Social Network Analysis: Online Anomaly Detection and Graphical Model Selection

Corinne Horn
Advisor: Rebecca Willett

Senior Thesis for Graduation with Departmental Distinction
Electrical and Computer Engineering, Duke University

April 20, 2011

Abstract

There is a demand for computational methods that can extract meaningful patterns from social networks in real time. However, these networks can be extremely large and volatile, and brute force algorithms for high-dimensional data analysis are intractable as high costs and poor run-time preclude many real-world applications. I present two online (sequentially updating) strategies that from learn from observations of a network in order to accomplish some task. Furthermore, both online methods analyze binary data, making them memory-efficient and accommodating to exceptionally large datasets. One method detects unusual or deviant behavior while utilizing expert feedback to adjust to a continuously evolving definition of alarming behavior. The second method learns relationships among individuals (via graphical model selection) that underpin otherwise indiscernible social patterns. In this paper I primarily focus on my contributions to the two projects. I applied the theoretical framework from the anomaly detection scheme to two real-world datasets. Then, after familiarizing myself with relevant literature, I formulated, analyzed, and tested my graphical model selection algorithm on simulated and real world data.

1 Introduction

The Information Age is characterized by an explosion of large volumes of raw data, particularly data that involves social interactions in large groups. As more information becomes available, one might ask what useful patterns could be extracted from this influx of data, and is there anything useful one could learn about a group by observing the individual behavior of its members? The first examples of large social networks that come to mind are usually websites such as Facebook, Twitter, and LinkedIn, that aim to form a cyber network amongst users. However, these networking websites only comprise a tiny fraction of relevant social groups that could be analyzed: interesting social patterns are likely to appear in the everyday contexts, such as the workplace, politics, traffic

patterns, and other substantial situations such as communication patterns among terrorist cells and rebel groups.

As more of our daily lives are spent on digital devices, it has become increasingly easier to collect individual behavior for a range of features in a variety of contexts. Most companies have their own intranet, which collects email communication information for each employee. The government publishes the Senate's voting records, and Facebook stores the collection of profiles that have been viewed by every user. Our cellphone history reveals who we communicate with. This information, if processed effectively, can offer valuable insight to otherwise indiscernible social patterns that underpin the behavior we exhibit on a daily basis.

There exist a variety of methods for learning useful patterns in datasets. However, there are a few complications in the social network context that can cause traditional machine learning or statistical learning approaches to perform poorly. First, social networks can be extremely large, and brute force algorithms for high-dimensional data analysis are intractable, as high costs and poor run-time preclude many real-world applications. The analysis tools I discuss in this paper exploit low dimensional structure intrinsic to systems governed by social behavior. Furthermore, social relationships can be extremely volatile, so a successful approach needs to accommodate a dynamic network structure (where the underlying patterns change over time). Provided this setting, I developed methods that analyze *a sequence of observations*, and update an appropriate model in real time. This approach does not analyze a collection of data all at once, but as it becomes available to the user. Such algorithms are called *online methods*, and they are useful because they can be implemented in real-world applications.

I worked with Dr. Rebecca Willett and Dr. Maxim Raginsky to design two methods that encode complex information into concise, tractable models in order to facilitate the analysis of real-world social networks. In both projects, the goal was to use high dimensional data to quickly and accurately update a set of model parameters that balance between previous data and the current observation. Provided these model parameters, it is possible to identify useful patterns in the network. The first method detects unusual (henceforth referred to as anomalous) behavior by updating a probability model that represents normal behavior for the entire group, and uses expert feedback to adapt a threshold for flagging alarmingly deviant behavior. The second method aims to learn social relationships among individuals by optimizing the parameters for a graphical model that most likely represents the topology of the social group. I elaborate the mathematical formulation and analysis of these two projects, as well as describe my contributions in detail, in sections 3 and 4, respectively.

From a high level, both algorithms provide three pieces of information about the state of the social network at any point in time:

- parameters characterizing a *probability model* that incorporates all the data up to time t,
- a *score* for how well the current state of the network fits the estimated model,
- and a *worst case guarantee* for how well the estimated model could deviate from the true underlying behavior, provided a reasonable set of assumptions on the structure of the network.

Furthermore, both tools intake data of the same format, and consequently can be applied concurrently without additional pre-processing.

In this paper, I will present these two methods as follows: In section 2, I will formalize the format of input data and justify the assumptions that are imposed on social network characteristics. In section 3, I briefly outline the theoretical approach to the anomaly detection task and discuss

the experimental procedure and results I obtained. In section 4, I will follow an analogous format to present the model selection method for inferring social relationships, but I further elaborate the theoretical framework to reflect my level of contribution. Finally, in section 5, I summarize my findings and suggest future directions for further research. I will formalize mathematical notation as new terms are introduced.

2 Social Networks in the Online Setting

The goal was to develop methods that would be capable of learning social patterns without high-computational analysis, such as contextual parsing of messages or monitoring pairwise interactions among group members. Instead, the following methods interpret behavior from a sequence of binary states of everyone in the social group over time, such as attendance, participation, or votes. Monitoring binary actions allows the information to be encoded into a p-dimensional vector, where p is the number of group members, and accommodates exceptionally large datasets that would otherwise be computationally expensive and/or time consuming to analyze.

Consider a dynamic social network consisting of p individuals where the behavior of every individual at any point in time is categorized as one of two actions, represented by value of -1 or 1 (I use -1 instead of 0 for mathematical convenience). At time t, I call the p-dimensional binary vector a *co-occurrence* x_t, where $x_t \in \{-1, 1\}^p$. As time evolves I observe new instantiations $x_{1,\ldots,t}$ of some unknown, underlying process. By analyzing the sequence of co-occurrences over time, I would like to generate a temporally-evolving model that mostly accurately incorporates all the data.

The observed behavior of each individual is limited to two actions because it allows for efficient representation of the state of the network; the co-occurrence for a network with a large dimension p can be encoded in a p-bit binary vector. Furthermore, this assumption that every individual has one of two actions available to him is valid in many contexts, such as voting, communication behavior, and meeting patterns. For example, suppose I wished to monitor communication behavior via email correspondences. Some computationally intensive approaches might require parsing the text or monitoring pairwise behavior (such as who emailed whom). However, my approaches extract information simply by observing which members of the network send or receive an email message within a particular time period. This example is discussed in further detail in the 3.2 section, specifically as it relates to my analysis of the Enron email database.

I limit the size of the network to p individuals. This means that new members cannot be introduced throughout the duration of the analysis. While p must remain fixed, I place no limitation on the size of the dimension p itself. In fact, these methods aim to exploit the inherent low-dimensional structure of social dynamics as a means for efficiently processing large-dimensional influxes of data.

These methods perform analysis in real time using sequential updates as they become available. In this online setting, each observation is analyzed one at a time, and the updated model parameters are calculated from the single observation at time t. Online methods respond in real-time to sequential inputs, and offer more flexibility (with little to no total performance trade-off) than batch methods, which analyze the data all at once. In addition to analyzing real-time data, online methods are capable of iterating through large, static datasets. In fact, online methods are preferable for exceptionally large amounts of data as they are memory efficient; this means that they do not

require all the information to be stored in memory at once. Online methods are also more readily adaptable for real-world applications such as video analysis, search engine technology, and data collection. Such settings are frequent contexts for social behavioral analysis.

3 Anomaly Detection

I first started collaborating with Dr. Willett and Dr. Raginsky as they finalized an online method for detecting anomalies in social networks by observing a sequence of co-occurrences, which was a specific application of a theoretical framework they called **F**iltering and **H**edging for **T**ime-varying **A**nomaly reco**GN**ition, or FHTAGN. My contributions to this project involved applying the theoretical framework to real-world datasets, interpreting and displaying the results, and writing sections of the journal and conference papers. I studied the Enron email corpus (1999-2002) as well as the comprehensive voting records from the US Senate (2003-2010) as both data sets embodied long-term behavior of complex social networks with supporting text documents. For completeness, I present an overview of framework below, then elaborate on my contributions to the project.

3.1 Outline of Approach

My first task was to familiarize myself with the theory developed so far. Understanding the motivation for the features in the FHTAGN framework ultimately proved invaluable while I was working on my own, second research project. In summary, FHTAGN is comprised of two primary components:

- *Filtering:* the sequential process of updating *beliefs* or *likelihoods* on the next state of the system based on the observed past.
- *Hedging:* the sequential process of flagging potential anomalies by comparing the belief against a time-varying threshold.

