Matrix Completion Techniques For Anomaly Detection in Network Attacks

---

A Thesis
Presented to
Department of Statistical Science
Duke University

James C. Wu

May 2018

Approved for the
Bachelor of Science in Statistical Science

Peter D. Hoff

Jerry P. Reiter

Galen Reeves

Mine Cetinkaya-Rundel, DUS

Acknowledgements

I thank my advisor, Professor Peter Hoff, and the Director of Undergraduate Studies, Professor Mine Cetinkaya-Rundel, for their guidance in this project. I also thank Duke University's Statistics Department for supervising this project and the Office of Information Technology for providing the dataset. Most of all I thank my parents for their continued unwavering support in all my endeavors.

Table of Contents

Abstract

The goal of this paper is to identify novel methods for detecting anomalies in network IP data. The data is comprised of four continuous features (source bytes, destination bytes, source packets, destination packets) divided by their respective source port and destination port combinations. Thus, the data is represented as a 3-dimensional tensor $T \in \mathbb{R}^{m \times n \times 4}$, where m is the number of source ports, n is the number of destination ports, and 4 is the number of numerical features. Each cell in T, t_{ijk} stores the mean of the observations of continuous feature k between source port at index i and destination port at index j. This paper proposes three techniques for generating means to fill in the missing cells in T, thereby completing the tensor, so as to provide reasonable estimates for new observations between every possible port combination. In the context of anomaly detection, new observations between ports that do not align closely with their corresponding estimate in T are considered anomalies. The first technique uses a low-rank singular value decomposition algorithm for completing individual matrix slices of the tensor. The second defines a statistical model for the values in T and uses a Bayesian Gibbs sampling procedure to simulate missing cells in individual matrix slices of T. Finally, the third approach extends the first and second approaches to completing the tensor all at once, rather than with completing individual matrices.

The goal of this paper is to identify novel methods for detecting anomalies in network IP data. The data is comprised of four continuous features (source bytes, destination bytes, source packets, destination packets) divided by their respective source port and destination port combinations. Thus, the data is represented as a 3-dimensional tensor $T \in \mathbb{R}^{m \times n \times 4}$, where m is the number of source ports, n is the number of destination ports, and 4 is the number of numerical features. Each cell in T, t_{ijk} stores the mean of the observations of continuous feature k between source port at index i and destination port at index j. This paper proposes three techniques for generating means to fill in the missing cells in T, thereby completing the tensor, so as to provide reasonable estimates for new observations between every possible port combination. In the context of anomaly detection, new observations between ports that do not align closely with their corresponding estimate in T are considered anomalies. The first technique uses a low-rank singular value decomposition algorithm for completing individual matrix slices of the tensor. The second defines a statistical model for the values in T and uses a Bayesian Gibbs sampling procedure to simulate missing cells in individual matrix slices of T. Finally, the third approach extends the first and second approaches to completing the tensor all at once, rather than with completing individual matrices.

Chapter 1

Introduction

1.1 Anomaly Detection

Anomaly detection is the identification of unusual patterns or observations that do not conform to expected behavior in a dataset. Anomalies can be broadly categorized into three categories:

-Point anomalies: A single instance of data is anomalous if it's too far off from the rest. For example detecting credit card fraud based on a single spending spree that represents the credit card being stolen and used.

-Contextual anomalies: The abnormality is context specific. This type of anomaly is common in time-series data. For instance, high spending on food and gifts every day during the holiday season is normal, but may be considered unusual otherwise.

-Collective anomalies: A set of data observations that when collectively assessed helps in detecting anomalies. For instance, repeated pings from a certain IP address to a port connection on a hosted network may be classified as a port scanner, which often preludes a network attack.

1.2 Network Attacks

Network security is becoming increasingly relevant as the flow of data, bandwith of transactions, and user dependency on hosted networks increase. As entire networks grow in nodes and complexity, attackers gain easier entry points of access to the network. The most benign of attackers attempt to shutdown networks (e.g. causing a website to shutdown with repeated pings to its server), while more malicious attempts involve hijacking the server to publish the attacker's own content or stealing unsecured data from the server, thus compromising the privacy of the network's users.

Network attackers follow a specific three step strategy when gathering intelligence on a network, the most important component of which is scanning. Network scanning is a procedure for identifying active hosts on a network. An attacker uses two particular types of scans, ping sweeps and port scans, to find information about the specific IP addresses that can be accessed over the Internet, their target's operating systems, system architecture, and the services running on each node/computer in the network.

These scanning methods leave digital signatures in the networks they evaluate because they apply specific pings that are then stored in the host's network logs. Thus, identifying a scanner or scanners from the millions of observed pings available in the network's logs is an anomaly detection problem. In particular, because the data is unlabeled, meaning it is unclear which observations are actually scanners and which are just standard user behavior, unsupervised approaches are necessary for tackling the problem.

The goal of this paper is to devise and evaluate techniques that use existing data to define expected behavior between ports. New observations between port connections that are far away from the defined expected behavior may be considered anomalies and investigated for whether they are a form of network attack.

1.3 Network Dataset

This particular dataset is from Duke University's Office of Information Technology (OIT), and it includes 1048575 observations in their network traffic during a five minute period in February 2017.

Argus is the open source network security tool that was used to collect the dataset. Argus focuses data collection on the interaction between different network ports. A network port is a number that identifies one side of a connection between two computers. Computers use port numbers to determine to which process or application a data message should be delivered. There exist 65,535 TCP (Transmission Control Protocol) ports. Using TCP, the computer sending the data connects directly to the computer it is sending data to, and stays connected for the duration of the transfer. Both computers have their own port number to identify their connection. In this dataset, the connections between two computers, known as sessions, are grouped by the IP address of the sender (the source). The bytes and packet values that are transmitted between two computers are accumulative over the set duration of the session's existence. Thus, each observation in the dataset contains data from a single session between two computers (the source address and destination address), each on their own source and destination port. Each session leaves a record with five numerical features and eight categorical features, which are described below.

1.3.1 Features

The networks dataset contains 13 features, 8 categorical and 5 numerical. The features are:

Continuous:

- StartTime (Start Time): the time when the observation is logged.
- SrcBytes (Source Bytes): the total number of bytes sent in the session
- SrcPkts (Source Packets): the number of packets sent in the session
- DstBytes (Destination Bytes): the total number of bytes received in the session
- DstPkts (Destination Packets): the number of packets received in the session

Note, the destination packets and bytes features do not have the same values as their source counterparts because the connections are compressed and decompressed into different forms and byte sizes when sent. For instance, it is possible for the number of destination packets to be larger than source packets. It is also possible for information to be lost during the connection.

Categorical:

- Flgs (connection flag): flow state flags seen in transaction between the two addresses.
- Proto (network protocol): specifies the rules used for information exchange via network addresses. Transmission Control Protocol (TCP) uses a set of rules to exchange messages with other Internet points at the information packet level, and Internet Protocol (IP) uses a set of rules to send and receive messages at the Internet address level.
- SrcAddr (Source Address): the IP address of the connection's source computer.
- DstAddr (Destination Address): the IP address of the connection's destination computer.
- Sport (Source Port): the network port number of the connection's source computer. A port numbers identifies the specific process to which a network message is forwarded when it arrives at a server.
- Dport (Destination Port): the network port number of the connection's destination.
- Dir (direction): the direction of the connection.
- State (connection state): a categorical assessment of the current phase in the transaction when the timestamp is recorded.

The addresses have been anonymized for security reasons.

1.3.2 Removing Features

The Argus wiki and the OIT manual provide key insights into the structure and nature of the data. Each session has its own start time but does not have a recorded end time. Furthermore the protocol in this dataset is always TCP protocol and the direction is

always to the right (i.e. Source to Destination). This information supports dropping "Proto", "StartTime", and "Dir" from the dataset for future analysis because they do not present any information regarding whether an observation can be considered an anomaly. Furthermore, the "State" and "Flgs" features may not be reliable because Argus occasionally resets the state data statistics and fails to assign connection flags to many connections during monitoring, so "State" and "Flgs" are also dropped.

1.3.3 Status Quo Solution

OIT's current solution for detecting scanners relies on specific domain knowledge gathered from diagnostics programs and data analysis completed on existing data. They prevent scanners by blocking IP addresses that violate certain rules. The specific conditional checks in these rules are private for security reasons, but they are similar to evaluating the size of transactions and detecting repeated connections between particular ports.

In this solution, any observation that does not fit within the constraints specified by the rules is classified as an anomaly and its source IP is blocked or investigated. While this solution presents a methodical way for banning IP addresses, it is inflexible, prone to detecting false negatives, and fails to detect observations that may be within the parameter constraints of the rules. The solution lacks a way to detect anomalies with respect to the parameters that are unspecified in the rules or combinations of parameters.

1.4 Problem Formulation

Preliminary data analysis indicated that there may exist patterns and regularities between different port combinations. For instance, a particular source and destination port may frequently contain large byte transactions in their connections. Devising a systematic way to identify expected or "regular" interactions between particular combinations may present outliers that can be further investigated for scanner behavior.

This approach to the anomaly detection problem reduces the dataset to the values of the four continuous features, SrcBytes, SrcPkts, DstBytes, DstPkts, observed across different source port (SrcPort) and destination port (DstPort) combinations. The data can be represented as a 3-dimensional tensor $T \in \mathbb{R}^{m \times n \times 4}$ where m represents the number of source ports, n represents the number of destination ports, and 4 accounts for the four continuous features in the dataset. Each cell, t_{ijk}, contains the mean of all the observations observed between the source port at index i and destination port at index j. In the cases where the combination of i and j is not observed in the dataset, t_{ijk} is considered missing (NA). Note, the data is collected in a way where either all

four continuous features are observed, or none are observed, i.e. a missing cell, t_{ij1} indicates t_{ij2}, t_{ij3}, and t_{ij4} are also missing.

The goal of this paper is to devise and assess techniques for calculating a reasonable estimate for the missing cells in T to create the completed tensor $T' \in \mathbb{R}^{m \times n \times 4}$. As new observations are observed for combinations of source ports at index i and destination ports at index j, the t'_{ijk} values can be interpreted as an approximation for the expected behavior for that particular port combination. Observations with continuous features that are a certain threshold away from t'_{ijk} may be marked as anomalies and investigated further.

