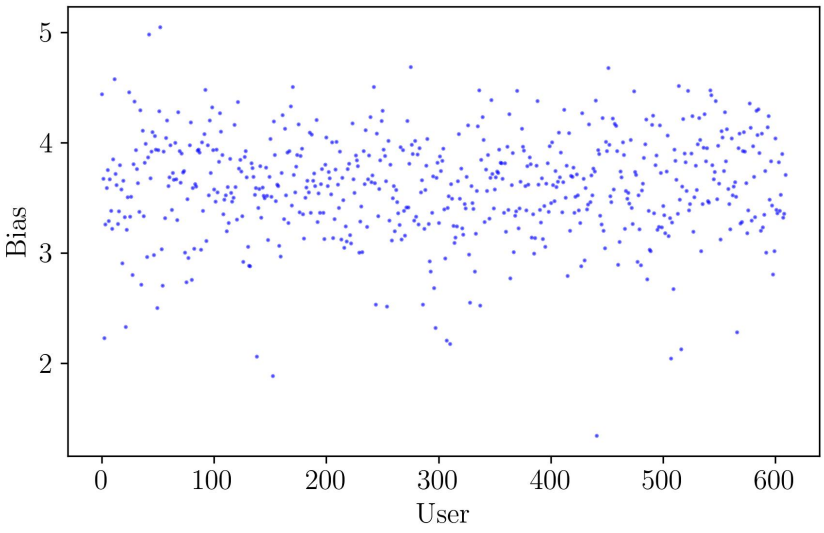

Figure 7: ALS learned user biases

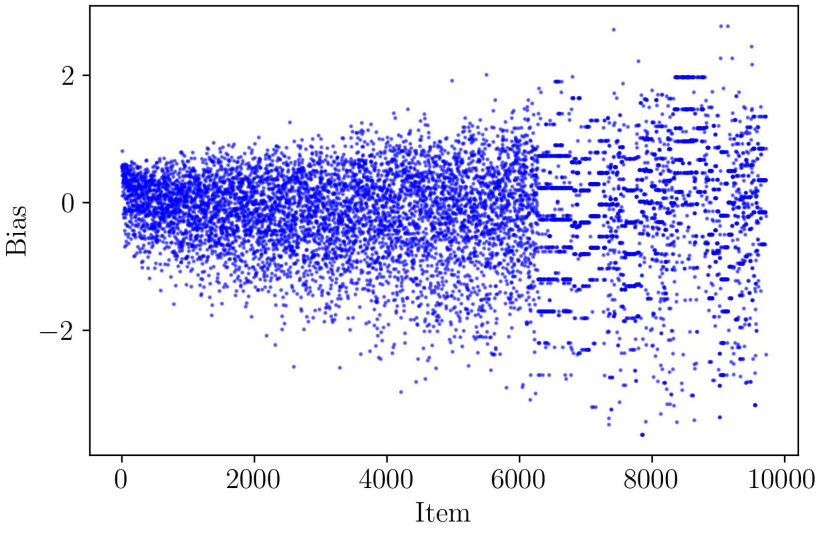

Figure 8: ALS learned item biases

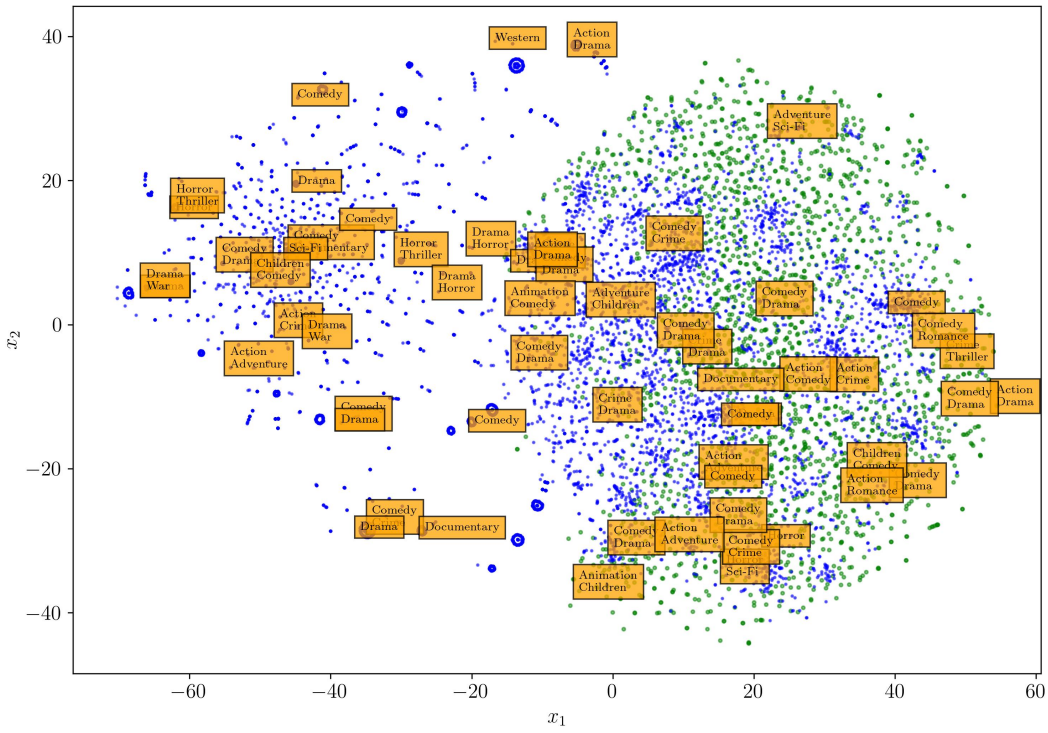

Figure 9: ALS item vectors represented in two dimensions using t-SNE

In figure 9, t-distributed stochastic neighbour embedding (t-SNE) was used to assist in visualising the high dimensional latent space. In short, this method aims to preserve the distance relationship between vectors in high dimensions in a low dimensional representation. In the figure it can be seen that there is a large cluster of movies, and lots of rogue dots away from the main cluster. The green dots represent the movies that have 80% of the ratings. The blue dots represent the rest. It is important to note that all of the green dots appear inside of the main cluster. The blue dots that appear outside the cluster are movies that have very few or no ratings.

Since this latent embedding is purely the result of a collaborative filtering approach, this graph makes sense. The blue dots ostracised from the main cluster are so because no collaboration pertaining to them has taken place, hence the algorithm does not really know what to do with them. It is not gauranteed that similar movies will cluster together within the main cluster; what is expected is that movies are related through the users that enjoyed them. Movies that are enjoyed similarly by users tend to cluster closer together than otherwise. This dynamic is, of course, subject to interruption by users with great variance in what they enjoy, and users with more than one individual using the profile.