Since social networks can be extremely large, the first project aimed to detect anomalies simply from observing the participants of various forms of communication, as opposed to parsing and analyzing the content of these communications. In the FHTAGN framework, detecting anomalies from co-occurrences is a specific case where the model is the product of Bernoulli marginals.

At each time step, FHTAGN observes the state of the system x_t and would like to infer whether x_t is anomalous relative to the sequence $x^{t-1} = \{x_1, ..., x_{t-1}\}$. This inference is represented by a binary decision $\widehat{y}_t$, where $\widehat{y}_t$ can either be -1 (implying x_t is normal) or $+1$ (x_t is anomalous). The 'hat' denotes that the variable is estimated from our framework, as opposed to a true label (which is represented as y_t). In order to predict $\widehat{y}_t$, I assign a 'likelihood' $\widehat{p}(x_t)$ that the observation x_t was derived from the exponential family distribution that was generated by x^{t-1}. This likelihood is then compared to an adaptive threshold τ_t, which decides how to label $\widehat{y}_t$. After predicting the anomaly label, the decision engine may request expert feedback y_t with a probability inversely proportional to its confidence about its decision, and use that information to adjust its decision-making threshold τ_t in the future. Finally, the model parameters are updated using a mirror descent strategy on a convex hypothesis space to balance between the previous data and the current observation x_t. The process is re-iterated as each new observation becomes available. The method also provides a

bound on the worst-case performance compared to any static or time-varying comparator in the hypothesis set.

The method can be concisely expressed in following algorithm, where η is the learning rate of the adaptive threshold τ_t, and C tunes how sensitive the decision-engine is to asking for feedback.

Algorithm 1 Label-efficient anomaly detection [1]

Parameters: real numbers $\eta > 0$, $C > 0$
Initialize: $\tau_1 = 0$
for $t = 1, 2, \ldots$ **do**
  Observe x_t
  Compute the estimated likelihood $\widehat{p}(x_t)$
  if $-\log \widehat{p}(x_t) > \tau_t$ **then** Flag x_t as an anomaly: let $\widehat{y}_t = 1$
  else Let $\widehat{y}_t = -1$
  end if
  Draw a Bernoulli random variable U_t that inversely depends on the confidence that the label is correct: $\Pr[U_t = 1 | U^{t-1}] = 1/(1 + C| - \log \widehat{p}(x_t) - \tau_t|)$
  if $U_t = 1$ **then** Request feedback y_t and let $\tau_{t+1} = \tau_t + \eta y_t$
  else Let $\tau_{t+1} = \tau_t$
  end if
end for

3.2 Experiments

My primary contributions to this project involved applying Algorithm 1 to real-world datasets and interpreting and displaying the results.

3.2.1 Enron E-mail Dataset

First, I conducted experiments using the Enron e-mail database, which is publicly available at `http://www.cs.cmu.edu/~enron`. This dataset should provide insight because Enrons publicized collapse provides justification for anomaly verification. I used SQL to organize the data into relevant tables, then printed the information into text files that I loaded in MATLAB. For each message that was sent within the company's intranet, I retrieved the recipient's email address, sender's email address, message subject and body (for generating expert feedback), and timestamp. The Enron corpus consists of approximately 500,000 e-mails involving 151 known employees and more than 75, 000 distinct addresses, between the years 1999 and 2002.

One problem I encountered was how to partition the data into co-occurrences. One (naive) option was to consider each thread individually: everyone who participated in a message thread was deemed active for that thread, and inactive otherwise. There were two dilemmas with this approach. First, the duration of a thread was never known at any point in time. As a result, the participants of a thread at time t could be analyzed, but if another employee responded to the thread at time $(t + 1)$, the thread was re-analyzed. This was not only redundant, but detrimental to the performance of the algorithm because it inappropriately weighted certain threads. Furthermore, spam messages in this setting confused the algorithm, as it would normally expect relatively sparse observations.

My next approach was to partition the data by day. I use email timestamps in order to record users what were active in each day, either sending or receiving emails. This was done for $1,177$ days, starting from January 1, 1999. This strategy worked relatively well, except that Saturday and Sunday of every week saw relatively low participation. Consequently, these days were frequent false alarms. Finally, I consolidated each weekend's emails into the preceding Fridays observation vector, resulting in a total of 902 days in the dataset. In this setting, the dimensionality was $d = 75,511$.

I used MATLAB to perform Algorithm 1 on this dataset. I had to use trial and error to tune the feedback parameter C, which adjusted how sensitive the decision engine was in regards to requesting feedback, as well as the learning rate η to get the optimal performance. Feedback was requested according to Algorithm 1, and $C = 0.0079$ and $\eta = 14500$ in the results below (Figure 1, and Table 1).

The algorithm relies on infrequent feedback from an oracle so that it can adjust its threshold decision τ if FHTAGN is assigning incorrect labels. However, there are no definitive labels intrinsic to this dataset, so I had to devise a method for generating 'true' labels that would most likely reflect unusual behavior within the company. Since this approach analyzes the participants within the network, I parsed email messages to assign oracle feedback as follows. If feedback is requested at time t, I generate word count vectors h_t using the $12,000$ most frequently appearing words (to avoid memory issues and misspelled words) for days $t-10, ..., t$. Then I average the difference in word counts between day t and each of previous 10 days to generate an error term e_t:

$$e_t = \frac{1}{10} \sum_{i=t-10}^{t-1} \|h_t - h_i\|_1$$

where $\|x\|_1 = \sum_i |x_i|$ denotes the ℓ-1 norm. When the prediction error e_t is sufficiently high, I consider day t to be anomalous according to the expert. Note that this expert only needs to process a limited amount of data to deliver feedback, thus reducing the total amount of computational resources needed to open, decrypt, transcribe, or translate documents.

The prediction error e_t and the threshold τ_t determining y_t is plotted in Figure 1, right bottom, but note that only a fraction of these values need to be computed to run FHTAGN. A variety of other expert systems for determining anomalous emails or documents based on their contents could be used instead of our keyword predictor, and investigating other expert systems could be an avenue for future research. The results are summarized in Table 1 and Figure 1. As predicted, FHTAGN performs very well relative to the best static threshold which could be chosen in hindsight at time T with full knowledge of all filtering outputs and feedback provided to FHTAGN.

As shown in the left plot in Figure 1, the threshold τ_t adapts to feedback regarding false alarms and missed anomalies from the oracle. Moreover, some of the true anomalies in this example are contextual in that they do not always correspond to large likelihood values, but rather to large values relative to neighboring observations. Letting the threshold τ_t change over time allows FHTAGN to adapt to an experts evolving notion of what is anomalous.

The right upper plot of Figure 1 shows the probability of requesting feedback over time and the days on which feedback is requested; as expected, feedback is less likely when the likelihood is very far from the current threshold choice τ_t. There are a total of 91 feedback requests over 902 days, and because of the sliding window used by the oracle to determine the true labels y_t, a total over 523 of the 902 days required text parsing (and, generally speaking, any overhead

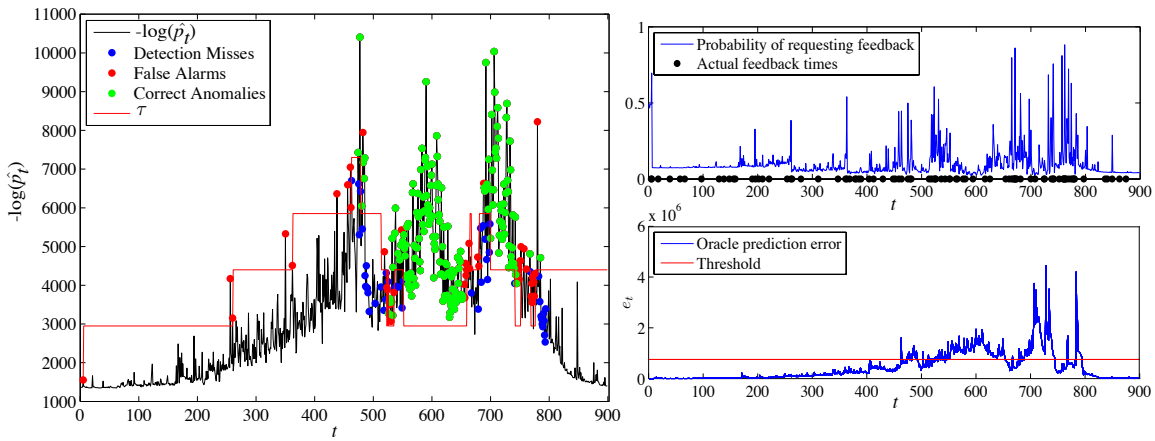

Figure 1: Online anomaly detection results on Enron corpus. Left plot displays filtering output, locations of missed anomalies (as declared by our expert), false positives, and correctly identified anomalies, as well as time-varying threshold τ_t. Upper right plot displays the probability U_t of requesting feedback, where black circles indicate the locations where feedback was provided. Lower right plot displays expert prediction error e_t (from contextual evidence within a 10-day sliding window) compared to a static threshold to assign y_t.

