A Comparative Study of Collaborative Filtering Recommendation Systems Using Algorithms to Impute Large Sparse Matrices

by

Steven Christopher Lindo

Submitted in partial fulfillment
Of the requirements for the degree of
Doctor of Professional Studies
In Computing

At

Ivan G. Seidenberg School of Computer Science and Information Systems

Pace University

September 2016

We hereby certify that this dissertation, submitted by **Steven C. Lindo** has successfully satisfied all the requirements for the degree of the Doctor of Professional Studies in Computing"

Dr. Ronald Frank
Chairperson of Dissertation Committee

September 27, 2016
Date

Dr. Charles Tappert
Dissertation Committee Member

September 27, 2016
Date

Dr. Fred Grossman
Dissertation Committee Member

September 27, 2016
Date

Seidenberg School of Computer Science and Information Systems

Pace University

An Abstract

A Comparative Study of Collaborative Filtering Recommendation Systems Using Algorithms to Impute Large Sparse Matrices

by

Steven Christopher Lindo

Submitted in partial fulfillment
Of the requirements for the degree of
Doctor of Professional Studies
In Computing

September 2016

In the modern era of computing, recommendation systems are a key component for enterprise systems and consumer applications including e-commerce and web applications. The challenge for these systems is the *accuracy* and *quality* of the calculations especially when dealing with sparse amounts of data. This research provides an empirical study of the problems associated with a sparse matrix when encountered by collaborative filtering recommendation systems. The research will conduct a comparative analysis of different algorithms used to address the issue of sparse data while *trying* to predict and prescribe (recommend) the optimal choice to users. The research will compare statistical techniques used to impute data with other estimations techniques for predicting missing values. The research will show why Matrix Factorization, Maximum Likelihood Estimation (MLE) or Gradient optimization methods work better for large sparse matrices, over simple mean, sub-group mean or regression methods.

Acknowledgements

Knowledge may be gained through our life experiences but wisdom is gained from observation and discernment. Research is more than just a disciplined systematic process; it is the key to clarity and understanding through observation, thus resulting in wisdom.

I am truly grateful to have found a career doing something I love. Something I have studied since before I was freshman in college. I feel that is it is my obligation to give back and do my part to be a good steward to the industry and science that has cared for me for over twenty-six years.

I would like to acknowledge and thank the entire faculty at Pace University. They were all very supportive helpful to me the entire time. My special thanks to Dr. Fred Grossman for his guidance and for challenging us to raise the bar for ourselves. I would also like to thank Dr. Tappert for providing many instructions and insights that had me thinking while trapped by traffic on the long ride back to my domicile. Thanks to Dr. Tao for his leadership and commitment to the program and students.

I learned a long time ago, that having a good advisor and mentor improves your chances of success in any endeavor. I was lucky enough to have Dr. Ron Frank as my advisor. His depth of knowledge surprised me at every turn, as he carefully guided me through my research. A great deal of my success at Pace University can be attributed a man who I have come to admire, Dr. Frank, thank you.

I would also like to acknowledge and thank my cohorts, the class of 2016. I believe that we are the model for all other classes, with true care and consideration for each other and sharing our struggle and celebrating our successes during the entire

program.

I am blessed to have a large support network of friends and family members who lift me up in prayer each and every day. I cannot thank them enough for all their support. To my mother, Alice Lindo, who has given me more than I could ever repay, I love you. For believing in me and for always being there for me, success or failure. To my father, who understands the value of higher education and made sure that we were striving to be the best that we can be. To Mr. and Mrs. Merton Owens for their constant encouragement and support. Thank you.

Finally, to my children Briannah, Brittany and Brandon for checking my grades and saw to it that I was studying. They have raised a good father. To my wife Janice for being there to support me, not only these past three years but always. To God Almighty, for his grace and blessings upon me, I give my eternal gratitude.

