Research in Industrial Projects for Students

Sponsor

IBM

Final Report

Social Network Analytics

Student Members

Edward Chang (Project Manager)

Edward Dewey

Maksim Tsikhanovich

Seçkin Şahin

Academic Mentor

Blake Hunter

Sponsoring Mentors

Vikas Sindhwani

Aurélie C Lozano

Prem Melville

Dmitriy Katz-Rogozhnikov

Date: August 20, 2010

This project was jointly supported by IBM and NSF Grant DMS-0439872.

Abstract

Twitter has become increasingly important to consumers and businesses as people turn to social media to read and form opinions about products and companies. How can a business track what people are talking about and who is most influential in these discussions? We investigated topic detection and causal influence modeling in microblogs, producing some novel algorithms and applying them to Twitter. We present a new Non-negative Matrix Factorization algorithm which uses 1-norm regularization and simplex projection for sparse encoding of tweets over sparse topic vectors. We show a way of automatically choosing regularization and constraint parameters simultaneously, using a two-dimensional generalization of the L-curve method. Our algorithm compares favorably with Hoyer's (JMLR 2002) sparse NMF algorithm. These models are utilized to convert Twitter content from a word-based representation to a topic-based representation, which requires less memory to represent and makes our further analysis computationally feasible. We then apply Multivariate Group Orthogonal Matching Pursuit (MGOMP) to detect influential twitter accounts based on notions of Granger Causality. We investigate how different choices for the number of topics, discretization of time, and time lag affect the algorithm's performance on Twitter data, benchmarked against retweet rates.

Acknowledgments

This report would not have been possible without the extensive support of the IBM mentor team: Vikas Sindhwani, Aurelie Lozano, Prem Melville, and Dmitriy Katz-Roghozhnikov. They have provided invaluable guidance at every step of the research process, from helping us first understand the problem formulations to enabling us overcome the obstacles that we faced to giving us advice about how best to present our findings. Blake Hunter, our academic mentor, has also been crucial to the success of this project, particularly during the initial and final stages and for providing inspiration and motivation throughout. We would also like to thank everyone at IPAM who has made RIPS possible for building the infrastructure and support for all of our projects.

Contents

REFERENCES

List of Figures

List of Tables

Chapter 1

Introduction

Social media have become very important as a source of information and opinion, and IBM has been developing and implementing mathematical tools to understand it. Two major directions in the study of social media are *Topic Modeling*, the effort to understand online content as arising from a mixture of topics, and *Influence Modeling*, the effort to understand how different online authors influence each other. Our work is at the intersection of these two efforts.

Topic Modeling

Our approach to topic modeling uses non-negative matrix factorization (NMF). NMF is used to represent content (such as a picture of a face, or in our case, a blog post) as a mixture of its parts (such as facial organs or, in our case, topics).

Given a database with some sort of content, one begins by representing everything in that database as a vector. For example a blog post using words from a set $L = \{w_1, w_2, ..., w_K\}$ may be represented as a vector $(a_1, a_2, , a_K)$ where a_i is the (weighted) frequency with which word w_i occurs in the blog post. This gives a set S of vectors corresponding to the database of content that one is trying to model. One then searches for a set of vectors $B = \{b_1, , b_p\}$ such that every element of S may be approximately realized as a sum of vectors in B with positive coefficients; this positivity condition ensures that, if the process is successful, each element of the database may be viewed as a mixture of parts corresponding to the elements of B. Mathematically, this amounts to approximately factoring a large N x k matrix X with non-negative entries as $X \approx UV$, where U is an N x p matrix, V is a p x k matrix, and both have non-negative entries. For ease of later computations, and in order to more easily give an intuitive meaning to the solution, one may require U to be as sparse as is possible without sacrificing too much accuracy. In the blog example this would mean that each blog post be modeled as a mixture of as few topics as possible. One refers to this as "sparse NMF". A short review of NMF techniques is available in [13]; for a more detailed treatment see [10].