	FHTAGN	Best static threshold
number of errors	73	143
number of false alarms	35	96
number of misses	38	47

Table 1: A summary of results comparing FHTAGN to the best static threshold.

associated with processing the documents). I researched the most news-worthy events in Enron's recent history to see if the method could detect turmoil within the company. In fact, I discovered that some of the most anomalous events detected by FHTAGN correspond to historical events. For instance, consider the following occasions:

- **Dec. 1, 2000:** Days before "California faces unprecedented energy alert" (Dec. 7) and energy commodity trading deregulated in Congress (Dec. 15). See `http://www.pbs.org/wgbh/pages/frontline/shows/blackout/california/timeline.html`.
- **May 9, 2001:** "California Utility Says Prices of Gas Were Inflated" by Enron collaborator in El Paso, blackouts affect upwards of 167,000 Enron customers. See `http://archives.cnn.com/2001/us/05/08/calif.power.crisis.02`.
- **Oct. 18, 2001:** Enron reports $618M third quarter loss, followed later by major correction `http://www.justice.gov/enron/exhibit/04-27/BBC-0001/Images/24379.001.PDF`.

3.2.2 Senate Voting Record

I also ran this algorithm on the US Senate roll call voting records from the 108th-111th Congress (2003-2010), which are available online at `www.senate.gov`. Since the data before 2003 was incomplete, I restricted my analysis to the voting record from 2003-2010. The dimension of the data is 100 Senators voting on a total of 2,429 bills. Each of the bills is put to a vote sequentially, and the votes are recorded as $+1$ for voting with the majority, and -1 for voting against the majority. I only collected data from the Senate because Senators serve 6-year terms (compared to Representatives who only serve 2 year terms) and I wanted to minimize the turnover rate of social players. I used a perl script to collect the bill number, the date, a brief description, the final result, and every Senator's vote. 33% of the seats went up for re-election every 2 years, and there were 141 distinct Senators in the 6 years, 3 month span of the data. For the purposes of this experiment, I considered the vote as being cast by the seat itself. Therefore, when one Senator replaced another I considered them the same social character. The dimension of this data set was 100×2429, and the algorithm only took a few seconds to run. This experiment used the same MATLAB script as the Enron experiment. The results can be seen in Figure 2.

Analysis of the Senate voting records indicated unusual behavior peaking every 2 years in response to elections, particularly coinciding with the commencement of the Obama administration in 2008. There is a relatively high average 'anomalous likelihood' from 2008-2010, and these bills correspond the to controversial health care bills.

Using these datasets, I was able to verify the anomalies detected in social network communications by FHTAGN are indicative of anomalous events of interest to the social network members.

4 Graphical Model Selection

As I grew familiar with techniques from my first research project, I acknowledged a caveat in the original framework; we modeled the behavior of individuals under the assumption that each participated independently with identical distribution from our exponential model. However, this is clearly not the case for social networks, as relationships and internal factors play a large role in the display of social behavior. Consequently, I conducted my own independent research project to model these intra-network dependencies under the guidance of Dr. Willett and Dr. Raginsky.

4.1 Problem Formulation

Within a social network, the behavior of one's acquaintances is likely to influence the behavior of that individual. I model these relationships using a graph, where vertices represent individuals and edges represent a relationship, or mutual influence, shared by the two people. My goal is to learn these relationships from a sequence of co-occurrences, or in other words, to learn the parameters for a graphical model that embodies the network's social structure. As before, I assume the behavior of each individual is restricted to one of two possible actions, so the observation vector at time t will be of the form $x_t \in \{-1, 1\}^p$. This assumption is valid in many contexts, as discussed before. Additionally, remaining contexts may categorize a collection of behaviors into two disjoint sets, thereby restricting one's action to a binary case.

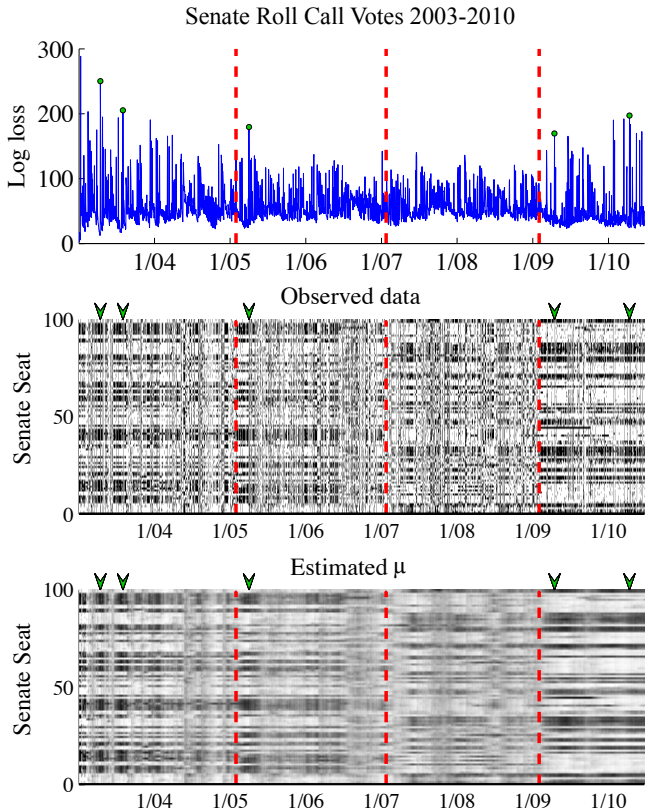

Figure 2: Online anomaly detection results on the Senate roll call record from 2003-2010. The top plot displays the anomalous likelihood $-\log \widehat{p}(x_t)$ of each vote. The bills with the highest likelihood are marked with a green tag. The dashed red line represents a turnover in 33% of the seats. The middle plot displays a graphical representation of the votes. The vertical axis represents the 100 seats. If a seat voted against the majority at time t, its coordinate it marked with a black pixel. The bottom plot is a visual representation of the predicted voting behavior for each seat. The green tags highlight how the actual voting behavior deviated from the expected behavior for those bills.

As in the previous project, the topology of the network can be dynamic, since it is reasonable to assume that the mutual influence between pairs of individuals will evolve over time. For convenience, I call the set of individuals connected to vertex r by an edge on the graph a *neighborhood* (N_r) for person r. Furthermore, I impose a maximum neighborhood size d such that $d \ll p$, and I assume the behavior of an individual is determined solely from its neighbors and its own intrinsic properties; the remaining individuals have negligible influence. This assumption is quite reasonable considering the social context of the problem, and will ensure that the model parameters $\theta \in \mathbb{R}^{p \times p}$ will be sparse. Sparse models are memory-efficient and reduce computational complexity in the parameter update step, which is critical for improving an online method's performance.

In summary, the structure of the network will be encoded in a binary, time-varying, sparse, online pairwise Markov random field, which in statistical physics is known as an Ising model. The pairwise Markov random field property is common in statistical analysis of social networks [2], [3], and it allows the algorithm to predict the behavior of an individual solely from its sparse neighborhood. This approach exploits the low dimensional structure of Ising models to simply the computational analysis, even for large datasets. Then, using convex optimization techniques, we specify an update equation that incorporates the current observation x_t into the model parameters θ with each iteration. The offline (batch) version of this problem has been investigated by Banerjee et al. [4], Ravikumar et al. [5], and Hofling and Tibshirani [6]. Finally, the ultimate goal is to obtain performance guarantees for how well this method will perform against any time-varying strategy.