1.5 Introduction of Methods

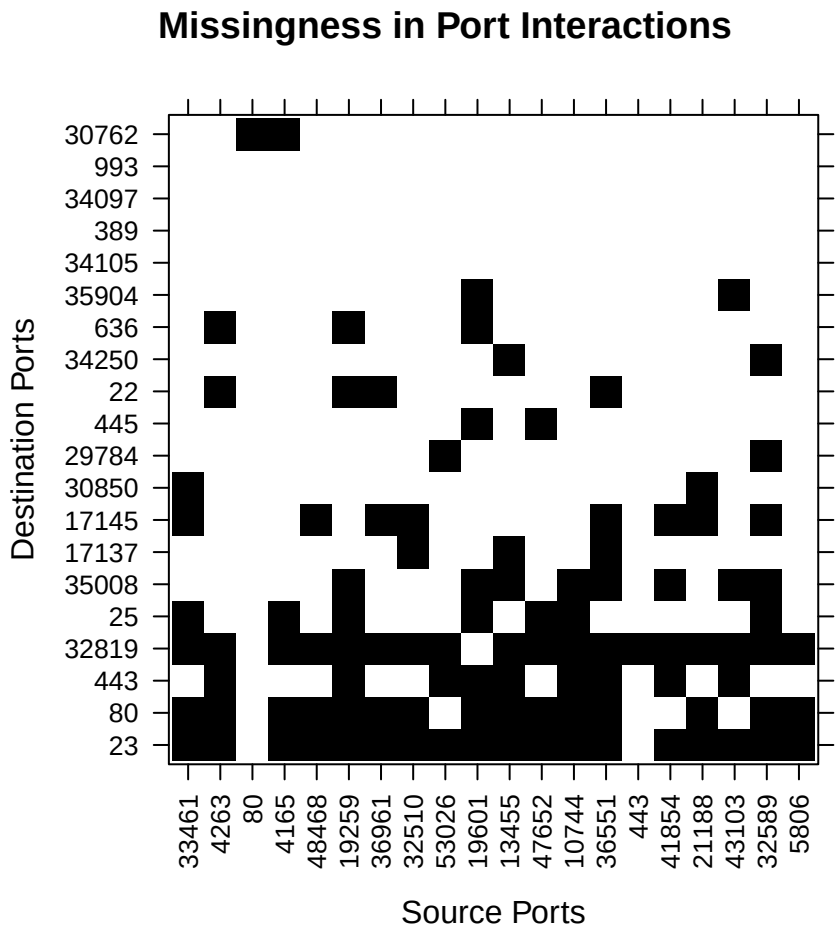
Chapters 3, 4, and 5 discuss three methods for completing the tensor T. The first two techniques slice T into four matrices divided by the four continuous features: $Y^{(1)}, Y^{(2)}, Y^{(3)}, Y^{(4)} \in \mathbb{R}^{m \times n}$. Because both techniques apply to each matrix separately, the techniques will refer to a general matrix Y, which represents any of $Y^{(1)}, Y^{(2)}, Y^{(3)}, Y^{(4)}$. Each $Y^{(k)}$ has missingness because not every source port interacts with every destination port. Chapter three considers an iterative approach using an alternating least squares technique and the best low-rank approximation of Y to calculate estimates for the missing values of the singular value decomposition of Y. While the approach does not consider the variable sample sizes and variances for each port combination, essentially treating each cell as a scalar value rather than a mean of observations, it is the fastest technique of the three and provides reasonable performance metrics. Chapter four shores up the weaknesses of chapter three by defining an additive statistical model that accounts for the variable sample sizes and variances of the observations in each port combination. This model is generalized to a weighted least squares problem, and a Bayesian approach is used to create a Gibbs Sampler to iteratively simulate the row factors and column factors with their respective variances of the model. Each approach is validated on simulated data where the ground truth is known to verify correctness before being applied to the actual networks dataset. Finally, chapter five proposes a tensor completion technique that simulates cells in T without slicing the tensor. This approach allows considers correlation and collinearity between the different continuous features and relies on the PARAFAC tensor decomposition (as opposed to the two-dimensional matrix singular value decomposition).

Chapter 2

Modeling Port Relationships

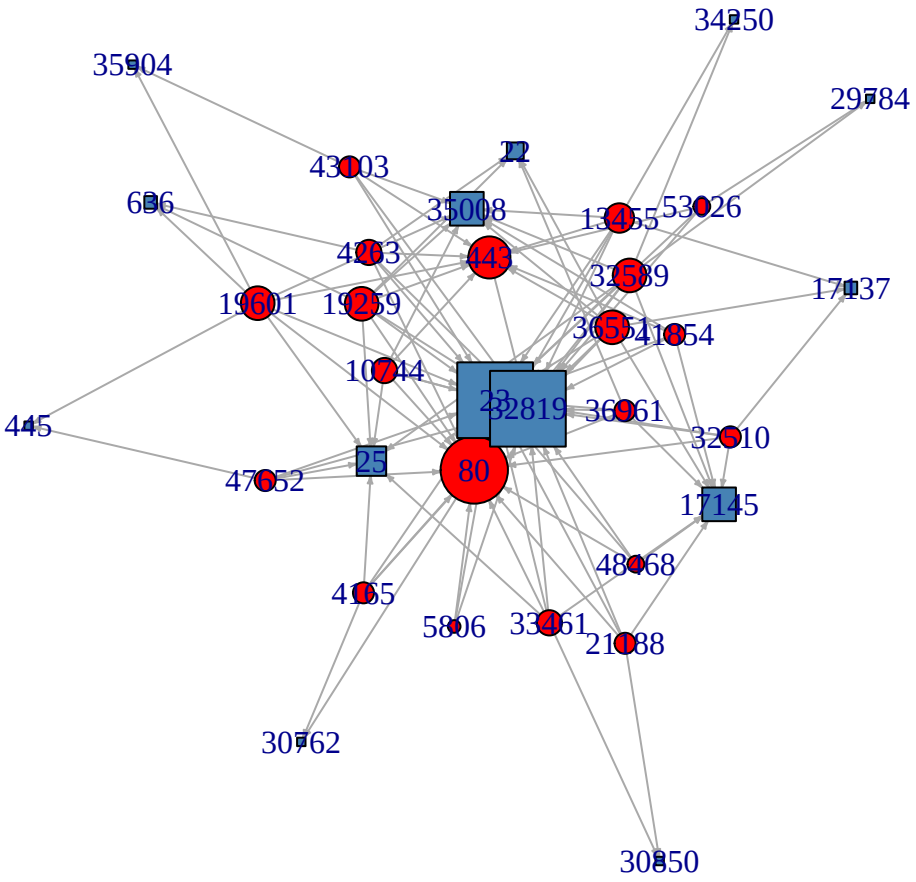
The following properties of T inform the matrix and tensor completion strategies in chapters 3, 4, and 5.

2.1 Missingness



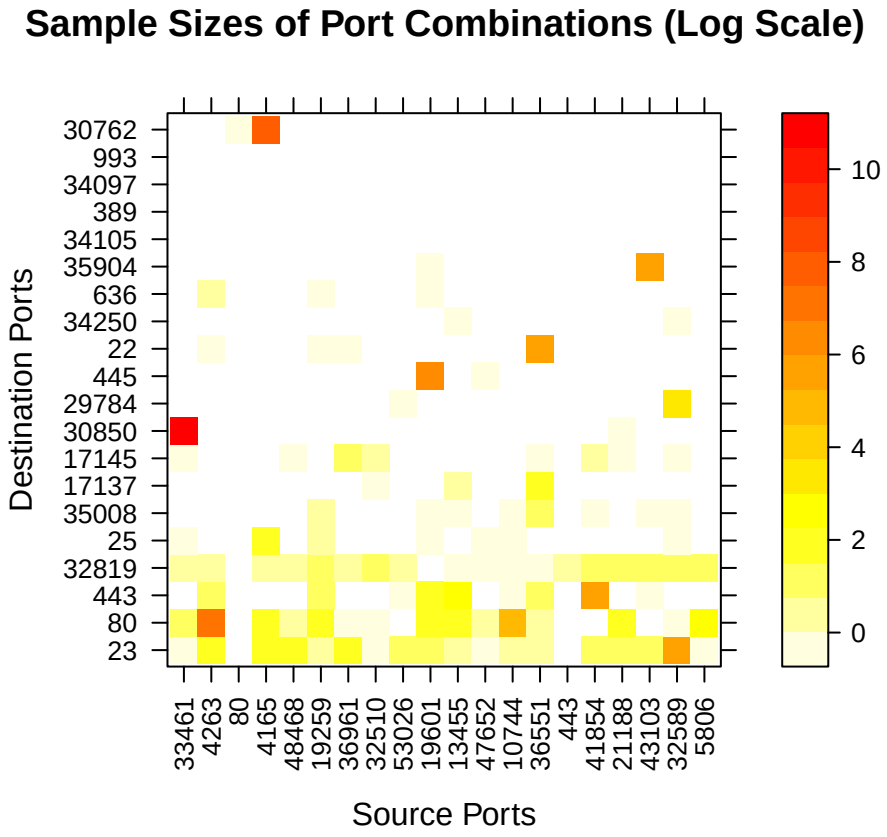
The above plot represents the missingness in the port combinations for pairings of the top 20 most used source ports and destination ports. The black cells represent missingness; of the 400 cells in the matrix, 295 (73.75%) of cells are missing observations. This single matrix slice can be extrapolated to missingness in ports throughout the entire tensor because the dataset is collected in a way such that either all four continuous features are observed, or none are observed. It is important to note that missingness is not uniform across source and destination port combinations. In the event that an entire row or column of port combinations is missing, the port at that respective index will need to be discarded because all three completion techniques depend on the row and column effects when simulating a value for a missing cell.

