**Brian Craft**

**CSC 577 Project**

**Overview**

I will explore and compare the performance of the Asymmetric SVD++, SVD++ and Aspect Modeling Recommenders. The first question I will address is:

1. Does ASVD++ perform worse than SVD++, but with better runtimes?

After reading Koren’s paper proposing ASVD++, I am curious to see if the lower complexity, as he suggests, 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.

I will tune ASVD++ and SVD++ recommenders using the 4 datasets describe below, aiming to predict ratings. My performance criteria will be MAE, MPE, RMSE, MSE and runtime, with an emphasis on runtime and MAE. While cross validation is preference, I am limited with respect to computing power.

The next question I will address is:

1. Do factorization models perform better then graphical models?

To do so, I will explore a comparison of ASVD++, SVD++ and Aspect Modeling Recommenders. Building on the above point, while matrix factorization appear powerful, runtime is a concern and I’d like to see how they compare. The performance criteria will be the same as with question 1.

**Data Description**

Below is a short discussion of the datasets I chose and how they were prepared. Initially, I chose to use Book Crossing dataset, but after doing some wrangling, I found the dataset to be insufficient to address my goals on it’s own, so I added in other 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. After converting the dataset to 5 core, there was 21,915 users, 39,702 items and 608,766 ratings, leaving a density less than .01%. A numeric key for each book was made, as the ISBN contained characters causing a Librec error. 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 with this configuration, the sparse makeup of the data and the binary ratings produced results that were not sufficient to aid in my questions. Therefore, I further scrubbed the data to 5 core for explicit ratings only. This left me with 141,081 ratings for 11,334 books and 14,220 users.
* 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. This dataset had 6,040 users, 3,706 items and 1,000,208 ratings. This left a rating matrix with a density of 4.4%, making this the densest of the datasets. No further cleaning was required.
* 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. The Yelp data was already 5 core, leaving 8,043 users, 5,199 items and 141,454 ratings, leaving a density of .3%. No further cleaning was required.
* Amazon Reviews: This dataset is ratings from 1995 until 2003, specifically for the home a kitchen section. 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 were 551,628 ratings, leaving a density less than .01%.

**Results**

Below are the results to my experiments, using the random guess recommender as my baseline. Note, the tuned hyper parameter values are listed in the tables. 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 count 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? The results above tend towards no. With respect to the Book Crossing dataset, the runtime and accuracy was worse for ASVD++. With respect to the Yelp data, the runtime was slightly better but there were less iterations and a worse performance. 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 and accuracy were worse.

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, relative to real world settings, 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. On a side note, potential gains from using ASVD++ are how new users are handled and the expandability of the results. This is not something I explored but could very well influence the answer to the question at hand. As well, Koren claims the method handles implicit data well, which in a real world is quite possibly the best data available as many sites don’t capture enough ratings to generate recommendations solely off that information.

Revisiting the second question, do factorization methods work better then graphical methods? With respect to the Book Crossing dataset, the runtime was better with comparable accuracy. 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 was best when using the graphical model. These results show strong evidence in favor of graphical models over factorization models in the scope of recommender systems.

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 for a recommender system. 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 indirectly but at a considerable resource cost.

Going into this project, I had this conception these advanced, fascinating models, in this case factorization methods, would outperform less flashy methods, in this case graphical models. However, that was proven to be wrong, as the graphical model performed well but not with a sacrifice of runtime.

In a real-world setting, recommender systems are not only difficult to implement, but difficult to scale. As well, factorization methods when using sparse, implicit data, at least in my experience on this project, yielded results that were not trustworthy, leading me to be skeptical of solving such a real-world problems with these methods. In my career, much of the appropriate data to use is implicit which, coupling with the above points, would, and will (as this is something I am tasked with this year at my company), lead me to explore graphical models for recommendation.

**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>