4.1.1 Ising Models

An Ising model consists of a set of vertices $\mathcal{V} = \{1, ..., p\}$ and edges $\mathcal{E} = \Theta_{\alpha\beta}$ for $\alpha, \beta = 1...p$. In the social network setting, the vertices represent the set of individuals and the behavior of person i at time t is represented by the binary status of the vertex $(x_t)_i \in \{-1, 1\}$. For notational convenience, we assume that the time dependence on x will be defined by context, so I will use the notation $x_i = (x_t)_i$. I represent the mutual influence of one person α with another person β by a signed, undirected edge $\theta_{\alpha,\beta}$. The structure of this graph exhibits conditional independence assumptions among the nonzero subsets of the sequential p-dimensional discrete random variables, and ensures sparsity on the edge set. For an Ising model with a known set of parameters $\theta_{\alpha,\beta}$, the state of the network $x = (x_\alpha : \alpha \in \mathcal{V})$ is given by the probability distribution function:

$$\mathbb{P}_\theta(x) = \frac{1}{Z(\theta)} \exp\left[\sum_{\alpha,\beta\in\mathcal{V}} \theta_{\alpha\beta} x_\alpha x_\beta\right]. \tag{1}$$

In our formulation, the $p \times p$ matrix $\theta = (\theta_{\alpha\beta})_{\alpha,\beta\in\mathcal{V}}$ is symmetric ($\theta_{\alpha\beta} = \theta_{\beta\alpha}$) and if $\theta_{\alpha\beta} = 0$ then person α possesses no influence over the behavior of person β. Otherwise, the correlation of behavior between a pair of individuals can either be positive or negative, and vary in magnitude as indicated by the edge between the pair of vertices. The function $Z(\theta)$ is the partition function and ensures the distribution sums to 1.

My goal is to update θ for every time step; therefore I denote the edge set for my model at time t as $\widehat{\theta}_t$. Ideally, I would update $\widehat{\theta}_t$ by optimizing (1) over all parameters $\theta \in \Theta$ in some hypothesis class Θ. The problem of undirected graphical model selection for discrete variables has attracted considerable attention because of the complication arising from typical evaluation metrics. Optimal models are selected on the basis of having the highest probability given the data, therefore all possible graphical structures must be considered. However, the number of graphical structures grows exponentially. Furthermore, computing the log partition function for any structure within an optimization routine would require summing over 2^p possible configurations of x for each possible θ, which is intractable. To sidestep this problem, I proposed extending an idea presented by Wainwright [5] to the online domain, where the simultaneous consideration of individual nodes conditioned on their nonzero neighbors constitutes a pseudo-likelihood metric for any graphical configuration. For any vertex $r \in \mathcal{V}$ and any edge set $\theta_r \in \mathbb{R}^p$ connected to r, the probability that

$x_r = +1$ is approximated (using the Markov property) by the behavior of the remaining nodes $x_{\setminus r}$:

$$\mathbb{P}_{\theta_r}(x_r|x_{\setminus r}) = \mathbb{P}_{\theta_r}(x_r|x_{N(r)}) = \frac{\exp\left(2x_r\theta_{rr} + 2\sum_{s\in N(r)}\theta_{rs}x_rx_s\right)}{\exp\left(2x_r\theta_{rr} + 2\sum_{s\in N(r)}\theta_{rs}x_rx_s\right) + 1} \tag{2}$$

Finding θ_r that maximizes (2) is a logistic regression problem. Furthermore, I assume that the neighborhood $N(r)$ is small with a maximum of d elements, so the calculation complexity is further reduced.

4.1.2 Minimizing the Loss

Now I need a method for quantifying how well a graphical model fits the previous data. Define the *data fit term* $\tilde{f}_{t,r}(x;\theta_r)$ of vertex r as the negative log of the likelihood function described in (2):

$$\begin{aligned}\tilde{f}_{t,r}(x;\theta_r) &= -\log\left(\mathbb{P}_{\theta_r}(x_r|x_{\setminus r})\right)\\ &= -2\theta_{rr}x_r - 2\sum_{s\neq r}\theta_{rs}x_rx_s + \varphi_{t,r}(x;\theta_r)\end{aligned}$$

where

$$\varphi_{t,r}(x;\theta_r) = \log\left[\exp\left(2\theta_{rr}x_r + 2\sum_{s\neq r}\theta_{rs}x_rx_s\right) + 1\right]. \tag{3}$$

Since $-\log(x)$ is a convex function, $\tilde{f}_{t,r}(\theta_r)$ is a convex function with respect to θ. To evaluate the likelihood of the entire graphical model for a parameter set $\theta \in \mathbb{R}^{p\times p}$, I sum the pseudolikelihoods over all p nodes to get the loss function associated with our parameter set θ at time t:

$$f_t(x;\theta) = \sum_{r=1}^{p}\tilde{f}_{t,r}(x);\theta_r) = \sum_{r=1}^{p}\left[-2\theta_{rr}x_r - 2\sum_{s\neq r}\theta_{rs}x_rx_s + \varphi_{t,r}(\theta_r)\right]. \tag{4}$$

From this point forward, I will express $f_t(x;\theta)$ as $f_t(\theta)$ for notational convenience. If I did not assume θ was sparse, it would suffice to find the next parameter $\widehat{\theta}_{t+1}$ by applying online optimization algorithms to the function $f_t(\theta)$. However, I would like to impose sparsity on the parameter set θ. A common approach is to add a penalty regularization term to the minimization problem, as it usually causes small elements in $\widehat{\theta}_t$ to drop to 0. The problem now takes the composite function $f_t(\theta) + r(\theta)$ as input to the optimization.

Since the pseudo-likelihood is convex, I could use one of a variety of online convex programming techniques available in the literature. My first attempt was to use a method developed by Langford et al. called the Truncated Gradient [7]. This optimization approach uses a combination of shrinking and thresholding every K rounds after performing gradient descent, as shown in Algorithm 2.

Algorithm 2 Regularized Online Learning via Truncated Gradient

Initialize $\widehat{\theta}^1 \in \mathbb{R}^{p\times p} = \mathbf{0}$
for $i = 1, 2, ...$ **do**
  Acquire new observation x_i and incur the loss $\ell_i(\widehat{\theta}_i) = f_i(\widehat{\theta}_i) + r(\widehat{\theta}_i)$
  Compute $g_i(\widehat{\theta}_i, x_i) = \nabla_1 \ell_i(\widehat{\theta}_i)$, a subgradient of $\ell(\widehat{\theta}_i)$ w.r.t. the first variable.
  Update via gradient descent: $\widehat{\theta}_{i+1} = \widehat{\theta}_i - \eta g_i(\widehat{\theta}_i, x_i)$
  Perform Truncation:
  if $\mod (i, K) = 0$ **then**

$$\widehat{\theta}_{i+1} = T(\widehat{\theta}_i, \eta g_i, \omega) = \begin{cases} \widehat{\theta}_{i+1} & \text{if } |\widehat{\theta}_{i+1}| \geq \omega \\ \widehat{\theta}_{i+1} - \eta g_i(\widehat{\theta}_i, x_i) & \text{if } \eta g_i(\widehat{\theta}_i, x_i) < \widehat{\theta}_{i+1} < \omega \\ 0 & \text{if } |\widehat{\theta}_{i+1}| \leq \omega \\ \widehat{\theta}_{i+1} + \eta g_i(\widehat{\theta}_i, x_i) & \text{if } -\omega < \widehat{\theta}_{i+1} < -\eta g_i(\widehat{\theta}_i, x_i) \end{cases}$$

  end if
end for

I saw good results in the experiments, but their technique for bounding the performance of this method was not easily adaptable to the time-varying case. Since I was primarily interested in tracking the performance for a dynamic system, I next considered a regularized dual averaging (RDA) method presented by Xiao [8]. The idea behind RDA is that the update step depends on an averaged arbitrary subgradient of the likelihood function, which lives in a dual space. This method performs well in a variety of settings, but it also was hard to prove any performance bounds in the time-varying case. A slight variation of RDA is called Composite Objective Mirror Descent (COMID), and was developed by Duchi [9]. This was the optimization strategy I used in my time-varying performance analysis and experiments. Since this optimization method is critical to my analysis, I describe Mirror Descent and Composite Objective Mirror Descent (Algorithm 3) briefly in the following section.

4.1.3 Composite Objective Mirror Descent

The Mirror Descent algorithm (MD) [10], [11] is an iterative method for optimizing a convex function $\ell : \Theta \to \mathbb{R}$. Using Mirror Descent, it is possible to tightly bound a term called the regret, defined as

$$R_T(\widehat{\theta}_t; \ell) = \sum_{t=1}^{T} \ell_t(\widehat{\theta}_t) - \inf_{\theta\in\Theta} \sum_{t=1}^{T} \ell_t(\theta) \tag{5}$$

where $\{\widehat{\theta}_t\}$ is the sequence generated by mirror descent, ℓ_t are convex functions, and Θ is the convex set of possible parameters. The concept of regret is elaborated in section 4.1.4.