2.2 Port Connections



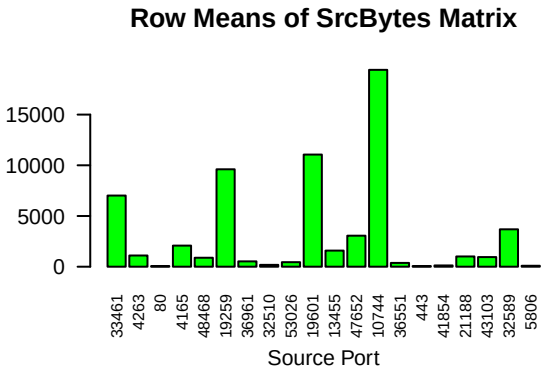
The above ports network graph displays the pairings between the top twenty source ports (red circles) and the top destination ports (blue squares). When a square and circle are connected it indicates that there exist observations for this particular port combination in the dataset. The size of each node reflects the number of paired observations that were observed using that particular port. Clearly, not every source port is paired with every destination port and vice versa (not every node is connected to every other node). These missing combinations reflect missing cells in the T tensor, and consequently they correspond to the combinations that require values to be estimated

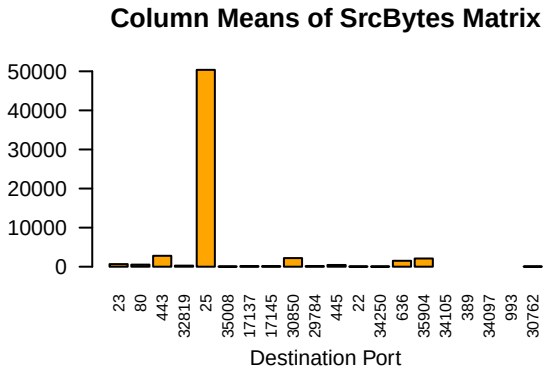
2.3 Varied Sample Sizes

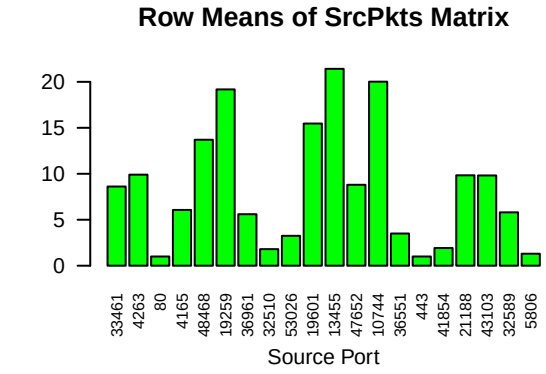


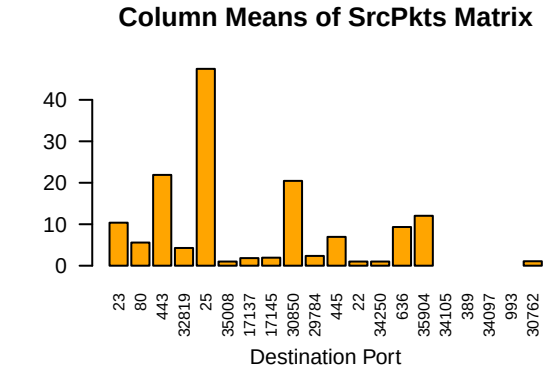
To better visualize the sample sizes first displayed in the network graph, the above heat plot represents the sample sizes for each source-destination port combination on a log scale for clarity. It is again clear there are certain combinations that have many observations (range of 35000 on the original scale), while most of the observations are 0, indicating no observations were observed and the corresponding cell in T is missing, or near 0, indicating few observations were observed. The large variation in sample sizes again suggests a simulation technique that accounts for sample size of a missing cell's related row and column cells is necessary.

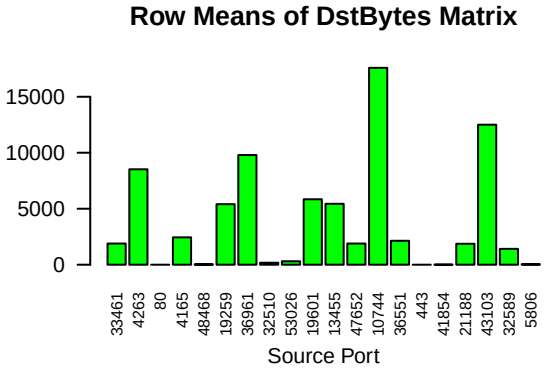
2.4 Row and Column Properties

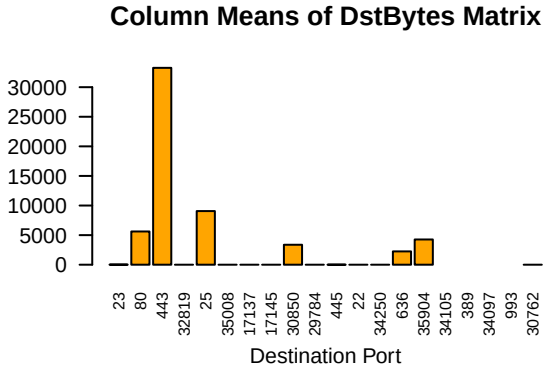


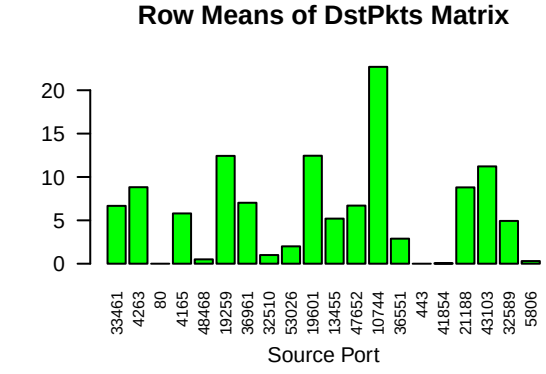


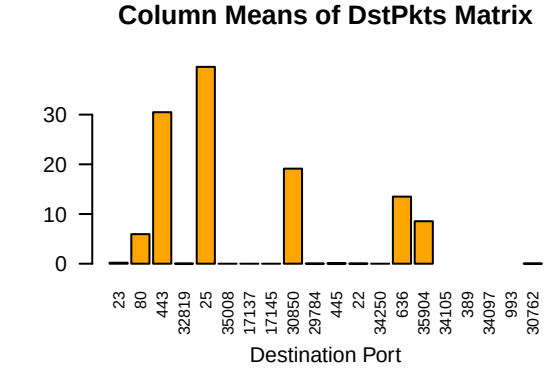








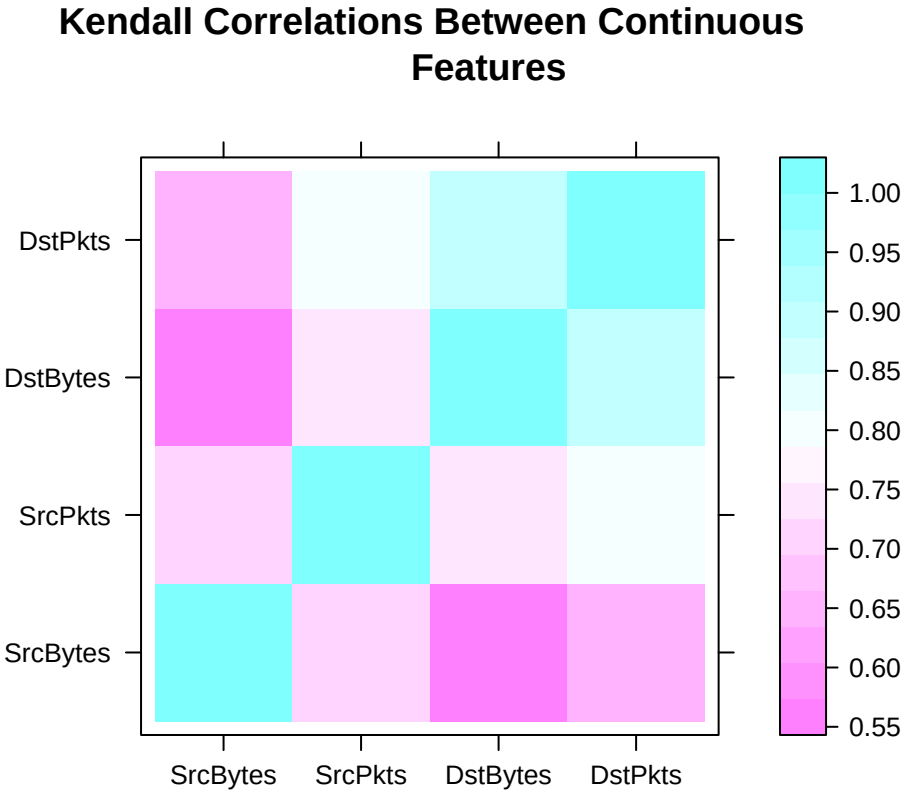




The bar plots above represent the row and column means of the continuous features for each slice of the tensor. These row means and column means inform simulation techniques for the missing cells within those respective rows and columns. There exist clear outliers in the means for certain rows and columns. This outlier behavior is undoubtedly caused by outliers existing within the cells in that particular row or column. These outliers exist because each cell represents the mean of all observations that occured within a particular port combination, regardless of sample size (i.e. some cells may have a few large observations, resulting in a large mean that skews the cell's row and column mean). Thus, cells that only have a few observed observations have a disproportionately large effect on their respective row and column mean.

These outliers may cause problems with simulating missing values in that row or column because the outliers will have a disproportionately large effect on the simulated value than the other observations, which are more close to the median in the missing value's row or column. This behavior suggests that the completion techniques that take into account variances among the row and column means and the number of samples observed for each port combination will result in higher quality estimations for the missing port combinations. In particular, least squares methods fail to take into account the ddiferent amounts of information available in each cell. If a cell only has one observation, it should not be treated with equal weight to a cell that is actually the mean of 1000 observations.

2.5 Correlations



The matrix above describes the Kendall rank correlations (commonly referred to as Kendall's tau coefficent) between the four continuous features in the dataset. Intuitively, the Kendall correlation between two features will be high when observations have a similar rank (i.e. relative position label of observations within the variable: 1st, 2nd, 3rd, etc.) between the two variables, and low when observations have a dissimilar rank between the two variables. The range of correlations is [-1, 1]. Kendall correlation was selected as a measure because it evaluates ranks between observations, as opposed to Pearson, which is more susceptible to outliers in the dataset (large byte and packet observations in the continuous features skewed the Pearson measures).

It is clear there exist strong correlations between the four continuous features, DstBytes and DstPkts in particular. This behavior suggests a technique that produces estimates for the missing cells with all four features considered at once (i.e. a technique that simulates the entire tensor as a whole rather than in slices) will also be valuable.

