ABSTRACT

ELSHEIMER II, DAVID BRUCE DURHAM. Flexible Machine Learning Methods with a Multinomial Choice Model and their Applications to Financial Markets. (Under the direction of Wenbin Lu).

A desirable feature of multinomial choice models is the ability to measure the association between multiple choices. Developments in recent years have seen the proposal of several new econometric models which capture this behavior. In addition, the capability to capture the effects of underlying structures in econometric data is desirable. Tensor regression methods and neural network methods can be utilized to model underlying structures. We introduce two methods for modeling joint multinomial choices when data has complex structures that should be preserved.

We propose an extension to a Multivariate Binomial Logit framework that utilizes Tensor Regression methods to model matrix covariates. The effectiveness of the Multivariate Binomial Logit Tensor Regression model (MVBNL-TR) is assessed with a study of simulated results, as well as two real data econometric applications. We also introduce a method which modifies the approach put forth in MVBNL-TR by using neural network methods instead of tensor regression methods to capture the effects of underlying structure. This method follows an Alternating Neural Network Logistic Regression (ANN-LR) algorithm. We assess the model for several different network structures of varying width and depth using the same econometric data sets used to assess MVBNL-TR. We demonstrate the utility of the MVBNL-TR and ANN-LR methods compared to existing an existing multivariate binomial choice model proposed in the literature, as well as models which do not assume interdependence between multinomial choices.

Flexible Machine Learning Methods with a Multinomial Choice Model and their Applications to Financial Markets

by
David Bruce Durham Elsheimer II

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Statistics

Raleigh, North Carolina
2024

APPROVED BY:

Jung-Ying Tzeng	Xinge Jeng
Tao Pang	Wenbin Lu Chair of Advisory Committee

DEDICATION

To my wife, Nikki

BIOGRAPHY

David Elsheimer grew up in Clayton, North Carolina, raised lovingly by his parents Bruce and Cindy. After graduating high school, he completed courses at Johnston Community College before transferring to North Carolina State University. After years of seeing his father's work at SAS, he was inspired to follow in his father's footsteps as a statistical software tester, and chose to study Statistics. At N.C. State, he graduated summa cum laude in 2018, earning Bachelor of Science degrees in Statistics and Applied Mathematics, and chose to continue his time at N.C. State by pursuing a PhD in Statistics. In 2021, David was granted the opportunity to work as an analytics software tester at JMP Statistical Discovery as a Graduate Research Assistant. In the same semester that he started working for JMP, David met his wife, Nikki Abija, whom he dated throughout his doctoral studies before marrying her in August 2023. David will begin working full time at JMP as an analytics software tester after completing his doctorate in 2024.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Dr. Wenbin Lu. His support and guidance throughout this process have been vital to my success in graduate school. It has been an privilege to learn from him. In addition, I would like to thank my advisory committee members Dr. Jung-Ying Tzeng, Dr. Xinge Jeng, and Dr. Tao Pang for their time and useful feedback.

I would like to thank all of the faculty, staff and students in the Department of Statistics who have been instrumental in my growth over the years, from my time as an undergraduate to my time as a doctoral student. Dr. Justin Post, Dr.Herle McGowan, Dr. Jonathan Duggins, Dr. Eric Chi, and Dr. Jacqueline Hughes-Oliver have all shaped how I think and act as a statistician and their teaching has been an invaluable part of my training.

I'd also like to thank my many classmates who helped make the experience of pursuing a PhD enjoyable. The camaraderie amongst my 2018 cohort consisting of but not limited to Nicholas Larsen, Ethan Davis, Sarah Fairfax, Grace Rhodes, Andrew Hollis, Cole Manschot, Kade Young, Kasia Dobrzycka, and Peter Norwood was vital to my success and forming a good foundation for the rest of grad school. I would also like to thank the students who came before my cohort. In particular, I'd like to thank Rebecca North and Michael McKibben for their friendship and leadership of the N.C. State Statistics Graduate Student Association at the start of my tenure as external Vice President. In addition, I especially would like to thank Jimmy Hickey for being a great friend and a constant source of insight, as well as a good chess rival.

Many thanks are due to my team at JMP. My manager, Nicole Jones has consistently supported me and helped to make work at JMP an exciting and rewarding experience. I would also like to thank James Koepfler and Jianfeng Ding.

I want to also thank Stephanie Hall, one of my first college instructors, who helped to instill a love for mathematics and great study habits early on in my academic career.

I have been incredibly blessed by my parents, whose unconditional love and support has been vital every step of the way. I would like to thank my dad, who has been an inspiration both as a statistician and as a father whose footsteps I aspire to follow. I want to thank my mom for spearheading my education and always pushing me to strive for my best. Both of my parents fostered a great learning environment, instilling the importance of hard work and education in life from a very young age.

Furthermore, I would like to thank my wife, Nikki. I met Nikki halfway through my time in graduate school, and I owe a great debt of gratitude to her for the encouragement and support I have received throughout our relationship. Most importantly, I give thanks to my Lord and Savior Jesus Christ, without whom this accomplishment could not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

Efficiently modeling multinomial choice data has been an area of focus in decision processes research for many years. One of the primary questions of interest is how to jointly model multiple multinomial choices at once. Several real world applications consist of multiple multinomial choices, demonstrating the need for methods which can model multiple choices simultaneously. Additionally, one may want to assume that the outcomes being modeled have a joint relationship. While models for individual multinomial choices have existed for decades, recent developments have enabled the modeling of joint multinomial choices in a computationally efficient way.

Early attempts to model multiple simultaneous outcomes were met with computational roadblocks. For example, Hausman and Wise (1978) propose the Multinomial Probit (MNP) approach. This approach requires solving high-dimension integrals in the parameter estimation step. The authors note that extending this method beyond five choices is not feasible. The MNP approach is infeasible even in the univariate case (Geweke et al. 1994, 1997), so a multivariate MNP model would be computationally inefficient. Other methods, such as the method put forth by Russell and Petersen (2000), rely on unrealistic assumptions about how the outcomes are observed. Some models which would otherwise be more computationally efficient rely on domain-specific assumptions which make it difficult to expand the model's utility to a wider range of applications. One such model is proposed by de Rooij and Kroonenberg (2003).

Bel and Paap (2014) propose a multivariate model for multinomial choices which allows for simultaneous, jointly related choices to be modeled. This model relies on realistic assumptions, and is computationally efficient. We will utilize this model as part of the basis for the models we propose in this thesis.

In recent years, complex, high-dimensional data has become readily available. Many methods have been proposed which make use of complex data without losing meaningful structure in the data. If multinomial choice outcomes have relevant independent variables that have a complex structure, it can be beneficial to consider models which efficiently models multiple multinomial outcomes simultaneously and make use of complex high-dimensional data. The desire to maintain complex data structures motivates a review of tensor regression methods, which allow for more complex and structured data to be used.

Earlier work by Zhou et al. (2013) has demonstrated the benefits of tensor regression methods over other existing methods for certain applications. Chang et al. (2021) proposes a tensor-based model that in conjunction with Bel and Paap (2014), forms the basis of the model we propose in Chapter 2. The tensor regression methods discussed in the literature review tend to focus on medical imaging (Zhou et al. 2013) or multi-omics data in the case of (Chang et al. 2021). However, utilizing these methods for other data applications is possible. In the scope of this thesis, we consider a stock market data application, where we consider the day-to-day changes in stock market price for several stocks as multiple associated multinomial choices. In the examples considered, we are only concerned with if a stock price increases, or decreases or remains static. With two discrete levels, this becomes a multivariate binomial problem. The stock market data is downloaded directly from Yahoo Finance utilizing the `get.hist.quote` function from the `tseries` software package in R.

There have been developments in choice modeling research in recent years which involve the implementation of neural networks in attempts to improve the predictive power of choice models. For instance, Sifringer et al. (2020) propose a multinomial logit model which implements a convolutional neural network in an intermediate step in an effort to improve the predictability of the model while maintaining interpretability. However, this model, along with others such as the model put forth by Arkoudi et al. (2023) are concerned with travel data, and are also not concerned with jointly modeling several different decisions simultaneously. This motivates further work to determine if the implementation of a neural network step can improve the predictive power of multivariate binomial logit methods.

In Chapter 2, we first conduct a review of the relevant work in the area of multivariate multinomial choice modeling. Additionally, we review models that handle ultrahigh-dimension data. In particular, we consider the improvements made by tensor regression models. Next, we propose a multivariate binomial logit tensor regression model which builds on the multivariate

multinomial choice model put forth in Bel and Paap (2014) and the tensor regression model discussed in Chang et al. (2021). This model will make use of the alternating parameter update idea in the updating of its matrix parameters. Departing from the least squares approach put forward in Chang et al. (2021), this approach will follow an alternating logistic regression framework to update the parameter estimates. We then conduct a simulation study comparing four models on various metrics. The models compared are the multivariate binomial logit model proposed by Bel and Paap (2014), the tensor regression extension (MVBNL-TR) to this model which we propose , and two simpler marginal models that assume independence between choices. These models will be compared to determine if the tensor regression extension benefits from the inclusion of more complex structures, and if it models the joint interdependence of the simulated choices well. The usefulness of the tensor regression model on two real stock data applications will also be discussed. In addition to this model, we propose an extension which modifies the framework. Instead of capturing underlying structure using tensor regression methods to estimate matrix parameters, we utilize convolutional neural networks to represent underlying structure.

This second method is the focus of Chapter 3. We discuss how these modifications will change the model algorithm and the computation of metrics for model performance which are introduced in Chapter 2. After this discussion, this new model's performance will be assessed for several different neural network structures. Both stock data examples considered in Chapter 2 will be used again for this analysis. In addition to assessing model performance on various neural network structures, we compare this neural network model's performance to the performance of the tensor regression model put forth in Chapter 2.

CHAPTER

2

ADVANTAGES AND LIMITATIONS OF MULTINOMIAL CHOICE MODELS AND TENSOR REGRESSION MODELS

2.1 Introduction

While decision processes have long been used in many areas of applied research, a common problem that many have worked to address involves the usage of multivariate multinomial choice data. It is common to utilize multinomial logit models (MNL) to describe data that consists of multinomial choices (McFadden 1983, Chapter 24). While suitable models for modeling single multinomial choices have existed for decades, modeling multiple simultaneous multinomial decisions is a more difficult task. This difficulty compounds due to the fact that such decisions can be correlated. (Bel and Paap 2014)

Developing the ability to model multiple multinomial decisions simultaneously has become a promising area of study in recent years. Difficulties exist that have made these models prohibitive to use for a multivariate case. As laid out by Bel and Paap (2014), the primary issues stem from the fact that computation of multiple simultaneous choices is generally infeasible in practice. However, models have been proposed that help to assess this issue.

High-dimensional data is becoming more and more prevalent and readily available for analysis, with burgeoning areas of research having arisen in recent years. Classical methods often underperform, due to the computational difficulty of carrying out traditional regression methods on ultra-high-dimensional data with complex structure. As discussed in Zhou et al. (2013), the improvements in data collection have improved the ability to capture the complex form of data in certain applications. In their study, this primarily concerned neuroimaging data. Traditional methods which treat covariates as vectors would lose the structure present in medical imaging data, which often stores data into multidimensional arrays that better describe the complex form of these data.

The need for methods which better capture this structure has led to the development of tensor regression, a new family of models that utilize the structure that is present in tensor covariates. Using this framework, the severity of the curse of dimensionality is lessened, leading to estimation and prediction that can be performed in an efficient manner (Zhou et al. 2013). This framework has frequently been used for medical imaging and multi-omics data (Zhou et al. 2013; Chang et al. 2021), but the tensor regression framework it provides can be used in other applications as well. An intriguing area of study to consider is utilizing advancements in both multinomial choice modeling and tensor regression methods to construct a model which is capable of modeling simultaneous multinomial choices utilizing more structured data.

Section 2.2 is comprised of a detailed overview of multinomial choice models, covering the early work in the field, as well as the current state of research and key developments that have furthered study. Furthermore, the methodology for such models is discussed, as well as some key results for select models. Section 2.3 is organized similarly, focusing instead on tensor regression methods. Sections 2.4 and 2.5 will propose an extension to multivariate multinomial choice modeling using a tensor regression approach and discuss the metrics by which to assess model performance. Section 2.6 will focus on the results of simulation studies for this proposed extension, whereas section 2.7 will focus on the results from multiple applications to real data. Lastly, section 2.8 will discuss conclusions and areas for future study.

2.2 Multinomial Choice Models

2.2.1 Early work

Multivariate logit (MVL) models have been present in literature for decades, with a binary MVL model being put forth by David Cox in 1972. As noted previously, modeling simultaneous multinomial decisions is more difficult. One proposed method is the correlated Multinomial Probit (MNP) approach, laid out in Hausman and Wise (1978). Prior to this paper, the widely

used method at the time for modeling multiple alternative choices is an extension of binary logit analysis, known as conditional logit analysis. Although computationally feasible, this method assumes the independence of irrelevant alternatives (IIA) property. (Hausman and Wise 1978, p. 404). This property is presented in detail by Hausman and Wise, but a summary is provided. Assume that there exists a set of three alternative choices. For an individual, U_i exists such that for the ith choice, $U_i = \bar{U}_i + \varepsilon_i$. Here $\bar{U}_i$ is the average value assigned to the alternative choice. The average is taken over all individuals who make decisions. ε_i is a random term representing how an individual's valuation of a choice deviates from the average. Then for the three choices, an individual would have $U_1 = \bar{U}_1 + \varepsilon_1$, $U_2 = \bar{U}_2 + \varepsilon_2$, and $U_3 = \bar{U}_3 + \varepsilon_3$. It is reasonable to suppose that one chooses the alternative that they value the most. If each alternative can be characterized by attributes Y and Z, then $U_1 = U(Y_1, Z_1) = \bar{U}(Y_1, Z_1) + \varepsilon_1$. U_2 and U_3 can be characterized similarly.

Consider a graph with axes Y and Z, with the points for each choice plotted on it. If two choices are close together in this space, then it is possible that an individual who values the first choice greater than the average value would value the second choice similarly. If this is truly the case, then the ε_i terms should be correlated. However, conditional logit models at the time of this paper were based on the assumption that these terms are independent. This assumption is unrealistic for many applications. Instead of a conditional logit framework, a model was proposed which does not assume this independence restriction. For a small number of alternative choices, the parameter estimation for this model is feasible. The parameter estimation for such models involves solving high-dimensional integrals. Hausman and Wise only discuss an example considering three choices, but note than extension to five choices is feasible. However, an extension to a larger number of choices is not feasible. Even in a univariate case, increasing the number of choices will render a heavy computational cost which makes the use of such a model infeasible (Geweke et al. 1994, 1997). As such, extending to a multivariate MNP model under this framework is ill-advised.

Other options exist, such as mixed logit models (Hensher and Greene 2003). As with multinomial probit models (Hausman and Wise 1978), these models do not rely on the often questionable assumption that choices are independent. However, this model still faces the a similar problem to MNP models, as it requires the computation of unconditional choice probabilities. A moderately large number of simultaneous decisions results in the required solving of integrals at a high computational cost. More feasible options exist, such as the Nested Multinomial Logit (NMNL) specification (Maddala 1986, Chapter 3). This model is tractable for large problems. The computation for this model involves sequential multinomial logit steps. Due to this, the model specification considers data to be decisions made sequentially. In practice, this is not necessarily the case. This limits the problems to which a NMNL method can be applied.

One could also use a MNL model for all potential combinations of the multinomial variables. However, the number of combinations quickly becomes large (see Amemiya (1978) and chapter 10 of Ben-Akiva and Lerman (1985)). With a large number of multinomial decisions, the number of parameters in the model increases greatly, and interpretation quickly becomes difficult. Estimation of parameters becomes infeasible, as the computation of choice probabilities also requires summing over all the possible outcomes.

Some of the more relevant models in recent years are the models put forth by Burda et al. (2008) and de Rooij and Kroonenberg (2003). The model put forth in Burda et al. (2008) is a Bayesian mixed logit-probit model for multinomial choice. This model has some key advantages, one being that the model avoids assuming the IIA. This is possible by using choice specific coefficients which are drawn from a multivariate distribution. (Burda et al. 2008, p. 1). The addition of choice-specific random variables to remove the need for the IIA assumption is the foundation of the "probit" part of the model, differentiating it from other logit models (Burda et al. 2008, p. 12). However, one drawback of this model is the individuals are making multiple choices on the same attribute. In a problem setting where one wanted to consider simultaneous multinomial choices for separate attributes, this would not be applicable.

The multivariate multinomial model posited by de Rooij and Kroonenberg (2003) is an insightful contribution to the work that has been done on simultaneous multinomial choice modelling. This model is designed specifically for the problem and data set discussed in the paper. One of the drawbacks of this is that the model cannot be applied to a general simultaneous multinomial case. Regardless, a review of its accomplishments is still in order.

2.2.2 Multinomial Choice Methods

The purpose of the model put forth by de Rooij and Kroonenberg is the analysis of discrete dyadic sequential behavior, with a focus on forecasting future behavior. Here, the dyadic group being considered is a therapist and a child engaged in Imagery play therapy (de Rooij and Kroonenberg 2003, p.463). The purpose of the study is to discover how a therapist engages with a child in this type of therapy and how new behavior can be forecasted using old behavior. Leading up to this study, the trend in the field was to utilize a log-linear method for analyzing these data. The use of these methods presented researchers with some difficult problems to solve. One of the drawbacks of using log-linear methods is it reduces the interpretability of the model. In application, the parameters of a log-linear model are not estimated, only the expected frequencies. Furthermore, interpretation of results is performed by analyzing standardized residuals from models rather than by interpreting parameters. As there tend to be a large number of parameters which need to be estimated, an alternative approach is generally

used. The important association effects are determined by using the expected frequencies. This results in less information that can be provided about how parameters are associated. (de Rooij and Kroonenberg 2003, p.464)

Two alternatives to the log-linear model help with handling interpretability problems. One method is weighted least squares, but the second approach is of more interest. Via a reasonably straightforward process, a log-linear model can be re-parametrized in such a way that it can be interpreted in terms of odds and odds ratios. This reparametrization forms a MNL model (de Rooij and Kroonenberg 2003, p.464). As one of the main drawbacks of this model is its data-specific application, it is good to know about the data and subjects that are considered in the scope of the analysis.

In Imagery play therapy, a child and therapist engage in a session, with their behaviors being observed and recorded at recurring intervals. The behaviors are coded to field-specific categories, but re-coded by de Rooij and Kroonenberg to four categories: Non play (N), Play Preparation (P), Functional play (F), and Imagery play (I). The key difference between functional and imagery play is that imagery play primarily consists of using imagination to create a play situation whereas functional play is related to play associated with movement of the toys; it focuses on the sensory qualities of play. (de Rooij and Kroonenberg 2003, p.468-469) In the study, 117 "dyads" (therapist-child pairs) were studied. 48 of the children were in therapy, with 69 not being in therapy. Some key assumptions were made about the data. Homogeneity across all therapist-child pairs was assumed, and the behavior at a given time t_0 is assumed to be able to be predicted from behavior at prior time points. The stability of predictability over the whole time sequence is assumed. Due to the fact that there are only four therapists considered in this study, the data are non-independent despite the fact that the therapists were equally distributed across all of the children in the study. In general, non-independence exists for this type of data. (de Rooij and Kroonenberg 2003, p.469).

One of the key differences in this approach is that the multinomial logit models discussed in this work are based on the log-linear models, instead of an extension of the binary logit model. The log-linear model is re-specified as a multivariate multinomial response model. This MV-MNL model models the log of the probability of the behavior of the dyad members at time t_0 conditional on their behavior at time t_{-1} : $\log \pi_{a_k^0 b_l^0 | a_i^{-1} b_j^{-1}}$ for behaviors a_k^0, b_l^0. (de Rooij and Kroonenberg 2003, p.471-472) The difference of two log-probabilities is able to be understood as the log-odds given the behavior at a previous time-point. Modeling the difference in these two probabilities forms the basis of the multivariate multinomial logit model form. Modeling such a response can be done by using a certain behavior category of both members of the dyad as a baseline to contrast against (a_K^0, b_L^0) (de Rooij and Kroonenberg 2003, p.472). Then the MV-MNL logit model can be written such that the response being modeled is $\log\left(\frac{\pi_{a_k^0 b_l^0 | a_i^{-1} b_j^{-1}}}{\pi_{a_K^0 b_L^0 | a_i^{-1} b_j^{-1}}}\right)$. The

primary difference between this model and the MV-MNL response model is that parameters are now interpreted in terms of the log odds of a certain category against the baseline. (de Rooij and Kroonenberg 2003, p.473)

One of the primary benefits of a MV-MNL logit model over the standard log-linear model is how it is more reasonable to interpret the parameters of such a model in contrast to having to examine and interpret the residuals of a log-linear model. The interpretation of odds is fairly straightforward, although there were still some drawbacks to this model. When fitting these models, one can encounter *structural zeroes*, which are observations which are structurally missing (de Rooij and Kroonenberg 2003, p.465). To obtain valid results, specifying a design matrix that avoids these structural issues is necessary (de Rooij and Kroonenberg 2003, p.498). This model serves as a good step forward for simultaneous multinomial choice modeling. However, due to its highly problem-specific nature, it is not useful in a broader context where dyadic data are not present.

Another model with an applied focus that should be discussed is put forth by Russell and Petersen. Motivated by the increase in data pertaining to shopping behaviors of consumers, the research put forth in this paper attempts to model how a consumer will select items from categories of products on a shopping trip. This response can be concisely referred to as *market basket choice*. This model permits a model of basket choice to be constructed using a set of conditional choice models which correspond to the basket items (Russell and Petersen 2000, p.367). Previous methods for market basket analysis operate under the assumption that sales across product categories are correlated in some way. Correlation patterns are used to make recommendations for how to market the products. However, this is a fairly simplistic way at looking at what is a complex system. (Russell and Petersen 2000, p.368)

A more insightful method of analysis is to identify where sources of cross-category dependence originate. When consumers enter a store, they see many products that can be purchased from various categories. The primary interest with these types of models is determining whether decisions in a basket framework are related. The linkages across categories which can be modeled will be of use for those wishing to market the product more effectively. There are two primary ways of explaining these linkages as laid out in the literature: store choice and global utility. Store choice models suggest that sales across categories are related due to the fact that the store consumer base will change across weeks, and that this changing mixture over time will create store-level cross-category correlation in the data. Global utility models instead propose that the dependence across categories is present within the process by which each consumer makes decisions. The model put forth by Russell and Petersen is a global utility model which attempts to model the basket of purchases by constructing linked choice models for the individual categories. This method has two key advantages. First, the model obtained is

parsimonious. Secondly, because the approach has the constituent parts being logistic in form, the market basket choice probabilities will follow a multivariate logistic distribution. (Russell and Petersen 2000, p.368-369)

To build this model, conditional choice specification is performed. This approach specifies *local* models (a choice model for each basket item) so that a global model (the choice of an entire basket) can be specified (Russell and Petersen 2000, p.369). To see how this works, an example should be considered. For $N = 4$ product categories A, B, C, and D, each basket consists of four random variables which correspond to the decisions to buy or not buy a product. As such, there are $2^4 = 16$ possible baskets. One of the contemporary methods for expressing the basket choice process is to directly specify the distribution $p(A, B, C, D)$, but this requires prior knowledge of how features interact and also it requires that all possible baskets be enumerated. Another possible method is to model the choices sequentially. As such, if choice D is considered first, C second, and so on, the following holds:

$$p(A, B, C, D) = p(A|B, C, D) \cdot p(B|C, D) \cdot p(C|D) \cdot p(D) \tag{2.1}$$

If sequentially modeling is a valid approach, the joint probability can be expressed as a product of sequential conditional probabilities. However, it is not realistic to assume that the order in which choices are made can always be observed. (Russell and Petersen 2000, p.370) The drawbacks of these methods serve as motivations for the conditional choice specification approach, which avoids the problems of directly specifying the joint distribution, as well as the issues of assuming that the particular order in which items are chosen is always available (Russell and Petersen 2000, p.370-371). By using the set of full conditional distributions $(p(A|B, C, D), p(B|A, C, D), p(C|A, B, D), p(D|A, B, C,))$, the properties of the joint distribution $p(A, B, C, D)$ can be inferred. This relies on the results of Besag (1974), which demonstrate the existence of a correspondence between the set of full conditional distributions and the joint distribution, as long as the full conditional distributions are consistent. This approach is very practical, as one can specify a choice process for one decision without having to specify the joint distribution simultaneously. (Russell and Petersen 2000, p.371).

A market basket of choices is defined as follows:

$$B(k, t) = \{C(1, k, t), \ldots, C(N, k, t)\} \tag{2.2}$$

$C(i, k, t)$ is a choice (0 or 1) denoting if the kth consumer buys a product in category i at the tth time point. The general form of the conditional choice models can be written as follows:

$$U(i,k,t) = \beta_i + HH_{ikt} + MIX_{ikt} + \sum_{j \neq i} \theta_{ijk} C(j,k,t) + \varepsilon(i,k,t) \tag{2.3}$$

In this framework, HH_{ikt} and MIX_{ikt} are specific to the basket analysis framework, and $\varepsilon(i,k,t)$ is a random error with mean 0. $\sum_{j \neq i} \theta_{ijk} C(j,k,t)$ is a term related to the association between the choice for category i and the real choice decisions present in all other categories. The term θ_{ijk} is an association parameter between product categories, where a positive value relates to positive association and a negative value relates to negative association. The probability of buying from category i conditional on other outcomes is assumed to be equivalent to $p(U(i,k,t) > 0)$. (Russell and Petersen 2000, p.372)

The random error is assumed to follow an extreme value distribution, and as such the conditional probability can be modeled in the following way (Russell and Petersen 2000, p.372-373):

$$Pr(C(i,k,t) = 1 | C(j,k,t) \text{ for } j \neq i) = (1 + \exp(-Z(i,k,t)))^{-1} \tag{2.4}$$

$$Z(i,k,t) = \beta_i + HH_{ikt} + MIX_{ikt} + \sum_{j \neq i} \theta_{ijk} C(j,k,t) \tag{2.5}$$

The form of the joint distribution can be described in the following way. First consider the variable $X(i,b)$ which has the value one if category i is in basket b, and zero otherwise. Then the probability of selecting a basket b is

$$Pr(B(k,t) = b) = \frac{\exp(\mu(b,k,t))}{\sum_{b^*} \exp(\mu(b^*,k,t))} \tag{2.6}$$

$$\mu(b,k,t) = \sum_i \beta_i + \sum_i HH_{ikt} + \sum_i MIX_{ikt} + \sum_{i<j} \theta_{ijk} X(i,b) X(j,b) \tag{2.7}$$

This model is capable of predicting the probability of selecting any of the 2^N possible baskets for a given day, but the only set of parameters required is those that are used in defining the conditional logit model. (Russell and Petersen 2000, p.373)

This model can be interpreted as a multivariate distribution for the vector of binary responses $\{C(1,k,t), \ldots, C(N,k,t)\}$. As such the joint probability can be given by equation 2.7 and the conditional probabilities are as outlined in equations 2.4-2.5. (Russell and Petersen 2000, p.374-375) While the parameters applicable to the model might vary based on application, this framework is very useful, and expanded upon by Bel and Paap (2014).

The first specification discussed by Bel and Paap is very similar to the ideas set forth by Russell and Petersen. There are a couple of differences in notation that simplify and generalize the model to more applications. Here, a joint set of binary choices for individual $i = 1, \ldots, N$ is

$Y_i = \{Y_{i1}, \ldots, Y_{ik}\}$. The set of all possible realizations for an individual is the set S. The choices can be correlated here, but the association parameter here is denoted by ψ. In comparison with equations 2.4 and 2.5, here

$$P(Y_{ik} = 1 | y_{il} \text{ for } l \neq k, X_i) = \frac{\exp(Z_{ik})}{1 + \exp(Z_{ik})} \tag{2.8}$$

where

$$Z_{ik} = \alpha_k + X_i \beta_k + \sum_{l \neq k} y_{il} \psi_{kl} \tag{2.9}$$

(Bel and Paap 2014, p. 4)

Here X_i denotes a vector of explanatory variables with the corresponding parameter vector denoted by β_k. α_k are intercepts specific to the possible choices, and ψ_{kl} are the association parameters, which capture the correlation between the kth and lth choices. As outlined by Besag (1974), the multinomial logit model for the joint distribution of Y_i is as follows:

$$Pr(Y_i = y_i | X_i) = \frac{\exp(\mu_{y_i})}{\sum_{s_i \in S} \exp(\mu_{s_i})} \tag{2.10}$$

where

$$\mu_{y_i} = \sum_{k=1}^{K} y_{ik}(\alpha_k + X_i \beta_k) + \sum_{l>k} y_{ik} y_{il} \psi_{kl} \tag{2.11}$$

(Bel and Paap 2014, p. 5)

Bel and Paap propose an extension from a binomial case to simultaneous multinomial decisions. For choice k, there are J_k possible outcomes. Now, individual i has Y_i defined such that $Y_{ik} = j$ which corresponds to individual i choosing j for choice k. Instead of 2^K possible outcomes, the number of outcomes is $\prod_{k=1}^{K} J_k$. Now the formula for the conditional probabilities is

$$Pr(Y_{ik} = j | y_{il} \text{ for } l \neq k, X_i) = \frac{\exp(Z_{ik,j})}{\sum_{l=1}^{J_k} \exp(Z_{ik,l})} \tag{2.12}$$

for

$$Z_{ik,j} = \alpha_{k,j} + X_i \beta_{k,j} + \sum_{l \neq k} \psi_{kl,j y_{il}} \tag{2.13}$$

(Bel and Paap 2014, p. 5-6) $\alpha_{k,j}$ are intercepts for the choice and jth outcome for a choice. X_i is again a explanatory variable vector with parameter vector $\beta_{k,j}$. The association parameter here $\psi_{kl,j y_{il}}$ is for the association between choosing outcome j for choice k and choosing outcome

y_{il} for choice l. Not all choices are identifiable here, resulting in identifiability issues. As such, some restrictions are placed on this model. $\alpha_{k,1}, \beta_{k,1} = 0$ for all k. Furthermore, symmetricity for the association parameters is assumed, $\psi_{kl,jh} = \psi_{lk,hj}$. Also, $\psi_{kl,j1} = \psi_{lk,1h} = 0$. With these restrictions imposed, a fairly straightforward extension of the binomial case is possible for all values of K and J_k. Besag's Theorem can again be applied, leading to a similar result as in (10) for

$$\mu_{y_i} = \sum_{K=1} \alpha_{k,y_{ik}} + X_i \beta_{k,y_{ik}} + \sum_{l>k} \psi_{kl,y_{ik}y_{il}} \tag{2.14}$$

(Bel and Paap 2014, p. 6)

A likelihood ratio test (LRT) is discussed for testing for independence in the multinomial decisions. The primary difference from a standard LRT is that as proposed in a prior work (Bel et al. 2014), a composite likelihood approach is used. This avoids the difficult summation over the total outcome space, and still provides consistent estimators at the expense of efficiency. (Varin et al. 2011)

In application, multinomial choice outcomes can be seen in a wide variety of disciplines, including social studies and economics (Russell and Petersen 2000; de Rooij and Kroonenberg 2003; Bel and Paap 2014). There are several key goals to keep in mind when modeling these outcomes. One of the primary goals is to forecast future choices by effectively utilizing knowledge about current and past choices, as described by de Rooij and Kroonenberg (2003). In the long and storied history of this area of research, many great strides have been made to develop models which are effective in forecasting future choices, when the number of choices made is particularly large. In traditional methods, computation quickly becomes difficult due to the exponential increase in outcome space that coincides with both an increase in the number of choices and the possible outcomes per choice. This is one of the motivations behind the multivariate multinomial model put forth by Bel and Paap (2014), a vital work in understanding the development of this class of models.