userId	movieId
44	Rumble in the Bronx (Hont faan kui) (1995)
44	Star Wars: Episode IV - A New Hope (1977)
44	Rock, The (1996)
44	Barb Wire (1996)
44	Star Wars: Episode VI - Return of the Jedi (1983)
44	First Strike (Police Story 4: First Strike) (G...
44	Private Parts (1997)
44	Chasing Amy (1997)
44	Four Rooms (1995)
44	Dead Man Walking (1995)
44	Beautiful Girls (1996)
44	Broken Arrow (1996)
44	Happy Gilmore (1996)
44	Executive Decision (1996)
44	Kids in the Hall: Brain Candy (1996)
44	Time to Kill, A (1996)
44	Tin Cup (1996)
44	1-900 (06) (1994)
44	People vs. Larry Flynt, The (1996)
44	Jerry Maguire (1996)

(a) Top true movies for user 44

	movieId
767	Producers, The (1968)
381	Zoolander (2001)
78	Austin Powers: The Spy Who Shagged Me (1999)
161	Monty Python's Life of Brian (1979)
1040	Fletch (1985)
98	Kill Bill: Vol. 2 (2004)
59	Dumb & Dumber (Dumb and Dumber) (1994)
7	Terminator 2: Judgment Day (1991)
1000	Pee-wee's Big Adventure (1985)
857	Easy Rider (1969)
583	Time Bandits (1981)
249	Young Frankenstein (1974)
66	Terminator, The (1984)
10	Star Wars: Episode V - The Empire Strikes Back...
670	Napoleon Dynamite (2004)
22	Lord of the Rings: The Return of the King, The...
135	Who Framed Roger Rabbit? (1988)
79	Clockwork Orange, A (1971)
949	Birdman: Or (The Unexpected Virtue of Ignoranc...
747	West Side Story (1961)