Chapter 3

Alternating Least Squares for Matrix Completion

This approach determines the best low-rank approximation for Y using the Eckart-Young-Mirsky Theorem to repeatedly generate the orthonormal matrices U and V of the singular value decomposition of Y ($Y = UDV^T$) in an alternating pattern. The technique is proven to correctly determine a low rank approximation on simulated matrices where the true rank is known. Once the optimal low rank is determined from the actual dataset the optimal low rank is used in the technique to generate estimates for the missing cells of Y. The technique's fit is assessed by comparing the fitted versus the observed values.

3.1 Related Work

A common approach to matrix completion revolves around the underlying assumption that there exists a low rank approximation for the data matrix, particularly in the case of high dimensional data. Hastie, Mazumder, Lee, and Zadeh (2014) devised a similar approach to the one presented in this chapter that fuses nuclear-norm-regularized matrix approximation (Candes and Tao, 2009, Mazumder, Hastie and Tibshirani, 2010) and maximum-margin matrix factorization (Srebro, Rennie and Jaakkola, 2005), resulting in a fast alternating least squares that relies on a low rank singular value decomposition to drive an efficient algorithm for large matrix factorization.

Similar techniques for matrix completion were employed heavily in the Netflix Challenge where competitors predicted ratings for movies by users that had not watched the movie based on the other ratings in the matrix of users and movies. The winning team, BellKor's Pragmatic Chaos, employed a low rank decomposition technique to reduce the incredibly large dataset, so that other algorithms could be applied without too much computational overhead (Andreas Toscher, Michael Jahrer, Robert M. Bell 2009).

Network datasets can range up to billions of observations (recall this particular dataset with 1 million observations was collected in just five minutes). Furthermore, there are up to 65000 possible network source and destination ports, so the resulting network tensor has large dimensions. Given the nature of the dataset and prior work in matrix completion, the technique in this chapter assumes a low rank decomposition to implement an alternating least squares completion technique.

3.2 Matrix Completion Algorithm

Let $F \in \mathbb{R}^{m \times n}$ be a sparse matrix that represents the frequencies of combinations, i.e $F[i, j]$ represents the number of observations for the $ith\ jth$ port combination Let $M \in \mathbb{R}^{m \times n}$ represents a boolean matrix of whether the corresponding Y values are missing. $Y[!M]$ represents all of the missing values in Y, so $m_{ij} = 0$ if $f_{iJ} = 0$.

The objective is

$$\min_{r} \sum_{i,j:F_{i,j}>0} (y_{i,j} - u_i D v_j^T)^2$$

where $UDV^{(k)T}$ represents the singular value decomposition of Y and r is the low rank approximation for Y. There are multiple steps to the matrix completion process:

3.2.1 ANOVA Initial Imputation

An analysis of variance (ANOVA) imputation is used to fill in the initial values for y_{ij}. This yields an additive model dependent upon the means of the present observations:

$$y_{ij} = a_i + b_j - \mu$$

whereμ is the overall mean of Y, a_i is the row mean, and b_j is column mean of y_{ij}.

3.2.2 Repeated Simulation

The repeated imputation procedure solves $Y^{(s)}[!M] = R_r(Y^{(s-1)})[!M]$ where $R_r(\cdot)$ is the best rank r approximation for the s-th step. For each step (s) the singular value decomposition decomposes

$$Y^{(s)} = U^{(s)} D^{(s)} V^{T(s)}$$

where D is a diagonal matrix of the singular values, U is the left singular vectors of Y and V is the right singular vectors of Y.

The Eckart-Young-Mirsky Theorem provides the best rank r approximation for the missing values in $Y^{(s+1)}$. Recall $Y[M]$ represents all of the missing values of Y. Applying the EYM theorem:

$$Y^{(s+1)}[M] = (U[, 1:r]^{(s)} D[, 1:r]^{(s)} V[, 1:r]^{T(s)})[M]$$

Where $U[, 1 : r]$ represents the first r columns of U and the same for D and V.

3.2.3 Convergence Criterion

The Eckart-Young-Mirsky rank approximation step is repeated until the relative difference between $Y^{(s+1)}$ and $Y^{(s)}$ falls below a set threshold, H. The relative difference threshold is expressed:

$$\frac{\|Y^{(s+1)} - Y^{(s)}\|_2}{\|Y^{(s)}\|_2} < H$$

where $\|Y\|_2$ is the Frobenius norm for matrices. The denominator of the expression ensures the convergence criterion is invariate to a scale change in the matrix itself.

3.3 Best Rank Approximation

To determine the best rank for approximating Y, Leave-One-Out Cross Validation (LOOCV) is used to generate prediction errors for each possible rank. LOOCV cycles through the observed values, setting each to NA (missing), and then performs the described matrix completion process. The prediction error is then calculated as some function of the difference between the imputed value and the true value. In this case, the algorithm records root mean square error

$$\sqrt{\frac{\sum(\hat{y}_{ij} - y_{ij})^2}{z}}$$

where z is the number of observations not missing.

3.4 Validation Against Simulated Data

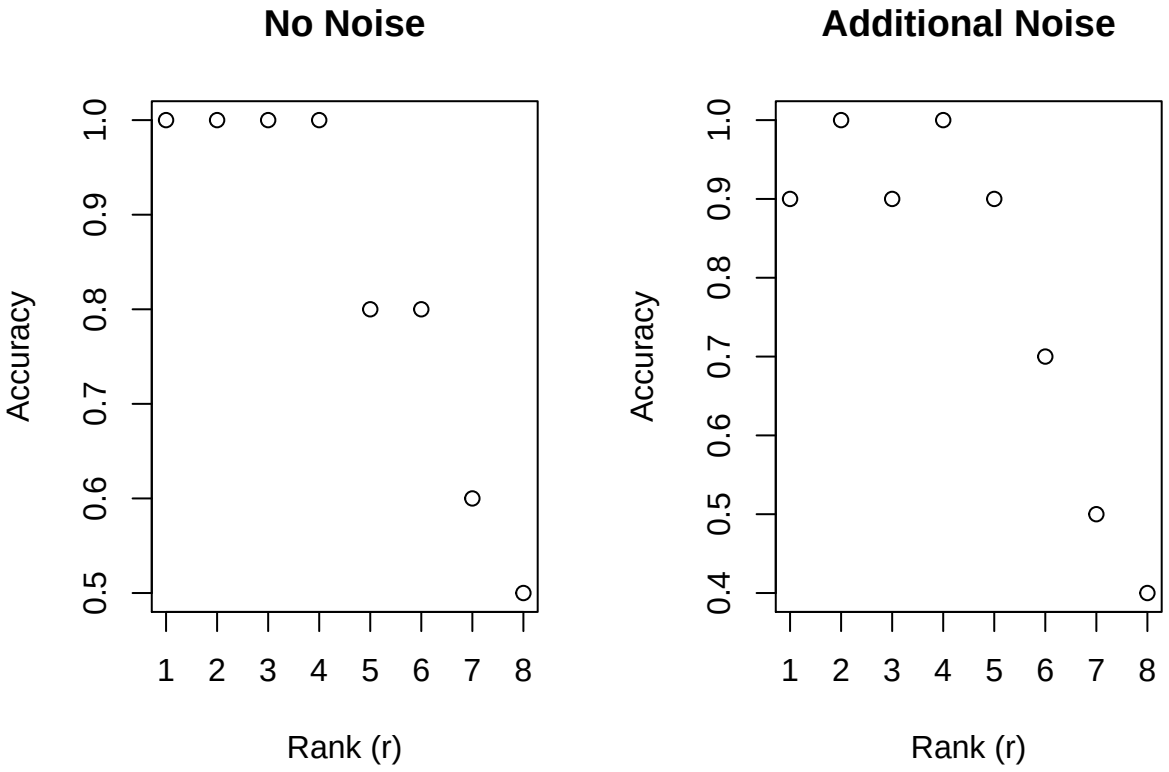
Before applying the algorithm on the real data it is useful to validate the algorithmic approach against simulated data where the true rank is already known.

3.4.1 Simulating a Low Rank Matrix

Taking the matrix product of two lower dimension matrices yields a higher dimension matrix with low rank. Explicitly, given matrix $A \in \mathbb{R}^{m \times r}$ and $B \in \mathbb{R}^{r \times n}$, $A \times B = C$ where $C \in \mathbb{R}^{m \times n}$ has rank r. Thus, when $r < m, r < n$ the matrix C has an optimal low rank that minimizes the root mean square error from the leave one out cross validation procedure. To add noise to the simulated matrix, C, simply add an error matrix, $E \in \mathbb{R}^{m \times n}$ sampled from a normal distribution.

This procedure provides a computationally efficient way to simulate many random low rank matrices to use as inputs for the validation procedure. In the case of simulated matrices, there are no missing entries, so the leave one out cross validation procedure sequentially removes each cell in the matrix, imputes its value using the rank being investigated, and considers the individual cell error as the difference between the true value and the imputed value. The overall root mean square error for the technique is then calculated with the aggregate each of these individual cell errors.

