**Brian Craft**

**CSC 577 Project**

**Data Description**

Below is a short overview of my datasets.

* 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.
* 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 method, perform better then statistical based methods?

Because the Book Crossing dataset is binary scores, I will evaluate the scores on precision and recall. For the other 3 datasets, 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 began with small learning rates and high iterations, monitoring the gain in loss and how it diminished overtime. Using this information, I slowly tuned the learning rate and iteration count. I tested different learning decay rates, but 1 offered the best results.

*Book Crossing*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Precision** | **Recall** | **AUC** |  | **Runtime (s)** |
| **Baseline** | | | | | |
| Random Guess |  |  |  |  | 5 |
| **Matrix Factorization** | | | | | |
| SVDPP | .002 | .0002 | .5 |  | 1629 |
| *Learning Rate: .001, Iterations: 50* | | | | | |
| ASVD++ |  |  |  |  | 908 |
| *Learning Rate: .0001, Iterations: 15* | | | | | |
| **Probabilistic Graphical** | | | | | |
| Aspect Modeling |  |  |  |  | 77 |
| *Learning Rate: .0001, Iterations: 15* | | | | | |

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, I was interested in exploring any runtime and accuracy difference in SVDP++ and ASVD++. With respect to the Book Crossing data, there was better runtime, when factoring in iterations, and little accuracy lost. With respect to the there was little gains in runtime but a large loss in accuracy. This pattern continued with the Movie Lens and Amazon datasets, interestingly enough, the runtime for ASVD++ was worse on the Amazon data when compared to SVD.

Stepping back, the answer to the question is simply no, that the tradeoff in runtime for accuracy is not worthwhile because I was unable to observe the improved runtime. However, I won’t be quick to discredit the ASVD++ system, as some of the additional value in this system is increased interpretability, handling new users and implicit feedback better and less complexity, which I assumed would result in reduced runtime. However, the results suggest differently.

One component that isn’t implemented in Librec, that could change these results, is batch gradient decent. Because each iteration of gradient decent must be performed on all data, runtimes will likely be higher. Adding in such a parameter could result in better runtimes.

Revisiting my second question, I wanted to look at how linear algebra based methods worked in comparison to statistically based methods. In the case of the book crossing, the difference was negligible so I will not focus on them. With respect to the Yelp dataset, the Aspect Modeling performed slightly worse with worse runtimes. With respect to the Movie Lens dataset, the model ran faster and performed nearly the best of the 3 methods. Lastly, looking at the Amazon dataset, the model was by far the best, in regards to accuracy and runtime. With these results in mind, I cannot say for certain which is better, but that in general, Aspect Modeling should increase runtime and perform on par with factorization methods.

When thinking further about this question, one aspect I didn’t explore but must consider, is interpretability. I find the factorization methods quite fascinating, and when factoring in gradient decent, the fascination compounds. But it’s challenging to interpret these results, even though we are extracting user and item vectors, it’s till not an easy cell. After looking at these results, I find the graphical model to be the preference, simply because they are powerful but offer easier interpretability as well.

Stepping back as a whole, all 3 models performed well above the baseline, and the accuracy measures were all acceptable, all being within a star.

The initial concern I had was the Book Crossing dataset, which provided uninteresting results that were almost too good. Therefore, I pivoted and chose an array of datasets to explore these algorithms, varying in size and density but all offering explicit ratings.

As well, the runtimes were not always consistent. When I initially ran the experiments, the runtimes were not logical, for instance lower iterations would take slightly longer. Therefore, I ran each final tuned experiment twice, with no programs open, to try and correct for this. These are the results you see above, which I am confident in. But initially this was an issue.

There was no batch gradient decent in Librec. While tuning, this led me to begin with lower iterations and higher learning rates to reduce runtime, however these configurations proved to not be the best starting point. Batch gradient decent would allow for smaller steps and larger iterations.

More due to scope, I didn’t explore important components of recommender systems, specifically diversity, interpretability and coverage. While my bubble of an experiment aided to a comparison of traditional performance measures, it’s quite possible these other measures would change the interpretation of my experiments.

My goal was to come away from this project with a better understanding of how linear algebra and statistics can be used to make recommendations. While it seems trivial, grasping SVD with gradient decent took me some time, but I feel confident I now understand how these factorization methods work. As well, I wanted to dive further into graph based recommenders, as it’s not a concept I feel was covered much thus far in my courses and I feel I accomplished such.

**Further Readings**

Below are papers proposing ASVD++ and Aspect Modeling.

<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>