Another important goal in the development of more advanced models for multinomial choices is that these choices are often likely to be correlated. At the time of authorship, Bel and Paap (2014) note that there are not a great deal of models designed to model this behavior. Capturing such behavior and being able to forecast future choices while also avoiding the computational burden that comes with modeling many choices has led to crucial developments in the field. One of the most intriguing models for simultaneous multinomial choice is that proposed by Bel and Paap (2014). Whereas prior contributions such as de Rooij and Kroonenberg (2003) have dealt with this area of interest, the model proposed there is designed with a particular applied problem in mind. In contrast Bel and Paap (2014) propose a Mul-

tivariate Multinomial Logit (MV-MNL) model which applies to a general case. Some models have avoided using the conditional logit specification despite its computational benefits due to some issues with model assumptions, such as the independence of irrelevant alternatives. However, as put forth by Russell and Petersen (2000) and Bel and Paap (2014), associations between choices can be modeled in a way which enables feasible computation without a required assumption that alternative choices are independent. Here a test for independence is proposed alongside the model. The framework put forth allows for feasible computation and parameter inference. Maximum (composite) likelihood estimators are consistent, and as demonstrated in Bel and Paap (2014), the loss in efficiency appears to be small with bias comparable to that of a standard MLE. Because of the general form put forth by Bel and Paap, this MV-MNL specification is widely applicable to problems which involve simultaneous multinomial choices. Furthermore, it avoids the shortcomings of models such as the one proposed by de Rooij and Kroonenberg (2003).

2.3 Tensor Regression Models

New developments in data collection capabilities have motivated the advent of high-dimensional methods that are capable of utilizing great amounts of data. One of the motivating problems with methods designed for high-dimensional data is that oftentimes these data can have inherent structure that more traditional methods do not utilize effectively. Such drawbacks in traditional methodology have motivated various methods which take advantage of data structures. One such method is tensor regression. The rise of this method has resulted in algorithms for estimation which are fast and scale well to large problems. (Zhou et al. 2013, p. 1)

2.3.1 Early work

To understand the motivation behind tensor regression, it is important to understand how the improvements of new data collection capabilities have shaped the demand for better modeling frameworks. One such application closely related to improvements in tensor regression is brain imaging. Brain imaging can be performed using various methods, such as magnetic resonance imaging (MRI). These methods have presented researchers with an interesting problem. With such a wealth of imaging data, which are inherently very large and complex, the standard statistical methods have proven to be insufficient for analysis. (Zhou et al. 2013, p. 1-2)

One of the primary research goals is to be able to establish some associative links between the brain image data and clinical traits of interest. In the relevant literature, there are approximately three different categories of statistical methods for establishing these relationships.

The first class of methods is referred to as *voxel*-based methods. (Zhou et al. 2013, p. 2) A voxel is volume element of the brain, representing a three-dimensional basic unit of data which is comprised of millions of neurons (Lazar 2008, p. 261). In these voxel-based methods, each voxel is a response, and certain clinical variables (e.g. demographic variables such as age and gender) are utilized as predictors (Zhou et al. 2013, p. 2). Then, a statistical parametric map of test statistics or p-values is generated across all the voxels (Zhou et al. 2013; Lazar 2008). For a subject (one brain) a map will be generated (Lazar 2008, p. 69). These maps will undergo additional processing to determine which voxels are active. Such a process is application specific, as the criteria one must consider can change. It is important to consider the assumptions made for this sort of model. Onset of a response is assumed to be immediate upon the presentation of a stimulus, and it is expected to cease as soon as the stimulus is stopped. Furthermore, variances in both the control and task conditions are assumed to be equal. While these assumptions may not be realistic, researchers note that the tests being considered can accommodate these assumptions. Some more problematic assumptions are made. Voxels are assumed to be independent from one another, and when analysis aggregates over time points, it is assumed that the measurements collected at different times are independent. Both of these independence assumptions are unrealistic, and in practice will be violated in all data sets. (Lazar 2008, p. 69) It is important to characterize this spatial dependence in some way. However, previous work to characterize this behavior has posed computational problems (Li et al. 2011, p .2). Some models attempt to answer this by spatially smoothing the data in such a way as to allow smoothing across both space (voxel location) and time. (Yue et al. 2010, p .2)

The second class of methods follows a functional data analysis framework. There has been some progress in this area, with an extension to two-dimensional image predictors having been proposed by Reiss and Ogden (2010). However, generalizing this method to higher dimensions is not a trivial undertaking, and requires more research. The third category performs dimension reduction, either by principal components analysis (PCA) or some other method. Next, a regression model is fit that is based on the top principal components (Caffo et al. 2010). While implementation of a two-stage method is straight-forward, this method is an unsupervised technique. As such, the principal components in the model might not be relevant. Due to the drawbacks of each of these methods, there is still much room for improvement. (Zhou et al. 2013, p. 2)

In contrast to the previous methods, the regression framework proposed in Zhou et al. (2013) treats a clinical outcome as the response, and uses the image data in the form of a tensor as covariates. Classical methods would vectorize such a tensor. This would result in a standard MRI image of size 256^3 requiring a model with over sixteen million regression parameters. Such an ultra-high dimension problem puts both computational and theoretical

guarantees for classical regression models into jeopardy. Such a step would also eliminate any useful spatial structure that the image might possess. (Zhou et al. 2013, p. 2) As such, tensor regression methods are a viable avenue for handling these data, as this helps to avoid violation of statistical guarantees and preserves structure useful for analysis.

2.3.2 Tensor Regression Methods

The notation used here is laid out clearly in section 2.1 of Zhou et al. (2013). In this setting, Y is a univariate response variable, $\boldsymbol{Z} \in \mathbb{R}^{p_0}$ is a covariate vector, and $\boldsymbol{X} \in \mathbb{R}^{p_1 \times \cdots \times p_D}$ is a D-dimensional tensor predictor. Consider the classic GLM framework, where Y is in an exponential family with the following probability mass function or density:

$$p(y|\theta,\phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right) \tag{2.15}$$

Here θ and ϕ denote dispersion parameters. This GLM relates $\boldsymbol{X} \in \mathbb{R}^p$ to $\mu = E(Y|\boldsymbol{X})$ using a link function $g(\mu) = \eta - \alpha + \boldsymbol{\beta}^\top \boldsymbol{X}$. For $D = 2$, $\boldsymbol{X} \in \mathbb{R}^{p_1 \times p_2}$ and the GLM considered would have the intuitive link function $g(\mu) = \alpha + \beta_1^\top \boldsymbol{X} \beta_2$. Note that due to how the Kronecker product operation works, $\beta_1^\top \boldsymbol{X} \beta_2 = (\beta_2 \otimes \beta_1)^\top \text{vec}(\boldsymbol{X})$ Expanding this out to a generalized dimension, Zhou et al. (2013) propose a GLM with the following link function:

$$g(\mu) = \alpha_\gamma^\top \boldsymbol{Z} + (\beta_D \otimes \cdots \otimes \beta_1)^\top \text{vec}(\boldsymbol{X}) \tag{2.16}$$

Here $\gamma \in \mathbb{R}^{p_0}$ and $\beta_d \in \mathbb{R}^{p_d}$. This model has the important advantage of reducing the tensor component dimensionality from the order of $\prod_d p_d$ to $\sum_d p_d$. For the MRI imaging example, the dimension is reduced from 256^3 to 768. This results in a substantial reduction in computation cost. However, this reduction in the parameter space does limit the capacity of the model to some extent, which motivates a more flexible tensor regression model. (Zhou et al. 2013, p. 3-5) This flexible model will have a coefficient tensor $\boldsymbol{B}$ which matches the dimension of $\boldsymbol{X}$. Here $g(\mu) = \alpha + \gamma^\top \boldsymbol{Z} + \langle \boldsymbol{B}, \boldsymbol{X} \rangle$. $\langle \boldsymbol{B}, \boldsymbol{X} \rangle = \text{vec}(\boldsymbol{B})^\top \text{vec}(\boldsymbol{X})$. However, in this model, $\boldsymbol{B}$ still has $\prod_{d=1}^{D} p_d$ parameters. Approximating $\boldsymbol{B}$ using a low rank approximation is an option. The proposed rank-R generalized linear tensor regression model has

$$g(\mu) = \alpha + \gamma^\top \boldsymbol{Z} \langle (\boldsymbol{B}_D \odot \cdots \odot \boldsymbol{B}_1)\mathbf{1}_R, \text{vec}(\boldsymbol{X}) \rangle \tag{2.17}$$

Here $\boldsymbol{B}_d = \left[\beta_d^{(1)}, \ldots, \beta_d^{(R)}\right] \in \mathbb{R}^{p_d \times R}$, and $\boldsymbol{B}_d \odot \cdots \odot \boldsymbol{B}_1 \in \mathbb{R}^{\prod_d p_d \times R}$ is a Khatri-Rao product. For this model, the number of parameters reduces to $p_0 + R\sum_d p_d$, in contrast to $p_0 + \prod_d p_d$. Even with such a large dimensionality reduction, reasonable approximation to low rank signals still

exists. To estimate the parameters of the model, maximum likelihood estimation is used. The algorithm used for maximizing the likelihood is a block relaxation algorithm (Lange 2010) which alternates between updating $(\alpha, \boldsymbol{\gamma})$ and B_d. For each iteration, updating block B_d is a classical GLM problem. For Gaussian models, the algorithm reduces to the alternating least squares method (de Leeuw et al. 1976).

Two of the key results from this algorithm worth noting are its convergence properties. The estimates generated by the algorithm will globally converge to a stationary point. Furthermore, the iterative parameter values generated by the algorithm are locally attracted to a local maximum. (Zhou et al. 2013, p. 8) Another area worth discussing is that even with greatly reduced dimension, the number of parameters often exceeds the small sample size in neuroimaging studies. As such, regularization can be useful for stabilizing the estimates. Implementing a penalty will result in the algorithm now involving a penalized GLM regression problem having to be fit to update $\boldsymbol{B}_d$. While conceptually possible to apply regularization to the full coefficient tensor $\boldsymbol{B}$, this results in further dimensionality issues. Other regularization methods such as including various parametric constraints can be considered. This low rank method helps lessen the curse of dimensionality that is present with ultra-high-dimensional data. Due to this, fast estimation and regularization can be performed. Despite the reduction in the parameter space, numerical analysis demonstrates that tensor regression methods work well at recovering features in the data. (Zhou et al. 2013, p. 13-14, 17)

Work has been done to further develop models that integrate tensor decomposition. Motivated by the complex structure of high dimensional multi-omics data used in gene-set integrative analysis, Chang et al. (2021) propose a tensor-based model to improve efficiency for inference. In this sort of analysis, the data is structured such that genes are examined on multiple platforms, which are designed to explain different aspects of the gene. The approaches for this sort of analysis consist of single-gene-single-platform analyses, single-gene-multi-platform analyses, multi-gene-single-platform analyses, and joint-based methods. Joint based methods regress outcomes on all variables. (Chang et al. 2021, p. 1-2) While joint-based methods generally outperform the other methods, previous work has encountered high-dimensional challenges. Previous work includes PCA methods (Meng et al. 2016), and penalized regression methods (Wu et al. 2019). To address such high-dimensional challenges without losing matrix structure, the approach put forth by Zhou et al. (2013) is utilized. For a gene-set framework, the data for a given individual will have a $P \times G$ structure, resulting in the data tensor having dimension $P \times G \times n$. Tensor-based methods retain the necessary structure to reveal associations between gene and platform levels. Prior methods tend to focus more on dimension reduction, feature extraction, and outcome prediction more than signal detection. (Chang et al. 2021) Methods to detect signals exist, but are designed to find global signals, or are based

on variable selection methods. Hung et al. (2016) utilizes a tensor model to perform variable screening, while Hung and Jou (2019) propose a global interaction test for a tensor regression model. Utilizing the framework in Zhou et al. (2013), an order-three rank-R tensor regression model is considered, with a tensor-based association being used to determine which variables should be prioritized and to help better understand the relationship between omics data and clinical outcomes (Chang et al. 2021, p. 2).

To briefly restate the model in terms of this problem, for a case with n samples, we denote y_i to be a continuous outcome for subject i. The order-3 tensor $\mathcal{X} \in \mathbb{R}^{P\times G\times n}$ stores the data for all of the samples, where P is the number of platforms and G is the number of genes. $\mathbf{X}_i$ is the ith slice of this tensor. Furthermore, z_i is a vector of covariates for sample i which includes the intercept. Then the order-2 tensor regression model is as follows:

$$y_i = \boldsymbol{z}_i^\top \boldsymbol{\beta} + \langle \mathbf{X}_i, \mathbf{B}\rangle + \varepsilon_i, \mathbf{B} = \mathbf{B}_1\mathbf{B}_2^\top \tag{2.18}$$

$\boldsymbol{\beta}$ is the parameter vector for the covariates, $\varepsilon_i \sim N(0,\sigma^2)$, $\mathbf{B} \in \mathbb{R}^{P\times G}$ is a parameter matrix, and $\langle\cdot,\cdot\rangle$ denotes the inner product where $\langle \mathbf{X}_i, \mathbf{B}\rangle = \text{vec}(\mathbf{X}_i)^T\text{vec}(\mathbf{B})$. Furthermore this model considers a rank-R tensor decomposition of $\mathbf{B}$, with $\mathbf{B} = \sum_{r=1}^{R} \mathbf{B}_1[,r]\mathbf{B}_2[,r]^\top = \mathbf{B}_1\mathbf{B}_2^\top$, $\mathbf{B}_1 \in \mathbb{R}^{P\times R}, \mathbf{B}_2 \in \mathbb{R}^{G\times R}$. (Chang et al. 2021, p. 2) To estimate results, an alternating least squares (ALS) algorithm is used to facilitate parameter estimation. To avoid solutions that find a local minima for the objective function rather than the global minima, multiple random initial values are considered. To determine the optimal rank, models are fit using rank r, and information criterion are used to select the optimal model. Both AIC and BIC are considered. The tensor-based test for association is a Wald test, where $H_0 : B_{pg} = 0$, and the test statistic is $T_{pg} = \frac{\hat{B}_{pg}}{\sqrt{[\sigma(\mathbf{C})]_{pg}}}$. $\sigma(\mathbf{C})$ is the variance-covariance matrix of $\hat{B}$, such that $[\sigma(\mathbf{C})]_{pg}$ is the variance for $\hat{B}_{pg}$. (Chang et al. 2021, p. 4)

Based on the results of simulation studies conducted in the paper, BIC has higher proportions of selecting true rank for large effect strength, but AIC performs better for weak effect strength. BIC tends to underestimate model rank, and AIC tends to overestimate. AIC is the recommended criterion to use, as it is more robust. (Chang et al. 2021, p. 4) Unlike most existing tensor-based tests, the test put forth here is variable-specific, enabling researchers to identify important variables with relative ease. One aspect of this model to note is that it focuses on continuous outcomes. While theoretically feasible, binary outcomes can be more challenging to implement numerically, and the convergence and stability of numerical properties can become complicated. (Chang et al. 2021, p. 6)

2.4 Proposed Model

Furthering the research in both multivariate multinomial choice modeling and tensor regression, the model proposed here utilizes both multivariate multinomial logit and tensor regression methods in an effort to better explain the joint relationship between multiple multinomial choices. The finer details of this multivariate multinomial logit model with tensor regression are provided in Appendix C . In summary, the model framework still is concerned with modeling multinomial choices. Here, binomial choices are considered, but the model could be extended for choices with a greater number of choices. For the binomial case, each joint outcome has a possible set of 2^K joint outcomes.

The stock-specific intercept term and association terms are the same as in Bel and Paap (2014). However, instead of using a vector covariate $\boldsymbol{\beta}_k$ for modeling the effects of the relative stock price change for previous days, a matrix covariate $\mathbf{B}_k$ is considered in an effort to better capture some underlying structure within the data. $\mathbf{B}_k$ is estimated in a way similar to Chang et al. (2021). Instead of an alternating least squares approach, an alternating logistic regression approach is used for parameter estimation. The model here is such that $Z_{ik} = \boldsymbol{\gamma}_{ik}^T\boldsymbol{\beta} + \langle \mathbf{X}_i, \mathbf{B}_k \rangle + \varepsilon_{ik}$, where $\mathbf{B}_k = \mathbf{B}_{k1}\mathbf{B}_{k2}^T$. Here, $\boldsymbol{\gamma}_{ik}^T\boldsymbol{\beta} = \alpha_k + \sum_{l \neq k} y_{il}\psi_{kl}$. $\boldsymbol{\gamma}_{ik}$ is a vector of ones and zeroes corresponding to the intercept and association parameters present for a particular day and stock choice. Additional notation and the estimation algorithm are provided in Appendix C.

In the simulation study, several types of models will be examined using a variety of simulation settings. The results of these simulations will be used to assess the performance of the models. Here, the models considered are the multivariate multinomial logit model put forth by Bel and Paap (2014) and a novel model that we are proposing that implements tensor regression in a MV-MNL logit framework. Also considered are two marginal models[1] which have have relaxed assumptions regarding the joint relationship of stocks. As marginal models, various degrees of between-stock independence are assumed. In the context of this simulation study, only the multivariate binomial logit (MVBNL) setting is assessed. For the real data examples, applications of tensor regression for multinomial choice models using financial data are considered.

A variety of simulation settings will be considered in the scope of this analysis. There are three primary ways for examining performance across these models. The two standard ways of assessing model performance are by computing the bias of the estimators by finding the difference between the mean estimates and the true values, and by computing the variance and standard deviation of the estimator from the truth. These two metrics will be used to assess the MVBNL and tensor regression models. A key point of interest is to examine how well the

[1] Marginal models explained in Appendix C .

models model the joint response. We can consider $Y_i = \{Y_{i1}, \ldots, Y_{iK}\}$ to be a joint set of choices or outcomes for an individual unit, such as days. For a given day i, there are 2^K unique sets of outcomes that could occur. We can analyze how well a model performs at predicting these outcomes by computing a model's multi-class area under its receiver operating characteristic (ROC) curve, referred to as a model's multi-class AUC[2]. The closer the multi-class AUC is to 1, the better a model is at classification. (Hand and Till 2001, p. 173)

The multi-class extension of AUC is the average of all unique AUC pairs. For 2^K possible outcomes, the multi-class AUC will be $\frac{2}{2^K(2^K-1)} \sum_{i<j} \hat{A}(i,j)$, where $\hat{A}(i,j)$ is the AUC for classes i, j (Hand and Till 2001). The computation of multi-class AUC over all possible outcomes is infeasible, because for large values of K there are more possible outcomes than days being considered. The data[3] are transformed so that only a top percentage of the outcomes sorted by frequency are considered. Other outcomes observed are re-coded to a baseline to which top outcomes are compared. In this simulation study, for training data the top 25% of outcomes are compared to each other and a baseline. For test data, the top 5% of outcomes are considered. The function `multiclass.roc` from the R package `pROC` is utilized. To compute AUC, the function requires a response vector, such as the top outcomes a day corresponds. A matrix giving the estimated probability of each outcome class occurring each day is also required.

For the tensor regression extension to the MVBNL model, information criteria such as AIC or BIC can be used to assess model quality. Based on prior simulation studies, AIC seems to be the preferable criterion to consider. (Chang et al. 2021, p. 4) The general form for AIC is $2 \cdot p - 2 \cdot \hat{L}$, where p is the number of parameters and $\hat{L}$ is the maximum value of the log-likelihood. For the AIC of the tensor regression extension to MVBNL model, the maximum value of the composite log likelihood is used in the computation.

In all of these results, the sample size for the data will be referred to as the total number of days. Each choice set Y_i will denote a set of K binomial choices on day i. For the standard MVBNL model, p denotes the number of parameters in each of the parameter vectors pertaining to each of the K choices. M denotes the the number of columns for a matrix covariate in the tensor regression extension of the MVBNL model. In this framework, the matrix covariates have rows corresponding to the K choices.

Two additional AUC metrics, co-movement AUC and conditional AUC, are also considered. The log-likelihood of observed estimated probability and the count of outcomes where the probability of the observed outcome is the maximum estimated probability are also metrics that are considered. These metrics are discussed in detail in section 2.5. For a real data example that focuses on applying the model to sector-specific stock data, co-movement mean squared

[2]Classical ROC and AUC are explained in detail in Appendix B.

[3]The data generation process is outlined Appendix A.

error (MSE) and co-movement concordance are considered. The details of these metrics are discussed in section 2.7.

2.5 Metrics

For the simulation study, the average of the metrics is considered. In some cases the standard deviations of the metrics are also discussed. In addition to AUC, the performance of joint MVBNL-TR model can be compared with simpler independent marginal models in other ways. A straightforward metric to consider is the count of cases where the observed estimated probability matches the maximum estimated probability for a day. The notation for this is as follows:

$$\text{Count}\left(\hat{P}(Y_i = y_i|\mathbf{X}_i) = \max_{s_i \in S}(\hat{P}(Y_i = s_i)|\mathbf{X}_i)\right), \tag{2.19}$$

where the joint probability follows the form expressed in 2.2.2, except that

$$\mu_{y_i} = \sum_{k=1}^{K} y_{ik}(\alpha_k + \langle X_i, \mathbf{B}_k\rangle + \sum_{l>k} y_{ik}\, y_{il}\, \psi_{kl} \tag{2.20}$$

Here y_i is the observed outcome and S denotes the space of possible outcomes. Having a high count for which this event occurs is a desirable outcome. The log-likelihood of observed estimated probability is also considered. Log-likelihood is considered instead of likelihood, as for large outcome probability spaces each individual probability will be small. Thus, the product of many probabilities will be a small value, often treated by software as being zero. Using the log-likelihood avoids this issue. Due to fact that probabilities are being considered, the log-transformed values will be negative. Thus, log-likelihoods closer to zero indicate better model performance. The log-likelihood for observed estimated probability is as follows:

$$\sum_{i=1}^{T} \log\left(\hat{P}(Y_i = y_i|\mathbf{X}_i)\right) \tag{2.21}$$

Conditional AUC is considered as well. Conditional AUC utilizes the conditional probability rather than the joint probability in classifying stock outcomes. Instead of measuring how well a model classifies a joint response, conditional AUC is concerned with how well a model classifies a stock-specific response given all other stock outcomes. Thus, the AUC for each of the K stocks is computed, instead of the multi-class AUC. For the marginal models, the conditional probability is just the marginal probability, as independence between stocks is

assumed. Thus the conditional probabilities would be

$$\hat{P}(Y_{ik}=1|X_i,\ldots)=\frac{\exp(\hat{\alpha}_k+\hat{\boldsymbol{\beta}}_k^{\top}X_{k[i,]})}{1+\exp(\hat{\alpha}_k+\hat{\boldsymbol{\beta}}_k^{\top}X_{k[i,]})},\hat{P}(Y_{ik}=1|X_i,\ldots)=\frac{\exp(\hat{\alpha}_k+\langle\mathbf{X}_i,\hat{\mathbf{B}}_k\rangle)}{1+\exp(\hat{\alpha}_k+\langle\mathbf{X}_i,\hat{\mathbf{B}}_k\rangle)} \tag{2.22}$$

for the first and second marginal models, which will be referred to as model model 1 and model 2. The response is simply the outcomes for stock K. Using this information all K conditional AUCs can be computed. For the joint model, the conditional probability can be derived from the conditional probability proposed in Bel and Paap (2014). The form proposed in that paper is

$$P(Y_{ik}=1|y_{il}\text{ for }l\neq k,\mathbf{X}_i)=\frac{\exp(Z_{ik})}{1+\exp(Z_{ik})}. \tag{2.23}$$

Due to differences between that model and in the MV-MNL-TR model, $Z_{ik}=\alpha_k+\langle\mathbf{X_i},\mathbf{B}_k\rangle+\sum_{l\neq k}y_{il}\psi_{kl}$ for the joint model. For all stocks and models, the probability matrix will consist of the positive outcome probability and its complement.

Additionally, a class of metrics that measure co-movement are considered. These metrics are co-movement AUC, co-movement mean squared error (MSE), and co-movement concordance. Co-movement tracks whether two responses move together in the same direction. For K stocks, there will be $K\cdot(K-1)/2$ pairs of outcomes that can potentially co-move. When two stocks for a stock pair move up or down in value together, the co-movement value is one, and zero otherwise. Thus, co-movement AUC is a metric which can give insight into how well a model classifies co-movement outcomes. To calculate co-movement AUC, the binary co-movement outcomes are computed according to the outcomes of the individual stocks. For the marginal models, the co-movement probability for stock pair (k,κ) is

$$P(y_{ik}=y_{i\kappa})=P(y_{ik}=1)P(y_{i\kappa}=1)+P(y_{ik}=0)P(y_{i\kappa}=0) \tag{2.24}$$

Calculating the co-movement estimated probability for the marginal cases is relatively straightforward. For the joint model, the process is more involved. The computation of the joint probability matrix relies on the entire outcome space S which consists of 2^K possible outcomes for every day. From this outcome space, an index can be created for (k,κ) to reflect co-movement. When the kth and κth stock co-move, ones are assigned, and when they do not, zeroes are assigned. For day i, the dot product of this index vector with the vector of probabilities for the joint outcome space for this day are computed. This quantity corresponds to the probability of co-movement for a (k,κ) pair on day i. The co-movement probability can then be computed for all days in this way, and the process is the same for all pairs. To compute

AUC for this quantity, all that is needed is the co-movement response, and a matrix consisting of the probability of co-movement, as well as the probability of non-co-movement, which is $1-P(y_{ik}=y_{i\kappa})$.

Co-movement MSE is defined in the following manner.

$$\text{MSE}=\frac{1}{T}\sum_{i=1}^{T}\left(\hat{P}_{model}(y_{ik}=y_{i\kappa})-P_{true}(y_{ik}=y_{i\kappa})\right)^2 \tag{2.25}$$

Here, $\hat{P}_{model}$ is the observed co-movement probability. P_{true} is slightly different. The true probability is one if co-movement is observed for a pair, and 0 if not. The lower this MSE, the better the model performance. As for co-movement concordance, it is an average across the days in the data set of the count of cases where co-movement outcome probability exceeds the probability of non-co-movement. Computing the concordance involves comparing every co-movement pair's probability with the probability of all non-co-movement pairs. The counts for a given day are normalized by a factor of $\binom{K}{2}$. To compute the concordance, the sets of co-movement and non-co-movement probabilities must be constructed. Then, the concordance is computed using the formula seen below.

$$\frac{1}{T}\sum_{i=1}^{T}\sum_{(k,\kappa)\in 1}\sum_{(j,j')\in 0}\frac{I(\hat{P}_{model}(y_{ik}=y_{i\kappa})>\hat{P}_{model}(y_{ij}\neq y_{ij'}))}{\binom{K}{2}} \tag{2.26}$$

The greater the concordance value, the stronger the predictive power of the model. Note that for every co-movement pair, if the probability of co-movement exceeds the probability of non-co-movement for all pairs, the concordance will be equivalent to $\frac{1}{T}\cdot T\cdot\binom{K}{2}\cdot\frac{\binom{K}{2}}{\binom{K}{2}}$. This means that the maximum value of the concordance is $\binom{K}{2}$.

2.6 Simulated Results

2.6.1 Multivariate Binomial Logit

First, consider the bias and standard deviation for the intercept parameters α_k for the model fit on training data. Recall that K denotes the number of stocks considered, and M denotes the length of the look-back window.

Table 2.1: MVBNL Summary for α_k for NSim = 100

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.132	-1.000	-0.132	1.408
α_2	-0.556	-0.500	-0.056	1.026
α_3	0.032	0.000	0.032	0.838
α_4	0.562	0.500	0.062	1.041
α_5	1.333	1.000	0.333	1.544

(a) $K = M = 5$, NDays = 500

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.074	-1.000	-0.074	1.345
α_2	-0.556	-0.500	-0.056	0.981
α_3	-0.031	0.000	-0.031	0.785
α_4	0.536	0.500	0.036	0.940
α_5	1.102	1.000	0.102	1.343

(b) $K = M = 5$, NDays = 1000

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.262	-1.000	-0.262	1.459
α_2	-0.824	-0.750	-0.074	1.087
α_3	-0.542	-0.500	-0.042	0.887
α_4	-0.289	-0.250	-0.039	0.747
α_5	-0.445	0.000	-0.445	1.074
α_6	-0.063	0.000	-0.063	0.767
α_7	0.264	0.250	0.014	0.733
α_8	0.596	0.500	0.096	0.889
α_9	0.917	0.750	0.167	1.129
α_{10}	1.190	1.000	0.190	1.403

(c) $K = 10, M = 5$, NDays = 1000

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.215	-1.000	-0.215	1.366
α_2	-0.849	-0.750	-0.099	1.076
α_3	-0.587	-0.500	-0.087	0.876
α_4	-0.286	-0.250	-0.036	0.773
α_5	-0.156	0.000	-0.156	0.732
α_6	-0.028	0.000	-0.028	0.720
α_7	0.298	0.250	0.048	0.725
α_8	0.605	0.500	0.105	0.897
α_9	0.891	0.750	0.141	1.081
α_{10}	1.166	1.000	0.166	1.3410

(d) $K = 10, M = 5$, NDays = 2000

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.534	-1.000	-0.534	1.823
α_2	-1.055	-0.750	-0.305	1.336
α_3	-0.671	-0.500	-0.171	1.002
α_4	-0.279	-0.250	-0.029	0.792
α_5	-1.088	0.000	-1.088	1.721
α_6	0.042	0.000	0.042	0.814
α_7	0.340	0.250	0.090	0.843
α_8	0.683	0.500	0.183	1.041
α_9	1.126	0.750	0.376	1.389
α_{10}	1.856	1.000	0.856	2.110

(e) $K = M = 10$, NDays = 1000

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.303	-1.000	-0.303	1.459
α_2	-0.914	-0.750	-0.164	1.139
α_3	-0.573	-0.500	-0.073	0.871
α_4	-0.303	-0.250	-0.053	0.696
α_5	-0.294	0.000	-0.294	0.880
α_6	-0.018	0.000	-0.018	0.722
α_7	0.245	0.250	-0.005	0.695
α_8	0.638	0.500	0.138	0.912
α_9	0.958	0.750	0.208	1.143
α_{10}	1.309	1.000	0.309	1.474

(f) $K = M = 10$, NDays = 2000

Where sample size is increased, the decrease in standard deviation and bias for the parameters is most prevalent for the case where $K = M = 5$. For a larger parameter space, this behavior is still present, but its effects are less uniform, as there are some cases where the standard deviation or bias will increase when sample size increases. For fixed sample size, an increase in the size in the parameter space will coincide with an increase in the bias and standard deviation of the parameters.