3.4.2 Approximating Optimal Rank

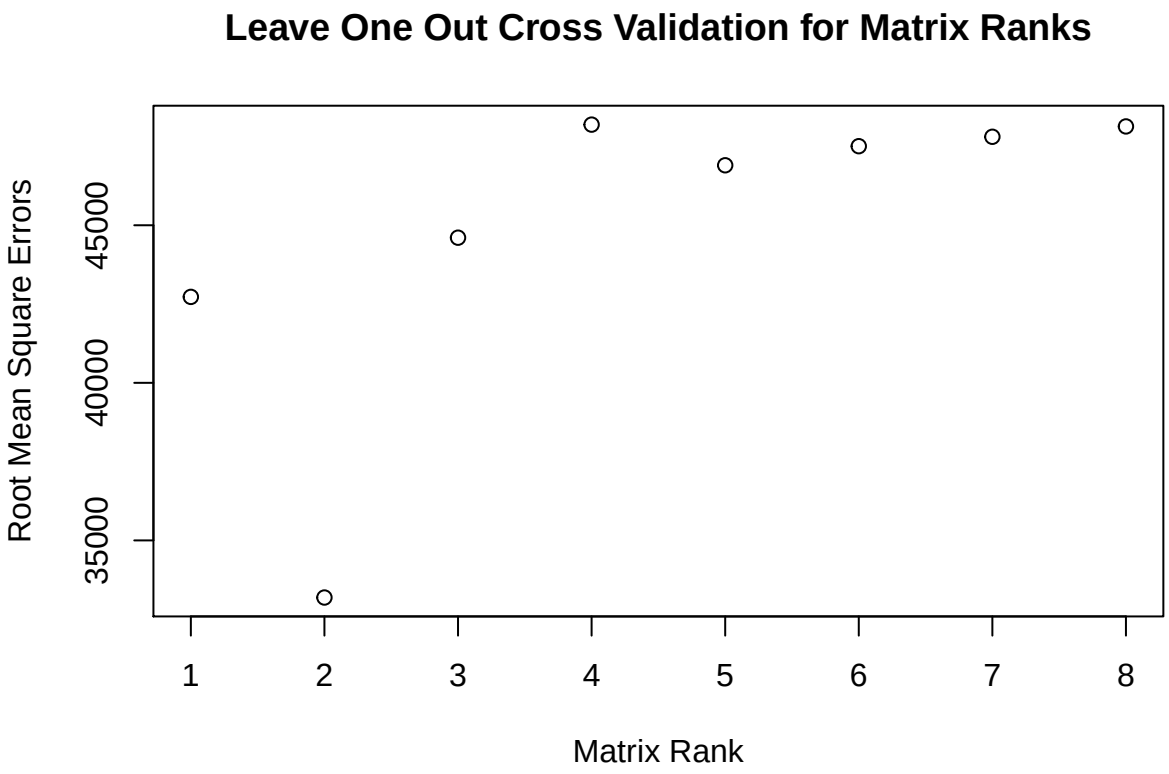


The above plots represent the accuracy of the matrix completion technique for matrices with true rank $r = 1, 2, ...8$. The accuracy is measured by simulating 10 random matrices (dimensions ranging from 74 to 100 values) with low rank of r for each value of r (80 simulated matrices total), and then running the leave one out cross validation procedure described above on the matrix C to generate a root mean square error for each possible rank. The accuracy is the calculated using the number of times the rank with the lowest error matches the true simulated rank divided by 10 (the number of trials with rank r). Note as the true rank becomes larger, the technique performs far worse at determining the true rank. This behavior is due to the fact that LOOCV attempts to find the rank that minimizes the root mean square error, not necessarily the true low rank approximation for a matrix. Higher true ranks tend to give higher out-of-sample validation error, so LOOCV will still select a low rank approximation for simulated matrices that have relatively high true ranks. For instance, a simulated

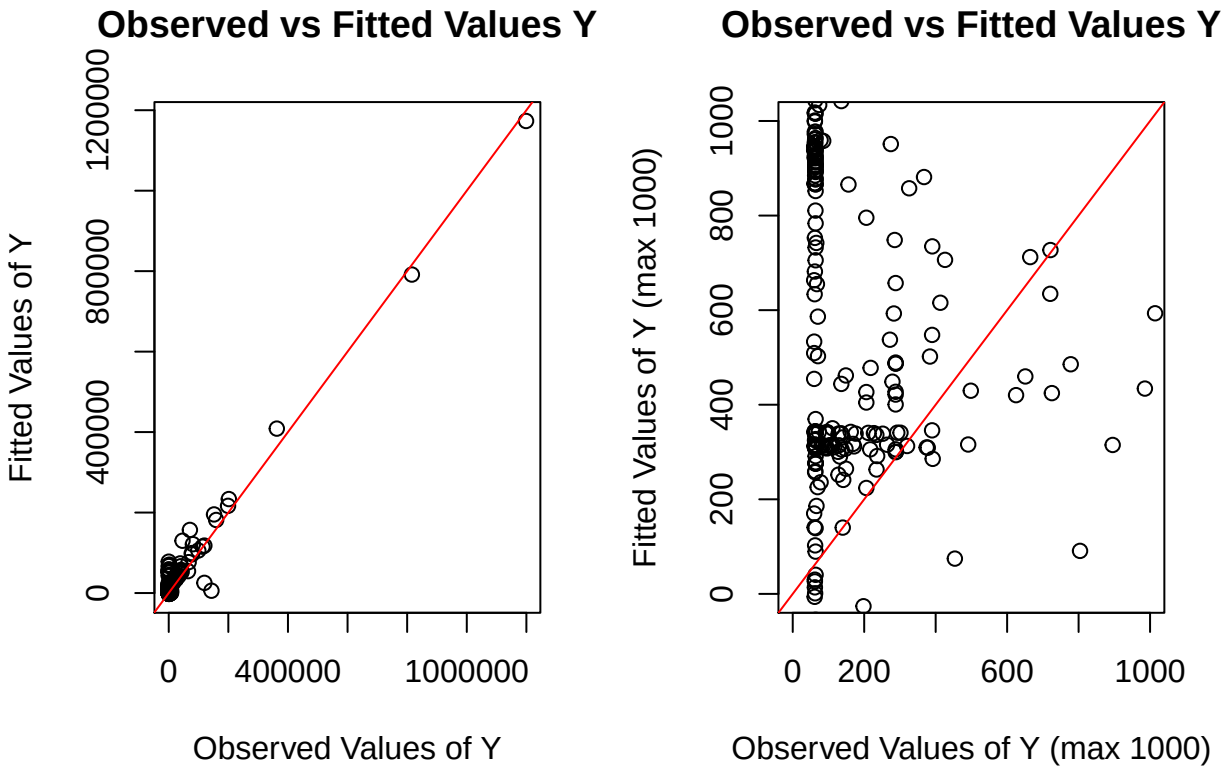
data matrix may have a true rank of 8, but it may also be very close to rank 2, which results in LOOCV selecting rank 2 as the optimal low rank approximation for minimizing error.

Furthermore, when noise is applied to each simulated matrix, C, (through the addition of a noise matrix E), the algorithm tends to perform worse at a majority of the attempted ranks. This is expected because the addition of noise to every cell in the matrix may obscure the true rank from the LOOCV procedure.

3.5 Results on Real Data



The above plot displays the root mean square errors from the leave one out cross validation process across different rank inputs into the algorithm. It's clear that rank 2 provides the best low-rank approximation for estimating missing values in Y using the alternating least squares algorithm. Thus, the dataset is fitted with the algorithm using rank 2 to estimate the missing values in Y.

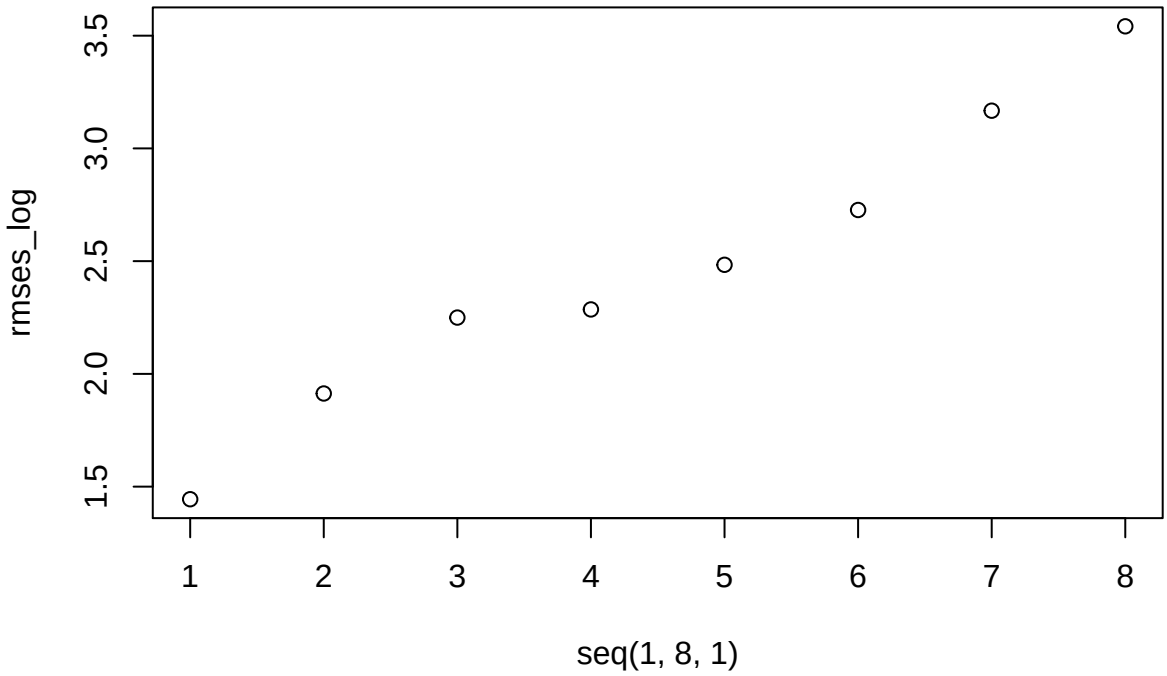


The two plots above display the true values of the Y matrix (i.e. the non-missing values) versus their corresponding fitted values using the alternating least squares algorithm with an input of rank 2. The first plot displays all values and shows a somewhat positive linear trend (an ideal fit of the true values would be a scatter of points following a linear relationship, represented in red). However, several outliers with large true values skew this dataset and cause the plot to appear linear. Closer examination of the true and fitted values smaller than 1000 (the plot on the right) reveals the relationship is far from the linear pattern.