(b) Top predicted movies for user 44

Figure 10: ALS results

	movieId	predicted rating
7378	Match Factory Girl, The (Tulitikkutehtaan tytt...	6.411793
8991	Galaxy of Terror (Quest) (1981)	6.404657
9099	Alien Contamination (1980)	6.404657
9456	Villain (1971)	6.185195
7751	Paterson	5.912556
8986	Looker (1981)	5.905180
9161	Master of the Flying Guillotine (Du bi quan wa...	5.905180
9461	Bossa Nova (2000)	5.826915
5453	Jetée, La (1962)	5.703233
4934	Come and See (Idi i smotri) (1985)	5.685184

Figure 11: Top predictions for user 44 including the movie biases

The movies recommended to the user via ALS make intuitive sense, they are matched well with the movies the user did enjoy. Figures 7 and 8 present scatterplots of the biases that were learned during ALS. The item biases are of particular interest. In figure 7, the items are ranked from most rated to least rated. There is a clear trend in the figure, the less ratings a movie has, the higher its bias becomes. This is rather problematic, since the system will often disproportionately recommend movies with very few ratings. The proposed remedy is to drop the item bias entirely when making predictions.

4.2 Bayesian Personalised Ranking

userId	movieId
44	Rumble in the Bronx (Hont faan kui) (1995)
44	Star Wars: Episode IV - A New Hope (1977)
44	Rock, The (1996)
44	Barb Wire (1996)
44	Star Wars: Episode VI - Return of the Jedi (1983)
44	First Strike (Police Story 4: First Strike) (G...
44	Private Parts (1997)
44	Chasing Amy (1997)
44	Four Rooms (1995)
44	Dead Man Walking (1995)
44	Beautiful Girls (1996)
44	Broken Arrow (1996)
44	Happy Gilmore (1996)
44	Executive Decision (1996)
44	Kids in the Hall: Brain Candy (1996)
44	Time to Kill, A (1996)
44	Tin Cup (1996)
44	1-900 (06) (1994)
44	People vs. Larry Flynt, The (1996)
44	Jerry Maguire (1996)

(a) Top true movies for user 44

	movieId
75	Twister (1996)
84	Willy Wonka & the Chocolate Factory (1971)
185	Nutty Professor, The (1996)
195	Mr. Holland's Opus (1995)
149	Wizard of Oz, The (1939)
387	Sabrina (1995)
1499	Sudden Death (1995)
280	Eraser (1996)
655	Miracle on 34th Street (1994)
350	Cool Hand Luke (1967)
261	Sense and Sensibility (1995)
241	In the Line of Fire (1993)
576	Striptease (1996)
485	Phenomenon (1996)
108	Harry Potter and the Sorcerer's Stone (a.k.a. ...
113	There's Something About Mary (1998)
47	Princess Bride, The (1987)
608	River Wild, The (1994)
12	Usual Suspects, The (1995)
154	Star Trek: First Contact (1996)