For the MVBNL model, a parameter vector of length $p = K \cdot M$ is considered in all cases. That is, each parameter vector $\boldsymbol{\beta}_k$ is of length $K \cdot M$. For $\hat{\beta}_k$, where $K = M = 5$, the standard

deviation decreases for almost all parameters when sample size is increased, with only a few exceptions with relatively small increases. Bias is also reduced for many of the parameters, but less uniformly so than standard deviation. For the smaller sample size, the magnitude of the bias does not exceed |0.13|. When the sample size is increased, the magnitude of the bias does not exceed |0.105|, resulting in an overall reduction in the maximum magnitude of the bias for the individual parameters. The variance decreases slightly from a magnitude of 1.414 to 1.352 when the sample size is increased. For larger parameter spaces, such as $K = 10, M = 5$, the bias and standard deviation stay fairly stable when sample size is increased. Some parameters have increased bias or standard deviation, but the differences are small. For a total sample size of 1000, the largest magnitude of the bias is |0.215|, and when the total sample size is increased to 2000, the magnitude of the bias is no larger than |0.197|. When the sample size is increased to 2000, the largest observed standard deviation for a parameter decreases from 1.425 to 1.393.

For $K = M = 10$, the bias is less stable, with several $\hat{\beta}_k$ having more increases in bias than decreases for increased sample size. The standard deviation however, decreases for the majority of parameters, with very few seeing any increase in standard deviation when the sample size is increased from 1000 to 2000. When the sample size is increased, the maximum magnitude of the bias increases from |0.964| to |1.890|, while the maximum standard deviation decreases from 2.294 to 1.521. In summary, when the parameter space is increased, the bias seems to increase as well. The standard deviations of the parameters also increase somewhat, but neither increase greatly. As the parameter space is held fixed and the sample size is increased, bias stays relatively stable, with some small increases for some parameters, but standard deviation consistently decreases.

With regards to the association parameters ψ_{kl}, when $K = M = 5$, the bias is fairly stable when the sample size is increased, with about half the association parameters seeing a decrease in bias. The maximum magnitude for the bias of the association parameters increases from |0.050| to |0.082|. There is a consistent decrease in the standard deviation, with the maximum standard deviation decreasing from 0.211 to 0.108. For $K = 10, M = 5$, the bias is fairly stable when sample size is increased, with less than half the bias terms increasing. The maximum magnitude of the bias decreases from |0.416| to |0.326|. There is a consistent reduction for the standard deviations of the association parameters, with the maximum standard deviation observed decreasing from 0.704 to 0.229. For $K = M = 10$, substantially less than half of the association parameters see an increase in bias when the sample size is increased, and the maximum magnitude of the bias is decreased substantially from |1.592| to |0.563|. There is again a consistent reduction in the standard deviation, with the largest standard deviation observed decreasing from 4.129 to 0.412, a sizeable reduction. To summarize, while the bias increases in magnitude as the parameter space increases, for fixed parameter space size it

decreases in magnitude for increases in sample size, with the exception of the case where $K = M = 5$. When the size of the parameter space is increased, the standard deviations of the association parameters also increase in magnitude, but for fixed parameter space, they consistently decrease as sample size increases.

A final way to assess model quality is by looking at the AUC for both training and test data across different sample sizes and parameter spaces. These will be analyzed at length in the next section, and compared to the AUC for the MVBNL-TR model.

2.6.2 Multivariate Binomial Logit with Tensor Regression

Before discussing simulation settings, it is important to properly define the multivariate binomial logit tensor regression (MVBNL-TR) model being proposed. In the model put forth by Bel and Paap (2014) for the binomial case, the conditional probability $P(Y_{ik} = 1 | y_{il} \text{ for } l \neq k, X_i) = \frac{\exp(Z_{ik})}{1+\exp(Z_{ik})}$. In the multivariate binomial case, $Z_{ik} = \alpha_k + X_i\beta_k + \sum_{l \neq k} y_{il}\psi_{kl}$. The actual realizations of Y_{il} are y_{il}. X_i denotes is a predictor vector, with a parameter vector β_k. α_k is an intercept parameter, and the association parameters for $l \neq k$ are ψ_{kl}. However, when integrating tensor regression into this framework, some aspects of this model will change.

While a general response Y_{ik} is still a kth binary choice for an day i, Z_{ik} will be defined differently, taking inspiration from Chang et al. (2021). We define the model as follows:

$$Z_{ik} = \boldsymbol{\gamma}_{ik}^{\top}\boldsymbol{\beta} + \langle \mathbf{X}_i, \mathbf{B}_k \rangle + \varepsilon_{ik}, \mathbf{B}_k = \mathbf{B}_{k1}\mathbf{B}_{k2}^{\top} \tag{2.27}$$

For a day i and choice k, $\boldsymbol{\gamma}_{ik}^{\top}\boldsymbol{\beta} = \alpha_k + \sum_{l \neq k} y_{il}\psi_{kl}$. Here $\boldsymbol{\gamma}_{ik}$ is a vector consisting of ones and zeroes corresponding the intercept and association parameters present for a particular day and choice.

There are several changes in notation from Bel and Paap (2014). Instead of referring to individual i, the model will refer to day i, with the total number of days being T. Furthermore, in the interest of considering an application for this model, the K "choices" for a given day i will be considered to be the increase (1) or decrease (0) of a stock prices from the previous day. In this model, the response Y_{ik} is of length $T \cdot K$, and it represents a binary response for if the stock k changes on day i. K will refer to the total number of stocks, with k referring to a specific stock. Here $\mathbf{X}_i$ will be a $K \times M$ slice of a $K \times M \times T$ tensor $\mathcal{X}$. Each slice contains the relative change in all K stock prices for the previous M days. Relative change is $\frac{\text{Stock Price}_i - \text{Stock Price}_{i-1}}{\text{Stock Price}_{i-1}}$.

Tensor regression methods thus far have utilized an alternating least squares (ALS) approach to parameter estimation. In this model, an alternating logistic regression (ALR) framework is considered. The details of the ALR algorithm, along with the design matrices used and the

details of parameter estimation are included in Appendix C. To assess the performance of this model, two comparison models are considered. For these models, instead of considering a MVBNL-TR model that models outcomes for different stocks jointly, we consider K marginal[4] models. Each model consists of a response vector $\boldsymbol{y}_k$ of length T containing responses for stock k denoting price increases or decreases from a previous day. In the first type of model, the parameters of interest are a stock-specific intercept α_k and a parameter vector $\boldsymbol{\beta}_k$ of length M relating to the covariates associated with the M-day look-back window of relative stock price change. Instead of slicing $\mathscr{X}$ on the individual days, it will be sliced on the stocks. Each stock will have an $M \times T$ matrix, but consider its transpose $\mathbf{X}_k$ which is $T \times M$. For day i and stock k, $Z_{ik} = \alpha_k + \boldsymbol{\beta}_k^\top \mathbf{X}_k[i,]$. The second comparison model is more complex. K marginal models are again considered, but for fixed k, $Z_{ik} = \alpha_k + \langle \mathbf{X}_i, \mathbf{B}_k \rangle$. This is similar to the original MVBNL-TR model definition, but without association terms. Instead of one joint MVBNL-TR model, K marginal models are considered. It is important to discuss how to compute the joint probability for these marginal models. These probabilities can be computed as follows: $\hat{P}(Y_i = y_i|\mathbf{X}_i,\ldots) = \hat{P}(Y_{i1} = y_{i1}|\mathbf{X}_i,\ldots) \times \cdots \times \hat{P}(Y_{iK} = y_{iK}|\mathbf{X}_i,\ldots)$. The marginal models have marginal probabilities that are the same as the conditional probabilities,

$$\hat{P}(Y_{ik} = 1|\mathbf{X}_k,\ldots) = \frac{\exp(\hat{\alpha}_k + \hat{\boldsymbol{\beta}}_k^\top \mathbf{X}_k[i,])}{1 + \exp(\hat{\alpha}_k + \hat{\boldsymbol{\beta}}_k^\top \mathbf{X}_k[i,])} \tag{2.28}$$

and

$$\hat{P}(Y_{ik} = 1|\mathbf{X}_k,\ldots) = \frac{\exp(\hat{\alpha}_k + \langle \mathbf{X}_i, \hat{\mathbf{B}}_k \rangle)}{1 + \exp(\hat{\alpha}_k + \langle \mathbf{X}_i, \hat{\mathbf{B}}_k \rangle)} \tag{2.29}$$

for marginal models 1 and 2 respectively.

For the tensor regression model, the number of parameters that have to be properly estimated is the sum of the $\alpha_k, \boldsymbol{\psi}, \mathbf{B}_{k1}$, and $\mathbf{B}_{k2}$ parameters. In total, this amounts to $K + \frac{K\cdot(K-1)}{2} + K \cdot (K \cdot R + M \cdot R)$ parameters. This must be taken into account when choosing the sample size for a batch of simulations. For each group of simulation settings, 100 simulations are run. For the tensor regression extension of the multivariate binomial logit model, the parameter setting combinations of $K = M = 5$, $K = 10$ and $M = 5$, $K = M = 10$ are considered. Here, both a rank-one and rank-three approach are considered for these parameter setting combinations. When a lower-rank approximation is used, there is a simpler parameter space. Increasing the rank should see some metrics improving, but at the cost of a more complex parameter space. For all of these combinations, the total number of days considered for each is 1000. The data in each case is partitioned into a 75%-25% training-test split, with 750 days in the training set and 250 in the test set. For the intercept parameters α_k, the bias and standard deviation of the

[4]Marginal models are explained in detail in section C.2.1 of the appendix

parameter estimates for the training data are summarized below.

Table 2.2: MVBNL-TR Summary for α_k for NSim = 100

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.025	-1.0	-0.025	0.362
α_2	-0.537	-0.5	-0.037	0.300
α_3	-0.028	0.0	-0.028	0.267
α_4	0.522	0.5	0.022	0.312
α_5	1.042	1.0	0.042	0.305

(a) $K = M = 5$, T = 1000, R = 1

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.007	-1.0	-0.007	0.450
α_2	-0.485	-0.5	0.015	0.342
α_3	0.011	0.0	0.011	0.283
α_4	0.528	0.5	0.028	0.304
α_5	1.097	1.0	0.097	0.373

(b) $K = M = 5$, T = 1000, R = 3

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.149	-1.00	-0.149	0.367
α_2	-0.758	-0.75	-0.008	0.283
α_3	-0.532	-0.50	-0.032	0.268
α_4	-0.277	-0.25	-0.027	0.261
α_5	-0.287	0.00	-0.287	0.559
α_6	-0.087	0.00	-0.087	0.344
α_7	0.220	0.25	-0.030	0.288
α_8	0.529	0.50	0.029	0.283
α_9	0.821	0.75	0.071	0.291
α_{10}	1.055	1.00	0.055	0.320

(c) $K = 10, M = 5$, T = 1000, R = 1

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.145	-1.00	-0.145	0.477
α_2	-0.777	-0.75	-0.027	0.429
α_3	-0.546	-0.50	-0.046	0.420
α_4	-0.295	-0.25	-0.045	0.319
α_5	0.014	0.00	0.014	0.423
α_6	0.048	0.00	0.048	0.346
α_7	0.341	0.25	0.091	0.302
α_8	0.530	0.50	0.030	0.324
α_9	0.735	0.75	-0.015	0.327
α_{10}	1.118	1.00	0.118	0.414

(d) $K = 10, M = 5$, T = 1000, R = 3

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.032	-1.00	-0.032	0.480
α_2	-0.820	-0.75	-0.070	0.406
α_3	-0.560	-0.50	-0.060	0.342
α_4	-0.241	-0.25	0.009	0.309
α_5	-0.124	0.00	-0.124	0.413
α_6	0.000	0.00	0.000	0.346
α_7	0.250	0.25	0.000	0.315
α_8	0.514	0.50	0.014	0.351
α_9	0.845	0.75	0.095	0.414
α_{10}	1.117	1.00	0.117	0.492

(e) $K = M = 10$, T = 1000, R = 1

	Mean $\hat{\alpha}$	True α	Bias($\hat{\alpha}$)	SD($\hat{\alpha}$)
α_1	-1.332	-1.00	-0.332	0.684
α_2	-0.964	-0.75	-0.214	0.573
α_3	-0.569	-0.50	-0.069	0.389
α_4	-0.256	-0.25	-0.006	0.337
α_5	-0.116	0.00	-0.116	0.479
α_6	0.005	0.00	0.005	0.393
α_7	0.287	0.25	0.037	0.373
α_8	0.483	0.50	-0.017	0.373
α_9	0.719	0.75	-0.031	0.418
α_{10}	1.100	1.00	0.100	0.500

(f) $K = M = 10$, T = 1000, R = 3

Both the bias and standard deviation are relatively stable. There are some increases in standard deviation when a rank approximation greater than one is performed. The bias is stable for rank increases, but decreases in some cases for fixed rank and increased parameter space. Because the true parameters here are identical to those in the MVBNL case where tensor

regression is not performed, the performance can be directly compared for the cases where $T = 1000$. The α_k estimates from the MVBNL-TR all have lower standard deviation, and most have decreased bias, with few exceptions. For the MVBNL-TR model where $K = 10, M = 5, R = 1$, $\text{Bias}(\alpha_6) = -0.087$, and $\text{Bias}(\alpha_7) = -0.03$. In the MVBNL model, these parameters have biases of -0.063 and 0.014 respectively. For the case where the rank is 3, the MVBNL-TR performs considerably better, with sizeable reductions in bias and standard deviation.

With regards to the matrix covariates $\mathbf{B}_k$, when the rank is increased, the standard deviation of the cells of the matrix covariate parameters tend to increase. There are a few cases where the standard deviation decreases, but the majority increase. For the case where $K = M = 5, R = 1$ the largest standard deviation is 0.203, corresponding to a cell of $\mathbf{B}_5$. When the rank is increased to 3, the largest standard deviation observed is 0.284, and is from a cell in $\mathbf{B}_1$. For $K = 10, M = 5$, and $K = M = 10$ there are similar increases for a rank increase. This increase is not surprising, as an increase in the rank of the approximation means that the true tensor decompositions of parameters $\mathbf{B}_k$ are becoming more complex.

The bias is fairly stable when the rank is increased, with some increases and some decreases notable across all the matrix parameters. When $K = M = 5$, the maximum magnitudes still increase when the rank of the true decomposition of $\mathbf{B}_k$ is increased. For $K = 10, M = 5$, when the rank increases there are some increases in the maximum magnitude of cells of B_k, but not for all k. The same also holds for $K = M = 10$. A direct comparison for fixed rank but increased parameter space is not applicable here in the same way as it is for α_k, ψ_{kl}, due to the different size and true parameter values for $\mathbf{B}_k$. However, by comparing the maximum standard deviations and magnitudes of biases across the many cells of these matrix parameters, it is evident that for fixed rank the bias and standard deviation are all fairly stable. There are some increases when the parameter space increases, but this is more noticeable for higher rank approximations, which is to be expected.

Lastly, it is insightful to compare the matrix parameters $\mathbf{B}_k$ to their vectorized counterparts $\boldsymbol{\beta}_k$ from the MVBNL model. These parameters share the same true values, so it is important to compare them in terms of bias and standard deviation. A simple way to compare the simpler MVBNL model to the model using the tensor regression extension is to consider all of the instances where bias and standard deviation are greater in the MVBNL-TR model. Having few instances of this behavior is desirable. There is only one instance of increased standard deviation across all simulation settings and rank setting combinations, indicating that the estimates for B_k produced by the tensor regression method have reduced variance in contrast to those produced by the multivariate binomial logit model which does not utilize tensor regression. We can compare these magnitude of the bias in these models by considering the average proportion of cases where the bias of $|\text{Bias}(\text{vec}(\mathbf{B}_k))|$ exceeds that of $|\text{Bias}((\boldsymbol{\beta}_k))|$ across

all k. The proportion in each case will involve the number of cases where this inequality holds over $K \times M$ total cases.

Table 2.3: Average Proportion(|Bias(vec($\mathbf{B}_k$))| - |Bias(($\boldsymbol{\beta}_k$))| > 0)

	$R=1$	$R=3$
$K=M=5$	0.192	0.464
$K=10, M=5$	0.148	0.314
$K=M=10$	0.052	0.187

As seen above in Table 2.3, the average proportion for each case is low in all of the rank-one models. When the rank of the model is increased, the average proportion of cases where the bias in the tensor regression model exceeds that of the MVBNL model increases. However, the majority of parameters still have lower absolute bias in the tensor regression model. Additionally, as the parameter space increases, the average proportion of the bias decreases in both the rank-one and rank-three models. Both models estimate these parameters, whether or not the tensor regression extension is utilized. However, the decrease in standard deviation of $\mathbf{B}_k$ estimates for the tensor regression model is a key benefit of the tensor regression model.

With regards to ψ_{kl}, for $K=M=5$, when the rank is increased, the maximum magnitude of the bias increases from |0.028| to |0.079|, while the maximum standard deviation increases from 0.311 to 0.382. For $K=10, M=5$, when rank increases, the maximum magnitude of bias increases from |0.165| to |0.204|. The maximum standard deviation decreases from 0.629 to 0.430. Lastly, for $K=M=10$, the maximum magnitude of the bias decreases from |0.352| to |0.259|. The maximum standard deviation decreases from 0.641 to 0.572. Bias is fairly stable, but increases for fixed rank when the parameter space grows. Standard deviation behaves interestingly, with substantial increases when the parameter space first increases, but when it increases again the maximum standard deviation of the association parameters remains stable. When parameter space is held fixed but rank is increased, maximum standard deviation decreases, with the exception of the case where $K=M=5$. Bias increases for the first two parameter spaces but decreases for the third parameter space.

The proportion of cases where the bias and standard deviation of $\boldsymbol{\psi}$ parameters in the tensor regression model exceed the MVBNL model can be considered. Recall that there are a total of $K \cdot (K-1)/2$ $\boldsymbol{\psi}$ parameters. When the size of the parameter space is increased, the proportion of cases with greater bias increases from $K=M=5$ to $K=M=10$. However, with the exception of the rank-three case for $K=M=10$, the tensor regression model tends to have

less bias than the standard MVBNL model. The standard deviations are slightly higher in the tensor regression case for the majority of the association parameters. As the parameter space increases, this majority shrinks. These proportions can be seen in Table 2.4.

Table 2.4: Proportions for Bias/SD of MVBNL-TR association parameters exceeding MVBNL

	$R=1$	$R=3$
$K=M=5$	0.1	0.5
$K=10, M=5$	0.089	0.178
$K=M=10$	0.178	0.89

(a) Prop(|Bias($(\psi_{kl,TR})$)| - |Bias((ψ_{kl}))| > 0)

	$R=1$	$R=3$
$K=M=5$	1	1
$K=10, M=5$	0.978	0.911
$K=M=10$	0.711	0.833

(b) Prop(SD($(\psi_{kl,TR})$) - SD((ψ_{kl})) > 0)

Lastly, it is important to assess these models in terms of their quality using both AIC and AUC. AIC is a useful metric for model selection when comparing multiple models of varying parameter sizes, and is useful in conjunction with other knowledge about the data, such as the multi-class AUC. With regards to AIC, the log-likelihood and number of parameters will differ according to the model. For the MVBNL model, the number of parameters is

$$K + \frac{K \cdot (K-1)}{2} + K \cdot (K \cdot M). \tag{2.30}$$

For the MVBNL-TR model, instead of having $K \cdot (K \cdot M)$ for the K $\boldsymbol{\beta}_k$ parameters of length $p = K \cdot M$, the model considers the size of the tensor decomposition $\mathbf{B}_k = \mathbf{B}_{k1}\mathbf{B}_{k2}^{\top}$. Thus, for each $\mathbf{B}_k$, the number of parameters considered is $K \cdot R + M \cdot R$. Therefore, the parameter size for that model is

$$K + \frac{K \cdot (K-1)}{2} + K \cdot (K \cdot R + M \cdot R). \tag{2.31}$$

For the MVBNL model, the composite log likelihood with MLEs is

$$\sum_{i=1}^{T}\sum_{k=1}^{K}\sum_{j=0}^{1} I(Y_{ik} = j) \log \hat{P}(Y_{ik} = j | y_{il} \text{ for } l \neq k, X_i) \tag{2.32}$$

(Bel and Paap 2014, p. 9) Here, $\hat{P}(Y_{ik} = j | y_{il}$ for $l \neq k, X_i)$ corresponds to the conditional probability for such an outcome with the MLE estimates for the parameters substituted in. For the MVBNL model,

$$\hat{P}(Y_{ik} = 1 | y_{il} \text{ for } l \neq k, X_i) = \frac{\exp(\hat{Z}_{ik})}{1 + \exp(\hat{Z}_{ik})}, \tag{2.33}$$

where

$$\hat{Z}_{ik} = \hat{\alpha}_k + X_i \hat{\boldsymbol{\beta}}_k + \sum_{l \neq k} y_{il} \hat{\psi}_{kl} \tag{2.34}$$

The change for the tensor regression extension is fairly straightforward, with the second term of $\hat{Z}_{ik}$ being $\langle \mathbf{X}_i, \hat{\mathbf{B}}_k \rangle$ instead of $X_i \hat{\boldsymbol{\beta}}_k$.

In the rank-one case, the tensor regression model has a lower AIC in all cases considered. For the rank-three approximations, the MVBNL model has a lower AIC than the tensor regression case. This can be seen below in Table 2.5. Having a lower AIC score is desirable, but this should be considered in conjunction with the results for multi-class AUC. It is necessary to compare the tensor regression extension to both the baseline MVBNL model, as well as the two marginal models used to assess the performance of the MVBNL-TR model. All models have the same regime for grouping responses. For the training data, $0.25 \cdot \min(T, 2^K)$ of the most frequently observed classes are present in the re-coded response as their respective class, but all remaining responses are grouped together as a baseline level. This process is much the same in the test data, except that the top $0.05 \cdot \min(T, 2^K)$ of responses is considered. This recoding is done due to the computational difficulties that occur with increases in K.

Table 2.5: Average AIC

	$R=1$		$R=3$	
	BP[1]	TR[2]	BP	TR
$K=M=5$	3968.23	3894.37	3719.03	3778.43
$K=10, M=5$	7380.33	7065.00	7374.91	7857.09
$K=M=10$	7032.19	6362.63	7349.18	7452.10

[1] MVBNL model (Bel and Paap 2014)

[2] Tensor regression extension to MVBNL

With most of the models, there is a clear increase in multi-class AUC as the parameter space increases, for both training and test data, as seen below in Table 2.6. The only exception to this pattern can be observed for the AUC for the test data for marginal model 1 when $R=3$. Marginal model 1 performs the worst out of all the models, for both training and test data. The tensor regression extension outperforms marginal model 2 in every case, but for the training data, the baseline MVBNL model outperforms the MVBNL-TR model. This result is not wholly unexpected, as the parameter space is considerably larger for the baseline model. As such, this model is more prone to over-fitting the data, and as such it might over-perform the tensor

regression extension. When the rank is increased, the tensor regression model has improved training set performance on average. While the baseline model still outperforms the tensor regression model for the training data, the differences in average AUC are considerably less.

Table 2.6: Average AUC, NSim = 100, Total Days = 1000

		$R=1$				$R=3$			
		M1	M2	BP	TR	M1	M2	BP	TR
Train	$K=M=5$	0.613	0.721	0.738	0.724	0.563	0.765	0.768	0.766
	$K=10, M=5$	0.629	0.839	0.883	0.852	0.595	0.884	0.891	0.888
	$K=M=10$	0.654	0.886	0.944	0.895	0.621	0.917	0.941	0.920
Test	$K=M=5$	0.586	0.663	0.658	0.670	0.559	0.710	0.714	0.715
	$K=10, M=5$	0.613	0.818	0.808	0.827	0.559	0.807	0.815	0.811
	$K=M=10$	0.630	0.864	0.837	0.868	0.556	0.826	0.820	0.830

For the test data, the MVBNL performs worse than marginal model 2 and MVBNL-TR in most cases, with average AUC only being higher for the case where $K=10,\ M=5$, and $R=3$. In all other cases, the tensor regression model has the highest average multi-class AUC for the test data. Given MVBNL-TR's larger AUC for test data and the model's lower average AIC score in low-rank cases, there is an apparent advantage to using the MVBNL-TR model for this situation. It clearly performs better than the marginal models for both training and test data, and outperforms the MVBNL model for most test data situations. It is also a simpler model than the MVBNL model. With lower or comparable AIC, it seems to be the case that the simpler MVBNL-TR model may be a better fit for these data.

Other metrics also warrant discussion. First, consider the average count of cases where observed estimated probability matches maximum estimated probability in Table 2.7. As the parameter space increases, the average count decreases. This is true for both marginal models as well as the tensor regression model. When the rank is increased, the average count decreases for larger parameter spaces, but increases in the $K=M=5$ case. In all cases, the tensor regression model outperforms both marginal models. The MVBNL-TR model having higher average count in most instances is desirable, but other metrics also need to be considered.

Table 2.7: Mean Count $\left(\hat{P}(Y_i = y_i|\mathbf{X}_i) = \max_{s_i \in S}(\hat{P}(Y_i = s_i)|\mathbf{X}_i)\right)$, NSim = 100, T = 1000

		$R=1$			$R=3$		
		M1	M2	TR	M1	M2	TR
Train	$K=M=5$	113.51	152.73	160.24	113.53	194.25	199.37
	$K=10, M=5$	17.2	52.22	59.43	11.24	47.67	51.91
	$K=M=10$	19.49	84.90	93.21	12.6973.90	77.40	
Test	$K=M=5$	34.64	45.68	47.84	34.00	55.08	55.94
	$K=10, M=5$	5.18	14.63	15.57	2.95	10.00	10.61
	$K=M=10$	5.25	24.44	25.45	2.98	12.41	12.77

The log-likelihood for observed estimated probability can also be used to assess model performance, as seen in Table 2.8. For each simulation setting, there are two types of probability log-likelihoods that are considered for the training and test data. The first uses the "full" response vector; that is, there is no re-coding done to the response as is done for the multi-class AUC calculations. The second type of probability log-likelihood uses the "reduced" response vector that is also utilized in the AUC calculations.

Table 2.8: Average Estimated Probability Log-likelihood, NSim = 100, Total Days = 1000

		$R=1$			$R=3$		
		M1	M2	TR	M1	M2	TR
Train	$K=M=5$	-2183	-1957	-1919	-2166	-1782	-1753
	$K=10, M=5$	-4463	-3605	-3463	-4695	-3602	-3519
	$K=M=10$	-4499	-3163	-3043	-4687	-3237	-3166
Top 25%	$K=M=5$	-1448	-1308	-1292	-1503	-1248	-1236
	$K=10, M=5$	-3750	-2986	-2874	-3667	-2764	-2705
	$K=M=10$	-3931	-2706	-2605	-3759	-2538	-2485
Test	$K=M=5$	-739.3	-673.4	-664.2	-734.5	-642.6	-636.4
	$K=10, M=5$	-1508	-1258	-1232	-1585	-1364	-1356
	$K=M=10$	-1538	-1131	-1118	-1601	-1350	-1350
Top 5%	$K=M=5$	-186.2	-176.6	-174.6	-195.8	-170.3	-169.3
	$K=10, M=5$	-698.2	-552.8	-536.6	-643.6	-526.8	-522.5
	$K=M=10$	-785	-529.0	-519.5	-694.8	-542.7	-539.8

In all of cases considered, the MVBNL-TR model performs better than the other models. With clear advantages to MVBNL-TR's performance in other scenarios, this is further evidence that the model is achieving desirable performance when applied to the appropriate data. Lastly, co-movement AUC and conditional AUC need to be discussed. For each of the $K \cdot (K-1)/2$ co-movement pairs, the mean and standard deviation for the AUC estimates are provided in Table 2.9.

Table 2.9: Co-movement AUC

	M1		M2		TR	
Pair	Mean	SD	Mean	SD	Mean	SD
Train (1,2)	0.512	0.004	0.594	0.007	0.600	0.007
(1,3)	0.545	0.002	0.587	0.007	0.588	0.007
(2,3)	0.510	0.004	0.537	0.012	0.538	0.012
(1,4)	0.597	0.004	0.701	0.012	0.703	0.012
(2,4)	0.527	0.003	0.585	0.007	0.586	0.007
(3,4)	0.534	0.002	0.554	0.009	0.555	0.010
(1,5)	0.627	0.007	0.775	0.030	0.781	0.031
(2,5)	0.530	0.003	0.609	0.006	0.615	0.006
(3,5)	0.537	0.002	0.568	0.009	0.571	0.008
(4,5)	0.611	0.005	0.676	0.009	0.676	0.009
Test (1,2)	0.499	0.005	0.567	0.008	0.576	0.008
(1,3)	0.537	0.004	0.565	0.008	0.567	0.008
(2,3)	0.496	0.005	0.518	0.013	0.519	0.013
(1,4)	0.588	0.006	0.688	0.015	0.691	0.015
(2,4)	0.503	0.005	0.563	0.008	0.567	0.008
(3,4)	0.508	0.004	0.513	0.014	0.510	0.016
(1,5)	0.615	0.009	0.761	0.034	0.768	0.036
(2,5)	0.502	0.005	0.582	0.007	0.591	0.008
(3,5)	0.507	0.004	0.527	0.012	0.520	0.013
(4,5)	0.604	0.008	0.662	0.012	0.661	0.012

Co-movement AUC is difficult to discuss concisely. For K stocks there will be $K \cdot (K-1)/2$ pairs of outcomes for which co-movement AUC is calculated. While the co-movement AUC can be briefly summarized for $K = 5$, the best way to summarize for the case where $K = 10$ is to provide some summary information about the average co-movement AUCs.

For $K = M = 5$, there appears to be a consistent decrease in the average co-movement AUC for the test data compared to the training data, as well as an increase in the standard deviation of the co-movement AUC for marginal model 1. Moreover, most of the co-movement AUCs are close to 0.5, with those that are greater only being around 0.6 to 0.62. Having such

low co-movement AUC performance indicates that the first marginal model does not perform very well at classifying if a pair of stocks co-move. For marginal model 2, the same behavior is observed, except that some pairs have higher co-movement AUC. Most of the co-movement pairs, still have fairly small AUC. For the joint model, the same behavior is again observed. The joint MVBNL-TR model are essentially the same. The standard deviation stays fairly stable, with some small increases or decreases seen between models. When the rank is increased, the maximum AUC decreases for both data sets for model 1, but increases for both marginal model 2 and the joint model. For marginal model 2, the mean of all $K \cdot (K-1)/2$ mean co-movement AUCs is 0.660, with the maximum being 0.850 for the training set. For the test set the mean is 0.614 with the maximum mean AUC being 0.828. For the joint model the mean of mean AUCs is 0.662, with the max being 0.8529, whereas the mean for the test set is 0.613 with the max being 0.8301.

