**Brian Craft**

**CSC 577 Project**

**Data Description**

Below is a short overview of my datasets. Initially, I chose to use Book Crossing, but after wrangling with the dataset, the ratings number was small so I chose complementary datasets to explore as well.

* Book Crossing: The dataset contains 271,858 users, 271,379 books and 1,149,780 ratings, leaving the matrix less than .1% dense. The data is from a 4 week web crawl from the book crossings website. As well, the data contains implicit and explicit ratings.
* Movie Lens: This dataset contains roughly 6,040 users, 3,900 movies and 1,000,209 ratings, leaving a ratings matrix that is about 4% dense. The data is gathered by GroupLens and the University of Minnesota and used for research purposes.
* Yelp: This dataset contains 1 million users, roughly 144,000 businesses and about 4.1 million reviews, leaving a ratings matrix that is less than .01% dense. The dataset was part of a data mining challenge offered by Yelp.
* Amazon Reviews: This dataset is ratings from 1995 until 2003, specifically for the home a kitchen section.

Below is a brief description of how the data was cleansed.

* Book Crossing: After converting the dataset to 5 core, there was 21,915 users, 39,702 items and 608,766 ratings. The ratings matrix was 870,069,330, leaving a density less than .01%. The key for the each item was an ISBN code, which had special characters that caused and error in Librec. Therefore, a numeric key was made. As well, because there was explicit data, anything rating 7 or above, on a 10 scale, was considered liked and converted to a 1, while lower ratings were 0. After testing, the sparse makeup of the data proved to be an issue, providing poor results that were not sufficient for interpretation. Therefore, I further scrubbed the data to 5 core, explicit ratings only. This left me with 141,081 ratings for 11,334 books and 14,220 users. While this is less then the 500K, I felt comfortable given some of the other datasets were larger.
* Movie Lens: This dataset had 6,040 users, 3,706 items and 1,000,208 ratings. This left a rating matrix with 22,384,240 cells and a density of 4.4%, making this the most dense of the datasets. No further cleaning was required.
* Yelp: The Yelp data was already 5 core, leaving 8,043 users, 5,199 items and 141,454 ratings. The ratings matrix was of 41,815,557 leaving a density of .3%. No further cleaning was required.
* Amazon Reviews: After converting the dataset to 5 core, there was 66,519 users and 28,237 items, leaving a ratings matrix of size 1,878,297,003. There was 551,628 ratings, leaving a density less than .01%.

**Approach**

I will explore the performance of the Asymmetric SVD++, SVD++ and Aspect Modeling Recommenders. The questions I will try to answer are:

1. Does ASVD++ perform worse then SVD++, but with better runtimes?
2. Do linear algebra based methods, perform better then statistical based methods?

I will use MAE, MPE, RMSE and MSE, as the prediction will be a score 1-5 scale.

After reading Koren’s paper proposing ASVD++, I am curious to see if the lower complexity, as he suggests vs. SVD++, will transfer to better runtimes. I have had impressive results when using SVD++, but the runtime is generally poor, which concerns me if I were to use this in a real world setting.

Building on that point, while matrix factorization methods are powerful, I am curious how they fair in comparison to statistical based approaches with respect to runtime and performance.

**Results**

Below are the results to my experiments, using the random guess as my baseline. Note, the tuned parameter values are below the results. For SVD++ the learning rate decay was left to 1 for all trials, which was omitted below.

For the matrix factorization methods, I favored small learning rates with larger iterations, as I didn’t want to overshoot the minimum. With SVD++, the iterations was tuned manually, by observing when the minimum was overshot and then adjusting. ASVD++ and Aspect Modeling seemed to have an early stopping mechanism that made it easier to tune. To note, there was other parameters to tune with SVD++, but I tried to limit the scope for this project.

*Book Crossing*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **MAE** | **MSE** | **RMSE** | **MPE** | **Runtime** |
| **Baseline** | | | | | |
| Random Guess | 3.57 | 19.07 | 4.36 | .99 | 4 |
| **Matrix Factorization** | | | | | |
| SVD++ | 1.28 | 2.87 | 1.69 | .98 | 67 |
| *Learning Rate: .01, Iterations: 30* | | | | | |
| ASVD++ | 1.41 | 3.16 | 1.77 | 1 | 77 |
| *Learning Rate: .01, Iterations: 30* | | | | | |
| **Probabilistic Graphical** | | | | | |
| Aspect Ratio | 1.40 | 3.16 | 1.77 | 1 | 32 |
| *Learning Rate: 01, Iterations: 25* | | | | | |

Table : Results using Book Crossing dataset

*Yelp*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **MAE** | **MSE** | **RMSE** | **MPE** | **Runtime (s)** |
| **Baseline** | | | | | |
| Random Guess | 1.81 | 4.94 | 2.22 | .99 | 3 |
| **Matrix Factorization** | | | | | |
| SVD++ | .79 | 1.03 | 1.01 | .98 | 47 |
| *Learning Rate: .0001, Iterations: 125* | | | | | |
| ASVD++ | .88 | 1.24 | 1.11 | 1 | 32 |
| *Learning Rate: .0001, Iterations: 25* | | | | | |
| **Probabilistic Graphical** | | | | | |
| Aspect Ratio | .83 | 1.23 | 1.11 | .95 | 109 |
| *Learning Rate: .0001, Iterations: 150* | | | | | |