Table of Contents

1 Introduction 9
1.1 Types of Recommendation Systems 11
1.2 Problem Summary 13
1.2.1 Problem of Missing Data 15
1.2.2 Sparse Matrix 15
1.3 Solutions for Dealing with Sparse Data 16
1.3.1 Drop the missing data 16
1.3.2 Impute the Missing Data 18
1.3.3 Impute with Simple Mean 18
1.3.4 Impute with Sub-Group Mean 19
1.3.5 Impute using Regression Estimation 19
1.3.6 Impute using Maximum Likelihood Estimation 19
1.4 Dimensionality Reduction and Latent Factor Models 21
1.4.1 Principle Component Analysis 22
1.5 Imputing by Gradient Methods. 32
1.6 Summary of Contributions 33
1.7 Dissertation Roadmap 34
2 Literature Review 35
2.1 Types of Recommendation Systems 36
2.1.1 Content-Based Systems 36
2.1.2 Collaborative-Filtering Systems (CF) 37
2.2 Sparse Matrix Problem 38
2.3 Big Data Revolution 40
3 Problem Solution Design 41
3.1 Research Summary 42
3.2 Research Data 43
3.3 Movie Lens Data Set 44
3.4 Generated Sample Data 46
4 Implementation Highlights 47
4.1 Skewed Distributions 52
4.2 Relevance and Significance of the Research 53
4.2.1 A Comparative Study 53
4.2.2 DAO Algorithm 54
4.3 Research Approach 55
4.4 Research Steps 56
4.4.1 Root Mean Square Error 58
4.5 Similarity Calculations: 59
4.5.1 Impute with Caution 60
4.5.2 Feedback Loop for Recommendation Systems 61
5 Experimental Validation 61
5.1 Implementation - Impute with the simple mean (μ) 62
5.2 Implementation - Impute with an appropriate subgroup mean 63
5.3 Implementation - Impute using a regression estimate (LSM) 63
5.4 Implementation - Impute using Maximum Likelihood Estimation (EM) 64

5.5 Implementation - Impute using Matrix Factorization 65
5.6 Implementation -Impute using Distribution Adjusted Optimization (DOA) 67
5.7 Results – m-series Test Data Sets 69
5.7.1 RMSE Comparison m-series Data Sets 72
5.8 Results – Movie Lens Data Sets 74
5.8.1 RMSE Comparison a / b Movie Lens Data Sets 75
5.8.2 RMSE Comparison u1 – u5 Movie Lens Data Sets 79
6 Conclusions 83
6.1 Summary of Observations and Conclusion 83
6.2 Observation of Results 84
6.3 Research Contributions 85
6.4 Future Work 85
7 Reference 87
Appendix A: System Specifications Research 91

List of Tables and Figures

Table 1: Sample Rating Matrix 13
Table 2: Sample Sparse Rating Matrix 15
Table 3: Population Statistics for Training Data (m) 70
Table 4: Sparseness % for Training Data 71
Table 5: Ratings Distribution for m-series Data Sets 71
Table 6: Rating Distribution for m-series Data Sets 72
Table 7: RMSE Comparison for m-series Data Sets 72
Table 8: Comparison of DAO modified algorithm 74
Table 9: Run-Time Comparison for m-series Data Sets 74
Table 10: u.data Ratings Distribution 75
Table 11: Rating Distribution for Training and Test 75
Table 12: Population Statistics a/b Data Sets 76
Table 13: ua and ub Sparse Analysis 76
Table 14: Ratings Distributions for ua - ub Training Data Sets 77
Table 15: RMSE Comparison ml a/b Data Sets 77
Table 16 RMSE Comparison ua and ub Data Sets 78
Table 17: Run-Time Comparison Movie Lens ab Data Sets 78
Table 18: Population Statistics for u1 - u5 Data Sets 79
Table 19: Sparse Analysis for u1 - u5 80
Table 20: Ratings Distribution u1 - u5 Data Sets 80
Table 21: Ratings Distribution u1 - u5 Data Sets Comparison 81
Table 22:RMSE Comparison u1 - u5 Data Sets 81
Table 23: RMSE Comparison u1 - u5 Data Sets - Chart 82
Table 24: Run-Time Comparison u1 - u5 Data Sets 83