Topic Modeling is very useful in grouping content by topic. Suppose that one wanted to group all twitter content by topic. merely looking for the presence of certain keywords would not work. For example, one would not want to lump content about cold war Honduras with content about clothing retail just because both include the string "banana republic". Topic modeling can avoid such pratfalls. As we will show, it can also be used to inform influence modeling.

Influence Modeling

Our approach to determining who is influential in online communities uses Multivariate Group Granger Causal Modeling (MGGCM), an algorithm that quantifies influence using the notion of Granger Causality. Granger Causality is based on the idea that a cause must precede its effect; thus, given two time series of data $A_1, A_2, A_3, ...$ and $B_1, B_2, B_3, ...$, if the time series variable A is said to causally affect B, then the past values of A should help predict the future values of B better than using the past values of B alone [5].

We can use this idea of causality to quantify the influence of one blogger upon another. Bloggers do not always explicitly cite each other or link to the blogs that influence them, so instead we look at the content of a blog itself and see how best we can predict it from the past content of all bloggers. Using multivariate regression, we can solve this problem to quantify this influence and construct a weighted directed graph of influence that shows how one blogger influences another. The Granger rank of a blogger is that blogger's weighted out-degree in the causal directed graph, and provides a measure of how influential that blogger is in the community. This approach was first presented in [14], and the authors have successfully applied Granger causal modeling to high-energy physics papers and blogs about the IBM Lotus software brand.

Data

In order to test our algorithm, we use 4 annotated data sets. Annotations was done by humans going through the datasets and classifying each document into one of k topic categories. The datasets are named and described in the following table:

20 ng Consists of about 20000 newsgroup documents, taken from 20 newsgroups. Each newsgroup is on a different topic, so the origin of each document in one of the 20 newsgroups provides a sorting of the documents into sets each consisting of one of the 20 topics [15].

bbc Consists of documents from the BBC news website corresponding to stories in five topical areas from 2004-2005: business, entertainment, politics, sport, and tech [6, 1].

reuters A subset of Reuters Corpus Volume I, consisting of newswires stories classified by hand into topic classes. Our subset consists of 10 of the optic classes, for a total of 7285 documents [16].

tdt2top30 Consists of news wire documents. Documents were annotated by humans into 100 topics. We use only the topics from the 30 largest topics [4].

Twitter Tweets collected over two month time period, from the 1000 twitter accounts which used the string "IBM" most often over that period. Unlike the other four datasets this dataset lacks any built-in document-topic associations. This dataset also has lower bounds on how often the content of a blogger in the dataset was retweeted.

The number of topics to be used is a parameter in NMF, and we did not make progress on how to choose this parameter. For ease of comparison in all experiments run on the four annotated datasets the number of topics used was equal to the number of topics in the human annotations: that is, 20 for ng20, 5 for bbc, 10 for reuters, and 30 for tdt2top30.

Innovations

We present a modification of Ngoc-Diep Ho's RRI algorithm for NMF which encourages a sparse solution. To our knowledge this is the first use of RRI with sparsity in mind. Our algorithm produces solutions with lower reconstruction error than the older sparse NMF algorithm of Patrick Hoyer [11]. We present an effective algorithm for model selection using a 2-dimensional generalization of the L-curve method.

We use our NMF algorithm to pre-process data for the MGGCM algorithm - instead of MGGCM running on vectors of words, we run it on vectors of topics. To our knowledge this is the first time this has been done, and it makes MGGCM feasible for very large datasets on a desktop computer.

Summary of this Report

In Chapter 2 we give a more detailed presentation of sparse NMF and present our implementations of it, and our results from applying it to Twitter datasets.

In Chapter 3 we discuss model selection problems for sparse NMF.

In Chapter 4 we benchmark our NMF algorithm by comparing it with the sparse NMF algorithm of Patrik Hoyer [11] and running a cross-validation test.

In Chapter 5 we discuss MGGCP in detail.

In Chapter 6 we discuss stopping criteria for MGGCP, motivated by results from simulated data sets.

In Chapter 7 use NMF to pre-process Twitter data and present the results of applying MGGCP to the pre-processed data.