Table 2.10: Mean of Mean Co-movement AUC

	$R=1$			$R=3$		
	M1	M2	TR	M1	M2	TR
Train $K=10, M=5$	0.545	0.675	0.678	0.527	0.674	0.674
$K=M=10$	0.552	0.727	0.729	0.534	0.720	0.719
Test $K=10, M=5$	0.532	0.654	0.646	0.513	0.615	0.616
$K=M=10$	0.533	0.700	0.700	0.513	0.632	0.632

Table 2.11: Maximum of Mean Co-movement AUC

	$R=1$			$R=3$		
	M1	M2	TR	M1	M2	TR
Train $K=10, M=5$	0.636	0.854	0.858	0.573	0.867	0.866
$K=M=10$	0.645	0.888	0.893	0.588	0.902	0.902
Test $K=10, M=5$	0.624	0.839	0.841	0.544	0.824	0.823
$K=M=10$	0.623	0.868	0.864	0.550	0.836	0.837

For $K=10, M=5$, consider Tables 2.10 and 2.11. In each case, the maximum and mean of the average co-movement AUC increase from marginal model 1 to marginal model 2, and again from marginal model 2 to the joint model, when $R=1$. When $R=3$, the behavior is very similar. The pairwise maximum and mean of mean co-movement AUC increase from marginal model 1 to 2, but the maximum co-movement AUC decreases slightly from model 2 to the joint model for the test data.

When $K=M=10$, $R=1$, there is a consistent increase from model 1 to the joint model for the pairwise maximum and mean co-movement AUC for the training data, whereas for model

2 and the joint model the co-movement AUC is essentially the same. For $K = M = 10$, $R = 3$, the pairwise maximum and mean AUC increase for the training data from model 1 to the joint model. For the test data, there is an increase in both the pairwise maximium and mean from model 1 to model 2, but essentially no change between model 2 and the joint model.

Overall, this metric seems to indicate that none of the models are that good at classifying co-movement outcomes. While the joint model being proposed has some improvements for the training data, marginal model 2 performs comparably for the test data. However, for all of these models, there are many co-movement pairs where the AUC is close to or less than 0.5, an indicator that the model is doing a poor job of classifying the response. What remains to be seen is how the models perform when classifying each stock outcome using conditional probability (see equation 2.23).

Conditional AUC is also considered in Tables 2.12-2.14. For both rank approximations, there is an increase in the mean conditional AUC for all stocks when model 2 is used to fit the data instead of model 1. Some however, see only a small increase, such as the third stock. The standard deviation of the conditional AUC increases for most stocks in the second marginal model and the joint model, but decreases for marginal model 1 for all stocks. Regardless of rank, using the joint model results in an increased mean conditional AUC for all stocks, and the standard deviation also remains stable, with only small changes.

Table 2.12: Conditional AUC, K = M = 5

	$R=1$						$R=3$					
	M1		M2		TR		M1		M2		TR	
Stock	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	0.681	0.018	0.829	0.038	0.853	0.038	0.588	0.005	0.913	0.077	0.923	0.070
2	0.498	0.013	0.604	0.026	0.641	0.024	0.516	0.004	0.673	0.023	0.693	0.022
3	0.497	0.013	0.502	0.054	0.520	0.057	0.495	0.005	0.497	0.056	0.510	0.059
4	0.584	0.007	0.651	0.021	0.672	0.021	0.516	0.005	0.553	0.042	0.591	0.037
5	0.625	0.010	0.841	0.041	0.865	0.041	0.591	0.007	0.759	0.029	0.792	0.031

For a larger parameter space such as the case where $K = 10$ and $M = 5$, the behavior is similar. From model 1 to model 2, there is an increase in the mean conditional AUC for most stocks. With regards to the joint model, it performs superior to model 2 or ties in in every case considered. The standard deviation increases from marginal model 1 to marginal model 2, but stays relatively stable when the joint model is fit, with some slight reductions being observed. This is true for both levels of R that are considered.

Table 2.13: Conditional AUC, K = 10, M = 5

	$R=1$						$R=3$					
	M1		M2		TR		M1		M2		TR	
Stock	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	0.642	0.009	0.885	0.028	0.914	0.026	0.569	0.003	0.921	0.069	0.929	0.055
2	0.598	0.004	0.798	0.012	0.815	0.010	0.551	0.003	0.845	0.040	0.848	0.030
3	0.530	0.004	0.702	0.012	0.710	0.015	0.514	0.002	0.720	0.024	0.720	0.019
4	0.500	0.008	0.623	0.026	0.623	0.036	0.501	0.002	0.568	0.037	0.570	0.042
5	0.574	0.008	0.737	0.015	0.836	0.016	0.517	0.004	0.603	0.035	0.715	0.018
6	0.532	0.005	0.612	0.030	0.689	0.019	0.533	0.003	0.531	0.051	0.597	0.036
7	0.564	0.004	0.689	0.014	0.721	0.014	0.533	0.001	0.556	0.041	0.594	0.035
8	0.562	0.003	0.708	0.010	0.744	0.010	0.536	0.001	0.599	0.032	0.643	0.025
9	0.588	0.003	0.822	0.015	0.846	0.013	0.517	0.003	0.764	0.026	0.782	0.020
10	0.573	0.003	0.897	0.032	0.916	0.028	0.533	0.002	0.901	0.061	0.909	0.048

For the cases where $K = M = 10$, there is again an increase in the mean conditional AUC when considering the results of marginal model 2 against marginal model 1, for both levels of R. For $R = 1$, there is a universal increase in mean AUC from model 2 when considering the joint model, but stocks 2, 3, and 5 see a noticeable decrease for $R = 3$. Stock 8 decreases is roughly equivalent between model 2 and the joint model. Again, there is an increase in the magnitude of the standard deviation of the conditional AUC for each stock when considering model 2 against model 1. However, when considering the joint model against model 2, the standard deviations are relatively stable. This is true for both levels of R considered.

Table 2.14: Conditional AUC, K = M = 10

	$R=1$						$R=3$					
	M1		M2		TR		M1		M2		TR	
Stock	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	0.653	0.011	0.939	0.030	0.950	0.026	0.572	0.004	0.935	0.063	0.945	0.070
2	0.596	0.004	0.880	0.019	0.886	0.015	0.551	0.003	0.888	0.046	0.884	0.048
3	0.542	0.003	0.774	0.012	0.778	0.012	0.505	0.002	0.789	0.025	0.771	0.026
4	0.500	0.008	0.635	0.039	0.635	0.050	0.504	0.002	0.630	0.031	0.603	0.037
5	0.559	0.006	0.772	0.015	0.860	0.013	0.518	0.004	0.611	0.038	0.599	0.040
6	0.526	0.005	0.651	0.037	0.711	0.026	0.531	0.002	0.531	0.065	0.531	0.064
7	0.569	0.003	0.727	0.018	0.747	0.017	0.528	0.002	0.576	0.044	0.567	0.048
8	0.587	0.004	0.784	0.013	0.804	0.012	0.547	0.002	0.628	0.032	0.627	0.034
9	0.597	0.003	0.897	0.021	0.908	0.018	0.524	0.002	0.787	0.024	0.801	0.029
10	0.586	0.004	0.949	0.034	0.955	0.028	0.530	0.002	0.901	0.051	0.918	0.060

With all of this information under consideration, there are some clear benefits to the multivariate binomial logit model with tensor regression. In the majority of cases it outperforms the marginal models in terms of multi-class AUC, estimated probability log-likelihood, count of instances where the observed estimated probability is the maximum estimated probability. It also performs well in terms of conditional AUC. None of the models do particularly well at properly classifying co-movement as a response, but this is not the primary goal of any of the models.

The MVBNL-TR model also does fairly well in contrast to the MVBNL model. In several cases, there are decreases in bias and standard deviation. Moreover, as the MVBNL-TR performs better on test data in the majority of cases and has a less dense parameter space and comparable AIC, it is a more desirable model for data which has an inherent structure that can make use of the tensor regression framework. To further examine the viability of this model, its performance on real stock data needs to be assessed.

2.7 Applications

2.7.1 High Market Capitalization Stock Example

To examine the performance of these models in the real world, we considered a five stock and ten stock price case over a period of 1500 trading days. For the five-stock case, the stocks considered are Apple, Microsoft, Google, Amazon, and Tesla. For the ten-stock case, Meta, Taiwan Semiconductor (TSMC), NVIDIA, Visa, and United Health Group Inc. are also considered. The window of data considered ends on March 24th, 2022. Other time-frames which did not include the COVID-19 Pandemic were assessed outside of the scope of this analysis, but there was not an apparent drawback to using data from this more volatile period. A 50/50 randomized split is performed on the data, so both the training and test data sets have 750 joint response outcomes. In this discussion, marginal models 1 and 2 are compared to the joint MVBNL-TR model. The metrics used for comparison are standard multi-class AUC, log-likelihood of observed estimated probability, the count of cases where the observed estimated probability matches the maximum probability, as well as co-movement and conditional AUC. Rank-one and rank-three approximations are considered for all parameter settings. The results for the multi-class AUC are given in Table 2.15.

Table 2.15: AUC, Total Days = 1500

		$R=1$			$R=3$		
		M1	M2	TR	M1	M2	TR
Train	$K=M=5$	0.571	0.588	0.607	0.571	0.614	0.629
	$K=10, M=5$	0.585	0.642	0.663	0.585	0.692	0.728
	$K=M=10$	0.594	0.685	0.686	0.594	0.739	0.773
Test	$K=M=5$	0.518	0.527	0.508	0.518	0.522	0.493
	$K=10, M=5$	0.543	0.514	0.518	0.543	0.529	0.510
	$K=M=10$	0.555	0.532	0.542	0.555	0.545	0.519

While the joint model displays increases in multi-class AUC over its marginal counterparts on the training data, it has poor performance on the test data. However, all the models proposed have low multi-class AUC for the test data. Despite the poor performance for AUC, the joint model has higher count of observed estimated probabilities matching the maximum estimated probabilities than the marginal models. This holds true for both the training and test data. Count also improves for the joint model when the rank is increased. The results for the count of observed maximum estimated probabilities are provided in Table 2.16.

Table 2.16: Count$\left(\hat{P}(Y_i = y_i|\mathbf{X}_i) = \max_{s_i \in S}(\hat{P}(Y_i = s_i)|\mathbf{X}_i)\right)$, T = 1500

		$R=1$			$R=3$		
		M1	M2	TR	M1	M2	TR
Train	$K=M=5$	102	83	160	102	94	172
	$K=10, M=5$	26	16	73	26	25	86
	$K=M=10$	17	11	71	17	21	82
Test	$K=M=5$	83	80	142	83	64	142
	$K=10, M=5$	22	7	61	22	15	50
	$K=M=10$	20	8	62	20	8	46

In every case considered, the joint MVBNL-TR model outperforms the marginal models in terms of the estimated probability log-likelihood. For each combination of parameter space, data set, and rank approximation, the log-likelihood for the joint model is consistently less negative. The log-likelihood is provided in Table 2.17.

Table 2.17: Estimated Probability Log-likelihood, T = 1500

		$R=1$			$R=3$		
		M1	M2	TR	M1	M2	TR
Train	$K=M=5$	-2569.7	-2548	-2169.7	-2569.7	-2524.7	-2138.7
	$K=10, M=5$	-5140.2	-5053.1	-4158.9	-5140.2	-4925.9	-4033.8
	$K=M=10$	-5101.3	-5007.2	-4133.1	-5101.3	-4792.9	-3968.8
Top 25%	$K=M=5$	-1695.7	-1679.2	-1310.8	-1695.7	-1668.6	-1298.2
	$K=10, M=5$	-4495.4	-4417.4	-3529.1	-4495.4	-4309.8	-3436.4
	$K=M=10$	-4459	-4378.4	-3517.3	-4459	-4204.8	-3392.3
Test	$K=M=5$	-2593.1	-2616.9	-2287.1	-2593.1	-2634.7	-2311
	$K=10, M=5$	-5174.3	-5277.9	-4490.7	-5174.3	-5432	-4703.4
	$K=M=10$	-5191.6	-5278	-4556.9	-5191.6	-5521.7	-4973.7
Top 5%	$K=M=5$	-958.5	-966.7	-685.7	-958.5	-979.1	-688.7
	$K=10, M=5$	-2664.6	-2721.8	-1926.3	-2664.6	-2815.5	-2017
	$K=M=10$	-2668.2	-2724	-1955	-2668.2	-2854.1	-2149.5

For co-movement AUC, first consider the case where $K=M=5$, detailed below in Table 2.18. In this parameter space, none of the proposed models perform well at classifying co-movement outcomes. Regardless of rank R, the co-movement AUC is low for all stock pairs, for both the training and test sets. Poor performance is expected, as similarly poor performance was seen for the simulated data. For larger parameter spaces, co-movement AUC increases slightly but overall performance is still poor.

Table 2.18: Co-movement AUC, K = M = 5

	$R=1$			$R=3$		
Pair	M1	M2	TR	M1	M2	TR
Train (1,2)	0.556	0.463	0.466	0.556	0.534	0.533
(1,3)	0.502	0.499	0.515	0.502	0.534	0.525
(2,3)	0.532	0.476	0.499	0.532	0.523	0.502
(1,4)	0.518	0.485	0.463	0.518	0.539	0.517
(2,4)	0.505	0.478	0.468	0.505	0.508	0.459
(3,4)	0.538	0.481	0.492	0.538	0.522	0.493
(1,5)	0.483	0.472	0.504	0.483	0.559	0.547
(2,5)	0.485	0.510	0.514	0.485	0.508	0.526
(3,5)	0.539	0.492	0.516	0.539	0.527	0.547
(4,5)	0.502	0.515	0.509	0.502	0.536	0.539
Test (1,2)	0.491	0.487	0.512	0.491	0.485	0.476
(1,3)	0.502	0.509	0.516	0.502	0.517	0.482
(2,3)	0.527	0.482	0.442	0.527	0.460	0.483
(1,4)	0.506	0.523	0.507	0.506	0.516	0.496
(2,4)	0.488	0.517	0.520	0.488	0.512	0.507
(3,4)	0.488	0.487	0.459	0.488	0.522	0.489
(1,5)	0.486	0.511	0.511	0.486	0.511	0.501
(2,5)	0.526	0.520	0.521	0.526	0.521	0.503
(3,5)	0.504	0.498	0.494	0.504	0.479	0.478
(4,5)	0.502	0.497	0.520	0.502	0.503	0.488

With regards to conditional AUC, both of the marginal models perform poorly. However, the joint model consistently performs better. For each stock, parameter space, and rank considered, the joint model displays strong performance at classifying stock price increases when conditioning on other stocks. This performance is noted in Tables 2.19 through 2.21.

Table 2.19: Conditional AUC, K=M=5

	$R=1$			$R=3$		
Stock	M1	M2	TR	M1	M2	TR
K = 1	0.521	0.523	0.790	0.521	0.525	0.786
K = 2	0.535	0.528	0.827	0.535	0.517	0.824
K = 3	0.514	0.505	0.819	0.514	0.496	0.810
K = 4	0.497	0.549	0.822	0.497	0.540	0.816
K = 5	0.513	0.481	0.641	0.513	0.492	0.638

Table 2.20: Conditional AUC, K=10, M=5

	$R=1$			$R=3$		
Stock	M1	M2	TR	M1	M2	TR
K = 1	0.521	0.516	0.810	0.521	0.533	0.799
K = 2	0.535	0.529	0.850	0.535	0.508	0.839
K = 3	0.514	0.475	0.853	0.514	0.500	0.826
K = 4	0.497	0.533	0.833	0.497	0.518	0.816
K = 5	0.513	0.467	0.665	0.513	0.493	0.654
K = 6	0.501	0.538	0.809	0.501	0.501	0.792
K = 7	0.540	0.557	0.790	0.540	0.549	0.773
K = 8	0.549	0.542	0.800	0.549	0.519	0.788
K = 9	0.537	0.528	0.784	0.537	0.518	0.769
K = 10	0.557	0.531	0.658	0.557	0.524	0.634

Table 2.21: Conditional AUC, K=M=10

	$R=1$			$R=3$		
Stock	M1	M2	TR	M1	M2	TR
K = 1	0.519	0.528	0.803	0.519	0.553	0.784
K = 2	0.510	0.515	0.849	0.510	0.508	0.825
K = 3	0.525	0.519	0.845	0.525	0.505	0.812
K = 4	0.519	0.547	0.830	0.519	0.569	0.812
K = 5	0.519	0.531	0.674	0.519	0.493	0.642
K = 6	0.505	0.505	0.805	0.505	0.506	0.780
K = 7	0.546	0.529	0.780	0.546	0.556	0.764
K = 8	0.545	0.550	0.796	0.545	0.546	0.778
K = 9	0.535	0.495	0.790	0.535	0.525	0.759
K = 10	0.574	0.543	0.656	0.574	0.562	0.634

Overall, the models perform as expected for real data for most of the metrics considered. However, the joint model performed worse than expected with regards to multi-class AUC. While seeing slightly weaker performance overall would not be unexpected, all models perform considerably worse on test data than their simulated counterparts. This discrepancy in performance between simulated and real data merits further study.

2.7.2 Agricultural Sector Example

In an effort to examine how this model performs for a sector-specific example, stocks in the agricultural sector or closely related sectors are considered. A 1500-day window with a 50/50 randomized training/test data split is considered. For the five-stock case, the stocks considered were Archer-Daniels-Midland Company, Bunge Limited, Caterpillar Inc., Deere & Company,

and The Scotts Miracle-Gro Company. When considering a ten-stock case, FMC Corporation, Calavo Growers, Inc., The Mosaic Company, Tyson Foods, Inc., and China Green Agriculture, Inc. are also considered. The multi-class AUCs for the various parameter spaces considered for this example can be seen below.

Table 2.22: AUC, Total Days = 1500

		$R=1$			$R=3$		
		M1	M2	TR	M1	M2	TR
Train	$K=M=5$	0.532	0.566	0.558	0.532	0.584	0.597
	$K=10, M=5$	0.580	0.647	0.657	0.580	0.693	0.710
	$K=M=10$	0.625	0.681	0.687	0.625	0.749	0.762
Test	$K=M=5$	0.501	0.508	0.506	0.501	0.524	0.521
	$K=10, M=5$	0.495	0.493	0.507	0.495	0.498	0.532
	$K=M=10$	0.471	0.492	0.511	0.471	0.486	0.509

For all of these combinations of parameter settings, the training set AUC is highest for the tensor regression model. Even with a more sector-specific example, the test data performance is still poor for all models. For the five-stock case, the test AUC for the tensor regression method performs poorer than the second marginal model. With the larger parameter spaces, however, the tensor regression model does demonstrate a slight advantage over the marginal models.

We can also consider different co-movement metrics than previously used. As seen in the previous real data case, co-movement AUC is not the most insightful metric. However, we can also consider other metrics such as co-movement MSE. The number of co-movement pairs changes with the number of stocks considered. Thus, it is better to look at the mean of the $(K \cdot (K-1)/2)$ co-movement MSEs at each parameter space setting. The mean of MSEs can be seen below.

Table 2.23: Average co-movement MSE, Total Days = 1500

		$R=1$			$R=3$		
		M1	M2	TR	M1	M2	TR
Train	$K=M=5$	0.250	0.249	0.224	0.250	0.248	0.223
	$K=10, M=5$	0.250	0.249	0.232	0.250	0.247	0.231
	$K=M=10$	0.250	0.249	0.232	0.250	0.244	0.231
Test	$K=M=5$	0.250	0.249	0.219	0.250	0.248	0.219
	$K=10, M=5$	0.250	0.250	0.230	0.250	0.249	0.231
	$K=M=10$	0.250	0.250	0.231	0.250	0.251	0.235

The average of co-movement MSEs across the (k,κ) pairs is fairly stable across the various parameter settings and models. The joint model has the lowest MSE out of any of the models for each combination of parameter settings. For the training data, increasing K increases the average MSE but that increasing M will result in a slight reduction to the average MSE. For the test data, this behavior is less consistent, with only model 1 seeing some reduction in average MSE when M is increased to 10. This reduction is minute. For the other models at both ranks, the average MSE increases as the parameter space grows larger. A concordance metric for co-movement is also considered. Concordance estimates can be seen below.

Table 2.24: Co-movement Concordance, Total Days = 1500

		$R=1$			$R=3$		
		M1	M2	TR	M1	M2	TR
Train	$K=M=5$	1.012	0.940	1.638	1.012	0.997	1.636
	$K=10, M=5$	4.600	5.156	9.282	4.600	5.455	9.203
	$K=M=10$	4.596	4.804	9.254	4.596	5.000	9.056
Test	$K=M=5$	0.992	0.915	1.553	0.992	0.975	1.547
	$K=10, M=5$	4.377	4.895	8.835	4.377	5.235	8.733
	$K=M=10$	4.484	4.660	8.806	4.484	4.884	8.546

As seen above, there is a clear benefit to the tensor regression model in terms of concordance, with the metric showing a significant increase in performance for larger parameter spaces. It seems to be the case for the training data that for both rank settings chosen, the concordance will increase when the number of stocks increases, but will decrease slightly when the look-back window increases in size. This behavior is also seen in the test data set, except for the first marginal model, where concordance strictly increases.

In the best scenario observed, the tensor regression model has concordance values on the test set that are 19.63% of the theoretical maximum. In the same scenario, the rank-one case for $K=10$ and $M=5$, the marginal models perform considerably worse, with marginal model 2 only attaining approximately 10.8% of its theoretical maximum.

From examining a stock-specific example, further benefits of the tensor regression model can be seen. By using more related stocks, some slight improvement in the multi-class AUC can be observed. Though test AUC is not very strong, it demonstrates better performance than the other models in the more complex cases. This behavior was not present for the more high market capitalization case, where the stocks were not necessarily closely related.

In addition to seeing these improvements, the tensor regression model also demonstrates stronger performance than the marginal models on different co-movement metrics than what

was considered previously. While the advantages demonstrated for co-movement MSE are relatively small, the improvements for concordance are very strong. Strong performance for the tensor regression model on the test data is also encouraging,

2.8 Conclusions and Areas for Future Study

After many different model proposals, each with their own advantages and limitations, recent strides in the field have resulted in intriguing and useful models such as the multivariate multinomial model proposed by Bel and Paap (2014) which avoids many of the limitations of prior methods, and is also a general framework which can be applied to various sorts of multinomial choice problems. With many developments in data collection in recent years, the need for models which preserve structure in the data has been met by various researchers. Tensor regression developments provide one of the most useful frameworks for future research. The new methods being proposed have made great improvements, with computationally feasible low rank approximations performing well at capturing the associations in the data.

Following up on avenues for future study put forth in tensor regression and multinomial choice modeling, one extension considered is a multivariate multinomial logit tensor regression model. Such a model permits matrix covariates akin to the framework put forth by Chang et al. (2021), with intercept and association parameters being stored in a vector covariate. Some departures from this method are made, as the algorithm that used to update intercept and association parameters follows an alternating logistic regression (ALR) framework to reflect the incorporation of a MV-MNL logit model.

The motivating example for this model is stock data. With a binary outcome reflecting whether a stock price has increased or decreased from the preceding day's value, the joint behavior of stock prices over time is modeled. Matrix covariates of financial data allow for more data to be considered when predicting stock price. Such covariates preserve structure present in the data that otherwise would not be utilized effectively.

The multivariate binomial logit model and its tensor regression extension are comparable on the metrics considered, and the tensor regression model outperforms the MVBNL model in most test data cases for multi-class AUC. This performance, coupled with the reasonably low AIC for the tensor regression model, demonstrates a strong advantage when using the tensor regression model for the appropriate setting. In addition, the MVBNL-TR model shows strong performance against the marginal models for the log-likelihood of estimated probability and the count of cases where observed and maximum estimated probability match. The tensor regression model also demonstrates good performance with classifying each individual stock when conditioning on other stocks. None of the proposed models do well at classifying co-

movement of stocks. Classifying co-movement is not the primary focus of this model, so poor performance on this metric is not an indicator of poor overall performance. The simulation results solidify this model as a strong candidate for situations which assume the joint inter-dependency of the different stocks.

When considering the real data applications of this model, most of this positive simulated performance is present when considering real stock data. For instance, in Tables 2.19-2.21, Conditional AUC shows decent performance. However, the tensor regression model performs worse than anticipated on test data for multi-class AUC. It is currently unclear why this large discrepancy is present. Although not discussed here, other time frames were considered, and similar performance was seen from all models. Thus, this does not seem to be poor performance due to economic troubles. This difference in performance needs to be looked into in more detail.

For a sector-specific case concerning agricultural sector related stocks, there are some slight improvements to the multi-class AUC performance. While in most cases the AUC is slightly lower in comparison to the high market capitalization stock case, the tensor regression model performs better on the test data than the marginal models for the ten-stock cases. In addition to this, the tensor regression model has lower average co-movement MSE and higher concordance than the marginal models. This behavior demonstrates that this model has some benefits over marginal models with regards to its ability to accurately predict if stock pairs will co-move. Other extensions to this model are also being considered. In the next chapter, a neural network extension to this model framework is going to be considered.

CHAPTER

3

ADVANTAGES AND LIMITATIONS OF AN ALTERNATING NEURAL NETWORK LOGISTIC REGRESSION MODEL

3.1 Introduction

As mentioned in the discussion of future work for the multivariate binomial logit tensor regression (MVBNL-TR) model, a neural network extension from that model framework is considered. The neural network extension being proposed will introduce several changes into the model framework, both in terms of how the update steps in the algorithm are designed and in terms of how metrics are computed. The new algorithm being proposed follows an alternating update framework, and consists of a neural network update step followed by a logistic regression update step. This new alternating model will be referred to an alternating neural network logistic regression model (ANN-LR). In the examples discussed here, a convolutional neural network is used, but other network structures could be considered in practice.

The goal of this discussion is to determine if using neural networks in an attempt to better capture the associations between stocks and the structure of the relative price change look-back window results in an improvement on the several model metrics considered for the MVBNL-TR

model. Necessary background information on existing work and basic model terminology will be provided in section 3.2. Section 3.3 will detail the algorithm being proposed, as well as the structure of the neural network updates considered in the neural network update step. Section 3.4 will include a discussion of how the metrics used for model assessment change for this new model. Section 3.5 will focus on the results from two applications using real financial data, and compares the results in both to results on the same data from the multivariate binomial logit tensor regression model. Lastly, section 3.6 will discuss conclusions and areas for further study.

3.2 Background

In recent years, work has been done in attempts to improve discrete choice models with machine learning methods. One approach put forth by Sifringer et al. (2020) divides the process of estimating choice models into a two-part process, one part informed by knowledge and the other part driven by data. The goal of this model is to increase the predictive power without loss of interpretability. (Sifringer et al. 2020, p 236) In the examples provided in this paper, the focus is on travel data, but the authors note that this model is general and could be expanded to other data applications that followed a multinomial logit framework (MNL). (Sifringer et al. 2020, p 252) Other models, such as the model proposed by Arkoudi et al. (2023) also focus on travel data applications.Note that this model is distinct from the model we are proposing, as it is only modeling one potential choice, rather than a number of distinct choices.

Following a MNL approach, the goal of the model is to predict the choice outcome given a set of alternatives, while being able to understand what leads to a specific choice. The MNL model has been a foundational work in choice analysis. McFadden (1974) suggests that an individual will aim to maximize the utility relative to their choice. Utility consists of two components, a deterministic utility V_{in}, and a random component ε_{in} which captures uncertainty that stems from the impossibility to fully model the choice context. Sifringer et al. propose the following model: The utility that individual n will associate with alternative i from a set of choices $\mathscr{C}_n$ is

$$U_{in} = V_{in} + \varepsilon_{in}, \tag{3.1}$$

$$V_{in} = \sum_{d} \beta_d \cdot x_{din} \tag{3.2}$$

Here $\boldsymbol{\beta}$ are the parameters associated with various explanatory variables $\boldsymbol{x} \in \mathscr{X}$. These variables describe observed attributes of the choice alternative, and characteristics of the individual. The standard MNL assumption is that error terms are independently and identically

distributed, following an Extreme Value Type I distribution with location parameter 0 and scale parameter 1. In other words, $\varepsilon_{in} \overset{i.i.d.}{\sim} EV(0,1)$. For this model, the probability of individual n selecting alternative i is

$$P_n(i) = \frac{e^{V_{in}}}{\sum_{j \in \mathscr{C}_n} e^{V_{jn}}}. \tag{3.3}$$

Parameters $\boldsymbol{\beta}$ are estimated by maximizing the log-likelihood function $\mathscr{L}$:

$$\mathscr{L} = \sum_{n=1}^{N} \sum_{i \in \mathscr{C}_n} y_{in} \log(P_n(i)), \tag{3.4}$$

where y_{in} is the observed choice variable. It is 1 if the choice is alternative i for individual n, and 0 otherwise. (Sifringer et al. 2020, p 238-239)

The process proposed by Sifringer et al. (2020) implementing the multinomial logit model as a neural network is insightful, and provides useful terminology vital for understanding neural network structure. As such, the relevant information is repeated here. A neural network is a function which will map an input x to an output U through several intermediate steps known as hidden layers $\boldsymbol{h}^{(j)}$.

$$U = \boldsymbol{h}^{(L)}(\boldsymbol{q}^{(L-1)}), \tag{3.5}$$

$$\boldsymbol{q}^{(j)} = \boldsymbol{h}^{(j)}(\boldsymbol{q}^{(j-1)}), \forall\, j = 1, \ldots, L, \tag{3.6}$$

such that $\boldsymbol{q}^{(0)} = \boldsymbol{x}$ and L is the last layer. The model proposed by Sifringer et al. (2020) utilizes a convolutional neural network (CNN). As defined in the paper, a CNN has weights in the shape of a filter to connect layer $\boldsymbol{h}^{(j)}$ to the next by applying a convolution. To briefly define a convolution, consider an input t with an output $x(t)$. Suppose the estimate is somewhat noisy, and a weighted average is desirable. With a weighting function $w(a)$, a new function s provides a smoothed estimate of x.

$$s(t) = \int x(a) w(t-a) da \tag{3.7}$$

Known as a convolution[1], this operation is usually denoted as follows s(t) = (x*w)(t). The value of the ith neuron (i.e. node, unit) in the next layer $(j+1)$ is expressed as

$$h_i^{(j+1)} = g\left(\sum_{k=0}^{d} h_{(s \cdot i+k)}^{(j)} \beta_k^{(j)} + \alpha_i^{(j)} \right), \tag{3.8}$$

where $\{\beta_1, \ldots, \beta_d\} = \boldsymbol{\beta}$ is the $(1 \times d)$ filter, s is the stride[2] of the convolution, α_i is a bias term, and $g(\cdot)$ is an activation function. For the purpose of the MNL model, one layer is used, the

[1] (Goodfellow et al. 2016, p 345)

[2] Definition in (Chollet 2017, p 127)

activation function is simply x, and the stride is s to d. Then probabilities are obtained with a softmax activation layer, such that

$$(\boldsymbol{\sigma}(\mathbf{V}_n))_i = \frac{e^{V_{in}}}{\sum_{j \in \mathscr{C}_n} e^{V_{jn}}} \tag{3.9}$$

whereas the output of the network is the categorical cross-entropy, such that

$$H_n(\boldsymbol{\sigma}, \mathbf{y}_n) = -\sum_{i \in \mathscr{C}_n} y_{in} \log(\sigma_i(\mathbf{V}_n)). \tag{3.10}$$

Minimizing equation 3.10 is equivalent to maximizing equation 3.4. This is how the reparametrization is performed. (Sifringer et al. 2020, p 239). This allows the model to remain reasonably interpretable, while greatly improving the predictability (Sifringer et al. 2020, p 253). This model demonstrates the potential usefulness of implementing neural networks to improve the predictability. However, how such a model would perform on the same data framework with the same metrics of interest as proposed in chapter 2 is unknown. This model also does not directly translate to a multivariate multinomial network, motivating the need for additional work.