Table : Results using Yelp dataset

*Movie Lens*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **MAE** | **MSE** | **RMSE** | **MPE** | **Runtime (s)** |
| **Baseline** | | | | | |
| Random Guess | 1.73 | 4.48 | 2.11 | .99 | 9 |
| **Matrix Factorization** | | | | | |
| SVD++ | .71 | .82 | .91 | .98 | 539 |
| *Learning Rate: .0001, Iterations: 35* | | | | | |
| ASVD++ | .93 | 1.24 | 1.11 | 1 | 458 |
| *Learning Rate: .001, Iterations: 25* | | | | | |
| **Probabilistic Graphical** | | | | | |
| Aspect Ratio | .72 | .85 | .92 | .99 | 344 |
| *Learning Rate: .0001, Iterations: 50* | | | | | |

Table : Results using Movie Lens dataset

*Amazon Review Data*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **MAE** | **MSE** | **RMSE** | **MPE** | **Runtime (s)** |
| **Baseline** | | | | | |
| Random Guess | 2.15 | 6.60 | 2.56 | .99 | 7 |
| **Matrix Factorization** | | | | | |
| SVD++ | .83 | 1.4 | 1.18 | .83 | 232 |
| *Learning Rate: .001, Iterations: 500* | | | | | |
| ASVD++ | .86 | 1.22 | 1.10 | 1 | 666 |
| *Learning Rate: .001, Iterations: 500* | | | | | |
| **Probabilistic Graphical** | | | | | |
| Aspect Ratio | .74 | 1.45 | 1.20 | .70 | 143 |
| *Learning Rate: .001, Iterations: 50* | | | | | |

Table : Results using Amazon dataset

**Conclusions**

To revisit my first question, does ASVD++ perform worse then SVD++, but with better runtimes? For my experiments, the premise is rejected. With respect to the Book Crossing dataset, the runtime and performance was worse for ASVD++, be it by a small portion. With respect to the Yelp data, the runtime was slightly better but there were less iterations. As well, the performance was slightly worse. With respect to the Movie Lens dataset, the performance was worse and the runtime was better, but with less iterations. Lastly, when using the Amazon dataset, the runtime was significantly worse and with slightly worse performance.

In Koren’s paper, he proposes exchanging user parameters, which in theory should lower the complexity. But in my datasets, the variance in the number of items and users isn’t a large amount, which could account for more consistent runtimes between SVD++ and ASVD++. As well, given the two methods are closely related it seemed probable the results would be similar from an accuracy perspective, which was indeed the case. In a scenario where the user count is far greater then item count, ASVD++ could be a better choice then SVD++, in the hopes of improved runtime but in theory similar accuracy, if my results are any indication.

One note is that some of the hypothesized gains in ASVD++ are how new users are handled and the expandability of the results. This is not something I explored but could very cause a rethinking of the results. As well, Koren claims the method handles implicit data well, which in a real world is quite possibly the best data available.

Secondly, do methods based on linear algebra perform better then statistical based methods? With respect to the Book Crossing dataset, the runtime for Aspect Modeling with comparable accuracy measures. When looking at the Yelp data, the accuracy was comparable but with worse runtime for Aspect Modeling, however more iterations were used. Looking at Movie Lens, the runtime was noticeably better with strong accuracy measures. Lastly, with respect to the Amazon dataset, the runtime and accuracy performed best. After looking at these results, I feel comfortable saying on these datasets a graph based approach is better then a matrix factorization method.

With respect to the graphical model, the accuracy was consistently in line with the matrix factorization methods, but more importantly, the gains in runtime were significant. However, implementing batch gradient decent, which isn’t in Librec, could reduce the runtime significantly for the factorization methods.

Going off on a bit of a tangent, I adjusted Librec to use rankings for the Amazon and Book Crossing datasets. As I began testing, gains in my loss didn’t transfer to performance gains with respect to Recall, Precision or AUC. In a real world setting, MAE and RMSE (among others) are not necessarily representative accuracy measures. Users will not like a system in production because the MAE was better by .1, hypothetically. Therefore, with factorization methods, the tuning is being done to a proxy that may or may not actually be improving, at a cost of resources.

Stepping back and looking at this project as a whole, I was a bit surprised at my post project perception of matrix factorization methods. At face value, they are fascinating and powerful, but the experiments have swayed my preference towards graph based methods mainly because they are simple and seem to be high performing with respect to accuracy and runtime.

**Source**

<http://www.cs.rochester.edu/twiki/pub/Main/HarpSeminar/Factorization_Meets_the_Neighborhood-_a_Multifaceted_Collaborative_Filtering_Model.pdf>

<https://pdfs.semanticscholar.org/a852/85aba983ebeeeb61d0859103adb8116597e3.pdf>