Chapter 2

Non-Negative Matrix Factorization

Let's begin by setting some notation. $\mathbb{R}^d_+$ denotes the set of vectors in $\mathbb{R}^d$ with non-negative components: $\{(v_1, v_2, ..., v_d) \in \mathbb{R}^d : v_j \geq 0\}$. Similarly, $\{\mathbb{R}^{a\times b}_+$ denotes the set of $a \times b$ matrices with real, non-negative entries. For any matrix X, the *Frobenius Norm* of X, $||X||_F$, is defined as $||X||_F = \sqrt{\sum_i \sum_j X^2_{i,j}}$. The ℓ_1-*norm* of X, $||X||_1$, is defined as $||X||_1 = \sum_i \sum_j |X_{i,j}|$. The ℓ_0-*norm* of X, $||X||_0$, is the number of non-zero entries in X. Despite it's name it is not a norm. A matrix whose entries are mostly equal to zero is said to be *sparse*.

Non-Negative Matrix Factorization (NMF) is a very general technique for parts-based learning. It can be applied whenever one has a set of n datapoints in $\mathbb{R}^d_+$. Let $X \in \mathbb{R}^{n\times d}_+$ be a matrix whose rows are the n datapoints. Let k be some integer (typically much smaller than d and n). In NMF, one looks for $W \in \mathbb{R}^{n\times k}_+$, $H \in \mathbb{R}^{k\times d}_+$ such that $||X - WH||^2_F$ is small (one could use other norms than the Frobenius norm here, but we shall always use the Frobenius norm to quantify the difference between X and WH). Thus, WH is an approximate factorization of X into two much smaller matrices. Let h_i, $i = 1...k$ be the columns of H and w_p, $p = 1...n$ be the rows of W.

We may interpret this result as saying that each row of X is approximately representable as a *mixture* of the vectors $h_1, ..., h_k$. So that when the rows of X are vectorized blog postings, all the rows of X are represented as mixtures of vectors h_t, which it is reasonable to identify with the topics discussed in the original blog postings.

2.1 Rank-one Residue Iterations as a Solution to NMF

In order to factor X we use a method called *Rank-one Residue Iterations* (RRI) due to Ngoc-Diep Ho[10], which is relatively new, and has not yet been applied to textual data. The key idea behind RRI is that when the reconstruction error $||X - WH||^2_F$ is minimized the reconstruction error contributed by each of W's columns is also minimized. The reverse is not necessarily true, nevertheless the assumption behind RRI is that minimizing the error contributed by each column of W will likely minimize the full reconstruction error.

2.1.1 RRI for Sparse NMF

The RRI algorithm we implement is the one proposed by Ho for minimizing the function

$$||X - WH||_F^2 + \lambda||W||_1 \tag{2.1}$$

. At each iteration the column of W being optimized, w_t is updated

$$w_t \leftarrow \frac{[R_t h_t^T - \lambda 1_{n\times 1}]_+}{||h_t||_2^2} \tag{2.2}$$

where $R_t = X - \sum_{i\neq t} w_i h_i^T$ is the residual that w_t ought to explain, h_t is the corresponding tth column of H, and $[v]_+$ is the projection of v onto the positive orthant. As a result in Equations 2.1 and 2.2 λ serves a parameter which zeros out increasingly larger entries of w_t as its value increases. Note that setting $\lambda = max_i \sum_j X_{ij}$ will force all entries of W to be set to 0. After optimizing each column of W, each column of H is optimized by a update rule similar to Equation 2.2 and then the process is repeated again until some convergence criterion is met.

2.1.2 Enforcing Constraints upon RRI

The base algorithm supplied by Ho is somewhat incomplete for our needs because the objective function we are interested in minimizing is

$$||X - WH||_F^2 + \lambda||W||_0 \tag{2.3}$$

while Ho's algorithm attempts to minimize the function in Equation 2.1. The main problem we run into is scaling, namely $WH = \alpha W \frac{1}{\alpha} H$. Due to this problem RRI may decrease $||W||_1$ without actually decreasing $||W||_0$. To eliminate this possibility we constrain the rows of H to have $||h||_1 = z$, where z is a parameter we choose. This problem reduces to projecting a vector onto the simplex $\{h : ||h||_1 = h\}$. Decreasing z tends to lead to a sparser solution but sometimes leads to higher reconstruction error (see Figure 3.2).