(b) Top predicted movies for user 44

Figure 12: BPR results

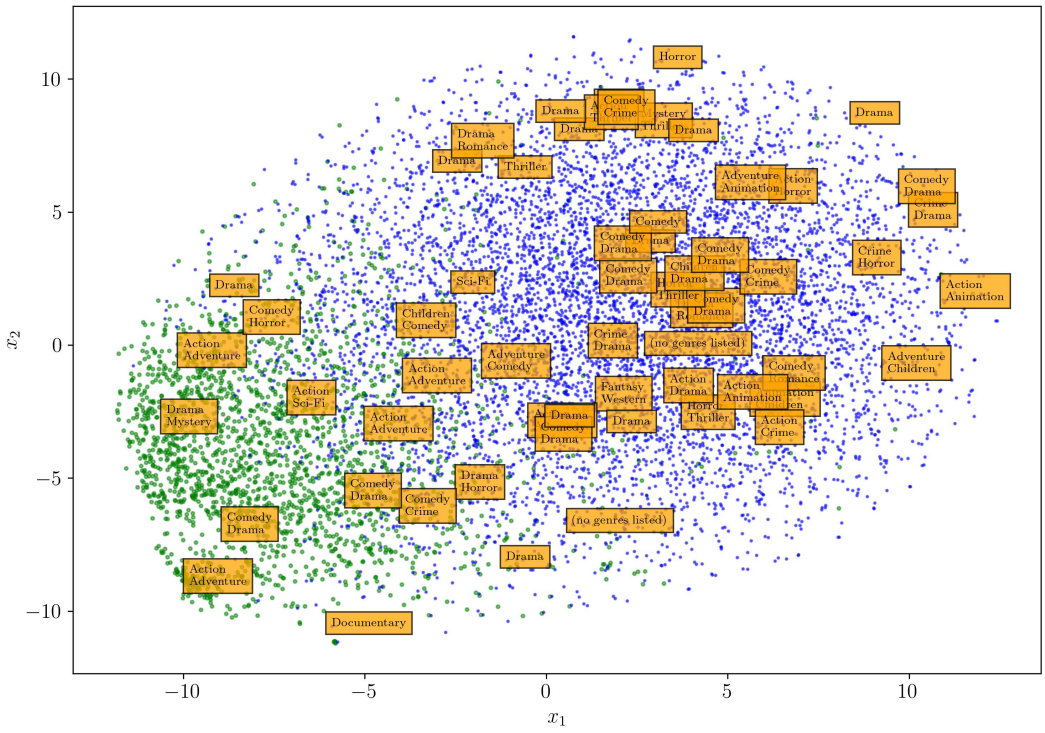

Figure 13: BPR item vectors represented in two dimensions using t-SNE

The movies recommended to the user via BPR make intuitive sense, they are matched well with the movies the user did enjoy aswell. However, BPR seems to have a preference for more popular movies over less popular movies.

5 A/B Testing

A/B Testing is an influential user experience research methodology where one conducts specific controlled tests with the objective of obtaining statistically signifcant results. A/B testing is particularly well suited for web applications, because the tests can be implemented immediately and regularly, streamlining the process of improving the service.

A mock up A/B test was conducted to show how it would be done in the context of online movie recommender systems. Note that a real A/B test should involve additional considerations [3]. The basic metholodgy of an A/B test is as follows:

1. Randomly uniformly distribute N users into 2 groups
2. The first group will be the control, and the second group the treatment
3. Nothing will change in the control group
4. A very particular change will be made to the user experience of the treatment group
5. The overall evaluation metric will be measured after some time has passed
6. A statistical evaluation will show whether or not the outcome of the experiment occured due to the change in the treatment group or due to random chance

Logs are also written to disk to keep a tidy record of the data collection pertaining to the test. For the ALS recommender system it is hypothesised that severly downweighing the item biases at inference time yields improved user experience in comparison to including the whole item bias. User experience refers to some metric indicating whether or not the user was engaged by their recommendations. In this case, it is simply a binary number indicating whether or not the user was satisied with their experience.

In this A/B test simulation, the users in the treatment group were simulated to have a higher probability of not giving positive feedback, in comparison to the users in the control group. All of the 610 users in the MovieLens 100k dataset participated in this trial.

The formulation is that $H_0 : p_A = p_B$ and $H_1 : p_A \neq p_B$; where the p's represent the proportion of users from each group that indicated they had a positive experience with their recommendations. In the simulated trial, there were 294 users in the control group and 316 users in the treatment group. In the control group 45% of users indicated satisfcation, and in the treatment group it was 72%.