I would like to generalize the mirror descent algorithm to the case where ℓ_t is composed of loss function f_t and a sparsity-inducing penalty term r, so that $\ell_t = f_t + r$. f_t is allowed to change over time but the regularizer r remains constant. Performing MD on the composite function ℓ_t can lead to poor performance; in the case when $r = \|\cdot\|_1$, MD does not directly lead to sparse updates [9]. Instead I follow the Composite Objective Mirror Descent strategy, where the name comes from the fact that I am attempting to minimize a composite objective function. In short, I perform MD on f_t

with the regularizer term only added to the update equation. This can also be viewed as performing MD on ℓ_t but without linearizing the regularizer r, and leads to the following update equation:

$$\widehat{\theta}_{t+1} = \arg\min_{\theta \in \Theta} \eta \langle f_t'(\widehat{\theta}_t, \theta) \rangle + B_\psi(\theta, \widehat{\theta}_t) + \eta r(\theta). \tag{6}$$

where $f_t'(\theta)$ denotes an arbitrary subgradient of f_t at θ, B_ψ is the Bregman divergence, and the step size η controls the trade-off between the minimization of the first order approximation of ℓ_t at $\widehat{\theta}_t$, and forcing the next update $\widehat{\theta}_{t+1}$ to lie close to $\widehat{\theta}_t$. Throughout, ψ designates a continuously differentiable function that is α-strongly convex w.r.t. a norm $\|\cdot\|$ on the set Θ. The Bregman divergence associated with ψ is

$$B_\psi(\theta_1, \theta_2) = \psi(\theta_1) - \psi(\theta_2) - \langle \nabla\psi(\theta_2), \theta_1 - \theta_2 \rangle,$$

and satisfies $B_\psi(\theta_1, \theta_2) \geq \frac{\alpha}{2}\|\theta_1 - \theta_2\|^2$ for some $\alpha > 0$. A vector $f_t'(\theta) \in \mathbb{R}^p$ is a subgradient of f_t at θ if for all $\bar{\theta} \in \text{dom} f_t$, $f_t(\bar{\theta}) \geq f_t(\theta) + \langle f_t'(\theta), \bar{\theta} - \theta \rangle$. If f_t is differentiable and convex, then its gradient is its subgradient. Convex functions that are nondifferentiable have a set of subgradients that lie tangent to the curve, but do not intersect the curve at any other point. Algorithm 3 outlines the general COMID approach.

Algorithm 3 Regularized Online Learning via Composite Objective Mirror Descent

Parameters: Choose a non-increasing $\eta_t > 0$, a continuously differentiable function $\psi(\cdot)$ that is α-strongly convex w.r.t. a norm $\|\cdot\|$, and a regularizer $r(\cdot)$.

for $i = 1, 2, ...$ **do**

Acquire new observation x_i and incur the loss $\ell_i(\widehat{\theta}_i) = f_i(\widehat{\theta}_i) + r(\widehat{\theta}_i)$

Compute $g_i(\widehat{\theta}_i) = \nabla_1 \ell_i(\widehat{\theta}_i)$, a subgradient of $\ell_i(\widehat{\theta}_i)$.

Update $\widehat{\theta}_i$ by solving

$$\widehat{\theta}_{i+1} = \arg\min_{\theta \in \Theta} \eta \langle g_i(\widehat{\theta}_i), \theta \rangle + B_\psi(\theta, \widehat{\theta}_i) + \eta r(\theta).$$

end for

4.1.4 Regret Bounds

Now, I focus on bounding the performance of COMID when the true, underlying model is allowed to vary with time. The goal is achieve low regret w.r.t. a time varying comparator $\boldsymbol{\theta}_t = \{\theta_i\}_{i=1...t}$ where $\theta_i \in \mathbb{R}^{p \times p}$. With every round of the online optimization I compute the regularized regret against $\boldsymbol{\theta}_t$, defined as

$$R_T(\widehat{\theta}_t, \theta_t) = \sum_{t=1}^{T} \left(f_t(\widehat{\theta}_t) + r(\widehat{\theta}_t) - f_t(\theta_t) - r(\theta_t) \right). \tag{7}$$

First, I bound the the behavior of $\ell_t = f_t + r$ for each time step in the algorithm. I then use the results to bound the regret R_T for any time-varying comparator.

Lemma 1 (Progress bounds for each step of algorithm, from Duchi's Lemma 1 [9]) *Let $\{\widehat{\theta}_t\}$ be defined using the update equation (6). Assume that $B_\psi(\cdot,\cdot)$ is α strongly convex w.r.t. a norm $\|\cdot\|$. Then for any $\theta \in \Theta$ and any regularizer term $r(\theta)$ we have*

$$f_t(\widehat{\theta}_t) - f_t(\theta) + r(\widehat{\theta}_{t+1}) - r(\theta) \leq \frac{1}{\eta}\left[B_\psi(\theta, \widehat{\theta}_t) - B_\psi(\theta, \widehat{\theta}_{t+1})\right] + \frac{\eta}{2\alpha}\|f_t'(\widehat{\theta}_t)\|_*^2 \tag{8}$$

where $\|\cdot\|_*$ is the dual norm. For any norm $\|\cdot\|$, the associated dual norm $\|\cdot\|_*$ is defined as

$$\|z\|_* = \sup\{z^T x \max \|x\| < 1\}.$$

Lemma 1 gives a worst case difference between the loss of an estimated model, $\ell_t(\widehat{\theta})$ and the loss of any other model θ is our hypothesis set Θ. This bound only holds for one point in time, but I am interested in the difference between the sequence of estimated models, and any time-varying comparator over all time.

Define the cumulative loss

$$L(\theta, T) \triangleq \sum_{t=1}^{T} f_t(\theta_t) + r(\theta_t)$$

and let

$$V_T(\boldsymbol{\theta}) \triangleq \sum_{t=1}^{T} \|\theta_t - \theta_{t+1}\| \tag{9}$$

be the variation of the time-varying comparator $\boldsymbol{\theta}$.

Theorem 1 (Regret against time varying strategies) *Let $\{\Theta_i\}_{i=1,\ldots,t}$ be a sequence of closed convex sets, and let $\widehat{\boldsymbol{\theta}}$ be the sequence of parameters computed using the update in (6) with $\eta_t = 1/\sqrt{t}$. Then, for any sequence $\boldsymbol{\theta} = \{\theta_i\}_{i=1,..,t}$ in $\{\Theta_i\}_{i=1,\ldots,t}$*

$$L_{\widehat{\theta},T}(x^T) \leq L_{\theta,T}(x^T) + D\left(1 + \sqrt{T+1}\right) + 4M\sqrt{T}V_T(\boldsymbol{\theta}) + \frac{G^2}{2\alpha}\left(2\sqrt{T} - 1\right). \tag{10}$$

where $G \triangleq \max_{\theta\in\Theta, f\in\mathcal{F}} \|f_t(\theta)\|_*$, $M \triangleq \frac{1}{2}\max_{\theta\in\Theta}\|\nabla\psi(\theta)\|$, $D \triangleq \max_{\theta,\theta'\in\Theta} B_\psi(\theta', \theta)$.

Proof: This proof follows the proof of Theorem 2 in FHTAGN [1]. Applying Lemma 1 to $\ell_t(\cdot)$, write

$$f_t(\widehat{\theta}_t) + r(\widehat{\theta}_{t+1}) \leq f_t(\theta_t) + r(\theta_t) + \frac{1}{\eta_t}\left(B_\psi(\theta_t, \widehat{\theta}_t) - B_\psi(\theta_t, \widehat{\theta}_{t+1}) + \Gamma_t\right) + \frac{\eta_t}{2\alpha}\|f_t'(\widehat{\theta}_t\|_*^2$$