Figure 1-1: Analytics Value Chain 9
Figure 1-2: NNMF Diagram 30
Figure 1-3: Bayes Net Diagram for PMF 31
Figure 4-1: Histogram Base File 51
Figure 4-2: Skewed Distribution 53
Figure 5-1: RMSE Comparison - Chart 73

1 Introduction

There is an extensive class of applications that involve predicting user responses to actions and suggesting items to match users. Such a system is called a recommendation (or recommender) system. Recommendation systems are having a big impact on web applications, e-commerce, and market intelligence systems [1] according to MIS Quarterly. Recommendation systems are mainly driven by data analytics. Data analytics is the process of turning raw data into information. There are three main categories or types of analytics:

1) Descriptive,

2) Predictive, and

3) *Prescriptive*.

The value chain for these type of analytics vary, but in general can be viewed as increased value with increasing complexity to create them. See the illustration below:

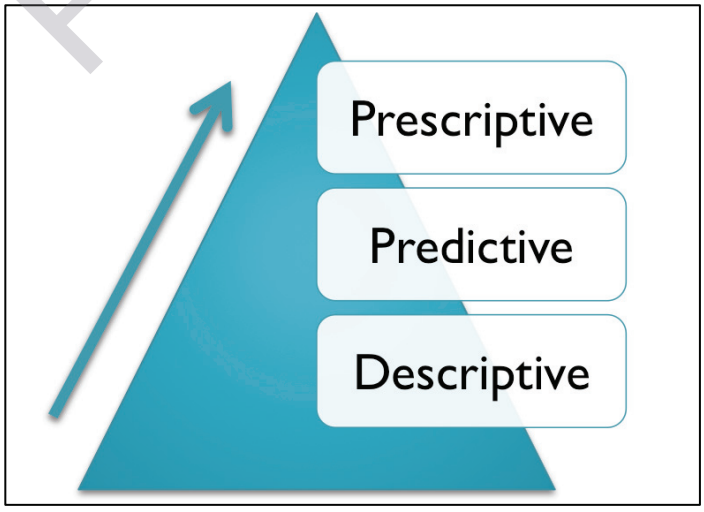

Figure 1-1: Analytics Value Chain

One type of analytic is not necessarily more important than the other, however they have different levels of value and one builds on the other. Perceived value is in the eye of the consumer of the item, in this case it is the information we call analytics.

Descriptive analytics are what legacy Business Intelligence (BI) systems use to be all about. That is, they provide reports on activities that happened in the past. The key there is the "past-tense" nature of the information. Predictive analytics attempts to provide information on when an event will happen in the ***future*** based on past information. Typical predictive modeling is done using regression analysis or other correlations methods with supervised rule based statistical approaches. The key thing to watch for when building predictive models is flawed data. Similar to other computer related data processing exercises, the principle of garbage-in-garbage-out (GIGO) holds true for analytics. So if erroneous data is used to predict the future, then the extrapolation too will be inaccurate.

Prescriptive analytics builds upon predictive analytics. It uses the predictions to "prescribe" action. Therefore, prescriptive analytics is synonymous with recommendations. Recommendations are prescribed to the user based on behavior and needs of the user in a particular context. In many situations with e-commerce application on the Internet the most difficult thing to discern is *context and intent*. For example, when a consumer makes a purchase on Amazon, is the consumer buying the item for himself or herself, or as a gift for someone else? What is their intent? There is no way of really knowing without first having insight to the context that they are in at that moment. Ideally knowing the end-users intent allows

recommendation systems to produces personalized suggestions or actions for their context. However, since knowing a consumer's intent is very difficult to capture, recommendation systems instead assumes that users will have similar preferences and taste, and also that there exist explicit and implicit features that are common to items and users tastes. This intuition is then used to build models for recommendation systems.

1.1 Types of Recommendation Systems

These recommendation systems use many different algorithms, components and technologies. We can classify these systems into two groups; content-based and collaborative=filtering systems.