3.3 Methods

The alternating neural network logistic regression model (ANN-LR) we are proposing shares some similarities with the multivariate binomial tensor regression model (MVBNL-TR). ANN-LR will consider the same types of data, and will also update parameters with alternating update steps. The tensor regression algorithm consists of three alternating steps. Each step is a logistic regression for either $\mathbf{B}_1, \mathbf{B}_2$, or $\boldsymbol{\beta}$. The $\boldsymbol{\beta}$ updates the α_k and $\boldsymbol{\psi}$ parameters. With the neural network method, several key changes are made. Only two update steps are considered for ANN-LR; a neural network update step and a logistic regression step. Furthermore, α_k parameters are not considered in this model. Therefore, the logistic regression step is only an update for $\boldsymbol{\psi}$. As such, the design matrix used for logistic regression will change.

The goal of the neural network step is to estimate network parameters $\boldsymbol{\theta}_k$. The neural network step also consists of K substeps. In each of these substeps, the responses for stock k are modeled using the same data tensor $\mathscr{X}$ used in the tensor regression model. Recall that the slices of this tensor $\mathbf{X}_i$ are $K \times M$ look-back windows of relative stock price change. Prior to performing the neural network step, an initial estimate of the association parameters $\boldsymbol{\psi}^{(0)}$ is computed, following the initial estimates used in the tensor regression model. $\boldsymbol{\psi}^{(0)} = (\boldsymbol{\Gamma}_{\psi}^{\top} \boldsymbol{\Gamma}_{\psi})^{-1} \boldsymbol{\Gamma}_{\psi}^{\top} \mathbf{y}$, where $\mathbf{y}$ is the $(T \cdot K) \times 1$ response matrix of all stock change outcomes. $\boldsymbol{\Gamma}_{\psi}$ is a submatrix of the design

matrix utilized in the tensor regression model, consisting of the columns of the design matrix relevant to the association parameters.

For each stock, a model is trained using the training data. The loss function used in training is the negative composite log-likelihood (NCLL) for stock k. The form of this NCLL for each network is

$$-\left(\sum_{i=1}^{T}\left(y_{ik}\cdot\log\left(\frac{\exp(Z_{ik})}{1+\exp(Z_{ik})}\right)+(1-y_{ik})\cdot\log\left(1-\frac{\exp(Z_{ik})}{1+\exp(Z_{ik})}\right)\right)\right) \tag{3.11}$$

where

$$Z_{ik}=y_{pred}+\sum_{l\neq k}y_{il}\psi_{kl} \tag{3.12}$$

Note that the Z_{ik} term differs from the Z_{ik} term in the tensor regression model (2.6.2). y_{pred} is the predicted output calculated by the network. After the network has been fully trained, a quantity $f_k(\mathbf{X}_i,\hat{\boldsymbol{\theta}}_k)$ is computed. This quantity is a $T\times 1$ matrix of the output from the network with its final weights after training. The ith entry for this quantity will be a scalar output of the model using input data $\mathbf{X}_i$. After training all k networks, these quantities are stored in a $T\times K$ matrix.

For the logistic regression step, α_k updates are no longer considered. Previously, the augmented design matrix included a submatrix $\boldsymbol{\Gamma}$ used for the logistic updates such that

$$\boldsymbol{\Gamma}^{\top}=\begin{bmatrix}\mathbf{I}_K & & & & \mathbf{I}_K & & & \\ \mathbf{y}'_{11} & \mathbf{y}'_{12} & \cdots & \mathbf{y}'_{1K} & \cdots & \mathbf{y}'_{T1} & \mathbf{y}'_{T2} & \cdots & \mathbf{y}'_{TK}\end{bmatrix} \tag{3.13}$$

No longer relevant, the columns making up the stacked identity matrices in $\boldsymbol{\Gamma}$ are removed. The design matrix used in the update is a new design matrix $\boldsymbol{\Gamma}_{\psi}$, such that

$$\boldsymbol{\Gamma}_{\psi}{}^{\top}=\begin{bmatrix}\mathbf{y}'_{11} & \mathbf{y}'_{12} & \cdots & \mathbf{y}'_{1K} & \cdots & \mathbf{y}'_{T1} & \mathbf{y}'_{T2} & \cdots & \mathbf{y}'_{TK}\end{bmatrix} \tag{3.14}$$

The offset for the logistic step is the matrix $f(\mathbf{X}_i,\hat{\boldsymbol{\theta}}_k^{(j)})=\begin{bmatrix}f_1(\mathbf{X}_i,\hat{\boldsymbol{\theta}}_1^{(j)}) & \ldots & f_K(\mathbf{X}_i,\hat{\boldsymbol{\theta}}_K^{(j)})\end{bmatrix}$. This alternating update process continues until convergence, or until the maximum number of iterations is reached. The convergence criterion used is the minimum of two criteria. The first criterion is maximum of the maximum absolute parameter differences, the criterion used in the tensor regression model. The second condition is the difference in composite log-likelihood between iterations. These conditions are represented in quantities 3.15 and 3.16.

$$\max(\max(|\hat{\boldsymbol{\theta}}_k^{(j+1)}-\hat{\boldsymbol{\theta}}_k^{(j)}|),\max(|\hat{\boldsymbol{\psi}}_{kl}^{(j+1)}-\hat{\boldsymbol{\psi}}_{kl}^{(j)}|)), \tag{3.15}$$

$$\ell(\hat{\boldsymbol{\theta}}^{(j+1)},\hat{\boldsymbol{\psi}}^{(j+1)},\mathbf{Y},\mathbf{X}_i)-\ell(\hat{\boldsymbol{\theta}}^{(j)},\hat{\boldsymbol{\psi}}^{(j)}\mathbf{Y},\mathbf{X}_i) \tag{3.16}$$

Note that $\ell(\ldots)$ in quantity 3.16 is the full composite log-likelihood over all K stocks, and has the form

$$\ell(\hat{\boldsymbol{\theta}}^{(j)}, \hat{\boldsymbol{\psi}}^{(j)}, \mathbf{Y}, \mathbf{X}_i) = \sum_{i=1}^{T}\sum_{k=1}^{K}\left(y_{ik}\cdot\log\left(\hat{P}_{ik}\right) + (1-y_{ik})\cdot\log\left(1-\hat{P}_{ik}\right)\right) \tag{3.17}$$

$$\hat{P}_{ik} = \frac{\exp(Z_{ik})}{1+\exp(Z_{ik})} \tag{3.18}$$

$$Z_{ik} = f_k(\mathbf{X}_i, \hat{\boldsymbol{\theta}}_k^{(j)}) + \sum_{l\neq k} y_{il}\hat{\boldsymbol{\psi}}_{kl}^{(j)} \tag{3.19}$$

Next, a detailed algorithm for the different steps of this alternating process is provided.

3.3.1 Algorithm

At a high level, the algorithm can be summarized as follows:

Algorithm 1 Neural Network in order-3 case implemented with an alternating Logistic Regression step

procedure ANN-LR($\mathbf{Y}, \mathscr{X}, \boldsymbol{\Gamma}_\psi$)
  Data pre-processing to arrive at $\mathbf{Y}, \mathscr{X}, \boldsymbol{\Gamma}_\psi$ ▷ Similar to process for MVBNL-TR
  Reshape $\mathscr{X}$ ▷ Add an "image" channel, $K \times M \times T$ becomes $T \times K \times M \times 1$
  Initialize $\boldsymbol{\psi}^{(0)}$ ▷ $\boldsymbol{\psi}^{(0)} = (\boldsymbol{\Gamma}_\psi^\top \boldsymbol{\Gamma}_\psi)^{-1}\boldsymbol{\Gamma}_\psi^\top \mathbf{y}$
  repeat
  Construct $\sum_{l\neq k} y_{il}\psi_{kl}^{(j)}$
  Initialize empty $T \times K$ matrix for $f(\mathbf{X}_i, \boldsymbol{\theta}_k^{(j)})$
  for k in $1:K$ **do** ▷ $\boldsymbol{\theta}$ Update Step
    $\boldsymbol{\theta}_k^{(j+1)} = NN(\mathbf{y}_k|\mathscr{X}, \mathbf{y}_{l\neq k}, \boldsymbol{\psi}^{(j)})$ ▷ Neural Network update step for stock k
  end for
  $\boldsymbol{\psi}^{(j+1)} = LR(\mathbf{y}|\boldsymbol{\Gamma}_\psi, f(\mathbf{X}_i, \boldsymbol{\theta}_k^{(j)}))$ ▷ Log. reg. update step, $\boldsymbol{\Gamma}_\psi$ data matrix, $f(\mathbf{X}_i, \boldsymbol{\theta}_k^{(j)})$ offsets
  until convergence or max iteration reached
end procedure

After the ANN-LR process is completed, the various metrics are computed. The process for computing these metrics is similar to how it is carried out for the tensor regression model. The changes in how metrics are calculated involves where $\alpha_k + \langle \mathbf{X}_i, \mathbf{B}_k\rangle$ was utilized in computation previously. Now, this expression is replaced by $f_k(\mathbf{X}_i, \boldsymbol{\theta}_k^{(j)})$. These changes will be discussed with the discussion of the metrics.

The process laid out in Algorithm 1 needs to be defined in more detailed pseudo-code. The portion of code below takes pre-processed stock data that is scraped from existing stock data. The input shape required for the neural network step is slightly different from the structure used in the tensor regression algorithm, so this data must be reshaped. Note that W is the name for $\mathbf{\Gamma}$ used in R code. Only the elements of $\mathbf{\Gamma}$ as they pertain to $\boldsymbol{\psi}$ are relevant so `W_psi` must be constructed. $\mathscr{X}$ is represented by `X`, with dimensions $K \times M \times T$.

Algorithm 2 Neural Network Regression in order-3 case implemented with an alternating Logistic Regression step

```
1: procedure ANNLR(Data, max_iter, tol, batch_size, epoch, steps, lr)
2:    '%L%' = function(X,y) solve(t(X)%*%X, t(X)%*%y)        ▷ Operator for computing (X^T X)^{-1} X^T y for inputs X, y
3:    y = Data$y; X = Data$X; W = Data$W                      ▷ (Y, 𝒳, Γ)
4:                                  ▷ W is portion of augmented design matrix related to α, ψ
5:    W_psi = W[,-(1:K)]                                      ▷ Γ_ψ
6:    X_rows=K;X_cols=M
7:    X = array_reshape(X,c(X_Rows, X_cols, dim(X)[3],1))
   ▷ Adds 1-dim channel
8:    testX = array_reshape(testX,c(X_Rows, X_cols, dim(testX)[3],1))
9:    X = aperm(X,c(3,1,2,4))    ▷ Permutes dimensions such that X is T × K × M × 1
10:   testX = aperm(testX,c(3,1,2,4))
11:   input_shape = c(X_rows, X_cols, 1)
12:   psi = W_psi%L%y                                         ▷ ψ^(0) = (Γ_ψ^T Γ_ψ)^{-1} Γ_ψ^T y
```

In the context of the algorithm, the $K \times M$ window of relative price changes is treated as a one-channel "image", which is why the array is reshaped for both training and test data, as seen in lines eight and nine. For the data tensor to be passed correctly to the network the time dimension needs to be the first dimension. This is why the data needs to have its dimensions permuted in lines 10 and 11. Then, the input shape for each slice is defined, which will be important for the input layer of the network. Line 13 demonstrates how $\boldsymbol{\psi}^{(0)}$ is computed.

Next, the iterative update loop commences. Lines 16-22 produce $\sum_{l \neq k} y_{il} \psi_{kl}$ for all days in the training set. Lines 24 through 27 create empty lists, but only for the first iteration of the update loop. The first list will be used to store the K separate networks trained between iterations. The second list gets the weights for the model at each iteration prior to updates, as

these will be used in computation of the convergence criterion. Line 28 initializes an empty matrix for storing each f_k term, whereas line 29 stores the responses and the $\sum_{l \neq k} y_{il}\psi_{kl}$ terms.

13:

14: **for** `ite_index in (1:max_iter)` **do** ▷ Begin for loop, $j =$ `ite_index-1`

15: `psi_fixed = matrix(0,K,K)` ▷ Constructing $\sum_{l \neq k} y_{il}\psi_{kl}$

16: `psi_fixed[lower.tri(psi_fixed, diag=FALSE)] = psi`
▷ Lower triangle of $K \times K$ matrix set to $\boldsymbol{\psi}^{(j)}$

17: `psi_fixed = t(psi_fixed)`

18: `psi_fixed = psi_fixed+ t(psi_fixed)` ▷ Make symmetric

19: `diag(psi_fixed) = rep(1,K)`

20:
```
get_y_psi_fixed = function(t){
            get_y_psi_fixed_k = function(k){
                Ymat[t,-k]%*%(psi_fixed[k,-k])}
            }
```
▷ Generates ith $\sum_{l \neq k} y_{il}\psi_{kl}$

21: `Ypsifixed = t(sapply(1:(Tmax), get_y_psi_fixed))`

22: ▷ Compute $\sum_{l \neq k} y_{il}\psi_{kl}$

23: **if** `ite_index == 1` **then**

24: `models_list = vector("list", K)` ▷ empty list of length K

25: `models_list_old = vector("list", K)` ▷ Storing old values (conv. check)

26: **end if**

27: `f_k_mat = Matrix(0,Tmax,K)` ▷ For $f_k(\mathbf{X}_i, \boldsymbol{\theta}_k^{(j+1)})$

28: `y_true = cbind(Ymat, Ypsifixed)`

29: **for** `k in 1:K` **do** ▷ $\forall k, \boldsymbol{\theta}^{(j+1)}$ update step

30: $\boldsymbol{\theta}_k^{(j+1)} = NN(\mathbf{Y}_k | \mathscr{X}, \mathbf{y}_{l \neq k}, \psi^{(j)})$ ▷ Neural network update, see Algorithm 3

31: **end for**

32: $\boldsymbol{\psi}^{(j+1)} = LR(\mathbf{y} | \boldsymbol{\Gamma}_{\boldsymbol{\psi}}, f(\mathbf{X}_i, \boldsymbol{\theta}_k^{(j+1)}))$ ▷ Logistic regression update step, Alg. 4

33: **until** convergence ▷ See algorithm 5 for convergence check

34: **end for**

35: **end procedure**

Algorithm 3 details an individual neural network update step (Alg. 2, Line 31). The conditional step in lines two through six will initialize a network structure according to K, M, width, depth, and dropout layers, if the algorithm is in the first iteration. Otherwise, this step

loads the previous iteration's final output for the network. Lines seven and eight store the model and its weights respectively for the computation of the composite log-likelihood and parameter difference convergence criteria. The `NCLL` procedure details the computation of the negative composite log-likelihood. Line 11 creates a name pointing to the Keras Python package software backend, which performs operations within the network models. To compute the loss function, the necessary quantities must be passed to the software from the data. Lines 12 and 13 partition `y_true` back into $\sum_{l \neq k} y_{il}\psi_{kl}$ and y_{ik} for stock k. The remainder of the procedure computes the negative composite log-likelihood as defined in equation 3.11.

Algorithm 3 Neural Network Update Step for kth stock

1: **procedure** NN($\mathbf{Y}_k|\mathscr{X}, \mathbf{y}_{l\neq k}, \boldsymbol{\psi}^{(j)}$)
2: **if** `ite_index == 1` **then**
3: `cnn_model` created using neural network structure with $\boldsymbol{\theta}_k^{(0)}$
4: **else**
5: `cnn_model` created using neural network structure with $\boldsymbol{\theta}_k^{(j)}$
6: **end if** ▷ Layer choice may vary, input shape $K \times M \times 1$
7: `cnn_model_old = cnn_model` ▷ Stores jth model for cll convergence calc.
8: `models_list_old = get_weights(cnn_model)`
9: ▷ Stores model weights for max-abs parameter diffs
10: **procedure** `ncll(y_true, y_pred)` ▷ Negative composite log likelihood
11: `Keras = backend()` ▷ Keras backend
12: `Ypsi = y_true[,(K+k)]`
13: `Yk = y_true[,k]`
14: `zhatprob = Keras$exp(y_pred+Ypsi)(1+Keras$exp(y_pred+Ypsi))`
15: `zhatprob0 = 1 - zhatprob`
16: `logzhatprob = Keras$log(zhatprob)`
17: `logzhatprob0 = Keras$log(zhatprob0)`
18: `nll = -1*(Keras$sum(Yk*logzhatprob + (1-Yk)*logzhatprob0))`
19: `nll` ▷ Negative log-likelihood
20: **end procedure**

The `ACCURACY_CUSTOM` metric checks and determines if the rounded predicted value $\hat{y}_{ik}$ matches y_{ik}. It then produces the proportion of cases where they match per epoch. An epoch is a pass over the dataset, splitting the training of data into phases. At the end of each epoch,

the loss and any related metrics are computed. (Chollet 2015) This metric is not used in the analysis conducted later. Line 27 denotes the compile step, where the NCLL loss is used. The Adam optimizer is used in analysis. Then, the model is fit. Line 29 shows how the model is stored after training, and line 30 shows how the kth column of $f(\mathbf{X}_i, \boldsymbol{\theta}_k^{(j+1)})$ is stored. Next, the logistic regression update must be performed, seen in Algorithm 4.

```
21:
22:    procedure accuracy_custom(y_true, y_pred)                    ▷ Accuracy
23:       y_true = y_true[,k]
24:       Keras = backend()                                          ▷ Keras backend
25:       Keras$mean(Keras$equal(y_true, Keras$round(y_pred)))
26:    end procedure
27:    cnn_model %>% compile(loss=ncll,
                             optimizer= optimizer_adam(learning_rate = lr),
                             metrics = accuracy_custom)              ▷ Compile step
28:    cnn_history = cnn_model %>% fit(X, y_true, batch_size = batch_size,
                             steps_per_epoch = steps_per_epoch,epochs = epochs)
29:    models_list[[k]] = cnn_model
  ▷ Store cnn_model for each θ_k^(j+1), starting point for updating θ_k in the next step
30:    f_k_mat[,k] = as.matrix(cnn_model(X))   ▷ Col. k of f(X_i, θ_k^(j+1)) is f_k(X_i, θ_k^(j+1))
31: end procedure
```

Algorithm 4 Logistic Regression Update Step at $(j+1)$th iteration

```
1: procedure LR(y, Γ_ψ, f(X_i, θ_k^(j+1)))                          ▷ Γ_ψ is W_psi
2:    offsets = c(t(f_k_mat))                  ▷ Creates offsets from all f_k using θ_k^(j+1))
3:    logit = glm(y ~ -1 + W_psi+offset(offsets),                   ▷ LR step
                          data=Data, family = "binomial")
4:    psi_new= logit$coefficients[1:(K*(K-1)/2)]                    ▷ ψ^(j+1)
5: end procedure
```

The last remaining step for one iteration of parameter updates is checking the convergence

criterion against a set tolerance level. Here, the criterion is the minimum of the maximum absolute parameter difference and the iterative log-likelihood difference. Respectively, these are quantities 3.15 and 3.16. The pseudo-code for computing these criteria and performing the check is as follows. Lines one and two compute the parameter differences. Lines 3 and 4 store the jth $f(\mathbf{X}_i, \boldsymbol{\theta}_k^{(j)})$ and $\sum_{l \neq k} y_{il} \psi_{kl}^{(j)}$, which are needed for computing the composite log-likelihood for step j. Lines six through eleven compute $\sum_{l \neq k} y_{il} \psi_{kl}^{(j+1)}$.

Algorithm 5 Convergence Check

```
1: theta_diffs = lapply( 1:K,
                  function(k) Map('-',
                  lapply(models_list, get_weights)[[k]],
                  models_list_old[[k]])
   ▷ Computes θ_k^(j+1) − θ_k^(j) for all k
2: psi_diffs = psi_new-psi                          ▷ Computes ψ^(j+1) − ψ^(j)
3: f_k_old = sapply(1:K, function(k) as.matrix(set_weights(cnn_model,
                  models_list_old[[k]])(X)))
   ▷ Computing all f_k(X_i, θ_k^(j))
4: Ypsifixed_old = Ypsifixed                        ▷ Storing Σ_{l≠k} y_il ψ_kl^(j)
5: psi_fixed_new = matrix(0,K,K)                    ▷ Constructing Σ_{l≠k} y_il ψ_kl^(j+1)
6: psi_fixed_new[lower.tri(psi_fixed_new, diag=FALSE)] = psi_new
   ▷ Upper triangle of K × K matrix set to ψ^(j+1)
7: psi_fixed_new = t(psi_fixed_new)
8: psi_fixed_new = psi_fixed_new+ t(psi_fixed_new)  ▷ Make symmetric
9: diag(psi_fixed_new) = rep(1,K)
10: get_y_psi_fixed_new = function(t){
                    get_y_psi_fixed_new_k = function(k){
                        Ymat[t,-k]%*%(psi_fixed_new[k,-k])}
                    }                               ▷ Generates ith Σ_{l≠k} y_il ψ_kl^(j+1)
11: Ypsifixed_new = t(sapply(1:(Tmax), get_y_psi_fixed_new))
```

Lines 12-19 are used to compute the new and old composite log-likelihood terms, and line 20 computes the final convergence criterion. Then a check is performed to determine if this convergence criterion is below the tolerance threshold needed to terminate the loop.

```
z_ik_old = sapply(1:K, function(k) f_k_old[,k]+ Ypsifixed_old[,k])
z_ik_new = sapply(1:K, function(k) f_k_mat[,k]+ Ypsifixed_new[,k])
zhatprob_old = exp(z_ik_old)/(1+exp(z_ik_old))
zhatprob_old0 = 1-zhatprob_old
zhatprob_new = exp(z_ik_new)/(1+exp(z_ik_new))
zhatprob_new0 = 1-zhatprob_new
cll_old = sum(Ymat*log(zhatprob_old) + (1-Ymat)*log(zhatprob_old0))
cll_new = sum(Ymat*log(zhatprob_new) + (1-Ymat)*log(zhatprob_new0))
test = min(max(max(unlist(lapply(1:K,
            function(k) max(unlist(lapply(lapply(theta_diffs[[k]],
                abs),max))))), max(abs(psi_diffs))), cll_new-cll_old)
```

▷ Computes $\max(\max(|\hat{\boldsymbol{\theta}}_k^{(j+1)} - \hat{\boldsymbol{\theta}}_k^{(j)}|), \max(|\hat{\boldsymbol{\psi}}_{kl}^{(j+1)} - \hat{\boldsymbol{\psi}}_{kl}^{(j)}|))$, $\ell^{(j+1)} - \ell^{(j)}$, uses the minimum as conv. criterion

```
psi = psi_new
```

if `test < tol` **then break**

end if ▷ Depending on convergence check, break update loop or return to top

Once the algorithm has been completed, there are some remaining steps needed prior to computing the metrics for model comparison. These steps are similar to the steps taken prior to computing the metrics for the tensor regression model, and their minor differences will be discussed alongside the discussion of those metrics.

3.3.2 Model Structure

In addition to algorithmic pseudocode, it is important to discuss several key aspects unique to a neural network approach. Using this approach, networks of different "depths" containing more layers can be considered. Network "width" can also be considered, where individual layers have more nodes. Different activation functions can also be considered. Other factors, such as different optimization functions and optimizer learning rates, are also options. Dropout layers, which randomly set some weights to zero, can be used as well.

In this analysis, performance is assessed at several settings of three different aspects of the model: model depth, model width, and the inclusion of dropout layers. Model performance for different learning rates and optimizers, as well as other options such as batch size were analyzed as well. However, the performance differences as result of changing these options was not considerable enough to include them in the scope of this discussion. For depth and width,

three levels were considered in each case. One dropout rate was considered and performance was compared against networks with no dropout layers. The general network structure in the simplest case can be seen below.

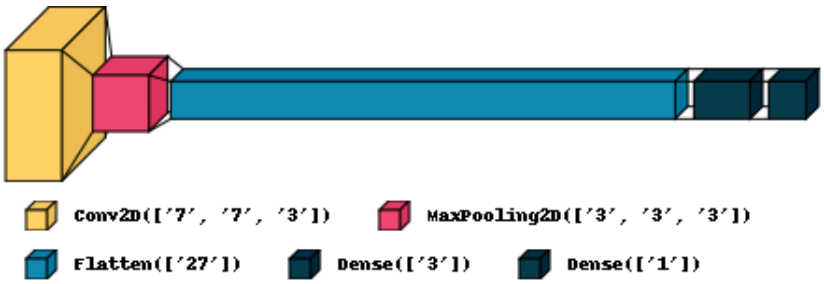

Figure 3.1: Example of Layer Structure

In this structure, the input layer is followed by a convolutional layer, then a pooling layer. Next are a flattening layer and a dense layer before a final output layer. When "depth" is changed, additional convolutional, pooling, and dense layers are added. When "width" is changed, the number of nodes or units in the dense layers is increased. The number of convolutional filters is also increased. When dropout layers are considered, they follow the dense layers.

Here, three depth options and three width options are considered. Shallow networks follow the basic structure seen above. Medium-depth networks have one set of additional layers, whereas deep networks in this context will have three sets of layers. At the narrowest setting "ultra-narrow", there are three filters in the convolution layers and three units in the dense layers. For "narrow" networks, four filters and eight units are used. At the widest setting, 32 filters and 64 units are used. The dropout rate used is 0.9. After a review of the metrics used, the results of the analysis will be discussed.

3.4 Metrics

With regards to the metrics, the same metrics are used for both the MVBNL-TR and ANN-LR models. The chief difference between how these metrics are computed is in how they depend on the model parameters. With no α_k parameters, and with $\mathbf{B}_k$ being replaced by $\boldsymbol{\theta}_k$, the steps which rely on the former parameters must now be adjusted to depend on the latter. For example, joint probability $P(Y_i = y_i|\mathbf{X}_i) = \frac{\exp(\mu_{y_i})}{\sum_{s_i \in S} \exp(\mu_{s_i})}$ now relies on μ_{y_i} such that

$$\mu_{y_i} = \sum_{k=1}^{K} y_{ik}(f_k(\mathbf{X}_i, \boldsymbol{\theta}_k)) + \sum_{l>k} y_{ik} y_{il} \psi_{kl} \tag{3.20}$$

Similarly, conditional probability, which has the form $P(Y_{ik} = 1|\mathbf{X}_i, \ldots) = \frac{\exp(Z_{ik})}{1+\exp(Z_{ik})}$, will have $Z_{ik} = f_k(\mathbf{X}_i, \boldsymbol{\theta}_k) + \sum_{l \neq k} y_{il} \psi_{kl}$. For test data metrics, a corresponding $f(\mathbf{X}_i, \boldsymbol{\theta}_k)$ is created for the test data. The elements of such a matrix are computed by getting the output for $f_k(\mathbf{X}_i, \boldsymbol{\theta}_k)$ when supplying the test data tensor. With key updates such as these, the model metrics can be computed and used to assess performance.

The estimated composite log-likelihood is expressed in the following way:

$$\ell(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\psi}}, \mathbf{Y}, \mathbf{X}_i) = \sum_{i=1}^{T} \sum_{k=1}^{K} \left(y_{ik} \cdot \log\left(\hat{P}_{ik}\right) + (1 - y_{ik}) \cdot \log\left(1 - \hat{P}_{ik}\right) \right) \tag{3.21}$$

$$\hat{P}_{ik} = \frac{\exp(Z_{ik})}{1 + \exp(Z_{ik})} \tag{3.22}$$

$$Z_{ik} = f_k(\mathbf{X}_i, \hat{\boldsymbol{\theta}}_k) + \sum_{l \neq k} y_{il} \hat{\boldsymbol{\psi}}_{kl} \tag{3.23}$$

Multi-class AUC, conditional AUC, and co-movement AUC, are all considered. Additionally, co-movement MSE, concordance, joint probability log-likelihood, and the frequency of maximum observed estimated probability are considered. All of these metrics were previously defined in section 2.4- 2.5. The changes to metrics involve the new conditional, joint, and co-movement probabilities. Model performance is compared using all of these metrics. Two sets of data are used for model analysis. These sets of data are the same real world examples that are are used in chapter 2. One set of data considers some of the companies with the highest market capitalization on the market. The other set of data is a sector-specific example, focusing on companies in the agricultural sector. Several different settings for ANN-LR will be compared during this discussion.

3.5 Analysis of Results

One thing to recall about the tensor regression model is its parameter size at different model settings. The number of parameters is $K + K \cdot (K-1)/2 + K \cdot R \cdot (K+M)$. Comparing all of the cases, the number of parameters in each model can be seen in Table 3.1. The number of parameters for the neural network model will be K times the number of network parameters plus $K \cdot (K-1)/2$ association parameters. The number of parameters for the network models can be seen in Tables 3.2 and 3.3. For the neural network model the number of parameters

only increases for different choices of K and M because of the number of networks K. As the networks deepen and widen, the number of parameters will also increase.

Table 3.1: MVBNL-TR Parameter Size, Total Days = 1500

	$R=1$	$R=3$
$K=M=5$	65	165
$K=10, M=5$	205	505
$K=M=10$	265	655

Table 3.2: ANN-LR Parameter Size, Total Days = 1500

	Ultra-Narrow		
	Shallow	Medium	Deep
$K=M=5$	435	615	675
$K=10, M=5$	895	1255	1375
$K=M=10$	895	1255	1375

Table 3.3: ANN-LR Parameter Size, Total Days = 1500

	Narrow			Wide		
	Shallow	Medium	Deep	Shallow	Medium	Deep
$K=M=5$	595	1055	2795	1515	39695	65775
$K=10, M=5$	1215	2135	4055	27255	79415	131575
$K=M=10$	1215	2135	4055	27255	79415	131575

3.5.1 High Market Capitalization Data

The data used in this first analysis comes from some of the stocks with highest market capitalization in recent years. For the five-stock case, Apple, Microsoft, Google, Amazon, and Tesla are included. When looking at the ten-stock case, Meta, Taiwan Semiconductor (TSMC), NVIDIA, Visa, and United Health Group Inc. are also included in the analysis.

When considering "ultra-narrow" networks, the medium-depth networks without dropout layers consistently perform the best, as seen in Table 3.4. When the networks widen, deep networks perform best for smaller ($M=5$) look-back windows. When $K=M=10$, medium networks peform the best. This can be seen in Table 3.5.