Adding and subtracting $B_\psi(\theta_{t+1}, \widehat{\theta}_{t+1})$ inside the parentheses and rearranging gives us

$$\begin{aligned}
&f_t(\widehat{\theta}_t) + r(\widehat{\theta}_{t+1}) - f_t(\theta_t) - r(\theta_t) \\
&\leq \frac{1}{\eta_t}\left(B_\psi(\theta_t, \widehat{\theta}_t) - B_\psi(\theta_{t+1}, \widehat{\theta}_{t+1}) + \Gamma_t\right) + \frac{\eta_t}{2\alpha}\|f_t'(\widehat{\theta}_t\|_*^2 \\
&\leq \Delta_t - \Delta_{t+1} + \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t}\right) B_\psi(\theta_{t+1}, \widehat{\theta}_{t+1}) + \frac{1}{\eta_t}\Gamma_t + \frac{\eta_t}{2\alpha}\|f_t'(\widehat{\theta}_t\|_*^2 \\
&\leq \Delta_t - \Delta_{t+1} + \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t}\right) D + \frac{1}{\eta_t}\Gamma_t + \frac{\eta_t}{2\alpha}\|f_t'(\widehat{\theta}_t\|_*^2
\end{aligned}$$

where I have defined $\Delta_t = (1/\eta_t)B_\psi(\theta_t, \widehat{\theta}_t)$ and $\Gamma_t = B_\psi(\theta_{t+1}, \widehat{\theta}_{t+1}) - B_\psi(\theta_t, \widehat{\theta}_{t+1})$. Next, I take a look at Γ_t:

$$\begin{aligned}\Gamma_t &= \psi(\theta_{t+1}) - \psi(\widehat{\theta}_{t+1}) - \langle \nabla\psi(\widehat{\theta}_{t+1}), \theta_{t+1} - \widehat{\theta}_{t+1}\rangle - \Big[\psi(\theta_t) - \psi(\widehat{\theta}_{t+1}) - \langle \nabla\psi(\widehat{\theta}_{t+1}), \theta_t - \widehat{\theta}_{t+1}\rangle\Big] \\ &= \psi(\theta_{t+1}) - \psi(\theta_t) + \langle \nabla\psi(\widehat{\theta}_{t+1}), \theta_t - \theta_{t+1}\rangle \\ &\leq 4M\|\theta_t - \theta_{t+1}\|.\end{aligned}$$

Combining everything and summing from $t = 1$ to $t = T$, I get

$$\begin{aligned}&\sum_{t=1}^{T}[f_t(\widehat{\theta}_t) + r(\widehat{\theta}_{t+1})] - \sum_{t=1}^{T}[f_t(\theta_t) + r(\theta_t)] \\ &\leq \sum_{t=1}^{T}(\Delta_t - \Delta_{t+1}) + \sum_{t=1}^{T}\left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t}\right)D + 4M\sum_{t=1}^{T}\frac{1}{\eta_t}\|\theta_t - \theta_{t+1}\| + \sum_{t=1}^{T}\frac{\eta_t}{2\alpha}\|f_t'(\widehat{\theta}_t)\|_*^2 \\ &\leq \Delta_1 - \Delta_{T+1} + \left(\frac{1}{\eta_{T+1}} - \frac{1}{\eta_1}\right)D + \frac{4M}{\eta_T}V_T(\boldsymbol{\theta}) + \frac{G^2}{2\alpha}\sum_{t=1}^{T}\eta_t \\ &\leq D + D\sqrt{T+1} + \frac{4M}{\eta_T}V_T(\boldsymbol{\theta}) + \frac{G^2}{2\alpha}(2\sqrt{T} - 1).\end{aligned}$$

where in the last time, I used the estimate $\sum_{t=1}^{T} t^{-1/2} \leq 1 + \int_1^T t^{-1/2}dt = 2\sqrt{T} - 1$.

Theorem 1 provides a worst case bound for the regret against any strategy, which is a powerful result. The tightness of this bound depends primarily on the choice of Bregman divergence. However, the question of how to choose the best Bregman divergence remains an open question. Surprisingly, this bound for the regret does not depend on the regularizer term $r(\theta)$ and it is sublinear with T if the variation $V_T(\boldsymbol{\theta})$ of the comparator is held constant. These results are comparable to the regret against static comparators of other online prediction strategies in the literature. However, our method has the added benefit of being able to track the dynamic comparator, and it is possible to show that this regret $R_T(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta}) \sim O(\sqrt{T}V_T(\boldsymbol{\theta}))$.

■

4.2 Derived Algorithm

In this section, I show how a specific instantiation of the framework is used to generate an algorithm for online updates of the Ising model. The squared Euclidean norm is an α-strongly convex Bregman divergence when $\alpha = 2$, so the dual norm $\|\cdot\|_* = \|\cdot\|_2^2$ is also the squared Euclidean norm. If I let $r(\theta) = \|\theta\|_1$ and $\eta_t = \frac{1}{\sqrt{t}}$, then I obtain the following optimization problem:

$$\widehat{\theta}_{t+1} = \arg\min_{\theta\in\Theta} \eta_t\langle f_t'(\widehat{\theta}_t), \theta\rangle + \|\theta\|_2^2 - \|\widehat{\theta}_t\|_2^2 - \langle 2\widehat{\theta}_t, \theta - \widehat{\theta}_t\rangle + \eta_t\|\theta\|_1.$$

This problem reduces to n independent scalar minimizations, and the jth minimization can be expressed in closed form as

$$\widehat{\theta}_{t+1}^j = \begin{cases} 0 & \text{if} \quad \|\eta_t g_t^j - 2\widehat{\theta}_t^j\| \leq \eta_t \\ -\frac{1}{2}(\eta_t g_t^j - 2\widehat{\theta}_t^j) - \eta_t \text{sgn}(\eta_t g_t^j - 2\widehat{\theta}_t^j) & \text{else} \end{cases} \tag{11}$$

This update method for this derived case is shown in Algorithm 4, and this is the algorithm I used in the experiments in section 4.3.

Algorithm 4 COMID Using the Squared Euclidean Norm and L1 Regularization

Parameters: Choose a non-increasing $\eta_t > 0$.
for $i = 1, 2, ...$ **do**
  Acquire new observation x_i and incur the loss $\ell_i(\widehat{\theta}_i) = f_i(\widehat{\theta}_i) + r(\widehat{\theta}_i)$
  Compute $g_i(\widehat{\theta}_i) = \nabla_1 \ell_i(\widehat{\theta}_i)$, a subgradient of $\ell(\widehat{\theta}_i, x_i)$.
  for j = 1,2, ..., p **do**

$$\widehat{\theta}^j_{i+1} = \begin{cases} 0 & \text{if} \quad \|\eta_i g^j_i - 2\widehat{\theta}^j_i\| \leq \eta_i \\ -\frac{1}{2}(\eta_i g^j_i - 2\widehat{\theta}^j_i) - \eta_i \text{sgn}(\eta_i g^j_i - 2\widehat{\theta}^j_i) & \text{else} \end{cases}$$

  end for
end for

For this derivation of COMID, $D = 4pd, \alpha = 2, G = 2p, M = pd$. Therefore

$$\begin{aligned} f_t(\theta) &= \sum_{r=1}^{p} \left[-2 \left(\theta_{rr} x_r + \sum_{s \neq r} \theta_{rs} x_r x_s \right) \right] + \sum_{r=1}^{p} \varphi_{t,r}(\theta_r) \\ &\leq 2dp + p \log\left(e^{2d} + 1\right), \end{aligned}$$

so $G = p(2dp + p\log(e^{2d} + 1))^2$. Therefore, according to Theorem 1, the regret for a fixed horizon T for $\widehat{\boldsymbol{\theta}} = \{\widehat{\theta}_i\}_{i=1,...T}$ computed by Algorithm 4 against any time-varying comparator $\boldsymbol{\theta} = \{\theta_i\}_{i=1,..,t}$ in $\{\Theta_i\}_{i=1,...,T}$ is bounded by

$$R_T(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \sum_{t=1}^{T} \left[L_{\widehat{\boldsymbol{\theta}},T} - L_{\boldsymbol{\theta},T} \right] \leq 4pd \left(1 + \sqrt{T+1}\right) + 4pd\sqrt{T} V_T(\boldsymbol{\theta}) + \frac{2p^2}{\alpha} \left(2\sqrt{T} - 1\right).$$

The updates in Algorithm 4 are special cases of other methods as well. In the case of this derivation, COMID reduces to a method developed by Duchi and Singer called FOBOS [12], as well as a special case of the Truncated Gradient (Algorithm 2) with $K = 1$ and $\omega = \infty$. Despite this particular algorithm's appearance in the literature, to the extent of my knowledge it has only been derived and bounded, but not tested on simulated or real world data. For this reason, I used it in my experiments. However, I did encounter a problem implementing this method, which I elaborate below.

4.3 Experiments

In this section, I provide both simulations on synthetic data, as well as computational experiments on the Senate voting record. The purpose of these experiments is to illustrate how well this method estimates relationships between members of a dynamic social network.

4.3.1 Simulations

To generate data, I created what would henceforth be considered the true, time-varying underlying edge set θ_t^*, where at any point in time, $\theta_t^* \in [-1, 1]^{p \times p}$ is sparse and symmetric with at most d nonzero elements in each row or column. I limited this simulation to the case where every entry lies on the interval $[-1, 1]$, but this is done without loss of generality since a finitely sized matrix must have bounded elements, and it is possible to scale θ_t^* to accommodate any range.