RRI is guaranteed to converge, but it is not guaranteed to converge to a global minimum since the problem is nnon-convex. In practice it may fail to get close even to a local minimum, for example, if H and W are initialized in a region where the objective function is relatively flat, so that the algorithm reaches its stopping condition too soon.

2.1.3 Overview of Our NMF Setup

Thus, factorizing X requires the following steps:

1. Model selection.
    - Choose k.
    - Choose λ.
    - Choose z.
2. Initialize W and H.
3. Perform RRI on X, W and H.

Our work focuses on the first 2 of these steps.

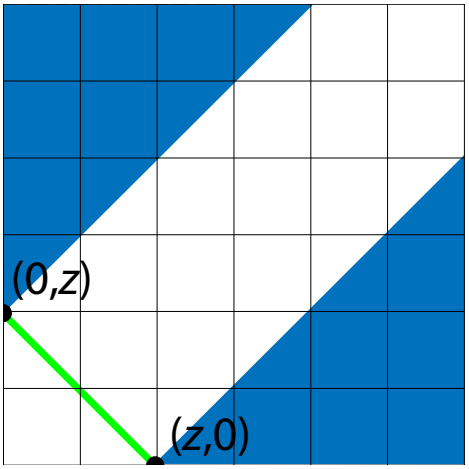


Figure 2.1: In 2 dimensions we are projecting points from the xy plane onto the 2-d simplex. A point in the blue region will be projected onto $(0, z)$ or $(0, z)$ and thus be sparsified. The blue region grows ass z shrinks, thus the tendency for this sparsification to occur is increased if we choose z smaller.

2.2 Benchmarking Against Human Annotations

The human annotations assign a single topic to each document, whereas our algorithm assigns many topics, with different weightings, to each document. Recall that the vector of weighted topics associated with each document is given by the row of W corresponding to that topic. In order to compare our algorithm's output to the human annotations we first go through each row of W, find the maximum entry in that row, and record which column the maximum entry was in. The number of this column is then the index of the topic most associated with that document by the algorithm. Let $t(m)$ be the number of the column containing the greatest entry of row m. Then the function $t(m)$ provides an assignment of exactly one topic to each document, which can be compared directly with the human annotations.

Let $p(m)$ be the topic associated with document m by the human annotators. Note that we do not know how (if at all) the topics chosen by our algorithm correspond to the topics used for the human annotations. Thus we cannot use a simple accuracy measure in order to compare t to p. Instead, we compute the *normalized mutual information (NMI)* of t and p.

For any two random variables T and P, the *mutual information (MI)* of T and P is defined

$$\mathrm{MI}(T, P) = H(T, P) - H(T|P) - H(P|T)$$

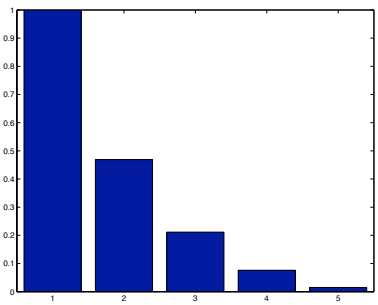
where $H(T, P)$ is the joint entropy of T and P and $H(T|P)$ is the conditional entropy of T upon P. In order to view t and p as random variables, we imagine choosing a document at random, with an equal probability of choosing any document, and taking the index topic associated with that document by t or p as the random variable. NMI is defined as

$$\mathrm{NMI}(T, P) = \frac{\mathrm{MI}(T, P}{\max(H(T), H(P))}$$

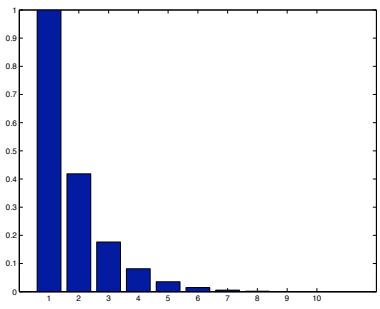
Intuitively, NMI measures the extent to which two random variables convey the same information [9].