Table 3.4: Composite Log-likelihood: Ultra-Narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	-2445.2	-2385.9	-2517.4	-2476.2	-2485.1	-2503.8
	$K = 10, M = 5$	-4821.3	-4788.5	-4743.4	-4925.2	-4973.3	-4961.8
	$K = M = 10$	-4757.4	-4742.2	-4756.2	-4878.4	-4971.4	-4965.5
Test	$K = M = 5$	-2480.8	-2421.9	-2546.7	-2509.1	-2516.4	-2534.1
	$K = 10, M = 5$	-4923.5	-4894.9	-4849.1	-5022.7	-5068.7	-5056.8
	$K = M = 10$	-4863.6	-4851.8	-4865.5	-4978.9	-5066.7	-5060.4

Table 3.5: Composite Log-likelihood: Narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	-2382.4	-2343.6	-2335.1	-2437.9	-2454.7	-2455.2
	$K = 10, M = 5$	-4697.8	-4666.7	-4660.1	-4793.0	-4897.9	-4898.3
	$K = M = 10$	-4664.3	-4660.7	-4661.5	-4724.6	-4894.3	-4897.5
Test	$K = M = 5$	-2418.3	-2381.1	-2372.9	-2438.0	-2486.1	-2487.3
	$K = 10, M = 5$	-4809.4	-4780.3	-4775.6	-4897.3	-4995.8	-4996.5
	$K = M = 10$	-4779.4	-4776.4	-4777.8	-4833.4	-4992.1	-4995.7

Table 3.6: Composite Log-likelihood: Wide Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	-2341.4	-2343.5	-2343.7	-2340.4	-2342.7	-2340.1
	$K = 10, M = 5$	-4665.7	-4666.5	-4666.7	-4665.6	-4666.5	-4662.0
	$K = M = 10$	-4666.5	-4666.6	-4666.7	-4666.6	-4666.5	-4661.2
Test	$K = M = 5$	-2378.9	-2380.9	-2381.1	-2377.9	-2380.2	-2377.7
	$K = 10, M = 5$	-4783.5	-4784.5	-4784.8	-4783.3	-4784.5	-4778.6
	$K = M = 10$	-4784.4	-4784.7	-4784.8	-4784.7	-4784.6	-4777.5

When considering wide networks, the deep network with dropout layers outperforms all other models, as seen in Table 3.6. Across all depth and dropout options, all of the models perform very similarly in terms of maximizing the CLL, with the deep dropout model only slightly outperforming the rest of the networks. It should also be noted that the networks here actually perform slightly worse overall at maximizing the CLL in contrast with the best models

in the narrow case. In addition to maximizing CLL best, the medium and deep narrow models without dropout layers also have a smaller parameter space than the wide networks. While still larger than the tensor regression model, the difference in the size of the parameter space is less drastic than in the wide network case.

Next, the multi-class AUC is considered in Tables 3.7-3.9. Across all options considered, multi-class AUC is fairly low. In addition to this, there are multiple cases where the AUC is lower for the training data than it is for the test data. These imbalanced results, in addition to the overall poor performance, indicate that the neural network model's utility for classifying joint outcomes is roughly the same as a random process that selects an outcome.

Table 3.7: Multi-class AUC: Ultra-Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	0.502	0.508	0.515	0.496	0.509	0.501
	$K = 10, M = 5$	0.505	0.517	0.516	0.512	0.531	0.500
	$K = M = 10$	0.524	0.498	0.474	0.512	0.499	0.481
Test	$K = M = 5$	0.478	0.490	0.516	0.470	0.490	0.521
	$K = 10, M = 5$	0.521	0.518	0.522	0.509	0.535	0.481
	$K = M = 10$	0.521	0.479	0.492	0.527	0.503	0.458

Table 3.8: Multi-class AUC: Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	0.490	0.502	0.488	0.488	0.522	0.490
	$K = 10, M = 5$	0.490	0.496	0.472	0.490	0.495	0.492
	$K = M = 10$	0.471	0.485	0.500	0.485	0.483	0.528
Test	$K = M = 5$	0.484	0.489	0.499	0.484	0.477	0.495
	$K = 10, M = 5$	0.506	0.492	0.519	0.516	0.516	0.511
	$K = M = 10$	0.497	0.480	0.499	0.498	0.528	0.482

Table 3.9: Multi-class AUC: Wide Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	0.515	0.484	0.487	0.511	0.485	0.487
	$K = 10, M = 5$	0.494	0.516	0.501	0.494	0.516	0.495
	$K = M = 10$	0.482	0.479	0.480	0.482	0.479	0.482
Test	$K = M = 5$	0.462	0.478	0.526	0.452	0.478	0.527
	$K = 10, M = 5$	0.550	0.463	0.528	0.547	0.463	0.532
	$K = M = 10$	0.496	0.484	0.504	0.496	0.484	0.510

With regards to the joint probability log-likelihood, the model performs best in many of the same scenarios that it performed best for the composite log-likelihood. For the data that was not re-coded, the pattern differs for the ultra-narrow case. Whereas the composite log-likelihood was maximized in the medium network for every choice of K and M, The deep networks outperform the medium-depth networks in terms of joint probability log-likelihood when $K = 10$ for the training data, and when $K = 10$ and $M = 5$ for the test data. When considering the narrow and wide networks, the best models are the same models for which the composite log-likelihood performed the best. As with the composite log-likelihood, the narrow networks that performed best maximize the joint probability log-likelihood better than the best-performing wide networks.

Table 3.10: Joint Probability Log-Likelihood: Ultra-Narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	-2634.2	-2594.9	-2683.5	-2654.7	-2660.8	-2672.5
	$K = 10, M = 5$	-5204.6	-5189.1	-5154.7	-5271.5	-5302.2	-5294.8
	$K = M = 10$	-5161.6	-5236.9	-5171	-5241.3	-5301.2	-5297.1
Test	$K = M = 5$	-2657.8	-2618.9	-2703.8	-2676.9	-2682.2	-2693.2
	$K = 10, M = 5$	-5260.2	-5284.9	-5212.6	-5324.6	-5354	-5346.3
	$K = M = 10$	-5220.1	-5163.4	-5231.5	-5296.4	-5352.9	-5348.6

Table 3.11: Joint Probability Log-Likelihood: Narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	-2593	-2567.2	-2561.7	-2629.9	-2640.4	-2640.7
	$K = 10, M = 5$	-5130.7	-5110	-5108.2	-5189.1	-5254.7	-5255.5
	$K = M = 10$	-5111.1	-5109.3	-5111.3	-5144.7	-5252.4	-5254.9
Test	$K = M = 5$	-2616.7	-2592.1	-2586.9	-2652	-2661.6	-2662.7
	$K = 10, M = 5$	-5192.2	-5173.3	-5172.8	-5245.9	-5307.8	-5308.9
	$K = M = 10$	-5175.4	-5174.1	-5176.6	-5204.5	-5305.4	-5308.3

Table 3.12: Joint Probability Log-Likelihood: Wide Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	-2574.8	-2578.5	-2578.8	-2573	-2577.1	-2572.6
	$K = 10, M = 5$	-5118.6	-5119.9	-5120.2	-5118.3	-5119.9	-5112.4
	$K = M = 10$	-5119.7	-5120.1	-5120.3	-5120.2	-5120	-5111
Test	$K = M = 5$	-2600	-2603.7	-2604	-2598.2	-2602.3	-2597.8
	$K = 10, M = 5$	-5185	-5186.6	-5186.9	-5184.7	-5186.6	-5177.9
	$K = M = 10$	-5186.3	-5186.8	-5187	-5186.7	-5186.7	-5176.2

When considering the re-coded data examples, the narrow networks again appear to be performing the best out of the networks considered. In the ultra-narrow case, the best performing network for $K = M = 10$ is the shallow network for the training data. Other than this difference, the metrics based on the re-coded responses match those utilizing the original responses.

Table 3.13: Joint Prob. LL (25%/5% Split): Ultra-Narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	-1713.7	-1677.8	-1753.6	-1731.3	-1736	-1745.9
	$K = 10, M = 5$	-4571.9	-4555.6	-4522.7	-4637.8	-4667.9	-4661.7
	$K = M = 10$	-4529.1	-4530	-4538.2	-4607	-4666.8	-4663.9
Test	$K = M = 5$	-969.9	-946.6	-999.4	-982.5	-986.2	-993.2
	$K = 10, M = 5$	-2626.8	-2615.5	-2591.5	-2678.2	-2701.9	-2696
	$K = M = 10$	-2595.4	-2595.1	-2602.7	-2655.9	-2701.1	-2697.7

Table 3.14: Joint Prob. LL (25%/5% Split): Narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	-1675.6	-1651.5	-1646.1	-1708.3	-1717.1	-1718.7
	$K = 10, M = 5$	-4498.9	-4478.8	-4476.8	-4556.6	-4622.3	-4622
	$K = M = 10$	-4479.6	-4477.8	-4479.6	-4512.8	-4620.1	-4621.5
Test	$K = M = 5$	-945	-929.5	-926.1	-967.6	-974.4	-974.2
	$K = 10, M = 5$	-2571	-2555.7	-2553.9	-2615.8	-2665.9	-2667.1
	$K = M = 10$	-2555.9	-2554.6	-2555.9	-2582.1	-2663.9	-2666.6

Table 3.15: Joint Prob. LL (25%/5% Split): Wide Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	-1653.4	-1655.6	-1655.9	-1652.3	-1654.8	-1652
	$K = 10, M = 5$	-4486.3	-4487.6	-4487.8	-4486.1	-4487.6	-4480.7
	$K = M = 10$	-4487.4	-4487.7	-4487.9	-4487.7	-4487.6	-4479.3
Test	$K = M = 5$	-931.9	-933.5	-933.6	-931	-932.9	-930.9
	$K = 10, M = 5$	-2560.4	-2561.3	-2561.4	-2560.3	-2561.3	-2556.5
	$K = M = 10$	-2561.1	-2561.4	-2561.5	-2561.4	-2561.3	-2555.6

Because the frequency of observing the maximum estimated probability does not change when network width increases, only the ultra-narrow results are provided. The count does not change when considering different network depths or the inclusion of dropout layers. As such, this metric alone does not provide any insight into which combination of network settings contributes the most towards model performance. The $K = M = 5$ case has the highest frequency of desired outcomes. As the tensor slices $\mathbf{X}_i$ become larger, the desired outcome becomes less frequent.

Table 3.16: Freq. Max Obs. Prob: Ultra-Narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	161	161	161	161	161	161
	$K = 10, M = 5$	77	77	77	77	77	77
	$K = M = 10$	77	77	77	77	77	77
Test	$K = M = 5$	151	151	151	151	151	151
	$K = 10, M = 5$	58	58	58	58	58	58
	$K = M = 10$	58	58	58	58	58	58

Conditional AUC is summarized in Tables 3.17-3.19 as the average conditional AUC across different stocks. The networks with dropout layers consistently under-perform in contrast to the networks with no dropout layers. For the narrowest networks considered, the medium depth option performs the best when $K = M$, but slightly worse in the case where $K = 10$ where $M = 5$. When considering wider networks, the conditional AUC noticeably improves. For wide networks, the networks with dropout layers have comparable AUC performance to those without dropout layers. The deep and wide networks without dropout layers perform the best overall: a moderate improvement over the deep and narrow networks, with a slight edge over medium-wide networks.

Table 3.17: Conditional AUC: Ultra-Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Test	$K = M = 5$	0.732	0.735	0.732	0.732	0.728	0.728
	$K = 10, M = 5$	0.723	0.737	0.738	0.708	0.701	0.703
	$K = M = 10$	0.732	0.744	0.739	0.715	0.703	0.702

Table 3.18: Conditional AUC: Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Test	$K = M = 5$	0.733	0.761	0.769	0.728	0.728	0.736
	$K = 10, M = 5$	0.745	0.748	0.751	0.733	0.714	0.716
	$K = M = 10$	0.750	0.751	0.752	0.741	0.714	0.716

Table 3.19: Conditional AUC: Wide Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Test	$K = M = 5$	0.778	0.781	0.781	0.778	0.780	0.779
	$K = 10, M = 5$	0.753	0.753	0.753	0.753	0.753	0.752
	$K = M = 10$	0.753	0.753	0.753	0.753	0.753	0.752

Co-movement is also studied and reported in Tables 3.20-3.22. In several of the cases

considered, co-movement AUC is highest when dropout layers are used. This is true for all of the ultra-narrow networks, and for all but one of the narrow networks. For wide networks, the deep network with no dropout layer performs the best when $K = M$, but for the ten stock case with a five day look-back window, the shallow dropout network performs better. For all the networks considered, the co-movement AUC is quite low. The issues exhibited for the multi-class AUC are present for co-movement AUC as well, indicating that the model does not classify co-movement very well.

Table 3.20: Co-movement AUC: Ultra-Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	0.467	0.480	0.512	0.468	0.490	0.516
	$K = 10, M = 5$	0.470	0.482	0.484	0.474	0.497	0.512
	$K = M = 10$	0.4728	0.475	0.467	0.473	0.487	0.503
Test	$K = M = 5$	0.463	0.475	0.513	0.458	0.490	0.539
	$K = 10, M = 5$	0.475	0.477	0.483	0.475	0.488	0.524
	$K = M = 10$	0.4729	0.476	0.465	0.469	0.477	0.500

Table 3.21: Co-movement AUC: Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	0.472	0.480	0.469	0.470	0.472	0.502
	$K = 10, M = 5$	0.475	0.482	0.479	0.477	0.489	0.497
	$K = M = 10$	0.476	0.471	0.477	0.475	0.497	0.484
Test	$K = M = 5$	0.462	0.472	0.452	0.454	0.470	0.460
	$K = 10, M = 5$	0.469	0.471	0.479	0.466	0.486	0.497
	$K = M = 10$	0.468	0.466	0.467	0.466	0.498	0.483

Table 3.22: Co-movement AUC: Wide Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	0.483	0.466	0.506	0.485	0.480	0.502
	$K = 10, M = 5$	0.493	0.478	0.494	0.508	0.478	0.480
	$K = M = 10$	0.484	0.484	0.506	0.493	0.484	0.479
Test	$K = M = 5$	0.454	0.475	0.539	0.452	0.480	0.527
	$K = 10, M = 5$	0.492	0.485	0.485	0.506	0.485	0.466
	$K = M = 10$	0.484	0.477	0.504	0.492	0.477	0.471

In terms of co-movement MSE, all the networks considered perform similarly, as seen in Tables 3.23-3.25. For ultra-narrow and narrow networks, the best performing models are the medium and deep networks, respectively. When considering wide networks, the performance is nearly the same across all options. In each case, when more stocks are considered the MSE increases slightly. However, when the look-back window increases in size, the MSE either stays the same or decreases slightly. Overall, there appear to be only minute changes when increasing network depth or width, or including dropout layers.

Table 3.23: Co-movement MSE: Ultra-narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	0.219	0.218	0.219	0.219	0.219	0.219
	$K = 10, M = 5$	0.227	0.226	0.226	0.227	0.227	0.227
	$K = M = 10$	0.226	0.226	0.226	0.227	0.227	0.227
Test	$K = M = 5$	0.221	0.220	0.221	0.222	0.221	0.222
	$K = 10, M = 5$	0.229	0.228	0.228	0.229	0.229	0.229
	$K = M = 10$	0.229	0.228	0.228	0.229	0.229	0.229

Table 3.24: Co-movement MSE: Narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	0.218	0.217	0.217	0.218	0.219	0.219
	$K = 10, M = 5$	0.226	0.225	0.225	0.226	0.227	0.227
	$K = M = 10$	0.225	0.225	0.225	0.226	0.227	0.227
Test	$K = M = 5$	0.220	0.220	0.219	0.221	0.221	0.221
	$K = 10, M = 5$	0.228	0.228	0.228	0.229	0.229	0.229
	$K = M = 10$	0.228	0.228	0.228	0.228	0.229	0.229

Table 3.25: Co-movement MSE: Wide Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	0.216	0.216	0.216	0.216	0.216	0.216
	$K = 10, M = 5$	0.225	0.225	0.225	0.225	0.225	0.225
	$K = M = 10$	0.225	0.225	0.225	0.225	0.225	0.225
Test	$K = M = 5$	0.219	0.218	0.218	0.219	0.218	0.219
	$K = 10, M = 5$	0.228	0.228	0.228	0.228	0.228	0.228
	$K = M = 10$	0.228	0.228	0.228	0.228	0.228	0.228

As the concordance does not change based on the width of the network, only the results for the ultra-narrow case need to be discussed. Dropout layers and increased depth do not change the concordance. The inclusion of more stocks results in a larger concordance, but increasing the look-back window does not change the results further. The concordance is higher for training data than it is for test data, which merits further investigation. Recall that the maximum value for concordance is a function of K, as such, this increase in concordance needs to be properly contextualized. For $K = 5$, the theoretical maximum for concordance is 10. When K is 10, the theoretical maximum is 45. Thus, while the concordance does indeed increase with K, its increase is actually relatively small. For example, whereas the training set concordance for $K = 5$ is 15.1% of its theoretical maximum, the concordance is at roughly 18.35% of its maximum for $K = 10$. For the test set, when $K = 5$, the concordance is 15.39% of its theoretical maximum, and 18.84% of its theoretical maximum when $K = 10$.

Table 3.26: Co-movement Concordance: Ultra-Narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	1.510	1.510	1.510	1.510	1.510	1.510
	$K = 10, M = 5$	8.256	8.256	8.256	8.256	8.256	8.256
	$K = M = 10$	8.256	8.256	8.256	8.256	8.256	8.256
Test	$K = M = 5$	1.539	1.539	1.539	1.539	1.539	1.539
	$K = 10, M = 5$	8.482	8.482	8.482	8.482	8.482	8.482
	$K = M = 10$	8.482	8.482	8.482	8.482	8.482	8.482

3.5.2 Agricultural Sector Data

The same agricultural sector stock data that is used for the tensor regression model in chapter 2 is used again in this discussion. Recall that for the five-stock case, the stocks considered are Archer-Daniels-Midland Company, Bunge Limited, Caterpillar Inc., Deere & Company, and The Scotts Miracle-Gro Company. When considering a ten-stock case, FMC Corporation, Calavo Growers, Inc., The Mosaic Company, Tyson Foods, Inc., and China Green Agriculture, Inc. are also included. In the ultra-narrow and narrow networks in Tables 3.27 and 3.28, the networks with no dropout layers consistently outperform those with dropout layers. Deep networks perform best the majority of the time, with the exception of the five-stock case for ultra-narrow networks. When considering narrow networks, medium and deep networks have similar composite log-likelihood values. There seems to be a substantial benefit to the initial widening step, as narrow network CLLs are consistently maximized greater than their ultra-narrow counterparts.

Table 3.27: Composite Log-likelihood: Ultra-Narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	-2536.1	-2485.5	-2601	-2563.6	-2572.5	-2586.5
	$K = 10, M = 5$	-5033.9	-5003.9	-4946.4	-5128.4	-5166	-5148.6
	$K = M = 10$	-4976.4	-4978.6	-4938.8	-5096.8	-5158.5	-5153.9
Test	$K = M = 5$	-2549	-2492	-2616.6	-2579.5	-2589.1	-2605.2
	$K = 10, M = 5$	-5089.2	-5046.6	-4992.5	-5190.4	-5228.2	-5211.6
	$K = M = 10$	-5026.3	-5019.4	-4983.7	-5156.8	-5220.7	-5217.2

Table 3.28: Composite Log-likelihood: Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	-2493.9	-2447.4	-2436.6	-2526.3	-2545.3	-2543.1
	$K = 10, M = 5$	-4920.7	-4888.2	-4884	-5006.5	-5090.3	-5093
	$K = M = 10$	-4885.5	-4887.6	-4883.8	-4957.5	-5089.7	-5092.2
Test	$K = M = 5$	-2501.1	-2449	-2437.3	-2537.9	-2558.5	-2559.2
	$K = 10, M = 5$	-4961	-4923.1	-4917.9	-5057.3	-5147.4	-5151.3
	$K = M = 10$	-4920.2	-4921.3	-4917.5	-5000	-5146.9	-5150.6

When considering wider networks in Table 3.29, the dropout networks have marginally better performance. At each choice of K and M, the difference between the minimum and maximum CLL is relatively small. While utilizing dropout layers for wide networks is the best option for maximizing the CLL, these maximized values are only slight improvements over the maximized values in the narrow case. The narrow models are comparably at maximizing the composite log-likelihood while also having a simpler parameter space.

Table 3.29: Composite Log-likelihood: Wide Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	-2442.5	-2443.6	-2443.7	-2441.8	-2443.4	-2442.7
	$K = 10, M = 5$	-4888.6	-4890.1	-4890.3	-4888.4	-4889.8	-4883.3
	$K = M = 10$	-4889.7	-4890.2	-4890.3	-4890	-4890.1	-4883.7
Test	$K = M = 5$	-2438.6	-2439.1	-2439.2	-2438.3	-2439	-2438.7
	$K = 10, M = 5$	-4921.8	-4923.1	-4923.3	-4921.6	-4922.9	-4917
	$K = M = 10$	-4922.8	-4923.2	-4923.4	-4923.1	-4923.2	-4917.4

As with the high market capitalization case, the agricultural stock sector data has consistently low multi-class AUC. When increasing width initially, the multi-class AUC improves slightly in several training cases while decreasing in test cases. This could indicate over-fitting. There are still several cases where the test AUC exceeds the training AUC. However, none of the models that performs best in training performs better on test data in the narrow case. In the ultra-narrow and wide cases (Tables 3.30 and 3.32), there is one case where the best performing training set model for a combination of K and M performs better on test data. These cases

are noted are noted in the tables. As with the CLL, it seems that there are some benefits to increasing the width, but when width is increased greatly, these benefits diminish. Overall, none of the models considered classify joint outcomes well.

Table 3.30: Multi-class AUC: Ultra-Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	0.481	0.483	0.508	0.479	0.515	0.508[1]
	$K = 10, M = 5$	0.493	0.488	0.470	0.502	0.493	0.503
	$K = M = 10$	0.493	0.492	0.499	0.499	0.524	0.506
Test	$K = M = 5$	0.458	0.481	0.484	0.466	0.488	0.534[1]
	$K = 10, M = 5$	0.479	0.509	0.480	0.481	0.518	0.499
	$K = M = 10$	0.487	0.511	0.481	0.471	0.502	0.534

[1] Test AUC exceeds Train AUC

Table 3.31: Multi-class AUC: Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	0.473	0.495	0.497	0.484	0.510	0.478
	$K = 10, M = 5$	0.494	0.494	0.489	0.497	0.503	0.491
	$K = M = 10$	0.507	0.496	0.503	0.491	0.495	0.495
Test	$K = M = 5$	0.454	0.485	0.456	0.453	0.490	0.465
	$K = 10, M = 5$	0.458	0.475	0.498	0.466	0.480	0.478
	$K = M = 10$	0.493	0.494	0.496	0.482	0.514	0.454

Table 3.32: Multi-class AUC: Wide Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	0.498	0.486	0.489	0.498	0.486	0.488
	$K = 10, M = 5$	0.483	0.499	0.496	0.483	0.499	0.499
	$K = M = 10$	0.507[1]	0.490	0.484	0.506	0.489	0.485
Test	$K = M = 5$	0.455	0.466	0.525	0.458	0.466	0.525
	$K = 10, M = 5$	0.530	0.499	0.469	0.533	0.499	0.459
	$K = M = 10$	0.527[1]	0.519	0.525	0.527	0.519	0.520

[1] Test AUC exceeds Train AUC

For most choices of number of stocks and look-back window size, the joint probability log-

likelihood is maximized for the same network as the composite log-likelihood. The exception to this pattern is for the narrow case where $K = M = 10$, seen in Table 3.34. For the training set, the shallow network maximizes the joint probability log-likelihood instead of the deep network.

Table 3.33: Joint Probability Log-Likelihood: Ultra-Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	-2662.5	-2633.1	-2707.7	-2679.9	-2686.3	-2694.2
	$K = 10, M = 5$	-5256.4	-5238.9	-5201.3	-5314.7	-5338.2	-5327.1
	$K = M = 10$	-5219.9	-5226.2	-5197.7	-5295.3	-5333.5	-5330.2
Test	$K = M = 5$	-2678.1	-2645.4	-2725.3	-2697.3	-2704.3	-2713.1
	$K = 10, M = 5$	-5295.9	-5271.9	-5235.7	-5358	-5381.6	-5371
	$K = M = 10$	-5255.5	-5257.7	-5231.6	-5337.1	-5377	-5374.3

Table 3.34: Joint Probability Log-Likelihood: Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	-2639.3	-2612	-2605.6	-2657.9	-2669.2	-2667.9
	$K = 10, M = 5$	-5187.5	-5166.3	-5166.3	-5240.2	-5291.9	-5293.4
	$K = M = 10$	-5166.3	-5168.1	-5166.6	-5209.3	-5291.5	-5292.9
Test	$K = M = 5$	-2652.3	-2621.3	-2614.2	-2673.1	-2685.4	-2685.5
	$K = 10, M = 5$	-5219.7	-5194.9	-5194.5	-5278.2	-5332.7	-5334.8
	$K = M = 10$	-5194.9	-5196.1	-5194.7	-5242	-5332.4	-5334.3

Table 3.35: Joint Probability Log-Likelihood: Wide Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	-2618.6	-2620.9	-2621.1	-2617.4	-2620.5	-2619.1
	$K = 10, M = 5$	-5172.8	-5174.6	-5174.9	-5172.4	-5174.3	-5166
	$K = M = 10$	-5174.1	-5174.7	-5174.9	-5174.5	-5174.7	-5166.5
Test	$K = M = 5$	-2624.9	-2626.8	-2627	-2623.8	-2626.5	-2625.3
	$K = 10, M = 5$	-5200.8	-5202.7	-5202.9	-5200.5	-5202.4	-5194
	$K = M = 10$	-5202.2	-5202.8	-5203	-5202.6	-5202.8	-5194.5

As with the composite log-likelihood, the log-likelihood here is maximized to a greater

extent when increasing width initially, but the improvements are small when going from the narrow to the wide network case. As increasing width a second time drastically expands the parameter space, this is inadvisable unless there are clear performance improvements when doing so. This behavior is also present in the re-coded response case in Tables 3.36-3.38. In this case, all network width options match the composite log-likelihood with regards to which network performs best.

Table 3.36: Joint Prob. LL (25%/5% Split): Ultra-Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	-1614.3	-1586.2	-1652.2	-1629.4	-1633.9	-1641.7
	$K = 10, M = 5$	-4495.8	-4475.8	-4441.2	-4553.2	-4576.1	-4566.7
	$K = M = 10$	-4459.2	-4462.2	-4437	-4532.9	-4571.6	-4569.7
Test	$K = M = 5$	-964.4	-940.4	-993.9	-977	-980.8	-987.5
	$K = 10, M = 5$	-2464.1	-2443.1	-2412.8	-2513.8	-2532	-2523.4
	$K = M = 10$	-2431.4	-2428.7	-2408.6	-2497.6	-2528.4	-2526

Table 3.37: Joint Prob. LL (25%/5% Split): Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	-1592.2	-1566	-1559.9	-1609.1	-1620.2	-1616.6
	$K = 10, M = 5$	-4427	-4406.3	-4405.4	-4479.9	-4532.5	-4532.3
	$K = M = 10$	-4405.8	-4407.2	-4405.6	-4448.6	-4532.1	-4531.8
Test	$K = M = 5$	-945.2	-922.7	-917.7	-960	-968.8	-967.9
	$K = 10, M = 5$	-2399.9	-2379.1	-2377.1	-2448.9	-2492.9	-2494.2
	$K = M = 10$	-2378.1	-2378.7	-2377	-2419.3	-2492.6	-2493.8

Table 3.38: Joint Prob. LL (25%/5% Split): Wide Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	-1568.2	-1569.7	-1569.9	-1567.4	-1569.5	-1568.6
	$K = 10, M = 5$	-4410.6	-4412.1	-4412.3	-4410.3	-4411.8	-4405
	$K = M = 10$	-4411.7	-4412.2	-4412.3	-4412	-4412.2	-4405.4
Test	$K = M = 5$	-921.4	-922.1	-922.1	-921	-922	-921.5
	$K = 10, M = 5$	-2380	-2380.9	-2381.1	-2379.8	-2380.8	-2376.6
	$K = M = 10$	-2380.7	-2381	-2381.1	-2380.9	-2381	-2376.9

In contrast to the high market capitalization case, the frequency of maximum observed estimated probability is consistently higher in test cases. When widening networks, this effect is exacerbated in the ten-stock cases as seen in the wide network case in Table 3.40. The narrow case is not provided in a table, as the frequency is the same as in the ultra-narrow case. It is unclear why the training set performance is worse than test set performance. Perhaps this could be behavior that is present in within-sector analyses. Such a problem merits further study. The frequency also does not change for a fixed width across depths or with the inclusion of a dropout layer. However, this is due to the fact that the observed probabilities change very little at these different settings, and thus the frequency is not affected.

Table 3.39: Count Obs. Prob = Max Prob.: Ultra-Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	120	120	120	120	120	120
	$K = 10, M = 5$	25	25	25	25	25	25
	$K = M = 10$	25	25	25	25	25	25
Test	$K = M = 5$	139	139	139	139	139	139
	$K = 10, M = 5$	29	29	29	29	29	29
	$K = M = 10$	29	29	29	29	29	29

Table 3.40: Count Obs. Prob = Max Prob.: Wide Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	120	120	120	120	120	120
	$K = 10, M = 5$	28	28	28	28	28	28
	$K = M = 10$	28	28	28	28	28	28
Test	$K = M = 5$	139	139	139	139	139	139
	$K = 10, M = 5$	45	45	45	45	45	45
	$K = M = 10$	45	45	45	45	45	45

There are several patterns worth noting with average conditional AUC in Tables 3.41-3.43. Networks without dropout layers often outperform their dropout counterparts. However, as the network structure widens, this performance gap decreases. In the wide case, the performances are close to equal, with the dropout networks outperforming the medium and deep networks when five stocks are being considered. Despite this near identical performance in the wide case,

a non-dropout network structure is the definitively superior model in terms of conditional AUC at each combination of K and M.

In the ultra-narrow case, the medium-depth structure is superior when $K = M$. However, the deep structure does a marginally better job classifying stocks conditionally in the case where $K = 10$ and $M = 5$. In the narrow case, the deep model consistently has a slight edge over other network structures. When considering wide networks, the shallowest network performs the best. However, its dropout counterpart is tied, and the conditional classification only improves slightly from the deep and narrow network. These findings are in line with the other metrics discussed so far.