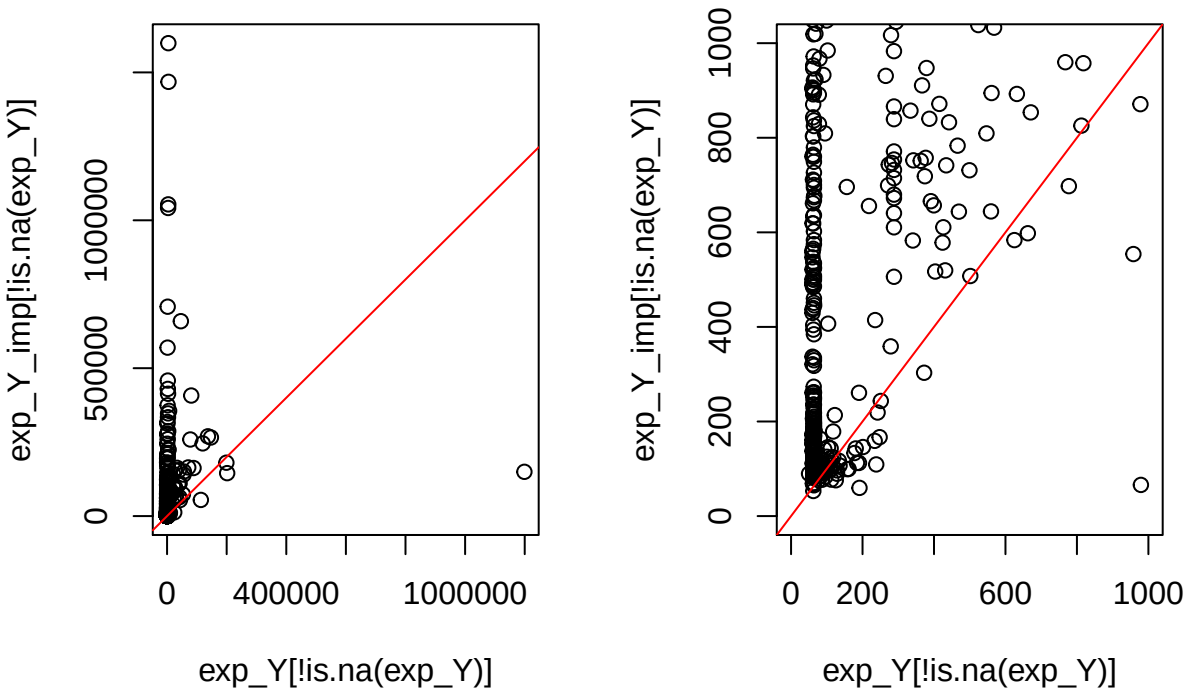
3.5.1 Scale Transformations

The poorly fitted results motivates a consideration of the scale of the data. The present algorithm uses the sample averages of the overall matrix as well as the row and column means when imputing each missing cell value. This reliance upon sample means leads to susceptiblity to outliers. Moreover exploratory data analysis reveals the dataset contains outliers, particularly in the SrcBytes and DstBytes measurements. Because outliers drive the sum of squares for the alternating least squares procedure, the poor fit on the data is unsurprising. Thus, a transformation of features in the dataset may be appropriate for improving the fit of the algorithm.

When a natural logarithm transformation is applied to the raw dataset before any simulation steps are taken, the alternating least squares imputation algorithm yields the following root mean square errors varied by rank.



Now the optimal rank from the LOOCV procedure is 1. The algorithm is run on the log transformed dataset with a low rank approximation of 1 and the fitted versus observed values are again compared.



The values above have been retransformed (exponentiated) to the dataset's original scale after the procedure was completed. The fit still appears to be quite poor and there is not much difference between the log transformed output versus the original non-transformed output.

This poor performance may largely be due to the fact the algorithm does not account for the variability in the sample size and variance in the observed interactions for each cell. Unlike the Netflix Competition, in which each cell of the matrix being completed contained only a single user rating of a movie, the matrix in this problem contains the mean of a variable number of observations corresponding to particular port combinations.

Chapter 4

A Bayesian Approach to Matrix Completion

The previous section's results reflected the need for a completion strategy that accounts for the variability in the number of observations observed for each port combination when imputing that particular combination's cell. The previous technique fails to take into account the differing sample size and variance in each cell's observations, so the algorithm treated each y_{ij} as a single value rather than the mean and variance of a vector of observations. The following section constructs a statistical model that takes the sample size of observations and their variances for each cell into account and repeatedly simulates values for the missing cells using a Gibbs Sampling procedure. The sampling procedure relies upon first building a general model for simulating the row and column facors with their respective standard deviations. After calculating the full conditionals for the parameters of this general model, the overall procedure repeatedly simulates values from these full conditional distributions, alternating between simulating the matrix of row factors and the matrix of column factors along with their respective standard deviations. Note, this technique again slices the tensor into the four separate matrices, $Y^{(1)}, Y^{(2)}, Y^{(3)}, Y^{(4)} \in \mathbb{R}^{m \times n}$ (referred to as Y in general), and the model can be applied to each matrix $Y^{(k)}$ separately.

4.1 Related Work

Bayesian methods are becoming increasingly popular as a matrix completion technique for large scale datasets. Work by Zhou, Wang, Chen, Paisley, Dunson and Carin (2010) indicate Gibbs Sampling provides an efficient solution to large scale problems and yields "predictions as well as a measure of confidence in each prediction." Their paper considers algorithm performance in several datasets of varying scale and relationship between variables, and the results indicate strong performance compared to other common approaches. Granted, this approach considers non-parametric Bayesian matrix completion, while the Gibbs Sampler in this chapter relies upon constructing the full

conditional distributions for parameters in a defined statistical model. Nevertheless, the hypothesis remains that Gibbs Sampling provides an efficient and effective solution for matrix completion. Mai and Alquier also support this claim in their paper "A Bayesian Approach for Noisy Matrix Completion: Optimal Rate under General Sampling Distribution" (2014), in which they construct a Bayesian estimator that relies upon the premise that "Bayesian methods for low-rank matrix completion with noise have been shown to be very efficient computationally." They apply this technique to the Netflix competition dataset as a case study.

4.2 Statistical Model for Port Relationships

The following statistical model is defined for the cells in Y:

$$y_{ij} = u_i^T v_j + \frac{\sigma_i \tau_j}{\sqrt{s_{ij}}} \epsilon_{ij}$$

where u_i represents the row factors, v_j represents the column factors, σ_i represents the standard deviation of each row in the matrix, τ_j represents the standard deviations of each column in the matrix, s_{ij} represents the sample size of observations observed for source port i and destination port j, and $\epsilon_{ij} \sim N(0, 1)$. Fixing the j values in the analysis (i.e. v_j and τ_j are known) enables the model to be rewritten in the form of a weighted least squares model for estimating u_i and σ_i. Similarly, when i is fixed, the model can be rewritten to estimate v_j and τ_j. To demonstrate this property, the procedure for estimating u_i and τ_j given known values for v_j and τ_j is described below. The same procedure is possible for v_j and τ_j when u_i and σ_i are known.

4.3 General Linear Model for Simulating Row and Column Factors

Varying $j = 1...n$ the model above yields the following cell values:

$$y_{i1} = u_i^T v_1 + \frac{\sigma_i \tau_1}{\sqrt{s_{i1}}} \epsilon_{i1}$$

$$...$$

$$y_{in} = u_i^T v_n + \frac{\sigma_i \tau_n}{\sqrt{s_{in}}} \epsilon_{in}$$

Vectorizing all of these equations varied across $j = 1...n$ yields:

$$\vec{y_i} = V u_i + \sigma_i W^{1/2} \vec{\epsilon}$$

where $V \in \mathbb{R}^{n \times p}$ is the matrix of column factors (p is the dimension of the latent factors), and $W \in \mathbb{R}^{n \times n}$ is the diagonal matrix of weights, such that

$$V = \begin{bmatrix} v_1^T - \\ v_2^T - \\ \dots \\ v_n^T - \end{bmatrix}, W = \begin{bmatrix} w_1 & & \\ & \ddots & \\ & & w_n \end{bmatrix} = \begin{bmatrix} \frac{\tau_1^2}{s_{11}} & & \\ & \ddots & \\ & & \frac{\tau_n^2}{s_{nn}} \end{bmatrix}$$

Note τ^2 refers to the variance, variance being the square of the standard deviation.

This model can be rewritten in a general form:

$$\vec{y} = X\beta + \sigma W^{1/2}\epsilon$$

where X represents V, β represents u_i and σ represents σ_i.

This is a modified form of the Generalized Least Squares Model (GLS), which gives a weighted least squares estimate of β, and it is appropriate when the error terms are not independent and identically distributed. Bayesian analysis of this problem provides similar parameter estimates to GLS, and both ordinary least squares and GLS provide unbiased parameter estimates of β with the latter giving estimates with a lower variance because the non-Bayes estimator serves as a limit of the Bayes estimator.

The full conditional distributions of the random variables β and σ^2 (note σ is squared in the model) for this case are described below:

$$\{\beta \mid X, \vec{y}, \sigma^2\} \sim MVN(\beta_n, \Sigma_n)$$

$$\{\sigma^2 \mid X, \vec{y}, \beta\} \sim IG(\frac{\nu_0 + n}{2}, \frac{v_0\sigma_0^2 + SSR_W}{2})$$

where MVN represents the Multivariate Normal Distribution, and IG represents the Inverse Gamma distribution.

$$\begin{cases} \Sigma_n = (X^T W^{-1} X/\sigma^2 + \Sigma_0^{-1})^{-1} \\ \beta_n = \Sigma_n (X^T W^{-1} y/\sigma^2 + \Sigma_0^{-1}\beta_0) \end{cases}$$

$$SSR_W = (y - X\beta)^T W^{-1} (y - X\beta)$$

The formulation for the closed form full conditional distributions for the β and σ^2 parameters are based upon the general full conditionals established for a regression model with correlated errors (Hoff 2009). These particular full conditional formulations are a special case of this model where W is a diagonal matrix and so the covariance matrix is diagonal. This general regression model formulation also conveniently specifies the prior distributions.

The remaining variables in the closed form full conditionals come from the parameter's prior distributions, which are defined as follows:

$$\beta \sim MVN(\beta_0, \Sigma_0)$$

$$\sigma^2 \sim IG(\frac{\nu_0}{2}, \frac{\upsilon_0}{2}\sigma_0^2)$$

The initial values for the prior distributions are set as: $\beta_0 = 0$, $\Sigma_0 = \gamma^2 I$ where γ^2 is a large number and I is the $m \times n$ identity matrix, $\nu_0 = 2$, $\sigma_0^2 = 1$. This results in a diffuse prior for β that spreads out the density, and a noninformative prior for σ^2.

Work by Alquier, Cottet, Chopin, Rousseau (2014) reveal that a standard approach to assigning priors in Bayesian Matrix completion "is to assign an inverse gamma prior to the singular values of a certain singular value decomposition of the matrix of interest; this prior is conjugate. However, [they] show that two other types of priors (again for the singular values) may be conjugate for this model: a gamma prior, and a discrete prior. Conjugacy is very convenient, as it makes it possible to implement either Gibbs sampling or Variational Bayes." In the case of this problem, the distributions of the priors are defined to be diffuse (β) and noninformative (σ^2), so that the effects of the priors on the posteriors are limited when compared to the effects of the observed data.