One drawback of this method is that we throw away a lot of the information in W before comparing with the human annotations. In particular, if many rows of W has no clear maximum then the NMI might be misleadingly low. On the other hand, if most rows of W do have a clear maximum then it is natural to see the column containing that maximum as the unique topic associated with the document corresponding to that row. In order to see which is the case, we factored $X \approx WH$ with the optimal parameters chosen by our model-choosing algorithm (see below). We then scaled each row w_t of W so that $||w_t||_1 = 1$. We re-ordered the entries of each row from highest to lowest, and then averaged each column and plotted it (see figure 2.2. The resulting graph therefore shows the relative sizes of the highest, second highest, etc. entries of a typical row of W. The graph shows a sharp decrease from the highest entry to the next highest, suggesting that our method of comparison is valid.

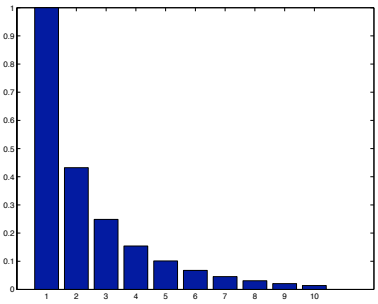
In the following, the "NMI" of a factorization of an annotated data set X as $X \approx WH$ shall always refer to $\mathrm{NMI}(t, p)$ as detailed above.



(a) bbc



(b) reuters



(c) tdt2top30

Figure 2.2: Relative sizes of the entries of an average row of W.

Chapter 3

Initialization and Model Selection

3.1 Initialization

The first problem we must address is how to initialize the matrices W and H. Our algorithm is very sensitive to the initialization of W and H. This is for three main reasons. First, if we initialize in a region of $\mathbb{R}^{n\times k} \times \mathbb{R}^{k\times d}$ where the objective function is very flat without actually being at a minimum, the algorithm may terminate too soon. Second, the NMF problem is non-convex, as can be seen by the fact that if (W, H) is a global minimum of the objective function, then so is $(QW, Q^{-1}H)$, where Q is any invertible matrix that does not change the 1-norms of the rows of H (e.g. a permutation matrix). Therefore, we expect the solution found to be sensitive to the starting conditions. Third, there is no reason to think that all local minima are global minima, and even if the first problem does not occur, the algorithm will find only local minima.

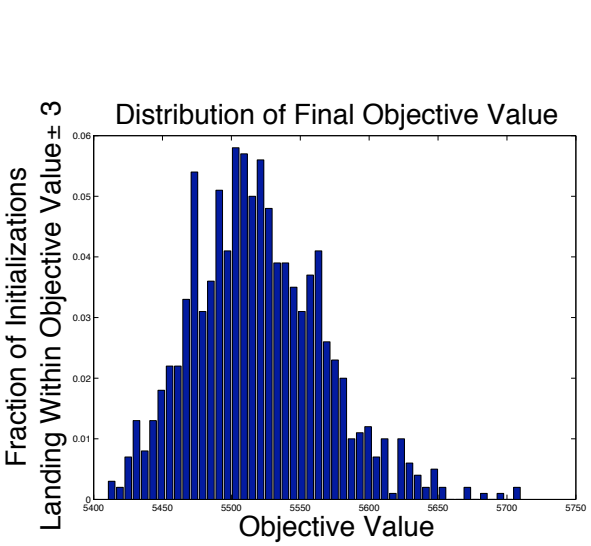
We resolve this problem by randomly selecting 4 initializations, running RRI on each, and choosing the run with the lowest objective value. We have no good reason for doing 4 initializations rather than 5, and one possible avenue of further research might be to see how many initializations one must do to have a given probability that the best run is within some distance of the actual minimum (or of the minimum over all possible runs).