Table 3.41: Conditional AUC: Ultra-Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Test	$K = M = 5$	0.655	0.701	0.675	0.648	0.645	0.642
	$K = 10, M = 5$	0.669	0.681	0.683	0.640	0.632	0.633
	$K = M = 10$	0.684	0.689	0.685	0.648	0.634	0.632

Table 3.42: Conditional AUC: Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Test	$K = M = 5$	0.709	0.735	0.743	0.683	0.646	0.649
	$K = 10, M = 5$	0.696	0.703	0.705	0.673	0.652	0.649
	$K = M = 10$	0.705	0.705	0.706	0.690	0.652	0.650

Table 3.43: Conditional AUC: Wide Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Test	$K = M = 5$	0.751	0.746	0.745	0.751	0.751	0.751
	$K = 10, M = 5$	0.707	0.707	0.707	0.707	0.707	0.706
	$K = M = 10$	0.707	0.707	0.707	0.707	0.707	0.707

In contrast to several other metrics, dropout network models have greater pairwise average

co-movement AUC in the ultra-narrow and narrow models in most cases, as seen in Tables 3.44 and 3.45. The models without dropout layers perform better in the majority of the cases considered for wide networks, as seen in Table 3.46. For ultra-narrow networks, the training co-movement AUC exceeds that of the test AUC in the ten-stock cases. However, in each case the models perform poorly. In the narrow case, all models classify co-movement worse than ultra-narrow models.

Table 3.44: Co-movement AUC: Ultra-Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	0.472	0.477	0.497	0.473	0.484	0.512
	$K = 10, M = 5$	0.482	0.477	0.477	0.483	0.478	0.510
	$K = M = 10$	0.473	0.474	0.475	0.474	0.487	0.498
Test	$K = M = 5$	0.464	0.482	0.509	0.463	0.493	0.514
	$K = 10, M = 5$	0.488	0.489	0.488	0.490	0.490	0.504
	$K = M = 10$	0.482	0.491	0.484	0.480	0.494	0.494

Table 3.45: Co-movement AUC: Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	0.456	0.473	0.483	0.458	0.471	0.473
	$K = 10, M = 5$	0.476	0.480	0.479	0.478	0.495	0.489
	$K = M = 10$	0.464	0.475	0.469	0.470	0.496	0.488
Test	$K = M = 5$	0.473	0.480	0.459	0.469	0.473	0.484
	$K = 10, M = 5$	0.483	0.488	0.491	0.486	0.497	0.494
	$K = M = 10$	0.478	0.488	0.480	0.484	0.498	0.489

Table 3.46: Co-movement AUC: Wide Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	0.473	0.496	0.518	0.476	0.499	0.514
	$K = 10, M = 5$	0.480	0.506	0.506	0.486	0.508	0.492
	$K = M = 10$	0.499	0.501	0.508	0.506	0.500	0.490
Test	$K = M = 5$	0.468	0.472	0.511	0.468	0.475	0.505
	$K = 10, M = 5$	0.490	0.496	0.497	0.494	0.497	0.497
	$K = M = 10$	0.489	0.504	0.507	0.495	0.503	0.493

When considering wide network structures, there is an improvement in the co-movement AUC over the narrow networks in the majority of cases. Additionally, for deep networks without dropout layers, the training set co-movement AUC outperforms the test data, and in the cases where $K = M$, the test AUC is great than 0.5. Despite this, overall co-movement AUC is still poor, indicating that the models do not classify co-movement well.

Co-movement MSE is mostly consistent across different network widths. As the width increases, at each choice of K and M, the MSE remains roughly the same. The deep network structure with no dropout layers will perform better than the other network structures when considering wider structures, but these improvements are small. In the test cases, the MSE is also lower than in the training case. Determining possible causes for this discrepancy is an area for future study.

Table 3.47: Co-movement MSE: Ultra-narrow Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	0.232	0.231	0.232	0.234	0.232	0.233
	$K = 10, M = 5$	0.240	0.240	0.240	0.241	0.241	0.241
	$K = M = 10$	0.240	0.240	0.240	0.240	0.241	0.241
Test	$K = M = 5$	0.231	0.229	0.231	0.231	0.231	0.232
	$K = 10, M = 5$	0.240	0.240	0.239	0.240	0.240	0.240
	$K = M = 10$	0.240	0.239	0.239	0.240	0.240	0.240

Table 3.48: Co-movement MSE: Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	0.231	0.231	0.230	0.232	0.232	0.233
	$K = 10, M = 5$	0.240	0.239	0.239	0.240	0.240	0.240
	$K = M = 10$	0.239	0.239	0.239	0.240	0.240	0.240
Test	$K = M = 5$	0.229	0.228	0.229	0.230	0.231	0.231
	$K = 10, M = 5$	0.239	0.239	0.239	0.240	0.240	0.240
	$K = M = 10$	0.239	0.239	0.239	0.239	0.240	0.240

Table 3.49: Co-movement MSE: Wide Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	0.230	0.230	0.229	0.230	0.230	0.230
	$K = 10, M = 5$	0.239	0.239	0.239	0.239	0.239	0.239
	$K = M = 10$	0.239	0.239	0.239	0.239	0.239	0.239
Test	$K = M = 5$	0.227	0.227	0.227	0.227	0.227	0.227
	$K = 10, M = 5$	0.238	0.238	0.238	0.238	0.238	0.239
	$K = M = 10$	0.238	0.238	0.238	0.238	0.238	0.239

Unlike the high market capitalization example, the concordance changes in several cases for the agricultural data. When $K = 10$, the concordance changes with depth, when M is increased, and when the width is changed. Recall that the theoretical maximum of concordance is $\binom{K}{2}$. In the ultra-narrow case, the concordance reaches approximately 19.2% of its theoretical maximum in the training case, for medium networks when $K = M = 10$.

Table 3.50: Co-movement Concordance: Ultra-Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	1.638	1.638	1.638	1.638	1.638	1.638
	$K = 10, M = 5$	7.853	8.241	8.189	7.729	7.695	7.731
	$K = M = 10$	7.906	8.636	8.473	7.794	7.732	7.721
Test	$K = M = 5$	1.555	1.555	1.555	1.555	1.555	1.555
	$K = 10, M = 5$	7.491	7.861	7.813	7.369	7.335	7.731
	$K = M = 10$	7.542	8.237	8.079	7.423	7.371	7.361

Table 3.51: Co-movement Concordance: Narrow Networks

		No Dropout			**Dropout**		
		S	M	D	S	M	D
Train	$K = M = 5$	1.638	1.638	1.638	1.638	1.638	1.638
	$K = 10, M = 5$	8.645	8.700	8.902	8.045	7.816	7.856
	$K = M = 10$	8.820	8.846	8.924	8.240	7.816	7.856
Test	$K = M = 5$	1.555	1.555	1.555	1.555	1.555	1.555
	$K = 10, M = 5$	8.247	8.297	8.489	7.672	7.453	7.494
	$K = M = 10$	8.410	8.844	8.509	7.861	7.454	7.494

When the width is increased, the deep and narrow networks tend to outperform their shallower counterparts. Here, the $K = M = 10$ training concordance value is approximately 19.83% of its theoretical maximum. As in the ultra-narrow case, the training concordance exceeds the test concordance in every case, but the test concordance performs similarly well, The best-performing test case occurs for the medium-depth network where $K = M = 10$. Here the concordance reaches 19.65% of its theoretical maximum. When increasing the width even further, the maximum concordance only increases slightly in the training case, and the maximum test case performance falls. However, the performance across all depths and across both dropout and standard networks is fairly uniform, unlike in narrower networks. The medium depth network with no dropout layer performs the best, reaching 20.11% of its theoretical maximum, while the corresponding test concordance reaches 19.18% percent of the theoretical maximum. With only slightly higher concordance performance in the training case and performance drops in the test case, the narrow networks may be more desirable than wide networks due to their reduced complexity.

Table 3.52: Co-movement Concordance: Wide Networks

		No Dropout			Dropout		
		S	M	D	S	M	D
Train	$K = M = 5$	1.638	1.638	1.638	1.638	1.638	1.638
	$K = 10, M = 5$	9.022	9.050	9.050	9.022	9.039	8.909
	$K = M = 10$	9.039	9.051	9.050	9.044	9.050	8.924
Test	$K = M = 5$	1.555	1.555	1.555	1.555	1.555	1.555
	$K = 10, M = 5$	8.605	8.631	8.631	8.605	8.621	8.495
	$K = M = 10$	8.621	8.631	8.631	8.626	8.631	8.508

3.5.3 Tensor Regression Comparisons

Several key comparisons between the tensor regression model and neural network model must be made. Both real data cases will be considered. In neither case are composite log-likelihood and joint probability log-likelihood compared across models. Due to different parameters estimated between the two models, the log-likelihoods are not directly comparable.

For the high market capitalization case, none of the network structures considered consistently outperform the tensor regression models at classifying joint outcomes. On the training set, the tensor regression regularly outperforms the neural network model. Despite this, there are some neural network structures that will outperform the tensor regression model for some of the test cases. For instance, consider the multi-class AUC for the deep and wide network in contrast to the tensor regression model. While both the rank-one and rank-three tensor regression approximations have a higher training AUC, the deep and wide network classifies joint responses better than either tensor regression model when a five day look-back window is used.

Table 3.53: Multi-class AUC

		MVBNL-TR		**ANN-LR**
		R = 1	R = 3	D/W
Train	$K = M = 5$	0.607	0.629	0.487
	$K = 10, M = 5$	0.663	0.728	0.501
	$K = M = 10$	0.686	0.772	0.480
Test	$K = M = 5$	0.508	0.493	0.526
	$K = 10, M = 5$	0.518	0.510	0.528
	$K = M = 10$	0.542	0.519	0.504

With regards to the frequency of observing maximum estimated probability, any of the networks considered will perform comparably to the tensor regression networks. In the training cases, the neural network model performs better than the rank-one tensor model but worse than the rank-three model. On the test set, this relationship is reversed. Additionally in the five stock case, the neural network model is superior to both of the tensor regression models.

Table 3.54: Observed Maximum Probability Frequency

		MVBNL-TR		**ANN-LR**
		R = 1	R = 3	All
Train	$K = M = 5$	160	172	161
	$K = 10, M = 5$	73	86	77
	$K = M = 10$	71	82	77
Test	$K = M = 5$	142	142	151
	$K = 10, M = 5$	61	50	58
	$K = M = 10$	62	46	58

The tensor regression models outperform the neural network model in the majority of cases in terms of conditional AUC, as shown in Table 3.55. The only instances where network models have a greater average conditional AUC are several wide networks for the five-stock setting. The medium and deep wide networks both have greater conditional AUC, and the medium-depth dropout network has the same conditional AUC as the rank-one network. Consider the deep and wide network. When five stocks are considered, the neural network model is marginally better at classifying a stock's outcome, given the other stock outcomes. However,

there is a performance loss when the parameter space is expanded, whereas the rank-one tensor regression model actually improves at conditional classification as the parameter space increases. This could be a drawback of using the larger and more complex parameter space of the neural network model. This is something that can be studied further.

Table 3.55: Conditional AUC

		MVBNL-TR		**ANN-LR**
		R = 1	R = 3	D/W
Test	$K = M = 5$	0.780	0.775	0.781
	$K = 10, M = 5$	0.785	0.770	0.753
	$K = M = 10$	0.783	0.759	0.753

With regards to co-movement metrics, both models perform poorly in terms of classifying co-movement. This can be seen below in Table 3.56. The tensor regression models classify co-movement outcomes better in several training set cases. Despite this, the neural network models can substantially outperform the tensor regression models in some test set cases. This can be seen for the ultra-narrow deep network with dropout layers.

Table 3.56: Co-movement AUC

		MVBNL-TR		**ANN-LR**
		R = 1	R = 3	UN/D/Dr
Train	$K = M = 5$	0.495	0.519	0.516
	$K = 10, M = 5$	0.511	0.519	0.512
	$K = M = 10$	0.510	0.545	0.503
Test	$K = M = 5$	0.500	0.490	0.539
	$K = 10, M = 5$	0.482	0.488	0.524
	$K = M = 10$	0.501	0.492	0.500

While co-movement AUC is slightly higher for the rank-three model in the training case, the test AUC is substantially higher in the five-day look-back window cases for the neural network model. When $K = M = 10$, the co-movement AUCs are very close but the tensor regression model performs slightly better. Regardless of this, both models have poor co-movement AUC. Additionally, the co-movement AUC is greater for some of the test cases than in the training

cases for the neural network model.

In terms of co-movement MSE, the neural network models have consistently higher estimates for MSE. These differences are small in most cases. This indicates that while neural network models are performing comparably, the tensor regression models are slightly better at minimizing the MSE. The neural network models consistently outperform both tensor regression models in terms of maximizing the co-movement concordance. This can be seen in even the simplest neural network models, such as the shallow ultra-narrow network.

Table 3.57: Co-movement Concordance

		MVBNL-TR		**ANN-LR**
		R = 1	R = 3	S/UN
Train	$K = M = 5$	1.509	1.506	1.510
	$K = 10, M = 5$	8.250	8.202	8.256
	$K = M = 10$	8.248	8.151	8.256
Test	$K = M = 5$	1.538	1.538	1.539
	$K = 10, M = 5$	8.477	8.421	8.482
	$K = M = 10$	8.465	8.335	8.482

In the agricultural stock case, the tensor regression models have stronger training set multi-class AUC performance. However, some ANN-LR models have stronger test set performance. For instance, the shallow and wide dropout model outperforms the tensor regression model on the test set in the ten stock case. Despite this better test set performance, the training-test set discrepancy is still present in the multi-class AUC for the neural network model. The fact that test set performance exceeds training set performance needs to be investigated further.

Table 3.58: Multi-class AUC

		MVBNL-TR		**ANN-LR**
		R = 1	R = 3	S/W/Dr
Train	$K = M = 5$	0.558	0.597	0.498
	$K = 10, M = 5$	0.657	0.710	0.483
	$K = M = 10$	0.687	0.762	0.506
Test	$K = M = 5$	0.506	0.521	0.458
	$K = 10, M = 5$	0.507	0.532	0.533
	$K = M = 10$	0.511	0.509	0.527

The frequency of observing the maximum estimated probability also faces a similar discrepancy between training and test data performance. This can be seen in Table 3.59 in the shallow and wide ANN-LR case, as well as each of the tensor regression cases. In each case, there are fewer times when the maximum estimated probability is observed on the training set than on the test set. While good test performance is desirable, better performance with a completely new set of data is unexpected. This persists for several metrics, and merits further study.

Table 3.59: Observed Maximum Probability Frequency

		MVBNL-TR		**ANN-LR**
		R = 1	R = 3	S/W
Train	$K = M = 5$	126	144	120
	$K = 10, M = 5$	43	42	28
	$K = M = 10$	45	49	28
Test	$K = M = 5$	133	144	139
	$K = 10, M = 5$	54	49	45
	$K = M = 10$	48	42	45

The average conditional AUC is consistently lower for the neural network models. In terms of measuring how well ANN-LR is classifying co-movement, peak model co-movement AUC lies between the performance of the rank-one and rank-three approximation tensor regression models in the test case. Note that both models are relatively poor at classifying co-movement.

Table 3.60: Co-movement AUC

		MVBNL-TR		**ANN-LR**
		R = 1	R = 3	S/W
Train	$K = M = 5$	0.515	0.528	0.518
	$K = 10, M = 5$	0.512	0.531	0.506
	$K = M = 10$	0.525	0.547	0.508
Test	$K = M = 5$	0.510	0.515	0.511
	$K = 10, M = 5$	0.495	0.505	0.497
	$K = M = 10$	0.499	0.511	0.507

In terms of minimizing co-movement MSE, the neural network models are consistently worse. For concordance, the tensor regression models perform slightly better than the neural network models in most cases. However, in the five stock case, the neural network model performs the same or better. When $K = M = 5$, the concordance using the training data is 1.638 for both models. When evaluating the fit on the test set, the concordance in the five stock neural network model case is 1.555, slightly larger than 1.553, the concordance value produced by the tensor regression model.

3.6 Conclusions

3.6.1 Summary of ANN-LR Results

There are several important results that need to be emphasized. For several metrics, it seems as if the narrow-width networks seem to be the best choice of neural network model. As increasing in width from the narrow to the wide case greatly increases the number of parameters in the model, the metrics would need to improve a good deal to justify choosing a wider model. Most of the metrics only improve slightly when increasing to a wider case. Some models actually worsen on some metrics. For instance, the composite log-likelihood is maximized better in the narrow case than the wide case. The same is true for the joint probability log-likelihood. For some of the metrics, such as conditional AUC and co-movement AUC, the metrics improve as the model is widened, but the improvements are not very large. In the high market capitalization study, the concordance stays the same across all width options, as does the frequency of observing the maximum probability. Co-movement and joint multi-class AUC are at their worst in the narrow case. These metrics are also poor at all settings, so poorer performance in the narrow case should hold less weight on model choice.

When considering the agricultural sector case, the patterns for the composite and joint probability log-likelihoods are similar. Conditional AUC and co-movement MSE also indicate only small improvements from utilizing the wider models. The frequency of observing the maximum probability does improve when increasing to the wide case. Again, co-movement and multi-class AUC perform poor on both training and test sets at all network width options. Due to this, poor classification performance in the narrow case should not be the sole factor motivating model choice. In terms of co-movement concordance, the improvement from choosing the narrow case over the ultra-narrow case is more pronounced than further extending the scope of the network to a wide structure.

From analyzing the results of the ANN-LR model options for the two data sets considered, the narrow network structure seems to be the superior choice. Of the depth options possible,

when considering narrow network performance, the deep networks without dropout layers generally perform the best, or at least similar to other network depths. However, their performance in relation to the tensor regression models must also be considered.

3.6.2 Summary of Model Comparisons

First the high market capitalization data should be considered. In terms of multi-class and co-movement AUC, the neural network model performs worse on the training data, but better in some test data cases. For both metrics the neural network models are superior on test data when the look-back window is five days long. When five stocks are considered, there are neural network models which are superior for conditional AUC. Co-movement concordance is also marginally greater for the neural network model. The multi-class AUC and co-movement AUC are superior in the neural network model for the deep wide model and the deep ultra-narrow dropout model, respectively. The conditional AUC case listed where the neural network model is superior for $K = M = 5$ is also the deep wide network model. The concordance is the same across all width options for the model, so the neural network model benefit would is present for all width options. To gain a slight performance edge, it seems that the choice of model depends on the metrics one chooses to prioritize. Narrower models are acceptable when considering co-movement concordance, AUC, MSE, as well as the frequency of observing the maximum probability. However, for better multi-class AUC and conditional AUC, wider models are needed.

For the agricultural stock data, wide models are also best for comparable multi-class AUC and co-movement AUC on the test data. However, co-movement AUC is consistently worse for neural network models in this sector-specific case. Additionally, conditional AUC is worse on average for the neural network models. Co-movement concordance is higher for narrow and wide neural network models when $K = M = 5$, but tensor regression models perform better for larger K and M.

Overall, when compared to the tensor regression models, the neural network model performance is only superior in a few cases. Several of these cases involve wide networks. Wide network models have considerably larger parameter spaces to consider. The wider networks primary strengths were marginal improvements in joint outcome and conditional outcome classification. The improvements are minimal, and choosing models based on higher frequency of observing the maximum probability and co-movement concordance would motivate choosing a simpler model. When deciding between which models to use, one has to consider the size of the parameter space in conjunction with which metrics are most relevant for assessing performance benefits. From this one can then make a decision on which model to use.

3.6.3 Areas for Future Study

There are multiple areas that merit further study. As highlighted in the discussion of the results, there are many instances of test set performance being greater than the training set performance. It is desirable for the model fit on the training data to also perform similarly well when evaluated on the test data. However, having consistently higher test set performance is unusual. Several of the cases where better test performance is observed involve AUC metrics. As the training AUC is consistently near and sometimes below 0.5 for these cases, this indicates that randomly choosing between two outcomes would have similar performance. Having test set performance that is marginally better than training set performance in such a case may be possible. Another possibility that merits further investigation is the effect of the re-coded response data on metrics for test set data. The top 5% of responses are not re-coded in any of the test set cases. This is due to the fact that in the simulated case, a 75%-25% training/test split is used. The smaller data set with have fewer possible outcomes and therefore less of the total outcome space is likely to be observed. However, in cases with a 50%-50% even split, perhaps the responses could be re-coded using a less stringent criterion.

Assessing model performance for cases involving more stocks is another area for future study. While more stocks will result in longer computation times, the computation hurdles that are more difficult to surmount involve the creation of the total sampling space. This step involves creating a matrix to represent the sampling space consisting of 2^K possible outcomes for K stocks. This matrix is vital for the computation of the joint probabilities, which in turn is necessary for several metrics. The principal idea behind generating this matrix is creating a binary expansion for a vector of integers 0 through $2^K - 1$. The function `intToBin` in `R.utils` is what is used to perform this step. Perhaps a function in C++ with a wrapper function in R would accelerate computation. This is an option that can be explored.

When $K = 20$, the sample space matrix will have over a million rows. This is difficult to store in memory. Any matrix computation that uses this matrix will be affected. Storing the matrix as a sparse matrix will reduce some of the memory load, but in an example involving twenty stocks, the size of S would be reduced from 167.8 MB to 125.8 MB. To compare, when $K = 10$, S is around 82.14 kilobytes when stored in memory. As a matrix S has order $2^K \times K$, doubling K will result in matrix that is $2^{(K+1)}$ larger. When increasing K from 10 to 20, the proposed size multiplier would be 2048. 2048×82.14 KB is approximately 168.2 MB, close to what is observed.

The full sampling space will be necessary for the computation of joint probability. Performing matrix operations with this many rows will also slow computation for the metrics considered. The reason that the full space is needed is because the joint probability $P(\mathbf{Y}_i = \mathbf{y}_i) = \frac{\exp(\mu_{y_i})}{\sum_{s_i \in S} \exp(\mu_{s_i})}$ can not be computed on the basis of which outcomes are present in the data and which are not. Until being fully computed, $\sum_{s_i \in S} \exp(\mu_{s_i})$ is an unknown quantity. Thus, probability

computation can't be simplified based on which s_i are present in $\mathbf{Y}$.

Lastly, different network structures can be considered. It is possible that different structures can form better representations of the relationships between multiple stocks. A change in proposed network structure would result in different structures being used in the neural network update step. Other than updating the network structures being used, the overall alternating update framework would remain similar to what is currently proposed.

REFERENCES

Amemiya, T. (1978). On a two-step estimation of a multivariate logit model. *Journal of Econometrics*, 8(1):13–21.

Arkoudi, I., Krueger, R., Azevedo, C. L., and Pereira, F. C. (2023). Combining discrete choice models and neural networks through embeddings: Formulation, interpretability and performance. *Transportation Research Part B: Methodological*, 175:102783.

Bel, K., Fok, D., and Paap, R. (2014). Parameter estimation in multivariate logit models with many binary choices. Econometric Institute Research Papers EI 2014-25, Erasmus University Rotterdam, Erasmus School of Economics (ESE), Econometric Institute.

Bel, K. and Paap, R. (2014). A multivariate model for multinomial choices. Econometric Institute Research Papers EI 2014-26, Erasmus University Rotterdam, Erasmus School of Economics (ESE), Econometric Institute.

Ben-Akiva, M. and Lerman, S. R. (1985). *Discrete choice analysis: Theory and application to travel demand*. The MIT Press, Boston, MA.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):pp. 192–236.

Burda, M., Harding, M., and Hausman, J. (2008). A bayesian mixed logit-probit model for multinomial choice. *Journal of Econometrics*, 147(2):232–246.

Caffo, B. S., Crainiceanu, C. M., Verduzco, G., Joel, S., Mostofsky, S. H., Bassett, S. S., and Pekar, J. J. (2010). Two-stage decompositions for the analysis of functional connectivity for fmri with application to alzheimer's disease risk. *Neuroimage*, 51:1140–1149.

Chang, S.-M., Yang, M., Lu, W., Huang, Y.-J., Huang, Y., Hung, H., Miecznikowski, J. C., Lu, T.-P., and Tzeng, J.-Y. (2021). Gene-set integrative analysis of multi-omics data using tensor-based association test. *Bioinformatics*, 37(16):2259–2265.

Chollet, F. (2015). Keras. `https://github.com/fchollet/keras`.

Chollet, F. (2017). *Deep Learning with Python*. Manning. `http://www.deeplearningbook.org`.

de Leeuw, J., Young, F., and Takane, Y. (1976). Additive structure in qualitative data: An alternating least squares method with optimal scaling features. *Psychometrika*, 41:471–503.

de Rooij, M. and Kroonenberg, P. (2003). Multivariate multinomial logit models for dyadic sequential interaction data. *Multivariate Behavioral Research*, 38:463–504.

Geweke, J., Keane, M., and Runkle, D. (1994). Alternative computational approaches to inference in the multinomial probit model. *The Review of Economics and Statistics*, 76(4):609–632.

Geweke, J., Keane, M., and Runkle, D. E. (1997). Statistical inference in the multinomial multiperiod probit model. *Journal of Econometrics*, 80(1):125–165.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Hand, D. J. and Till, R. J. (2001). A simple generalisation of the area under the roc curve for multiple class classification problems. *Mach. Learn.*, 45(2):171–186.

Hausman, J. and Wise, D. (1978). A conditional probit model for qualitative choice: Discrete decisions recognizing interdependence and heterogeneous preferences. *Econometrica*, 46(2):403–26.

Hensher, D. and Greene, W. (2003). The mixed logit model: The state of practice. *Transportation*, 30(2):133–176.

Hung, H. and Jou, Z.-Y. (2019). A low rank-based estimation-testing procedure for matrix-covariate regression. *Statistica Sinica*, **29**(2):1025–1046.

Hung, H., Lin, Y.-T., Chen, P., Wang, C.-C., Huang, S.-Y., and Tzeng, J.-Y. (2016). Detection of gene–gene interactions using multistage sparse and low-rank regression. *Biometrics*, 72:85–94.

Lange, K. (2010). *Numerical Analysis for Statisticians, Second Edition*. Statistics and Computing. Springer; New York.

Lazar, N. (2008). *The Statistical Analysis of Functional MRI Data*. Springer; New York.

Li, Y., Zhu, H., Shen, D., Lin, W., Gilmore, J. H., and Ibrahim, J. G. (2011). Multiscale adaptive regression models for neuroimaging data. *Journal of the Royal Statistical Society Series B*, 73(4):559–578.

Maddala, G. S. (1986). *Limited-Dependent and Qualitative Variables in Econometrics*. Cambridge University Press.

McFadden, D. (1974). The measurement of urban travel demand. *Journal of Public Economics*, 3(4):303–328.

McFadden, D. (1983). *Handbook of Econometrics*. North-Holland Publishing Company, Amsterdam.

Meng, C., Zeleznik, O. A., Thallinger, G. G., Kuster, B., Gholami, A. M., and Culhane, A. C. (2016). Dimension reduction techniques for the integrative analysis of multi-omics data. *Briefings in Bioinformatics*, 17(4):628–641.

Reiss, P. T. and Ogden, R. T. (2010). Functional generalized linear models with images as predictors. *Biometrics*, 66:61–69.

Russell, G. J. and Petersen, A. (2000). Analysis of cross category dependence in market basket selection. *Journal of Retailing*, 76(3):367–392.

Sifringer, B., Lurkin, V., and Alahi, A. (2020). Enhancing discrete choice models with representation learning. *Transportation Research Part B: Methodological*, 140:236–261.

Varin, C., Reid, N., and Firth, D. (2011). An overview of composite likelihood methods. *Statistica Sinica*, 21(1):5–42.

Wu, C., Zhou, F., Ren, J., Li, X., Jiang, Y., and Ma, S. (2019). A selective review of multi-level omics data integration using variable selection. *High-Throughput*, 8:4.

Yue, Y., Loh, J. M., and Lundquist, M. A. (2010). Adaptive spatial smoothing of fmri images. *Statistics and its Interface*, 3:3–13.

Zhou, H., Li, L., and Zhu, H. (2013). Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552.

APPENDICES

APPENDIX

A

DATA SIMULATION

This appendix provides a summary of the simulated data generation process.

Creating of data tensor $\mathscr{X}$ with day slices $\mathbf{X}_i$:

1. An index of length T is initialized
2. 75% of the index is randomly sampled and assigned to the training set, the remainder to the test set
3. Perform K random uniform (U(-1,1)) draws of length $T+M+1$, these are to simulate the relative price changes for the stocks, resulting in a $K \times (T+M+1)$ matrix (T is the length of training/test combined)
4. A $K \times M \times T$ data tensor $\mathscr{X}$ is constructed.
5. These slices are then arranged in a $K \times M \times \frac{4}{3} \cdot T_{max}$ data tensor, which is then split into the first T_{max} days (training set) and remaining $\frac{1}{3} \cdot T_{max}$ days (test set) by randomly sampling T_{max} slices from the third dimension full data tensor, which is the training set, and the test set is comprised of the remaining slices

To simulate the response vector $\boldsymbol{y}$, there are several intermediate steps which must be completed first. The steps are similar for the MV-BNL model and its tensor regression extension. In the simulation of $\boldsymbol{y}$, the true parameter values $\mathbf{B}_k, \boldsymbol{\alpha}, \boldsymbol{\psi}$ are used. For the tensor regression case, the generation of the response is as follows:

1. $\langle \mathbf{X}_i, \mathbf{B}_k \rangle$ is constructed for all i days and k stocks.

2. For all days and possible outcomes compute $P(Y_i = y_i|X_i) = \frac{\exp(\mu_{y_i})}{\sum_{s_i \in S} \exp(\mu_{s_i})}$, $\mu_{y_i} = \sum_{k=1}^{K} y_{ik}(\alpha_k + \langle \mathbf{X}_i, \mathbf{B}_k \rangle) + \sum_{l>k} y_{ik} y_{il} \psi_{kl}$, these results will be stored in a $T_{max} \times 2^K$ probability matrix S.

3. For each day, one of 2^K outcomes s_i (of length K, denoting possible binary response for K stocks) from a total realization space S is generated. This outcome is generated by randomly drawing one outcome from 1 to 2^K where the probability of drawing such an outcome corresponds to the probability vector for each day (row i of matrix S.

The same approach can be followed for constructing the response in the test set, using the proper data for computation. The process is similar for the MV-BNL model, except that $\mu_{y_i} = \sum_{k=1}^{K} y_{ik}(\alpha_k + X_i \beta_k) + \sum_{l>k} y_{ik} y_{il} \psi_{kl}$.

APPENDIX

B

METRICS

Bias and standard deviation of the estimators are straightforwardly calculated. Bias for any of the parameters considered is simply the difference between the average of the estimates for a given parameter over all simulations and the true parameter. Similarly, the variance of a parameter estimate here is $Var(\hat{\theta}) = \frac{1}{n}\sum_j(\hat{\theta}_j - \theta)^2$, where θ_j denotes the estimate of parameter θ in simulation j.

The AIC for both models is constructed in the following manner: AIC = 2·Parameters $-2\cdot\hat{L}$, where $\hat{L}$ is the maximum value of the composite log-likelihood. The number of parameters for the MV-BNL model is $K + K\cdot(K-1)/2 + K\cdot p$. In the simulation study comparison case is, $p = K\cdot M$. For the tensor regression extension, the number of parameters is $K + K\cdot(K-1)/2 + K\cdot(K\cdot R + M\cdot R)$.