The following are typical models and approaches for the two types of recommendations systems. [11]

- *Content-based systems*

  Models and approach build profiles:

  - Use features to predict response
  - User features: Age, gender, geo-location, visit pattern, …
  - Item features: Category, keywords, topics, entities, …
  - Linear regression, Bayes Net, SVM, tree/forest methods, mixture models

- *Challenges*:

- Bottleneck: Need predictive features
- Difficult to capture signals at granular levels.
- Cannot distinguish between users/items having same feature vectors

- *Collaborative filtering systems*
  - *Models and Approaches* mostly mathematical:
    - Make recommendation based on past user-item interaction
    - User-user, item-item, machine learning, linear factor models,
    - Good performance for users and items with enough data
  - *Challenges*:
    - Performs poorly with sparse matrix
    - Does not naturally handle new users and new items (cold-start). [12]

Examples of recommendation system applications can be found in numerous areas, especially e-commerce applications or document recommendation systems. Below are two broad examples of recommendation systems:

1. Offering news articles to on-line newspaper readers, based on a prediction of reader interests: A recommendation systems can build user profiles and then suggesting documents to read. Essentially predicting topics of interest and finding relevant documents or news articles related to these topics.

2. Recommending movies and/or restaurants to users to dine or watch base on a ratings scale. Recommended purchase can be emailed, or messaged to your mobile phone, for purchases from Amazon, BestBuy or other retailers.

1.2 Problem Summary

The goal of a recommendation system, in the context of a web application or e-commerce application, is to predict what a customer would be willing to purchase and "suggest" it to them. Most of the times the suggestion is based on prior purchase patterns or purchase patterns of "similar" users. The most common problem for recommendation systems is the need for "meaningful" data that can be used to build the prescriptive model. When data is missing, some algorithms struggle to deliver accurate results. This research has identified a specific problem to address in this study known as the "Sparse Matrix Problem".

As mentioned before, this research will focus mainly on Collaborative Filtering (CF) recommendation systems. The interaction between the users and the items is the key input into a CF system. These interactions are typically transformed into a rating matrix. A collaborative-filtering rating matrix is based on *U* number of users that have rated *N* number of items, creating a *U* x *N* matrix. For example Table 1: Sample Rating Matrix, simulates a rating matrix for 5 restaurants, R1 – R5, and ratings from 4 customers, C1 – C4.

Table 1: Sample Rating Matrix

		Restaurants				
		R1	R2	R3	R4	R5

Consumers	C1	5	3	4	5	1
	C2	1	5	2	1	4
	C3	4	4	1	2	2
	C4	5	3	1	4	1

In this example the rating matrix is fully populated as all the consumers have visited and rated all five (5) restaurants. By examining the data we can clearly see that consumer (C1) and consumer (C4) "like" similar restaurants. We do not need to know the characteristics of the restaurant or the cuisine or the background and profiles of consumer (C1) or consumer (C4). Therefore if consumer1 (C1) visits a new restaurant (R6) and give it a high rating of 5, our recommendation systems would obviously recommend restaurant (R6) to consumer (C4). As previously stated, the intuition is that there is some latent feature of restaurant R6 that makes it favorable to consumer C1 and makes it a "good" recommendation for consumer (C4). This rating matrix is sometimes called the interaction matrix because it captures the interaction between the users and the items.

When the rating matrix if fully populated CF recommendation systems will perform as well as can be expected and with very reasonably results. However, if the rating matrix has missing data the computations for collaborative filtering systems will produce high margins of error, thus recommending items to users that seems quite irrational to their context and intent.

1.2.1 Problem of Missing Data

Data *sparseness*, *scalability* and *prediction* quality have been recognized as the three most crucial challenges for recommendation systems [35]. The variable that affects the quality of prediction the most is missing data. Why? The reason being is that without the right level of density in the rating matrix the computational methods used by these recommendation systems will produce suggestions that are highly disproportionate and will appear very confusing to the end-user. This issue affects both content-based and collaborative-filtering recommendation systems.

In terms of the collaborative-filtering systems, this problem is defined as a sparse matrix problem. Sparse means "thinly dispersed or scattered" [13]. In the context of this research the general definition of *sparse* that we will use is: a large matrix with many zero elements.