The z-test statistic can be calculated,

$$\begin{aligned} z &= \frac{p_A - p_B}{\sqrt{\frac{pq}{n_A} + \frac{pq}{n_B}}} \\ z &= 6.668 \end{aligned} \tag{8}$$

Since $z > 1.96$, the results are at the least signifcant at 5%.

6 Discussion

6.1 Overview

Overall, it seems that BPR is biased towards popular movies, and there is no immediately obvious way to tame this bias. On the other hand, ALS is much more open to change and adaptation; ALS can also offer more diverse movie recommendations than BPR. A great strength of ALS is its apparent proclivity to scaling. It is almost trivial to distribute to ALS computation and storage to a network of computers, each processing computations for ALS in parallel.

6.2 Future work

There are a number of points that must be looked at in future work on this topic. In short they are

- Training and analysis with the 25 million MovieLens dataset. This is anticipated to increase the quality of the predictions, as well as give new insight into analysing the data as a whole. In modern machine learning, big data is not always the answer; in the case of collaborative filtering recommender systems, it still is.
- Introduction of additional content-based filtering. Adding some additional embedding to the item vectors that indicate what type of movie it is (genres, etc.) should be looked at, perhaps using a VQVAE.
- JAX implementation. JAX offers the tantalising prospect of easily parallelising and distributing computations using a numpy-like syntax. Implementing ALS using JAX types, as well as *pmap* and *vmap* will speed up the algorithm considerably, as well as make the implementation more suitable for deployment at scale.

References

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[2] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian Personalized Ranking from Implicit Feedback," 2012. [Online]. Available: https://arxiv.org/abs/1205.2618

[3] T. Crook, B. Frasca, R. Kohavi, and R. Longbotham, "Seven pitfalls to avoid when running controlled experiments on the web," in *KDD*, 2009.

Appendix A: Derivations

Alternating Least Squares

We are interested in estimating the user and item vectors given the dataset of interactions. Considering Bayes' rule this can be written in the following way

$$p(\mathbf{u}_m, \mathbf{v}_n, b_m^{(u)}, b_n^{(i)} \mid r_{mn}) \propto p(r_{mn} \mid \mathbf{u}_m, \mathbf{v}_n, b_m^{(u)}, b_n^{(i)}) \cdot p(\mathbf{u}_m, \mathbf{v}_n, b_m^{(u)}, b_n^{(i)}) \tag{9}$$

If we assume the parameters $\mathbf{u}_m, \mathbf{v}_n, b_m^{(u)}, b_n^{(i)}$ are independent of one another, this expression becomes,

$$\propto p(r_{mn} \mid \mathbf{u}_m, \mathbf{v}_n, b_m^{(u)}, b_n^{(i)}) \cdot p(\mathbf{u}_m) \cdot p(\mathbf{v}_n) \cdot p(b_m^{(u)}) \cdot p(b_n^{(i)}) \tag{10}$$

Now including all interaction pairs of users m and items n in the dataset

$$\propto \sum_{(m,n)} p(r_{mn} \mid \mathbf{u}_m, \mathbf{v}_n, b_m^{(u)}, b_n^{(i)}) \cdot p(\mathbf{u}_m) \cdot p(\mathbf{v}_n) \cdot p(b_m^{(u)}) \cdot p(b_n^{(i)}) \tag{11}$$

At this point it is convenient to apply the natural logarithm to the expression, this transforms multiplications into additions and facilitates more floating point stable computation. The expression is also negated to yield a loss function to be minimized as this is convention.

$$\begin{aligned} \mathcal{L} = -\sum_{(m,n)} & \log(p(r_{mn} \mid \mathbf{u}_m, \mathbf{v}_n, b_m^{(u)}, b_n^{(i)}) \\ & + \log(p(\mathbf{u}_m)) + \log(p(\mathbf{v}_n)) + \log(p(b_m^{(u)})) + \log(p(b_n^{(i)}))) \end{aligned} \tag{12}$$

It is assumed that this is a system of Gaussian random variables. Let the priors for the prediction be defined by

$$p(r_{mn} \mid \mathbf{u}_m, \mathbf{v}_n, b_m^{(u)}, b_n^{(i)}) = \mathcal{N}(r_{mn}; \mathbf{u}_m^T \mathbf{v}_n + b_m^{(u)} + b_n^{(i)}, \lambda^{-1}) \tag{13}$$