Random initialization of the matrices is performed in 5 steps:

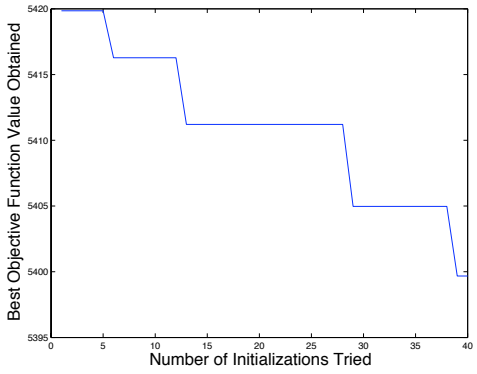
1. Assign each entry of H to be zero (with probability $1-p$) or non-zero (with probability p)
2. Choose the value of each non-zero entry of H from the uniform distribution on $[0, 1]$
3. Choose the value of each entry of W from the uniform distribution $[0, 1]$
4. Find the real number α which minimizes $||X - \alpha WH||_F$, and scale W and H by $W = \sqrt{\alpha}W$, $H = \sqrt{\alpha}H$
5. Perform simplex projection on the rows of H

We were unable to find a better initialization procedure but did not attack this problem systematically. For more information on initialization see [10].

In order to see how sensitive RRI is to initialization, we performed RRI 1000 times with 1000 different initializations on the datasets, with optimal λ and z chosen as described below. We then plotted the distribution of the final objective function attained. The results for the Twitter dataset with 30 topics are shown in figure 3.1(a).



(a) Distribution of the final objective function over 1000 random initializations



(b) Objective function attained versus number of initializations

Figure 3.1: Impact of initialization on final objective value obtained

As stated above, the algorithm performs l initializations and chooses the best one. In order to see how this affects the objective function eventually reached, we plotted objective function versus l, using the same dataset and same choices of k, z and λ. Such a plot depends on the random initializations chosen at each step, but some fairly typical results are shown in Figure 3.1(b)

3.2 Model Selection

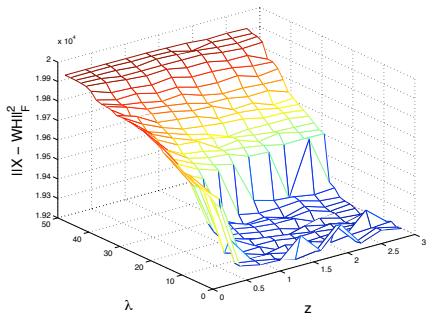
3.2.1 Effects of Model Selection on Accuracy and Sparsity

Choice of λ and z greatly impacts the results of the algorithm. To see the impact of model selection on the results of our algorithm, we plotted reconstruction error versus zero-norms of H and W for various values of λ and z in Figure 3.2.

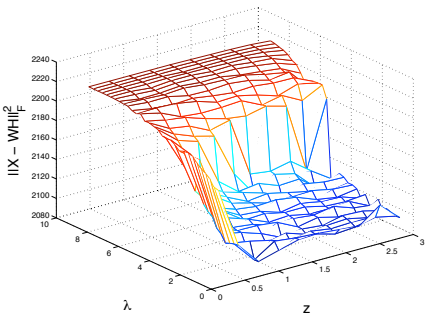
For all four data sets, λ has little effect on error while it is below a certain threshold. Once it reaches that threshold, we see a rapid increase of error with λ until the error reaches a plateau. This plateau occurs when the algorithm has begun to set $W = 0$, at which point $||X - WH||_F^2 = ||X||_F^2$. z has far less effect until it gets very low, at which point decreasing z is also associated with a rapid decrease in reconstruction accuracy. Though it is not clear in the figures, the lowest value of z checked is actually 0.1, not 0 (at $z = 0$, H is simply set equal to zero, and the reconstruction error is equal to $||X||_F^2$).