I performed experiments on two slightly different simulated datasets. The first had the following dimensions: $p = 100$, $d = 5$, and $n = 1500$. The true model θ_t^* was randomly assigned nonzero entries while symmetry and sparsity were imposed. θ_t^* was static except for three jumps at $t = 400, 700,$ and 1200. At these 3 jumps, the old parameters were thrown out and a completely random (under appropriate conditions) model was generated. The second model exhibited a systematic lattice structure that could be easily compared to the estimated model by the naked eye. It had dimension $p = 100$, $d = 5$, and $n = 500$, with one jump at $t = 250$. At this jump, the model deviated from a lattice with predictable, alternating values to a lattice with random values.

I then used Gibbs sampling to generate the data from the Markov random field until there was less than 1% deviation. I repeated the Gibbs sampling method for each observation vector $x_1, ..., x_n$ for the two models, and repeated the whole process 10 times so I could average my results. Generating the data took approximately 150 hours with a 2.4 GHz Intel Core 2 Duo processor.

Using MATLAB, I applied the COMID Algorithm 4 to both datasets, and averaged the results over 10 runs. However, I noticed a problem in the simulations; if one vertex in the graph became completely isolated, the thresholding rule would prevent that vertex from reconnecting with any other node for the duration of the experiment. As a result, most runs returned $\widehat{\theta}_t$ empty. In the case where I chose $r(\theta) = \| \cdot \|_1$ as the ℓ-1 norm, and the Bregman divergence to be the squared Euclidean norm $\| \cdot \|_2^2$, it just so happened that the argument minimization problem *could be* optimal when $\widehat{\theta}_t$ is 0, unless $f_t'(\widehat{\theta}_t)$ and η_t are carefully chosen to follow certain constraints.

However, getting a solution of $\widehat{\theta}_t$ = 0 for all t is uninteresting, even if it satisfies the regret bounds in Theorem 1. Since I applied this method several datasets, and I did not want to waste the computational power to find the perfect values for $f_t'(\widehat{\theta}_t)$ and η_t. Furthermore, if a vertex did come completely isolated, I did not necessarily want it to remain as such. As a result, I only performed truncation/shrinkage according to (11) every $K = 20$ rounds so that elements of $\widehat{\theta}_t$ could grow above the threshold and remain nonzero. This insight comes from Langford's truncated gradient framework [7]. Figure 3 are results from this improved algorithm on the large, random dataset.

I used $\eta_t = 1/\sqrt{t}$. This sequence of values for η_t was useful for proving the theoretical bounds, but in the simulations I noticed that the updated estimate $\widehat{\theta}_t$ responded increasingly slowly to underlying changes as time progressed. I suspect using a constant value for η would be appropriate for models that are expected to vary as t increases. Otherwise, observations for large t would have relatively minimal effect. I applied the same algorithm to the lattice-structured model, as shown in Figure 4. The comparison between θ_t^* and $\widehat{\theta}_t$ is much easier to verify as a recognizable shape.

To quantitatively compare the results for both datasets, I computed a difference matrix $\widehat{\theta}_t - \theta_t^*$ (where $t = 1190$ and 240 for the large and lattice datasets respectively), and counted the number of entries whose absolute value was less than a certain error threshold σ. This was the metric I used for evaluating how well Algorithm 4 recovered θ_t^* for the two datasets, and Table 2 summarized the percent correct versus error threshold σ for each of the two datasets at times $t = 1190$ and 240,

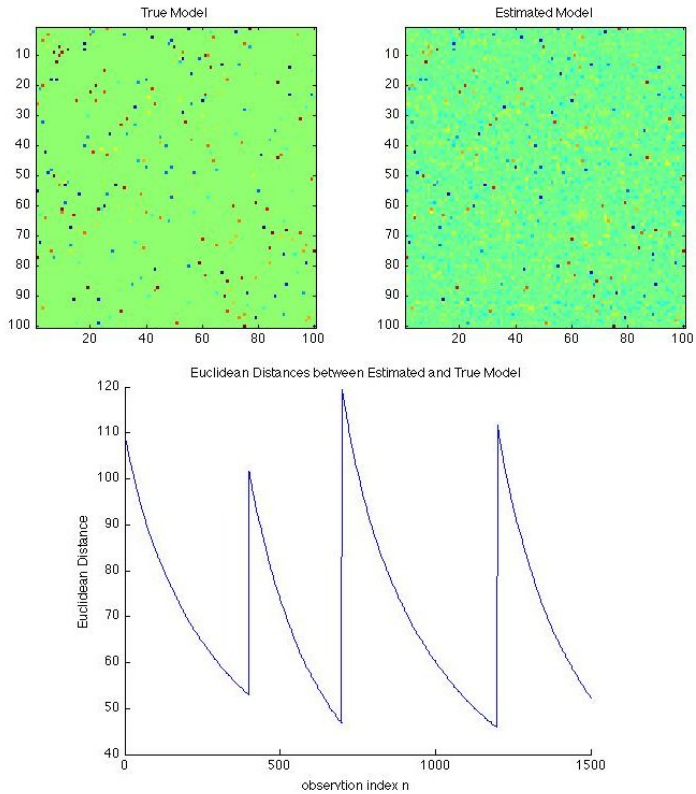

Figure 3: The bottom plot tracks the Euclidean distance between the true model (of edges characterizing the social structure) and the estimated model, also shown in the upper left plot. The large jumps at $t = 400, 700,$ and 1200 reflect the sharp variation in θ_t^* at those locations. The top two plots display a visual representation of the two $p \times p$ dimensional models at $t = 1190$. $\widehat{\theta}_{1190}$ (top right) has been iteratively updated with each new $\{x_t\}_{t=1,\ldots,1189}$ to estimate θ_{1190}^* (top left). The method was successful at recovering a majority of the nonzero features.

respectively.

	Error Threshold σ					
	0.5	**0.4**	**0.3**	**0.2**	**0.1**	**0.05**
Large (n = 1500)	99.95%	99.81%	99.43%	96.77%	80.16%	53.43%
Lattice (n = 500)	96.92%	96.18%	95.74%	95.10%	87.05%	66.56%

Table 2: This table provides the percent of θ^* recovered by $\widehat{\theta}$ for a given error threshold σ for the simulated experiments.

In the simulations, I did not compare the regret to the upper bound I derived in section 4.1.4. This is because the dimensionality was relatively small, with an easy-to-learn underlying model so the upperbound of the regret was grossly overestimated. Furthermore, the true model θ^* does not, in reality, optimize the regularized loss. The penalty term acts to induce sparsity, but some estimates that are more sparse than θ^* will have a lower loss than the model that generated the data,

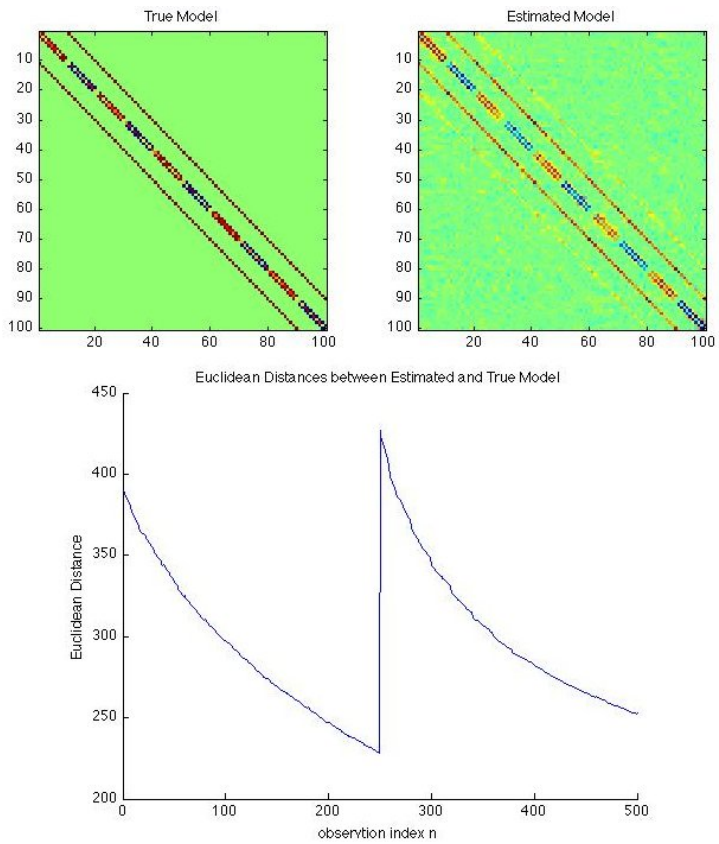

Figure 4: This bottom figure tracks the Euclidean distance between the true model (of edges characterizing the social structure) and the estimated model, also shown in the upper left figure. The jump at $t = 250$ reflects the sharp variation in θ_t^* at that location. The two upper figures are a visual representation of the two models at $t = 240$. $\widehat{\theta}_{240}$ (top right) has been iteratively updated to estimate θ_{240}^* (top left). It is quite easy to see that the method was successful at recovering a majority of the nonzero features.

providing a negative 'regret'. This was often the case in these simulations.