Since the prediction is assumed to be Gaussian, the log likelihood becomes

$$\log(p(r_{mn} \mid \mathbf{u}_m, \mathbf{v}_n, b_m^{(u)}, b_n^{(i)})) = -\frac{\lambda}{2} \sum_{(m,n)} \left(r_{mn} - \left(\mathbf{u}_m^T \mathbf{v}_n + b_m^{(u)} + b_n^{(i)}\right)\right)^2 + c \tag{14}$$

If these predictions are assumed to be Gaussian distributed random variables of the form

$$p(r_{mn} \mid \mathbf{u}_m, \mathbf{v}_n, b_m^{(u)}, b_n^{(i)}) = \mathcal{N}(r_{mn}; \mathbf{u}_m^T \mathbf{v}_n + b_m^{(u)} + b_n^{(i)}, \lambda^{-1}) \tag{15}$$

and the following priors are imposed on the user and item vectors

$$\begin{aligned} \log(p(\mathbf{u}_m)) &= \mathcal{N}(0, \tau^{-1}) \\ \log(p(\mathbf{v}_n)) &= \mathcal{N}(0, \tau^{-1}) \end{aligned} \tag{16}$$

Then the Bayesian log likelihood with regularisation terms can be express as

$$\begin{aligned}&\log(p(r_{mn} \mid \mathbf{u}_m, \mathbf{v}_n, b_m^{(u)}, b_n^{(i)})) + \log(p(\mathbf{u}_m)) + \log(p(\mathbf{v}_n)) = \\ &-\frac{\lambda}{2}\sum_{(m,n)}\left(r_{mn} - \left(\mathbf{u}_m^T\mathbf{v}_n + b_m^{(u)} + b_n^{(i)}\right)\right)^2 - \frac{\tau}{2}\sum_m \mathbf{u}_m^T\mathbf{u}_m - \frac{\tau}{2}\sum_n \mathbf{v}_n^T\mathbf{v}_n\end{aligned} \tag{17}$$

During optimisation, the objective is to learn the latent space such that similar user and item vectors occupy the same general high dimensional cones. In other words, the inner product of similar user vectors and item vectors should be large, and dissimilar ones small.There are several ways to go about finding these user and item vectors. Some manipulation of the loss function is required. The loss function for the users becomes

$$\begin{aligned}\mathcal{L}(\mathbf{u}_m, b_m^{(u)}) &= -\sum_{(m,n)}\left[-\frac{\lambda}{2}\left(\begin{bmatrix}\mathbf{u}_m & b_m^{(u)}\end{bmatrix}\begin{bmatrix}\mathbf{v}_n \\ 1\end{bmatrix} - b_n^{(i)}\right)^2 - \frac{\tau}{2}\begin{bmatrix}\mathbf{u}_m & b_m^{(u)}\end{bmatrix}\cdot\mathbf{I}^*\cdot\begin{bmatrix}\mathbf{u}_m \\ b_m^{(u)}\end{bmatrix}\right] \\ &= -\sum_{(m,n)}\left[-\frac{\lambda}{2}(x_{mn} - \mathbf{z}_m^T\mathbf{y}_n)^2 - \frac{\tau}{2}(\mathbf{z}_m^T\mathbf{I}^*\mathbf{z}_m)\right] \\ &= -\sum_{(m,n)}\left[-\frac{\lambda}{2}(x_{mn}^2 - 2x_{mn}\mathbf{z}_m^T\mathbf{y}_n + \mathbf{z}_m^T\mathbf{y}_n\mathbf{y}_n^T\mathbf{z}_m) - \frac{\tau}{2}(\mathbf{z}_m^T\mathbf{I}^*\mathbf{z}_m)\right] \\ &= -\sum_{(m,n)}\left[-\lambda\mathbf{z}_m^T x_{mn}\mathbf{y}_n - \frac{\mathbf{z}_m^T}{2}(\lambda\mathbf{y}_n\mathbf{y}_n^T + \tau\mathbf{I}^*\mathbf{z}_m)\right]\end{aligned} \tag{18}$$

where $\mathbf{I}^*$ is the identity matrix with the last element along the diagonal set to zero. This is because the prior on the bias is that it is zero.

