**Brian Craft**

**CSC 577 Project**

**Dataset**

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.

MoveiLens: 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

**Data Cleansing**

Book Crossing: Because the 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: This dataset was clean. The only adjustments were removing the time variable from the ratings data.

Yelp: The data was cleaned to only include users and businesses that had at least 5 reviews.

Amazon Reviews

**Proposal**

My project is focused on exploring the singular value decomposition methods implemented in Librec, specifically SVD++ and ASVD++. Furthermore, I will compare the performance of these models with that of the aspect modeling graphical system in Librec. To adequately explore these methods, I have chosen 4 datasets to draw comparison and conclusions from.

**Data Cleaning**

**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 |

In the case of book crossing, the models all performed quite well. Since the ratings were binary, the baseline of random guess performed as expected. The matrix factorization worked quite well, achieving very low scoring but with runtimes more than double that of the aspect modeling, which achieved even better results in half the time. One such reason I chose to gather more data, was I felt unfulfilled in my exploration of the SVD methods, which is a concept I wanted to explore in depth for this project. Simply looking at the above results, it’s difficult to access just how powerful these methods are. Taken at face value, the Aspect Modeling performed better.

***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 |

The Yelp data was far more useful in interpreting the value of these models. In the case of all 3 models, the MAE was below 1, meaning we are predicting within 1 star. The runtime was double for Aspect Ratio modeling, interestingly enough. While the ASVD++ had the best MAE, I would trade that for the lower computation cost of SVD++ given a larger dataset.

***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**

For the SVD++, the possible tuning parameters are the learning rate, the iterations, user and item regularization and the learning rate decay. For ASVD++ the learning rate and iterations are tuned. For aspect modeling the learning rate and iterations are also tuned. When tuning, I factored in the amount of time taken for each iteration, since batch gradient decent isn’t implemented, and the impact on the accuracy measures. Therefore, I tended to favor higher learning rates and lower iterations.

***Book Crossing:***

The models when using this dataset converged so quickly to highly accurate results. Therefore, little tuning was actually needed.

* SVD++
  + Iterations: 25
  + Learning Rate: .01
* ASVD++
  + Iterations: 25
  + Learning rate: .01
* Aspect Modeling
  + Iterations: 75
  + Learning Rate: .01

***Yelp***

To tune the SVD methods, I lowered the learning rate to .0001 and increased the iterations to 100. The models ended up converging and stopping early, leaving little tuning to be had. For the Aspect Modeling, I took a similar approach, by reducing the learning rate and increasing the iterations. Conversely to the other methods, this allowed me to achieve lower scores with more iterations.

* SVD++
  + Iterations: 25
  + Learning Rate: .01
* ASVD++
  + Iterations: 25
  + Learning rate: .01
* Aspect Modeling
  + Iterations: 75
  + Learning Rate: .01

***MovieLens***

To train the SVD methods, I chose the same approach as described above. However, these models didn’t converge but would overshoot the minimum cost. Manually, I was able to tune the learning rate and iterations to a good stopping point.

* SVD++
  + Iterations: .0001
  + Learning Rate: 25
* ASVD++
  + Iterations: .0001
  + Learning rate: 5
* Aspect Modeling
  + Iterations: .0001
  + Learning Rate: 15

***Amazon Review***

* SVD++
  + Iterations: 25
  + Learning Rate: .01
* ASVD++
  + Iterations: 25
  + Learning rate: .01
* Aspect Modeling
  + Iterations: 25
  + Learning Rate: .01