4.3.2 Senate Voting Record

I conclude my experiments by testing my derivation of COMID on the US Senate roll call voting records from the 108th-111th Congress (2003-2010). The details of this dataset were discussed in section 3.2.2.

First, I applied Algorithm 4 with $\eta = 1/\sqrt{t}$ to the sequence of binary data, and observed the evolution of social dynamics as revealed by the system in Figure 5. The loss function for the estimated model is shown in the top plot. As expected, the loss decays as more bills are analyzed, but there are small increases every 2 years when 33% of the seats are replaced, as indicated by the red dashed lines. First, I ran the algorithm and could make little sense of the results, which looked like an erratic checkerboard (bottom left plot in the figure).

Next, I reorganized these results by political party affiliation. I moved all of the Independents

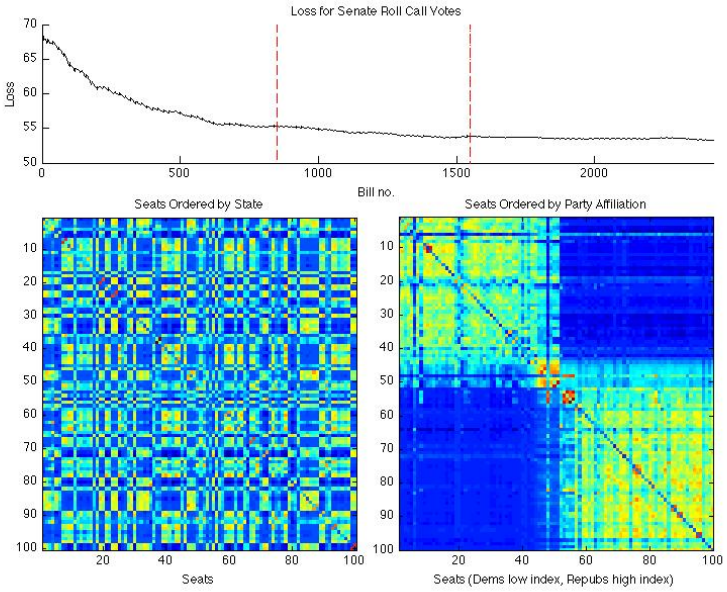

Figure 5: The top plot shows the loss for the estimated model $\widehat{\theta}_t$. The middle plot ordered the axes by state, with the two senators from each state numbered consecutively. The right most plot separates senators by party affiliation: Democrats are assigned low index seat numbers, and Republicans are assign high index seat numbers. Members of the same party tend to share positive mutual influences, while Senators from different parties tend to exhibit a negative influence.

to the lowest seat indexes, then Democrats to the lower-middle seat indexes, and Republicans to the higher seat indexes. It is clear that members of the same political party are more likely to share a positive mutual influence, while Senators with associated with different parties are more likely to exhibit a negative influence. This re-ordering seems to make order out of chaos, and is shown in the lower right plot.

There are two Senators for which the relationships seem unexpected in the lower right plot. The first is Joe Lieberman from Connecticut at index 6, and the second is Dianne Feinstein from California at index 48. Based on this model, we can infer that Lieberman's voting habits are minimally influenced by the voting behavior of the remaining Senators, while Feinstein's habits appear to be more congruent with Republicans. This analysis offers insight to how an individual actually votes, regardless of their political affiliation.

5 Conclusions and Future Work

In the paper, I presented two different online methods for finding patterns in social networks from high-dimensional, binary co-occurrence data. The first framework, called FHTAGN, detects anomalous or unusual behavior relative to previous observations by (1) *filtering* the current value, or computing its likelihood given the previous data, (2) *hedging*, or comparing this likelihood against a feedback-adaptive threshold, and (3) updating the exponential family probability distri-

bution via primal-dual updates to balance between the old and new data. While I was minimally involved in the framework design and analysis, I applied the algorithm to Enron e-mail database and Senate voting record, interpreted the results, and wrote sections of the journal and conference papers. To collect and pre-process the data, I utilized languages such as Perl and SQL. Once I formatted the datasets, I applied FHTAGN using MATLAB. Working on this project introduced me to the process of problem formulation, analysis, finding performance guarantees, and experimentation that characterizes this field of research.

Once I became slightly more familiar with the research process, I started my own research project under the guidance of Drs. Willett and Raginsky. My goal was to learn the structure of social relationships underpinning observable behavior in a social group by iteratively updating an Ising model using online convex programming techniques. Having reviewed the literature, I formalized and justified my assumptions, attempted to extend various existing methodologies to the online Ising model problem, proved how well my framework would perform against any time-varying comparator, and performed simulations and experiments to investigate its implementation in simulation and real datasets.

Despite the work I have accomplished, there still remain issues I would like to address in the future. First and foremost, I would like to address my problem of selecting a poor combination of Bregman divergence and a regularization term. Provided the time, I would research how others approached this problem (preliminary investigation revealed that most scientists did not perform experiments for this derivation, and simply included the update steps). I would also choose difference Bregman divergences and/or regularization penalties to derive alternative algorithms, and compare their performance.

I would also generate more simulated data with higher dimensionality p and a longer observation period n, so that regret analysis would be nontrivial (at least the simulated regret was well within the limit). Real world datasets provided limited insight to the behavior of regret because there is no true, underlying model with which to compare. I would also hope to find a way to interpret COMID's results on an extremely large dataset like the Enron e-mail database, where the social relationships are hard not already known and would need to be justified.

Finally, a more general question would be how to pick a regularizer term for a specific problem, and how it would affect any method's performance. There is currently a lot of trial and error to the science of selecting a regularizer that performs well in experiments. This question is large enough to be a completely new project.

5.1 Acknowledgements

I would like to thank Drs. Rebecca Willett and Maxim Raginsky for their support, guidance, and mentorship through these projects, and for their patience and invaluable feedback as I worked to become a effective researcher. I would also like to thank the members of the Networking and Imaging Sciences Lab for accepting me into their research group and engaging me in helpful discussions. Finally, I would like to thank Dean Absher and the Pratt Undergraduate Research Fellowship Program for introducing me to these exciting projects, and for providing me the opportunity and funding to conduct rigorous research as an undergraduate student.

References

[1] M. Raginsky, R. Willett, C. Horn, and R. Marcia, "Sequential anomaly detection in the presence of noise and limited feedback," *Submitted*, 2010.

[2] O. Frank and D. Strauss, "Markov graphs," *Journal of American Statistics Society*, vol. 81, no. 395, pp. 832–842, 1986.

[3] A. Goldenberg, A. X. Zheng, S. E. Fienberg, and E. M. Airoldi, "A survey of statistical network models," *Foundations and Trends in Machine Learning*, vol. 2, no. 2, pp. 1–117, 2009.

[4] O. Banerjee, L. El Ghaoui, and A. d'Aspremont, "Model selection through sparse maximum likelihood estimation for multivariate guassian or binary data," *Journal of Machine Learning Research*, vol. 9, pp. 485–516, 2008.

[5] Pradeep Ravikumar and Martin Wainwright, "High dimensional ising model selection using l1 regularized logistic regression," *Annals of Statistics*, 2000.

[6] H. Hofling and R. Tibshirani, "Estimation of sparse binary pairwaise markov random fields," *Annals of Statistics*, 2009.

[7] John Langford, Lihong Li, and Tong Zhang, "Sparse online learning via gradient descent," *Journal of Machine Learning Research*, vol. 10, pp. 777–801, March 2009.

[8] Lin Xiao, "Dual averaging methods for regularized stochastic learning and online optimization," *Journal of Machine Learning Research*, vol. 11, pp. 2543–2596, March 2010.

[9] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari, "Composite objective mirror descent," in *Conference on Learning Theory*, 2010.

[10] A. Nemirovsky and D. Yudin, *Problem complexity and method efficiency in optimiztion*, Wiley, 1983.

[11] Amir Beck and Marc Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex programming," *Operations Research Letters*, vol. 31, pp. 167–175, 2003.

[12] John Duchi and Yoram Singer, "Efficient online and batch learning using forward backward splitting," *Journal of Machine Learning Research*, vol. 10, pp. 2899–2934, 2009.