1.2.2 Sparse Matrix

Let us re-examine the rating matrix of customers and restaurants as a sparse matrix. Table 2: Sample Sparse Rating Matrix, represents the same interaction between customers and the restaurants, however notice the sparseness or missing ratings for some of the restaurants. The rating matrix has been modified to simulate the fact that customers have not visited all the restaurants and therefore have not rated all the restaurants in the matrix.

Table 2: Sample Sparse Rating Matrix

		Restaurants				
		R1	R2	R3	R4	R5

Consumers	C1	5		4		1
	C2	1	5		1	
	C3		4	1	2	2
	C4	5			4	1

Now, by examining the sparse data we cannot draw the same inference as before where we noted that consumer (C1) and consumer (C4) had similar taste in restaurants.

The conclusion is that a sparse matrix used in a collaborative filtering recommendation system makes it difficult to predict how well a user will like an item that is not scored; therefore it will not accurately recommend a reasonably action or item to the users.

1.3 Solutions for Dealing with Sparse Data

The reason we analyze data is so that we can derive insights and gain understanding based on the *truth* represented in the information. Typically we are looking for some correlation or pattern that gives us awareness of a new phenomenon. Then we use this knowledge gained from the derived data as evidence to defend our theories. However, missing data have always plagued data analysis and challenge our ability to draw meaningful conclusions and infer insights. There are several techniques for dealing with sparse or missing data.

1.3.1 Drop the missing data

When dealing with sparse data sets, the obvious thing to do first is to ignore

or drop the missing data from our analysis and hope the amount of data you have is good enough for your model to draw an accurate conclusion for your domain of study. You can approach dropping the data in a couple of ways. According to Allison [18] you can use methods corresponding to *Listwise, Casewise, or Pair-wise deletion of missing data.* Especially if the missing data is limited to a small number of the subjects, you may just choose to eliminate those cases from the analysis. For example, if there exist a population of data items, and the population is missing data on *any* of the variables used in the analysis, it is dropped completely. There are pros and cons with dropping data. The main benefit is that you eliminate partial data sets that could distort your calculations. The problem is that you may be eliminating very important pieces of data that you would otherwise need. Also if the population was very sparse and you eliminate the missing data then the remaining data may not be representative of the population being analyzed.

According to Allison, there are occasions where eliminating missing data is a plausible solution. He says that the [18] procedure is sensible if (and only if) the data points are randomly missing. In that case, each parameter, mean, variance, or standard deviation is an unbiased estimate of the corresponding population, however he warns against using it with logistic regression.

The conclusion is that dropping missing data from the population can be a practical approach. Allison [18] states that the worst case is that your mathematical model will tell you that based on the data there is no significant correlation identified by the data. So you would not use the results to make decisions. A good rule of thumb is that “no data” is better than “erroneous data” as someone making

decisions on erroneous data can led to serious misfortunes.

1.3.2 Impute the Missing Data

Another technique for dealing with missing data is to impute. To *Impute*: means to assign (a value) to something by *inference* from the value of the products or processes to which it contributes. Therefore it is reasonable that [18] another strategy, particularly for large amounts of missing data, is to substitute the missing data with some sort of conceivable "guess". In general terms, to impute data means to assign a value by inference or by a result that was extrapolated from the observed values. Fundamentally, when you impute the data of a sparse matrix you are filling in the missing data based on some algorithm or approach. The approaches for *data imputation* that will be compared in this research are:

- Impute with the simple mean (μ)
- Impute with an appropriate subgroup mean
- Impute using a regression estimate
- Impute using Maximum Likelihood Estimation (MLE)
- Impute using Matrix Factorization or Matrix Decomposition
- Impute using Optimization algorithms such as gradient descent/ascent.

1.3.3 Impute with Simple Mean

To impute or substitute the data with the overall population mean (μ) of the observed values is the easiest way to deal with missing data. [19] However, it should be used with caution because it can severely distort the distribution for this variable that could cause underestimates of the standard deviation. According to