4.3.1 Gibbs sampler for the General Linear Model

Following the formulation of the model and the definition of the priors, a general Gibbs sampler function is created to simulate samples from the full conditional of each parameter in the statistical model, which iteratively creates an approximate value for each cell.

The Gibbs sampler algorithm progresses as follows:

Let the parameters at step s be:

$\phi^{(s)} = \{\beta^{(s)}, \sigma^{2(s)}\}$

Sample $\beta^{(s+1)} \sim P(\beta \mid X, \vec{y}, \sigma^{2(k)})$

Sample $\sigma^2 \sim P(\sigma^2 \mid X, \vec{y}, \beta^{(s+1)})$

Set $\phi^{(s+1)} = \{\beta^{(s+1)}, \sigma^{2(k+1)}\}$

This Gibbs sampler serves as a general technique that can be used to simulate both the values of u_i and σ_i or v_j and τ_j depending on the inputs it is given because the formulation of both models are identical; they only differ by the the inputs, X and W, which are calculated, and y which is sliced directly from Y. In the context of the problem, this function can first be called repeatedly to simulate all of the rows in the matrix Y, then called repeatedly with updated inputs to simulate all of the columns of the matrix.

4.3.2 Validation on Simulated Data

Before using the general Gibbs sampler function in the overall procedure for simulating missing values in Y, it is necessary to validate the procedure's effectiveness on simulated data where the ground truth is known. As the algorithm runs, it stores a matrix of β vectors and a vector of σ^2 scalars. Thus, if $S = 50$, i.e. the algorithm samples 50 β and 50 σ, the final returned output will be

$$\beta = \begin{bmatrix} \beta_1^T - \\ \beta_2^T - \\ \dots \\ \beta_{50}^T - \end{bmatrix}, \vec{\sigma^2} = \begin{bmatrix} \sigma_1^2 \\ . \\ . \\ \sigma_{50}^2 \end{bmatrix}$$

Using random sampled values from the normal distribution for X and random sampled values from the exponential distribution for W (exponential distribution is used to ensure Σ_n is positive definite), it is possible to calculate values of $\vec{y}$ using a predefined $\beta*$ and $\sigma*$, which are known as the ground truth values for comparison:

$$\vec{y} = X\beta * + \sigma * W^{1/2}\epsilon$$

This $\vec{y}$, W, and X are used as inputs to the general Gibbs Sampler Function to generate a distribution of β's and a distribution σ^2's. The posterior means of these distributions are then computed and compared to recover the original values, $\beta*$ and $\sigma*$.

In particular, the posterior mean of σ^2 is calculated by taking the mean of the function's output of σ^2. This posterior mean is compared to the original $\sigma*$ used to generate $\vec{y}$. Repeatedly performing this procedure reveals the posterior mean only differs from the ground truth value by 1-2% in almost every single trial.

Recovering the original $\beta*$ provides a much more defined procedure for evaluating the performance of the Gibbs Sampler. First, the Bayes estimator (the posterior mean of generated βs) should be close to the GLS estimator and theoretical results state the GLS estimator serves as a good approximation for the true value. Furthermore, the variance matrix of the GLS estimator around the true value is

$$Var(\hat{\beta}_{GLS}) = \mathbb{E}[(\hat{\beta}_{GLS} - \beta*)(\hat{\beta}_{GLS} - \beta*)^T] = (X^T W^{-1} X/\sigma^2)^{-1}$$

Thus, after simulating many data sets and solving for the posterior mean estimator, $\hat{\beta}$, the variance of these simulated posterior means, $Var(\hat{\beta})$ should be close to $(X^T W^{-1} X/\sigma^2)^{-1}$. Moreover, the standard errors are calculated

$$SE(\hat{\beta}_{GLS}) = \sqrt{diag((X^T W^{-1} X/\sigma^2)^{-1})}$$

This provides a nominal 95% confidence interval for which to assess the performance of the model for recovering the original $\beta*$.

4.4 Full Sampling Procedure

The complete sampling procedure for imputing missing values uses the generalized Gibbs Sampler defined above to iteratively simulate missing values for the entire matrix Y. The procedure is described below:

Initialize σ_i for $i = 1...m$ and τ_j for $j = 1...n$ as the overall standard deviation of the $Y^{(k)}$ matrix. Initialize missing values of Y using the ANOVA imputation described in the previous section. Initialize the matrix of row factors $U \in \mathbb{R}^{m \times p}$ and the matrix of column factors $V \in \mathbb{R}^{n \times p}$.

Repeat the following:

1. For $i = 1...m$: Estimate u_i and σ_i using the generalized Gibbs Sampler. For the first iteration of the sampler, set X to the V matrix in the singular value decomposition of Y (truncated to be $n \times p$). For all future iterations, the use the stored V from the previous iteration as X. For the first iteration, use the initialized τ_j for $j = 1...n$ to calculate the diagonals for the W matrix. For all future iterations use the stored τ_j values from the previous iteratioWn to calculate W. Take the corresponding y_i directly from the Y matrix. Store the resulting sampled u_i's as rows of a matrix U, and the σ_i in a vector to use for simulating v_j and τ_j.

2. For $j = 1...n$: Estimate v_j and τ_j using the generalized Gibbs Sampler. Use the stored U from the previous step as the X input and use the stored τ_j to calculate the W input. Take the corresponding y_j directly from the Y matrix. Store the resulting sampled v_j's in a matrix V, and the τ_j in a vector, to use for simulating u_i and σ_j.

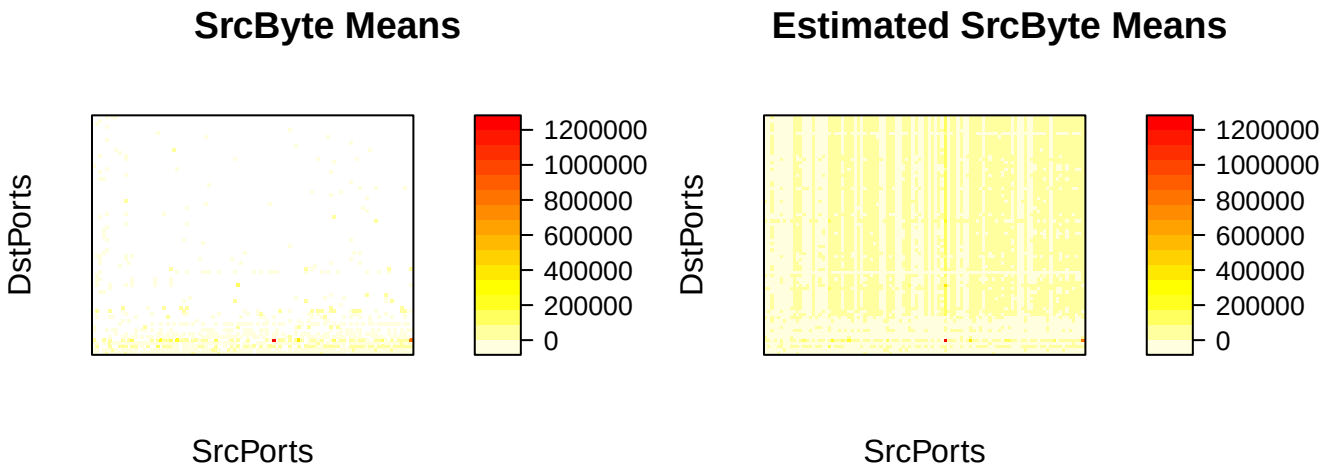
3. Estimate values for y_{ij} in Y that were missing in the original dataset by sampling from the normal distribution

$$y_{ij} \sim N(u_i^T v_j, \frac{\sigma_i^2 \tau_j^2}{s_{ij}})$$

4.4.1 Selecting the Dimension of Latent Factors

The dimension of the latent factors, p, is used to define the dimension of $\beta \in \mathbb{R}^{p \times 1}$ and consequently defines the dimensions of the row and column factor matrices U and V. Selecting p is a model selection choice similar to determining the optimal low rank approximation r in the previous section. Once again, Leave One Out Cross Validation may be used to detemine the optimal p given the observed data. In this technique, it is more computationally expensive than the previous technique to perform Leave One Out Cross Validation on the entire dataset, so K-Fold cross validation or randomly selecting a set number of observed cells to set to missing for determining p is also valid.

4.5 Results on Real Data



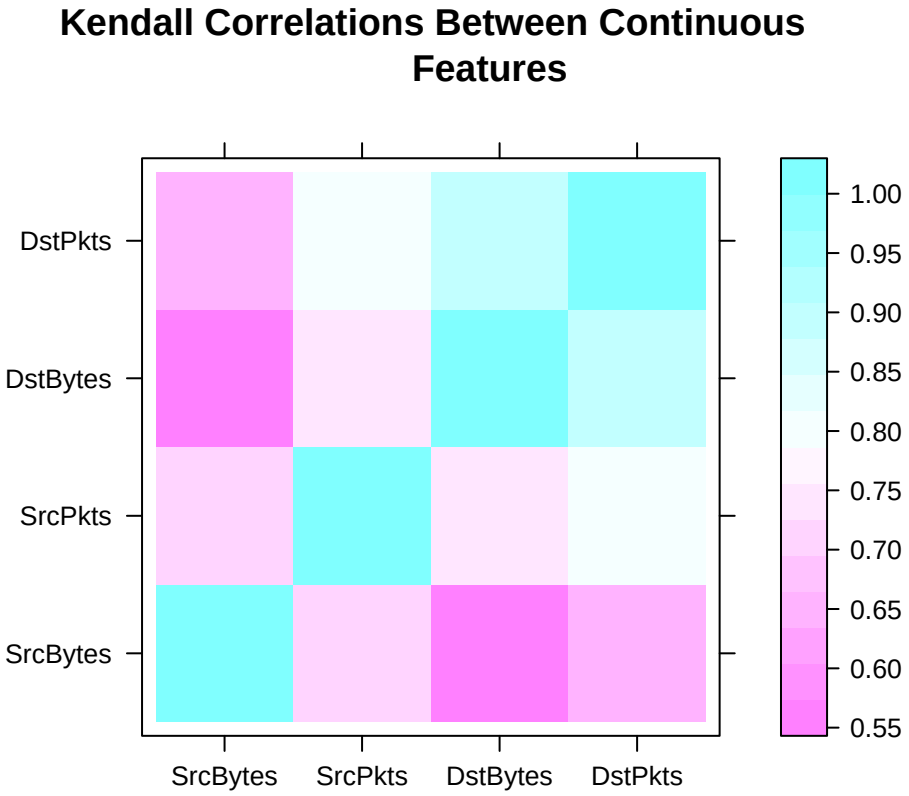
The above plots show the original matrix (left) and the completed matrix (right) using the full estimation procedure. The estimated means all fall within the range of the existing means (approximately 50000-400000). There exist several apparent patterns in the rows and columns of the estimated matrix, but these patterns are most likely the result of the full estimation procedure estimating U (matrix of row factors), V (matrix of column factors), then the missing values of Y. In terms of the networks data domain, it makes sense for certain ports to have more traffic (so the entire row/column is darker in the estimated matrix).