The definition of an ROC curve is as follows. Let $\hat{p}(x)$ denote the probability of x belonging to class 0. The ROC curve is a plot of the cumulative distribution function $G(\hat{p}) = G(\hat{p}(x)|1)$ on the vertical axis against $F(\hat{p}) = F(\hat{p}(x)|0)$ on the horizontal axis. These are cumulative distribution functions for corresponding probability functions $g(\hat{p}(x)|1)$, $f(\hat{p}(x)|0)$. These probability functions denote the estimated probability of belonging to class 0 for points belonging to class 1 and class 0, respectively. As this plot is bounded by the unit square, a diagonal line from (0,0) to (1,1) denotes a fit no better than random chance. This is due to the fact that the AUC in such an example is 0.5, generating 0 or 1 by random chance will be as effective at classifying the response correctly. (Hand and Till 2001, p. 173)

As laid out in the simulation study, multi-class AUC is the average of all unique AUC pairs. For 2^K possible outcomes, the multi-class AUC will be $\frac{2}{2^K(2^K-1)}\sum_{i<j}\hat{A}(i,j)$, where $\hat{A}(i,j)$ is the AUC for classes i, j (Hand and Till 2001). Computation for all possible outcomes is infeasible.

For large values of K there are more possible outcomes than days being considered. In this study, data is transformed so that only a top percentage of the outcomes are considered. Other outcomes observed are re-coded to a baseline to which top outcomes are compared. For training data the top 25% of outcomes are compared to each other and a baseline. For test data, the top 5% of outcomes are considered.

Having a multi-class AUC close to one is desired. To compute the multi-class AUC for these models, the function `multiclass.roc` from the R package `pROC` is utilized. To compute AUC, the function requires a response vector with multiple classes, as well a matrix giving the estimated probability of each class occurring each day.

APPENDIX

C

METHODS

C.1 Multivariate Binomial Logit Model

The methods for the multivariate binomial logit specification are laid out in Bel and Paap (2014) and have already been covered in the literature review. Finer details of the implementation used in the simulation study are included in the code section of this appendix.

C.2 Multivariate Binomial Logit Tensor Regression Model

There are some preliminary notation details that should be specified.

Recall that the model here is such that $Z_{ik} = \boldsymbol{\gamma}_{ik}^T \boldsymbol{\beta} + \langle \mathbf{X}_i, \mathbf{B}_k \rangle + \epsilon_{ik}, \mathbf{B}_k = \mathbf{B}_{k1}\mathbf{B}_{k2}^T$. $\boldsymbol{\gamma}_{ik}^T \boldsymbol{\beta} = \alpha_k + \sum_{l \neq k} y_{il}\psi_{kl}$. Here $\boldsymbol{\gamma}_{ik}$ is a vector containing 1's and 0's corresponding to the intercept and association parameters present for a particular day and stock choice.

$$
\begin{aligned}
\langle \mathbf{X}_i, \mathbf{B}_{k1}\mathbf{B}_{k2}^\top \rangle &= \text{tr}(\mathbf{X}_i \mathbf{B}_{k1}\mathbf{B}_{k2}^\top) \\
&= \text{tr}(\mathbf{B}_{k2}^\top \mathbf{X}_i^\top \mathbf{B}_{k1}) \\
\mathscr{A}(\mathbf{B}_{k1}^{(j)}) &= \begin{bmatrix} vec(\mathbf{X}_1^\top \mathbf{B_{k1}})^\top \\ \vdots \\ vec(\mathbf{X}_T^\top \mathbf{B_{k1}})^\top \end{bmatrix} \\
\mathscr{B}(\mathbf{B}_{k2}^{(j+1)}) &= \begin{bmatrix} vec(\mathbf{X}_1 \mathbf{B_{k2}})^\top \\ \vdots \\ vec(\mathbf{X}_T \mathbf{B_{k2}})^\top \end{bmatrix} \\
E(y_{ik}|\mathbf{X}_i, \boldsymbol{\gamma}_{ik}) &= \boldsymbol{\gamma}_{ik}^\top \boldsymbol{\beta} + \langle \mathbf{X}_i, \mathbf{B}_{k1}\mathbf{B}_{k2}^\top \rangle \\
&= \boldsymbol{\gamma}_{ik}^\top \boldsymbol{\beta} + \langle \mathbf{X}_i^\top \mathbf{B}_{k1}, \mathbf{B}_{k2} \rangle = \boldsymbol{\gamma}_{ik}^\top \boldsymbol{\beta} + \langle [\mathscr{A}(\mathbf{B}_{k1})]_i, \mathbf{B}_{k2} \rangle \\
&= \boldsymbol{\gamma}_{ik}^\top \boldsymbol{\beta} + \langle \mathbf{X}_i \mathbf{B}_{k2}, \mathbf{B}_k \rangle = \boldsymbol{\gamma}_{ik}^\top \boldsymbol{\beta} + \langle [\mathscr{B}(\mathbf{B}_{k2})]_i, \mathbf{B}_1 \rangle
\end{aligned}
$$

While $\mathscr{X}$ is a $K \times M \times T$ tensor, it is beneficial to see how the covariates for intercept and association parameters as well as the tensor covariates can be expressed using a full augmented design matrix. In practice, we will see that this design matrix is not used in full in a particular step of the algorithm. Instead, at times certain parameters are fixed. However, it is important to make clear what the constituent parts of the design matrix can be. This is seen below:

$$
\mathbf{X}^* = \left[\begin{array}{c|cccc}
\boldsymbol{\Gamma} & \text{vec}(\mathbf{X}_1)^\top & \mathbf{0}_{K\cdot M} & \dots & \mathbf{0}_{K\cdot M} \\
& \mathbf{0}_{K\cdot M} & \text{vec}(\mathbf{X}_1)^\top & \dots & \mathbf{0}_{K\cdot M} \\
& \vdots & \dots & \ddots & \vdots \\
& \mathbf{0}_{K\cdot M} & \cdots & \mathbf{0}_{K\cdot M} & \text{vec}(\mathbf{X}_1)^\top \\
& \vdots & \vdots & \vdots & \vdots \\
& \text{vec}(\mathbf{X}_T)^\top & \mathbf{0}_{K\cdot M} & \dots & \mathbf{0}_{K\cdot M} \\
& \mathbf{0}_{K\cdot M} & \text{vec}(\mathbf{X}_T)^\top & \dots & \mathbf{0}_{K\cdot M} \\
& \vdots & \dots & \ddots & \vdots \\
& \mathbf{0}_{K\cdot M} & \cdots & \mathbf{0}_{K\cdot M} & \text{vec}(\mathbf{X}_T)^\top
\end{array}\right] \tag{C.1}
$$

$$\boldsymbol{\Gamma} = \begin{bmatrix} \mathbf{I}_K & \mathbf{y}_{11}^{\top\prime} \\ & \mathbf{y}_{12}^{\top\prime} \\ & \vdots \\ & \mathbf{y}_{1K}^{\top\prime} \\ & \vdots \\ \mathbf{I}_K & \mathbf{y}_{T1}^{\top\prime} \\ & \mathbf{y}_{T2}^{\top\prime} \\ & \vdots \\ & \mathbf{y}_{TK}^{\top\prime} \end{bmatrix} \tag{C.2}$$

$$\boldsymbol{\beta}^* = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_K \\ \psi_{12} \\ \vdots \\ \psi_{1K} \\ \vdots \\ \psi_{(K-1)K} \\ \mathbf{B}_{1[,1]} \\ \vdots \\ \mathbf{B}_{1[,M]} \\ \vdots \\ \mathbf{B}_{K[,1]} \\ \vdots \\ \mathbf{B}_{K[,M]} \end{bmatrix} \tag{C.3}$$

For reference, the term $y_{ik}^{\top\prime}$ is of length $K \cdot (K-1)/2$. To understand of this vector, recall that $\boldsymbol{\psi}$ is a symmetric matrix. Only the upper triangle of the matrix (not including diagonal elements) needs to be estimated. Thus, the K elements of y_i are indexed to an element of $y_{ik}^{\top\prime}$ based on which $\boldsymbol{\psi}$ term they correspond to. This index is constructed by denoting the ordering the elements of the upper triangle of $\boldsymbol{\psi}$ in a row-wise fashion. To get the elements of the index relevant to a response, the process is simple. When $k = K$, the last column of the upper triangle of $\boldsymbol{\psi}$ denotes the corresponding index for $y_{ik}^{\top\prime}$ to which elements of y_i are assigned. For other k, this index is the kth column of the index matrix for rows 1 through k and the kth row for columns $k+1$ through K. However, for a given day i and choice k, the kth element is removed. For example, for $k = 1$ the indices of $y_{ik}^{\top\prime}$ that are selected contain the second through the Kth entries of y_i.

Consider the following trivial example where $K = 3$, where for day 1, $y_1 = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$. For

$Z_{11}, \sum_{l \neq k} y_{il}\psi_{kl} = \psi_{12} + \psi_{13}$. Using the method proposed to select indices, the first column for the first row, then the remaining columns for the first row are selected. Storing this in an augmented vector, the first response is not included. As such, the second two terms are included, but are assigned to the second and third elements of $y_{ik}^{\top\prime}$, as these are the indexed elements to which the response values y_{12}, y_{13} are assigned.

The second partition of $\mathbf{X}^*$ can be simplified. For each day i, the corresponding portion of the second partition of can be expressed as a block diagonal matrix. Here each block is of the form $\text{vec}(\mathbf{X}_i)^\top$. A block diagonal matrix can be expressed.

$$BD(\mathbf{X}_i) = \begin{bmatrix} \text{vec}(\mathbf{X}_i)^\top & \mathbf{0}_{K\cdot M} & \cdots & \mathbf{0}_{K\cdot M} \\ \mathbf{0}_{K\cdot M} & \text{vec}(\mathbf{X}_i)^\top & \cdots & \mathbf{0}_{K\cdot M} \\ \vdots & \cdots & \ddots & \vdots \\ \mathbf{0}_{K\cdot M} & \cdots & \mathbf{0}_{K\cdot M} & \text{vec}(\mathbf{X}_i)^\top \end{bmatrix} \mathbf{X}^* = \left[\begin{array}{c|c} \boldsymbol{\Gamma} & BD(\mathbf{X_1}) \\ & \vdots \\ & BD(\mathbf{X_T}) \end{array}\right]$$

The model being proposed here differs somewhat from the existing ALS model. As the goal is to modify this model to implement aspects of the model in Bel-Paap (2014), the model of interest remains a logistic model with a binary response. As such, while the alternating idea will persist in this model, the model being considered will instead be an Alternating Logistic Regression (ALR) model.

Algorithm 6 Tensor Regression in order-3 case implemented using Alternating Logistic Regression

procedure ALR($\mathbf{B}_1^{(0)}, \mathbf{B}_2^{(0)}, \boldsymbol{\beta}^{(0)}$) ▷ Initialize $\mathbf{B}_1^{(0)}, \mathbf{B}_2^{(0)}, \boldsymbol{\beta}^{(0)}$
  repeat
  for all k
  $\mathbf{B}_2^{(j+1)} = LR(\mathbf{B}_{12}, \ldots, \mathbf{B}_{K2}|, \mathbf{y}, \{\boldsymbol{\Gamma}, \mathscr{A}(\mathbf{B}_1^{(j)})\}, \boldsymbol{\beta}^{(j)})$
  $\mathbf{B}_1^{(j+1)} = LR(\mathbf{B}_{11}, \ldots, \mathbf{B}_{K1}|, \mathbf{y}, \{\boldsymbol{\Gamma}, \mathscr{B}(\mathbf{B}_2^{(j+1)})\}, \boldsymbol{\beta}^{(j)})$
  $\boldsymbol{\beta}^{(j+1)} = LR(\boldsymbol{\beta}|, \mathbf{y}, \boldsymbol{\Gamma}, \mathbf{B}_1^{(j+1)}, \mathbf{B}_2^{(j+1)})$
  until convergence
end procedure

$LR(\cdots|\cdots)$ here denotes a logistic regression problem with respect to the first variable given the other information. As such, the estimation of the coefficients in a tensor regression problem will be a series of these logistic regression problems.

To find $\mathbf{B}_2^{(j+1)} = LR(\mathbf{B}_{12}, \ldots, \mathbf{B}_{K2}|, \mathbf{y}, \{\boldsymbol{\Gamma}, \mathscr{A}(\mathbf{B}_1^{(j)})\}, \boldsymbol{\beta}^{(j)})$, there are a few preliminaries to consider. For this model, a logistic model is fit where the α_k and ψ parameters are held fixed. That is, these parameters need to be appropriately *offset*. In practice, one instructs software to offset data in a particular way. For this algorithm, the offset variables are fixed to the previous parameter

estimates. For the estimation of both $\mathbf{B}_{k1}$, $\mathbf{B}_{k2}$, the terms $\alpha_1,\ldots,\alpha_K,\psi_{12},\ldots,\psi_{(K-1)\cdot K}$ are held fixed. The offset is expressed as the sum of linear terms to be offset. This is acceptable as an offset of the sums here is treated the same way as the sum of offsets.

Here, we are considering the following term to be the offset:

$$\alpha_1^{(j)}\cdot\mathbf{\Gamma}_{[,1]}+\cdots\alpha_K^{(j)}\cdot\mathbf{\Gamma}_{[,K]}^{(j)}+\psi_{12}^{(j)}\cdot\mathbf{\Gamma}_{[,K+1]}+\cdots\psi_{K(K-1)}^{(j)}\cdot\mathbf{\Gamma}_{[,K+\frac{K\cdot(K-1)}{2}]}$$

When estimating $\mathbf{B}_2^{(j+1)}$, we consider a different augmented design matrix. To construct this design matrix, the term $\langle\mathbf{X}_i,\mathbf{B}\rangle$ in the model needs to be redefined.

$$\begin{aligned}\langle\mathbf{X}_i,\mathbf{B}_k\rangle &= tr(\mathbf{X}_i^\top\mathbf{B}_{k1}\mathbf{B}_{k2}^\top)=tr(\mathbf{B}_{k2}^\top\mathbf{X}_i^\top\mathbf{B}_{k1})\\ &=\langle\mathbf{X}_i^\top\mathbf{B}_{k1},\mathbf{B}_{k2}\rangle\\ &=\mathrm{vec}(\mathbf{X}_i^\top\mathbf{B}_{k1})^\top\mathrm{vec}(\mathbf{B}_{k2})\end{aligned}$$

Seeing as we can express $\langle\mathbf{X}_i,\mathbf{B}_k\rangle$ as $\langle\mathbf{X}_i^\top\mathbf{B}_{k1},\mathbf{B}_2\rangle$, we can similarly re-express this part of the design matrix to produce a new augmented design matrix and a new corresponding parameter vector. Thus, the corresponding augmented design matrix (dimension $(T\cdot K)\times(K\cdot M\cdot R)$) for this logistic model will be

$$\tilde{\mathbf{X}}=\begin{bmatrix}\mathrm{vec}(\mathbf{X}_1^\top\mathbf{B}_{11})^\top & \mathbf{0}_{M\cdot R} & \cdots & \mathbf{0}_{M\cdot R}\\ \mathbf{0}_{M\cdot R} & \mathrm{vec}(\mathbf{X}_1^\top\mathbf{B}_{21})^\top & \cdots & \mathbf{0}_{M\cdot R}\\ \vdots & \cdots & \ddots & \vdots\\ \mathbf{0}_{M\cdot R} & \cdots & \mathbf{0}_{M\cdot R} & \mathrm{vec}(\mathbf{X}_1^\top\mathbf{B}_{K1})^\top\\ \vdots & \vdots & \vdots & \vdots\\ \mathrm{vec}(\mathbf{X}_T^\top\mathbf{B}_{11})^\top & \mathbf{0}_{M\cdot R} & \cdots & \mathbf{0}_{M\cdot R}\\ \mathbf{0}_{M\cdot R} & \mathrm{vec}(\mathbf{X}_T^\top\mathbf{B}_{21})^\top & \cdots & \mathbf{0}_{M\cdot R}\\ \vdots & \cdots & \ddots & \vdots\\ \mathbf{0}_{M\cdot R} & \cdots & \mathbf{0}_{M\cdot R} & \mathrm{vec}(\mathbf{X}_T^\top\mathbf{B}_{K1})^\top\end{bmatrix} \tag{C.4}$$

$$\tilde{\boldsymbol{\beta}}=\begin{bmatrix}\mathbf{B}_{12[,1]}\\ \vdots\\ \mathbf{B}_{12[,R]}\\ \vdots\\ \mathbf{B}_{K2[,1]}\\ \vdots\\ \mathbf{B}_{K2[,R]}\end{bmatrix} \tag{C.5}$$

Using the elements of $\tilde{\boldsymbol{\beta}}$, the estimated matrix parameters $\mathbf{B}_{k2}^{(j+1)}$ can be constructed.

Next, we consider $\mathbf{B}_1^{(j+1)} = LR(\mathbf{B}_{11},\ldots,\mathbf{B}_{K1}|,\mathbf{y},\{\boldsymbol{\Gamma},\mathscr{B}(\mathbf{B}_2^{(j)})\},\boldsymbol{\beta}^{(j)})$. $\boldsymbol{\beta}^{(j)}$ is held fixed here as before, so the offsets do not change from the previous offsets.

$$\begin{aligned}\langle \mathbf{X}_i,\mathbf{B}_k\rangle &= tr(\mathbf{X}_i^\top \mathbf{B}_{k1}\mathbf{B}_{k2}^\top) = tr(\mathbf{B}_{k2}^\top \mathbf{X}_i^\top \mathbf{B}_{k1}) \\ &= tr(\mathbf{B}_{k1}^\top \mathbf{X}_i \mathbf{B}_{k2}) \\ &= \langle \mathbf{X}_i\mathbf{B}_{k2},\mathbf{B}_{k1}\rangle \\ &= \mathrm{vec}(\mathbf{X}_i\mathbf{B}_{k2})^\top \mathrm{vec}(\mathbf{B}_{k1})\end{aligned} \tag{C.6}$$

Thus, the corresponding augmented design matrix (dimension $(T\cdot K)\times(K\cdot K\cdot R)$) for this logistic model will be

$$\dot{\mathbf{X}} = \begin{bmatrix} \mathrm{vec}(\mathbf{X}_1\mathbf{B}_{12})^\top & \mathbf{0}_{K\cdot R} & \cdots & \mathbf{0}_{K\cdot R} \\ \mathbf{0}_{K\cdot R} & \mathrm{vec}(\mathbf{X}_1\mathbf{B}_{22})^\top & \cdots & \mathbf{0}_{K\cdot R} \\ \vdots & \cdots & \ddots & \vdots \\ \mathbf{0}_{K\cdot R} & \cdots & \mathbf{0}_{K\cdot R} & \mathrm{vec}(\mathbf{X}_1\mathbf{B}_{K2})^\top \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{vec}(\mathbf{X}_T\mathbf{B}_{12})^\top & \mathbf{0}_{K\cdot R} & \cdots & \mathbf{0}_{K\cdot R} \\ \mathbf{0}_{K\cdot R} & \mathrm{vec}(\mathbf{X}_T\mathbf{B}_{22})^\top & \cdots & \mathbf{0}_{K\cdot R} \\ \vdots & \cdots & \ddots & \vdots \\ \mathbf{0}_{K\cdot R} & \cdots & \mathbf{0}_{K\cdot R} & \mathrm{vec}(\mathbf{X}_T\mathbf{B}_{K2})^\top \end{bmatrix} \tag{C.7}$$

$$\dot{\boldsymbol{\beta}} = \begin{bmatrix} \mathbf{B}_{11[,1]} \\ \vdots \\ \mathbf{B}_{11[,R]} \\ \vdots \\ \mathbf{B}_{K1[,1]} \\ \vdots \\ \mathbf{B}_{K1[,R]} \end{bmatrix} \tag{C.8}$$

Using this augmented design matrix, another logistic regression model can be made, this time estimating the matrix parameters $\mathbf{B}_{k1}$. Using the elements of $\dot{\boldsymbol{\beta}}$, the estimated matrix parameters $\mathbf{B}_{k1}^{(j+1)}$ can be constructed.

Lastly, holding $\mathbf{B}^{(j+1)} = \mathbf{B}_1^{(j+1)}\mathbf{B}_2^{(j+1)\top}$ fixed, we can consider a logistic regression model using components of the augmented design matrix $\mathbf{X}^*$ again, and thus estimate $\boldsymbol{\beta}^{(j+1)}$ accordingly. The offsets in this case are different than before. Here, since we are fixing $\mathbf{B}^{(j)}$ the offset will be defined as follows:

$$\sum_{k'=1}^{K}\left(\sum_{k=1}^{K}\mathbf{B}_{k'[k,1]}\cdot\begin{bmatrix}BD(\mathbf{X_1})\\ \vdots \\ BD(\mathbf{X_T})\end{bmatrix}_{[,k]}+\cdots+\sum_{k=1}^{K}\mathbf{B}_{k'[k,M]}\cdot\begin{bmatrix}BD(\mathbf{X_1})\\ \vdots \\ BD(\mathbf{X_T})\end{bmatrix}_{[,K(M-1)+k]}\right) \tag{C.9}$$

As such, the corresponding design matrix for this step is

$$\mathbf{\Gamma}=\begin{bmatrix}\mathbf{I}_K & \mathbf{y}_{11}^{\top\prime}\\ & \mathbf{y}_{12}^{\top\prime}\\ & \vdots\\ & \mathbf{y}_{1K}^{\top\prime}\\ & \vdots\\ \mathbf{I}_K & \mathbf{y}_{T1}^{\top\prime}\\ & \mathbf{y}_{T2}^{\top\prime}\\ & \vdots\\ & \mathbf{y}_{TK}^{\top\prime}\end{bmatrix} \tag{C.10}$$

The parameter estimates here are

$$\boldsymbol{\beta}^{(j+1)}=\begin{bmatrix}\alpha_1\\ \vdots\\ \alpha_k\\ \psi_{12}\\ \vdots\\ \psi_{(K-1)K}\end{bmatrix} \tag{C.11}$$

C.2.1 Marginal Models

Furthermore, we want to consider two comparison models to which we will compare the usefulness of the model being proposed. First, for each of the comparison models, we consider K marginal models, each consisting of a response vector $\boldsymbol{y}_k$ which denotes T responses for stock k denoting if the price increases or decreases from the previous day. For a given day, the predictors are a stock-specific intercept and the relative stock price changes for stock k each day within a $M-$day look-back period. Because we are consider k different models, we want to split the $K\times M\times T$ data tensor K ways instead of T ways. This response vector and design matrix can be denoted as follows:

$$
\begin{aligned}
\boldsymbol{y}_k &= \begin{bmatrix} y_{1k} \\ \vdots \\ y_{Tk} \end{bmatrix} \\
\boldsymbol{X}_k &= \begin{bmatrix} 1 & X_{k[1,1]} & \cdots & X_{k[1,M]} \\ 1 & \vdots & \vdots & \vdots \\ 1 & X_{k[T,1]} & \cdots & X_{k[T,M]} \end{bmatrix} \\
\boldsymbol{\beta} &= \begin{bmatrix} \alpha_k \\ \boldsymbol{\beta}_k \end{bmatrix}
\end{aligned}
\tag{C.12}
$$

For both the training and test data, k models will be fit. For the purpose of comparing the marginal models with the joint model being proposed, these marginal models are reformed into a $T \times K$ matrix of responses, where each row corresponds to the responses for all K stocks on a given day. Each of these days will correspond to one of 2^K possible outcomes. Using the multi-class ROC method proposed by Hand and Till (2001), these cutoff values can be compared to the truth, and multi-class ROC curves can be constructed, with the area under the curve being computed for both training and test data. A similar process is considered for both comparison models, and this is how the models will be compared.

In a similar manner, we consider a second comparison model of greater complexity. In much the same way as before, we wish to consider K marginal models yet again. Here however, for fixed k, $Z_{ik} = \alpha_k + \langle \mathbf{X}_i, \mathbf{B}_k \rangle$. No association terms are considered. As such, the model can be considered as followed:

$$
\begin{aligned}
\boldsymbol{y}_k &= \begin{bmatrix} y_{1k} \\ \vdots \\ y_{Tk} \end{bmatrix} \\
\boldsymbol{X}_k &= \begin{bmatrix} 1 & \text{vec}(\mathbf{X}_1) \\ \vdots & \vdots \\ 1 & \text{vec}(\mathbf{X}_T) \end{bmatrix} \\
\boldsymbol{\beta} &= \begin{bmatrix} \alpha_k \\ \mathbf{B}_{k[,1]} \\ \vdots \\ \mathbf{B}_{k[,M]} \end{bmatrix}
\end{aligned}
\tag{C.13}
$$

This model, rather than performing a simple logistic regression, will follow the alternating logistic method proposed in the more complex model, albeit with different design matrices.

The differing design matrices will also require certain parameters to be offset depending on the offset. First, the update step for the matrix $\mathbf{B}_{k2}$ has the following design matrix and parameter vector:

$$\begin{aligned} \ddot{\mathbf{X}}_k &= \begin{bmatrix} \text{vec}(\mathbf{X}_1^\top \mathbf{B}_{k1})^\top \\ \vdots \\ \text{vec}(\mathbf{X}_T^\top \mathbf{B}_{k1})^\top \end{bmatrix} \\ \ddot{\boldsymbol{\beta}} &= \begin{bmatrix} B_{k2[,1]} \\ \vdots \\ B_{k2[,R]} \end{bmatrix} \end{aligned} \tag{C.14}$$

Next, the update step for the matrix $\mathbf{B}_{k1}$ has the following design matrix and parameter vector:

$$\begin{aligned} \dddot{\mathbf{X}}_k &= \begin{bmatrix} \text{vec}(\mathbf{X}_1 \mathbf{B}_{k2})^\top \\ \vdots \\ \text{vec}(\mathbf{X}_T \mathbf{B}_{k2})^\top \end{bmatrix} \\ \dddot{\boldsymbol{\beta}} &= \begin{bmatrix} B_{k1[,1]} \\ \vdots \\ B_{k1[,R]} \end{bmatrix} \end{aligned} \tag{C.15}$$

Both of these updates have the same offset:

$$\alpha_k^{(j)} \cdot \mathbf{1}_T \tag{C.16}$$

Once estimates for $\mathbf{B}_{k1}^{(j)}, \mathbf{B}_{k2}^{(j)}$ are obtained, their product $\mathbf{B}_k^{(j)} = \mathbf{B}_{k1}^{(j)} \cdot \mathbf{B}_{k2}^{(j)\top}$ is constructed. The design matrix here for α_k is merely $\left[\mathbf{1}_T\right]$, with the offsets being

$$\left(\sum_{k=1}^{K} \mathbf{B}_{k'[k,1]} \cdot \begin{bmatrix} \text{vec}(\mathbf{X}_1) \\ \vdots \\ \text{vec}(\mathbf{X}_T) \end{bmatrix}_{[,1]} + \cdots + \sum_{k=1}^{K} \mathbf{B}_{k'[k,M]} \cdot \begin{bmatrix} \text{vec}(\mathbf{X}_1) \\ \vdots \\ \text{vec}(\mathbf{X}_T) \end{bmatrix}_{[,KM]} \right) \tag{C.17}$$

Following along with this framework, the ALR process can be performed until a convergence criterion is reached. The resulting model can then be assessed by reconstructing a $T \times K$ matrix of responses, and constructing a $T \times 1$ vector where the K responses in a given day corresponds to one of 2^K potential outcomes. In this way, we can compare both models to the response by performing multi-class ROC analysis yet again. Then the AUC can be produced for both training and test data. These results will then be compared to the proposed model to assess the usefulness of this model at predicting the response.

It is beneficial to provide some details of how the AUC is computed. While the details of the multi-class method are discussed in Hand and Till (2001), it is important to detail how

the function is used. The function used in R to compute this quantity requires two quantities. First, it requires the true response for a given day (transformed from a vector of length K to a value from 1 to 2^K). The second argument is a matrix of estimated probabilities. This matrix of estimated probabilities has the dimensions $T \times 2^K$, where each row corresponds to a day i, with a corresponding probability for each of all 2^K possible outcomes for a given day.

For the model being proposed, this is fairly straightforward, to compute the estimated joint probability of a given outcome for day i, there is a small modification made to computing the joint probability for an outcome on that day.

$$\begin{aligned} P(\hat{Y}_i = y_i | X_i) &= \frac{\exp(\hat{\mu}_{y_i})}{\sum_{s_i \in S} \exp(\hat{\mu}_{s_i})} \\ \hat{\mu}_{y_i} &= \sum_{k=1}^{K} y_{ik}(\hat{\alpha}_k + \langle \mathbf{X}_i, \hat{\mathbf{B}}_k \rangle) + \sum_{l>k} y_{ik} y_{il} \hat{\psi}_{kl} \end{aligned} \tag{C.18}$$

It is fairly straightforward to generate this probability matrix, as it follows the same steps required to generate the true probability matrix that is required to generate the simulated Y_i. This process becomes more difficult for the two comparison models being considered. As both are marginal models, $P(\hat{Y}_i = y_i | X_i, \dots) = P(\hat{Y}_{i1} = y_{i1} | X_i, \dots) \times \cdots \times P(\hat{Y}_{iK} = y_{iK} | X_i, \dots)$.

For model 1,

$$\begin{aligned} P(\hat{Y}_{ik} = 1 | X_i, \dots) &= \frac{\exp(\hat{\alpha}_k + \hat{\boldsymbol{\beta}}_k^{\top} X_{k[i,]})}{1 + \exp(\hat{\alpha}_k + \hat{\boldsymbol{\beta}}_k^{\top} X_{k[i,]})} \\ P(\hat{Y}_{ik} = 0 | X_i, \dots) &= 1 - P(\hat{Y}_{ik} = 1 | X_i, \dots) \end{aligned} \tag{C.19}$$

For model 2,

$$\begin{aligned} P(\hat{Y}_{ik} = 1 | X_i, \dots) &= \frac{\exp(\hat{\alpha}_k + \langle \mathbf{X}_i, \hat{\mathbf{B}}_k \rangle)}{1 + \exp(\hat{\alpha}_k + \langle \mathbf{X}_i, \hat{\mathbf{B}}_k \rangle)} \\ P(\hat{Y}_{ik} = 0 | X_i, \dots) &= 1 - P(\hat{Y}_{ik} = 1 | X_i, \dots) \end{aligned} \tag{C.20}$$

While these terms are not inherently difficult to compute, it can be tedious to compute the products of these marginal probabilities. How these are computed can be seen in the code provided in section E.2.1. One key thing to note is that with the marginal models being considered, the joint probability only has to be computed for the outcomes of the realization space that we are interested in based on the index we use. Regardless of what reasoning is used to construct said index, the joint probability only needs to be computed for the indexed outcomes. The remaining outcomes in the realization space can be reassigned to a baseline category, which will have the probability of 1 minus the sum of the probabilities of indexed outcomes.

GLOSSARY

Batch Size Size of subset from sample data (Chollet 2017, p. 34).

Epoch Full pass over the dataset (Chollet 2017).

Learning rate Scalar term determining the size of the step (Goodfellow et al. 2016, p. 86).

Loss function How network measures its performance on training data (Chollet 2017, p. 29).

Optimizer function How the network updates itself based on data and loss function (Chollet 2017, p. 29).

Step Made in movement of algorithm to update loss function (Chollet 2017, p. 48).