Perhaps more relevant than the reconstruction error is the NMI: how well our factorization agrees with human annotations. The effect of λ and z on NMI is shown in Figure 3.3

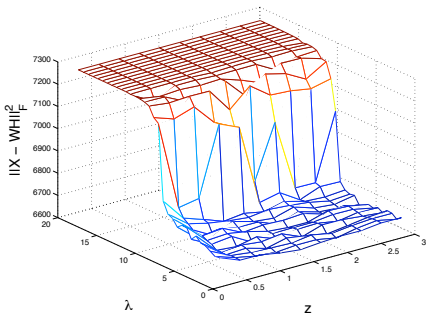
Comparing this to Figure 3.2, we see that the NMI shows behavior opposite the error,



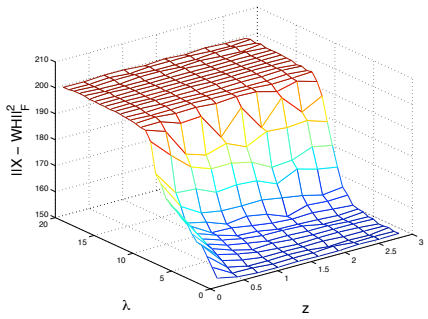
(a) 20ng



(b) bbc

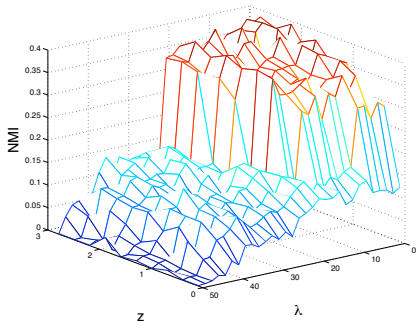


(c) reuters

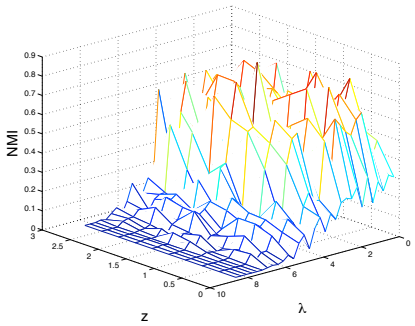


(d) tdt2top30

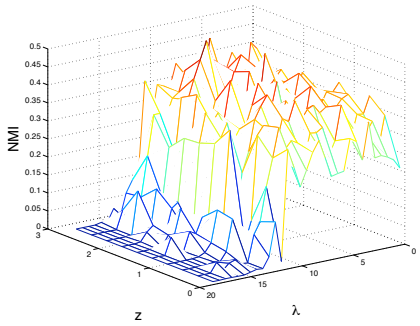
Figure 3.2: $||X - WH||_F^2$ vs λ and z



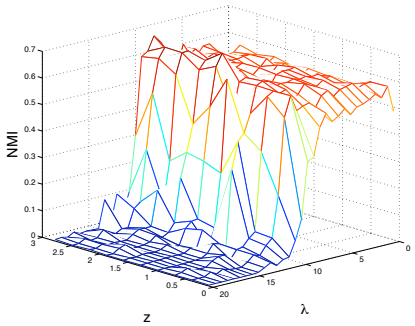
(a) 20ng



(b) bbc



(c) reuters



(d) tdt2top30

Figure 3.3: NMI vs λ and z

as might be expected. For low λ the NMI is relatively high. As we increase λ we eventually reach a threshold, after which the NMI falls off to zero, reaching zero once W starts being zeroed-out entirely. Very low z is also associated with a drop-off in NMI for precisely the same data sets in which a sudden increase in error is associated with very low z.

What these test have shown is that we have a fairly wide window of λ and z to choose from. Within this range the choice of λ and z has little effect on the error and the NMI. Outside this range there is a great deterioration in performance.

Next we will see how model selection affects $||W||_0$ (Figure 3.4). This test shows rather more variation between data-sets than the previous two. The 20ng dataset shows a very late drop-off in $||W||_0$: we see little change until λ is quite large. In particular, the drop off in $||W||_0$ occurs around $\lambda = 40$, whereas the drop-off in NMI occurs around $\lambda = 20$. Thus, there is no way to significantly decrease the zero norm of W without greatly diminishing NMI. On the other hand, tdt2top30 shows a decrease in $||W||_0$ for small values of λ. The other two datasets lie in between.