The fit still stands to improve using improved model selection, but in this case achieving the optimal sampling procedure is limited by available computational resources. For instance, using cross validation to select the optimal dimension of latent factors, p, would likely improve the fit of the model. However, running LOOCV using the sampling technique is computationally expensive. Moreover, increasing the overall number of iterations of the full estimation procedure (it is currently at $S = 100$) may also improve the model fit. Work done by Raftery and Lewis (1992) suggests that "reasonable accuracy [with a Gibbs Sampler] may often be achieved with 5,000 iterations or less; this can frequently be reduced to less than 1,000 if the posterior tails are known to be light."

Chapter 5

Tensor Completion

Recall the analysis of correlations between the continuous features in T in chapter 2.



The strong correlations between the individual continuous features suggests imputing the tensor T all at once may yield closer estimates than the previous two techniques, which sliced the matrices. By imputing the tensor as a whole, techniques that include effects that capture the relationships between features can take advantage of possible collinearity between the features.

The following section describes the way techniques in the previous sections may be

extended to full 3-dimensional tensor completion.

5.1 PARAFAC Decomposition

The PARAFAC decomposition expresses the tensor as:

$$T = \sum_{r=1}^{R} u_r \cdot v_r \cdot w_r$$

where r represents the rank approximation, $\cdot$ denotes the outer product of tensors, and $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{n \times r}$, and $W \in \mathbb{R}^{4 \times r}$. Each individual cell is expressed:

$$t_{ijk} = \sum_{r=1}^{R} U_{ri} \cdot V_{ri} \cdot W_{ri}$$

Applying this decomposition yields the objective

$$\min_{T'} \|T - T'\|$$

where

$$T' = \sum_{r=1}^{R} \lambda_r (u_r \cdot v_r \cdot w_r)$$

and λ_r is the regularization penalty.

The PARAFAC Decomposition also operates under several limiting assumptions. PARAFAC assumes the low rank matrix slices of the tensor are multiples of one another, which puts strong constraints on the similarity between the slices. There exists more flexibility with the model in chapter 4 because each matrix can be a different rank.

5.2 Statistical Model

The following model is proposed:

$$t_{ijk} \sim N(\mu_{ijk}, \frac{\sigma_{ijk}^2}{s_{ijk}})$$

where μ_{ijk} is the sample mean, s_{ijk} is the sample size of observations, and σ_{ijk}^2 is the sample variance of observations for source port i, destination port j, and continuous feature k.

Substituting these values into the Gaussian probability density function yields the likelihood:

$$\frac{s_{ijk}}{\sigma_{ijk}^2} \sum (\bar{t}_{ijk} - \mu_{ijk})^2$$

Applying the PARAFAC decomposition, μ_{ijk} is re-expressed:

$$\mu_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$$

Vectorizing the inputs in the likelihood yields:

$$\sum_{j} \sum_{k} [\bar{t}_{ijk} - a_i^T (b_j \cdot c_k)] \frac{s_{ijk}}{\sigma^2_{ijk}}$$

where $a_i \in \mathbb{R}^{m \times r}$, $b_j \in \mathbb{R}^{n \times r}$, and $c_k \in \mathbb{R}^{4 \times r}$. Summing across j and k in this case solves for the ith row slice of the tensor.

5.3 Future Work

Future work will use the PARAFAC decomposition and the Gaussian statistical model described above to extend the techniques described in chapters 3 and 4 to completing the tensor as a whole. The PARAFAC decomposition allows for low rank tensor completion on data with high degrees of missingness. Recent work by Yokota, Zhao, and Cichocki (2016) propose a "Smooth PARAFAC Decomposition for Tensor Completion" that "consider 'smoothness' constraints as well as low-rank approximations, and propose an efficient algorithm for performing tensor completion that is particularly powerful regarding visual data. The proposed method admits significant advantages, owing to the integration of smooth PARAFAC decomposition for incomplete tensors and the efficient selection of models in order to minimize the tensor rank." The statistical model for tensor completion mimics Chapter four's technique more closely. The full conditionals for the parameters being estimated in the model are constructed, and a Gibbs Sampler is created using these full conditionals to iteratively sample the three factor variables and their respective standard deviations described in the model. Like the previous techniques, each of these technique's validity will first be tested on simulated data where the ground truth is known before being applied to the real dataset.

Discussion

This paper has developed two techniques for matrix completion that can then be used to complete individual slices of a tensor. The first technique, an alternating least squares procedure relies upon the Eckart-Young-Mirsky theorem to implement low-rank singular value decomposition. Determining the optimal low rank is a model selection problem that is solved using leave one out cross validation to assess the quality of possible ranks on given datasets. The validity of this technique is tested using simulated datasets where a true low rank is known before it is applied to the actual dataset. Finally, the procedure is applied to the real networks dataset and its performance is evaluated by comparing the fitted values versus the true observed values. Upon noticing certain trends due to outliers in the comparison, a log transformation is conducted on the original dataset and the matrix is once again completed. The second technique addresses matrix completion in the presence of heteroscedasticity. It constructs a statistical model and uses Gibbs Sampling to iteratively sample values from the parameter's full distributions. It acknowledges the weaknesses of the first technique by including both the sample size and the standard deviation of the observations in each matrix cell in the model. This model ends up being very close to a Generalized Least Squares model and so the sampling procedure for the row and the column factors in the matrix is validated by examining the proximity of the variance of the β posterior means in many simulated datasets compared to the variance of the GLS estimator. The paper finally also proposes a solution to extend the two techniques to imputing the tensor iteratively as a whole, rather than individual slices.

For the empirical results throughout this paper $T \in \mathbb{R}^{100\times74\times4}$: the most used 100 source ports and their combinations with the most used 74 destination ports are considered. This was constructed by first taking the most used 100 source and destination ports, creating a $100 \times 100 \times 4$ tensor, and then removing the rows and columns where all values were completely missing because those rows and columns would not provide row means and column means respectively to inform the completion techniques. All of the modeling techniques are informed by the exploratory data analysis conducted on the original dataset prior to completion steps.

Application of these completion techniques to the networks dataset provides a reasonable estimate for the four continuous features at every possible port combination in the matrix. This completed tensor provides a basis for detecting anomalies in future

data. For instance, new observations for a certain port combination that fall outside a threshold of error for the estimated value in the tensor can be marked as as an anomaly and the source and destination IP may be flagged for further investigation. There are two major limitations of these techniques:

1. The sheer size of existing and future network data makes the computational efficiency of the techniques a large concern. The techniques would need to provide a reasonable estimate for every cell in a tensor that may extend up to $65535 \times 65535 \times 4$ (65535 is the number of overall ports). Furthermore, billions of observations may eventually become available, so recalculating initial means and standard deviations for the observations of each cell would need to be done carefully and to be completed in a reasonable amount of time. While Gibbs Sampling and alternating least squares have both been shown to be very efficient and are often employed for handling high dimensional and large scale datasets, the potential size of the networks datasets that may be fed into these techniques enforce the absolute need for computational efficiency.

2. The techniques do not consider the specific domain context with which each port is used. For instance, in TCP protocol, port number 22 is Secure Shell (SSH), which is one of the most popular ports for everyday users. Previous domain knowledge could perhaps inform reasonable estimates for each port combination. Some port combinations may be expected to have large byte and packet transfers, while others may be expected to have few transfers overall. Constructing additional models or features that incorporate domain knowledge regarding the particular port combinations will be useful for improving the model's effectiveness in detecting anomalies.

References

Allen, G. I., Grosenick, L., & Taylor, J. (2014). A generalized least-square matrix decomposition. *Journal of the American Statistical Association*, *109*(505), 145–159.

Alquier, V. C., Pierre; Cottet. (2014). Bayesian matrix completion: Prior specification.

Bailey, S. (2012). Principal component analysis with noisy and/or missing data.

G.H. Golub, G. S., Alan Hoffman. (1987). A generalization of the eckart-young-mirsky matrix approximation theorem. *Linear Algebra and Its Applications*, *88-89*, 317–327.

Hastie, R. L., Trevor; Mazumder. (2014). Matrix completion and low-rank svd via fast alternating least square.

Hoff, P. D. (2009). *A first course in bayesian statistical methods*. Springer-Verlag New York.

Raftery, A. E., & Lewis, S. (1992). How many iterations in the gibbs sampler? *In Bayesian Statistics 4*, 763–773.

Tatsuya Yokota, A. C., Qibin Zhao. (2016). Smooth parafac decomposition for tensor completion. *IEEE Transactions on Signal Processing Issue 20*, *64*(20), 5423–5436.

Toscher, A., Jahrer, M., & Bell, R. M. (2009). The bigchaos solution to the netflix grand prize.

Yang, D., Ma, Z., & Buja, A. (2014). A sparse singular value decomposition method for high-dimensional data. *Journal of Computational and Graphical Statistics*, *23*(4), 923–942. Retrieved from `https://doi.org/10.1080/10618600.2013.858632`

Zhou, M., Wang, C., Chen, M., Paisley, J., Dunson, D., & Carin, L. (2010). Nonparametric bayesian matrix completion. *2010 IEEE Sensor Array and Multichannel Signal Processing Workshop*, 213–216.