ALS alternates updating two sets of parameters that otherwise cannot be optimised simultaneously. To find the update rules for the user and item vectors the loss function for user is differentiated and set to zero,

$$\begin{aligned}0 &= -\sum_{(m,n)}\left[-\lambda\mathbf{z}_m^T x_{mn}\mathbf{y}_n - \mathbf{z}_m^T(\lambda\mathbf{y}_n\mathbf{y}_n^T + \tau\mathbf{I}^*\mathbf{z}_m)\right] \\ 0 &= \lambda\sum_n x_{mn}\mathbf{y}_n + \left(\lambda\sum_n \mathbf{y}_n\mathbf{y}_n^T + \tau\mathbf{I}^*\right)\mathbf{z}_m \\ \mathbf{z}_m &= \left(\lambda\sum_n \mathbf{y}_n\mathbf{y}_n^T + \tau\mathbf{I}^*\right)^{-1}\left(\lambda\sum_n x_{mn}\mathbf{y}_n\right)\end{aligned} \tag{19}$$

The derivation for the item vectors follows the exact same process and results in the same equation, except the users and items are swapped around. These equations can be alternated to update the user vectors and the item vectors until convergence has been achieved. The most desirable quality of ALS is it steps in the direction of the solution with optimal step length, since it is performing maximum a posteriori parameter estimation. In constrast to gradient descent, where the step length is suboptimal.

Regularisation parameter

The regularisation parameter τ of alternating least squares represent the priors for the user and item vectors.

$$
\begin{aligned}
\text{var}[\mathbf{u} \cdot \mathbf{v}] &= \mathbb{E}[\mathbf{u}^T \cdot \mathbf{v} \cdot \mathbf{v}^T \cdot \mathbf{u}] \\
&= \mathbb{E}_{\mathbf{u}}[\mathbf{u} \cdot \mathbb{E}_{\mathbf{v}}[\mathbf{v} \cdot \mathbf{v}^T] \cdot \mathbf{u}^T] \\
&= \mathbb{E}_{\mathbf{u}}\left[\mathbf{u} \cdot \left[\frac{1}{\tau} \cdot \mathbf{I}\right] \cdot \mathbf{u}^T\right] \\
&= \mathbb{E}_{\mathbf{u}}\left[\text{trace}\left[\mathbf{u} \cdot \frac{1}{\tau} \cdot \mathbf{I} \cdot \mathbf{u}^T\right]\right] \\
&= \text{trace}\left[\frac{1}{\tau} \cdot \mathbf{I} \cdot \mathbb{E}_{\mathbf{u}}\left[\mathbf{u} \cdot \mathbf{u}^T\right]\right] \\
&= \text{trace}\left[\frac{1}{\tau} \cdot \mathbf{I} \cdot \frac{1}{\tau} \cdot \mathbf{I}\right] \\
&= \text{trace}\left[\frac{1}{\tau^2} \cdot \mathbf{I}\right] \\
&= \frac{1}{\tau^2} \cdot \text{trace}[\mathbf{I}] \\
&= \frac{1}{\tau^2} \cdot k
\end{aligned}
\tag{20}
$$

This is then the variance of the $\mathbf{u} \cdot \mathbf{v}$ distribution. Since this is a Gaussian distribution, the desired value for τ can be obtained

$$
\begin{aligned}
\sigma^2 &= \frac{1}{\tau^2} \cdot k \\
\sigma &= \frac{1}{\tau} \cdot \sqrt{k} \\
\tau &= \frac{1}{\sigma} \cdot \sqrt{k} \\
&= \frac{1}{2.5} \cdot \sqrt{k} \\
&= \frac{3}{2.5} \cdot \sqrt{k}
\end{aligned}
\tag{21}
$$

Appendix B: Code

The Python implementations of all algorithms mentioned in this document can be found on my github.