|  |
| --- |
| **User Taste Prediction** |

**Annies AbduljaffarMatt Vail**

Student Student

Stanford University Stanford University

Stanford, CA 94305 Stanford, CA 94305

[anniesab@stanford.edu](mailto:anniesab@stanford.edu) [vailm@stanford.edu](mailto:vailm@stanford.edu)

**Abstract**

Recommendation engines are ubiquitous in today’s online world. Recommendation systems provide users with personalized suggestions for products or services. There are two basic types of recommendation engines: content based and collaborative filters. These approaches can also be combined to form hybrid systems. Collaborative filtering methods can be further categorized into neighborhood models that rely on user-user or item-item similarity and matrix factorization models that directly profile users and items according to a set of learned latent factors [1]. This implementation project investigates multiple methods for reducing training on large sparse ratings matrices including exploring various combinations of parameters on different models, preconditioning training data to transform a highly sparse matrix into a dense matrix, and implementing a basic momentum method for model optimization.

**1 Introduction**

Discuss the context of the problem, your motivation for looking at the problem, and clearly state the intended outcomes of the project (10%)

**1.1 Sub Intro If Necessary**

Intro continued

**2 Related Work**

Discuss at least 2 pieces of related literature (or software, if you're doing an implementation project) and how your project compares to them (5%)

**3 Data and Software Libraries**

This project was completed in Python 3.7.2\_1 and Cython 0.29.6 using standard libraries:

1. Os – miscellaneous operating system interfaces
2. SciPy – user friendly and efficient numerical routines including linear algebra methods such as dot product, vector norms, minimize, optimize, etc.
3. Numpy – scientific computing package
4. Pandas – easy-to-use data structures and analysis package
5. Matplotlib – visualization library
6. Seaborn – visualization library built on top of Matplotlib
7. Surprise – modeling frameworks including a Cython implementation of the “SVD” and “SVDpp” algorithms, parameter search, and cross validation.

The following modules were written from scratch for this project:

1. ‘load\_data.py’ – provides methods to load MovieLens data and filter based on number of ratings per user and per movie
2. ‘MF\_SGD\_momentum.py’ – extends the Surprise SVD and SVDpp algorithms to include momentum methods for optimization of model parameters
3. ‘benchmark.py’ – main project module; produces all of the relevant experimental results

Code was produced in the open-source Atom text editor on a Macbook Pro.

**4 Methods**

Discuss your methodology-- this includes but is not limited to architecture description, software pipeline, algorithm description, optimization scheme. If you use methods from the course notes we expect you to briefly describe the method in your own words, but it should not take up the majority of this section (i.e. your methods section should not be only a rehash of the course notes.) (20%)

**5 Experimental Results and Analysis**

Show your results, and if appropriate, analyze them. Note that the weighting between the two will change depending on your project type: for visualization projects and network exploration projects we'll weight experiments more heavily, but for implementation projects and theory projects we'll weight the analysis more heavily. (40%)

**6 Conclusion**

Implementing Conjugate Gradient on subsets of the matrix. Implementing other more sophisticated momentum methods. Clearly summarize your results and describe a few potential avenues for future work. (5%)

**7 Contributions**

This project was completed entirely by Annies Abduljaffar and Matt Vail.

Annie’s contribution –

Matt’s contribution –

**References**

|  |  |
| --- | --- |
| [1] | Y. Koren, "Factorization Meets the Neighborhood a Multifaceted Collaborative Filtering Model," 2008. |
| [2] | Aggrawal, Recommender Systems, Yorktown Heights, New York: Springer, 2016. |
| [3] | Y. M. B. J. C. Daniel A. Galron, "Deep Item Based Collaborative Filtering for Sparse Implicit Feedback," 2018. |
| [4] | H. S. S. Daniel D. Lee, "Algorithms for Non-Negative Matrix Factorization". |
| [5] | I. P. D. T. Gabor Takacs, "Applications of the Conjugate Gradient Method for Implicit Feedback Collaborative Filtering," 2011. |
| [6] | Y. Koren, "Factor in the Neighbors Scalable and Accurate Collaborative Filtering, Koren 2010," vol. 4, 2010. |
| [7] | S. Postmus, *Recommender System Techniques Applied to Netflix Movie Data, 2018,* Amsterdam, 2018. |
| [8] | Y. S. P. A. Prateek Sappadla, *Movie Recommender System - Project Report,* New York, New York, 2017. |
| [9] | A. M. Ruslan Salakhutdinov, "Probabilistic Matrix Factorization". |
| [10] | W. W. J. F. Sheng Wang, "Learning from Incomplete Ratings Using Non-negative Matrix Factorization". |
| [11] | J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain," 1994. |
| [12] | V.-T. Tran, "Dimensionality Reduction: SVD and its applications". |
| [13] | R. B. C. V. Yehuda Koren, "Recommender-Systems-[Netflix]," 2009. |
| [14] | S. M. Thomas George, "Scalable Collaborative Filtering Framework based on Co-clustering". |
| [15] | A. M. Daniel Lemire, "Slope One Predictors for Online Rating-Based Collaborative Filtering," *arXiv,* 2008. |
| [16] | H.-C. C. F. M. L. Robert H. Sues, "Stochastic Pre Conditioned Conjugate Gradient Method," *Probabilistic Engineering Mechanics,* pp. 175-182, 1992. |
| [17] | Y. Koren, "The BellKor Solution to the Netflix Grand Prize," 2009. |
| [18] | M. J. Andres Toscher, *The BigChaos Solution to the Netflix Grand Prize,* 2009. |
| [19] | M. C. Martin Piotte, *The Pragmatic Theory Solution to the Netflix Grand Prize, 2009.pdf,* 2009. |