**Brian Craft**

**CSC 577 Project**

**Dataset**

Initially I wanted to work with the SVD methods in librec and diversity metrics, as I am most interested in using gradient decent to fill in the components of a sparse matrix. However, when I began to test my initial dataset, the Book Crossing, I found it to be uninteresting as it was just binary attributes, like/dislike the book. AS well, I was not finding one dataset to be sufficient to explore the methods. Therefore, I pivoted from implementing the diversity evaluator, and gathered some other datasets to test the SVD methods, along with one probabilistic graphical method. Below are the new datasets.

Book Crossing: Made the dataset 5 core.

MoveiLens: I chose to use the 1M dataset, which contains about 1 million reviews.

Yelp: Leveraged the 5 core dataset from class.

Amazon Reviews Kitchen: Made the dataset 5 core.

I was trying to find some datasets out of the ordinary that were relatively friendly, but I was finding the options to be quite limited.

**Data Cleansing**

The Amazon and Yelp datasets were 5 cored. Because the Book Crossing data was so sparse, I chose to sample users and items that had at least 5 ratings each. Furthermore, the book id was an ISBN code, which had characters that were causing errors in Librec. Therefore, I created my own numeric key, on a 1 and up scale. MovieLens was useful as is.

**Results**

Below are the results to my experiments. I used the random guess recommender to create a baseline for each dataset.

***Book Crossing***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **MAE** | **MSE** | **RMSE** | **MPE** | **Runtime (s)** |
| **Baseline** | | | | | |
| Random Guess | .49 | .33 | .57 | .99 | 212 |
| **Matrix Factorization** | | | | | |
| SVDPP | .01 | .005 | .024 | .278 | 646 |
| ASVD++ | .02 | .001 | .04 | .38 | 582 |
| **Probabilistic Graphical** | | | | | |
| Aspect Modeling | < .0001 | < .0001 | < .0001 | < .0001 | 298 |

***Yelp***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **MAE** | **MSE** | **RMSE** | **MPE** | **Runtime (s)** |
| **Baseline** | | | | | |
| Random Guess | 1.81 | 4.94 | 2.22 | .99 | 3 |
| **Matrix Factorization** | | | | | |
| SVD++ | .88 | 1.24 | 1.11 | 1 | 34 |
| ASVD++ | .79 | 1.03 | 1.01 | .98 | 54 |
| **Probabilistic Graphical** | | | | | |
| Aspect Ratio | .83 | 1.23 | 1.11 | .95 | 106 |

***MovieLens***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 | 143 |
| ASVD++ | .93 | 1.24 | 1.11 | 1 | 458 |
| **Probabilistic Graphical** | | | | | |
| Aspect Ratio | .75 | .89 | .94 | .99 | 141 |

***Amazon Review Data***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **MAE** | **MSE** | **RMSE** | **MPE** | **Runtime (s)** |
| **Baseline** | | | | | |
| Random Guess | 2.15 | 6.60 | 2.56 | .99 | 7 |
| **Matrix Factorization** | | | | | |
| SVD++ | .76 | 1.15 | 1.07 | .91 | 32 |
| ASVD++ | .86 | 1.22 | 1.10 | 1 | 60 |
| **Probabilistic Graphical** | | | | | |
| Aspect Ratio | .74 | 1.45 | 1.20 | .70 | 120 |

**Model Tuning and Final Hyper Parameters**

I started my learning rates low with high iterations. However, 2 cases would emerge. The first, the model would converge and stop early. The second, the model would overshoot the minimum and I’d have to lower the iterations until it stopped manually at a good point. As well, because this doesn’t use batch gradient decent, runtime was a concern. So I tried to find a good mix of low learning rate and iterations. In the end, most of my models used roughly 25 iterations and a learninig rate of roughly .001.

**To Dos**

* Rerun some of the tests, as I was getting some odd runtimes. I want to confirm this is due to the model selection and not my computer being fussy.
* As well, in many cases my iterations were converging and causing early stopping. I’d like to investigate if Librec is doing this correctly.
* Explore how the metadata of each dataset could be oddly impacting the runtimes.
* Explore other implementations of SVD using gradient decent, perhaps in Python or R.