Figure 3.5 shows the same test for $||H||_0$, and here we start to see some surprises. As expected, z has a bigger effect on $||H||_0$ than on $||W||_0$, with $||H||_0$ decreasing for smaller z. As λ gets very large and W becomes the zero matrix, H becomes extremely dense. What is surprising is that $||H||_0$ is quite sensitive to λ as well, showing decreases for even small values of λ in all 4 tests. This is encouraging. The data on $||W||_0$ suggested that some datasets, such as 20ng, might not be suitable for sparse factorization: deterioration of NMI sets in long before any appreciable decrease in $||W||_0$. But in fact we see a decrease in $||H||_0$ before deterioration in NMI. So if we care about $||H||_0$ as well as $||W||_0$, sparse NMI can have appreciable benefits for all the datasets tested.

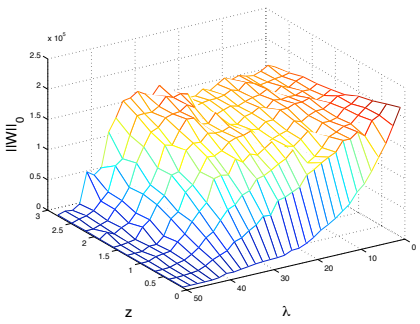
3.2.2 1-dimensional Corner Finding

One might expect that model selection would be a matter of taste. For applications where sparsity is more important, one ought to choose a higher λ and lower z, and for applications where accuracy is more important, one ought to choose a lower λ and a higher z. But our previous results have shown that $||H||_0$, and sometimes $||W||_0$, are sensitive to small values of λ, whereas for small values of λ we see little effect on reconstruction error and sparsity.

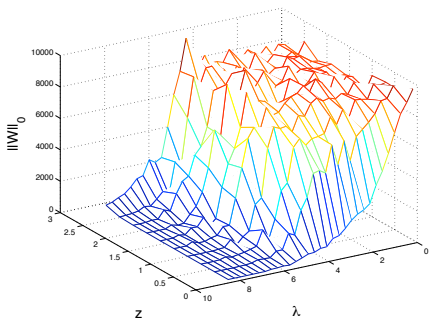
This is a situation that occurs relatively frequently in signal processing with regularization [8, 7]. In many signal processing applications, one wants to solve $X = WH + \epsilon$ for W, where X is a signal detected, W is a signal produced by some natural process, ϵ is noise, and H is a matrix depending on properties of the detector. This amounts to minimizing $||X - WH||_F$ over all choices of W, keeping H fixed. One often adds a regularization parameter to decrease the effect of noise upon one's solution, and one minimizes $||X - WH||_F^2 + \lambda||W||_1$ instead. In these applications it can be proven [8, 7] that the curve $||X - WH||_F^2$ versus $||W||_1$, parametrized by one's choice of λ, will have an L-shape, with the vertical portion of the L corresponding to regions where innacuracies are dominated by noise effects and the horizontal part corresponding to regions where inaccuracies are dominated by oversmoothing. For most applications, one ought to choose the λ parameterizing the corner of the L. Thus, detecting the corner of this characteristic L curve is a classic procedure for model selection.

Our situation is a little different. First, we are interested in 0-norms rather than in 1-norms. Second, we do not fix H, and we want to get H as sparse as possible. Nonetheless, our situation is similar to the one in which the L-curve is traditionally applied, and it is reasonable to explore L-curve ideas here as well.

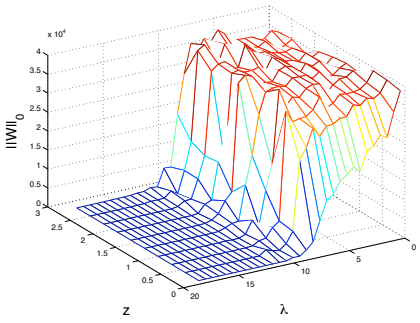
First we try plotting the curve of $||W||_0$ versus reconstruction error realized by our



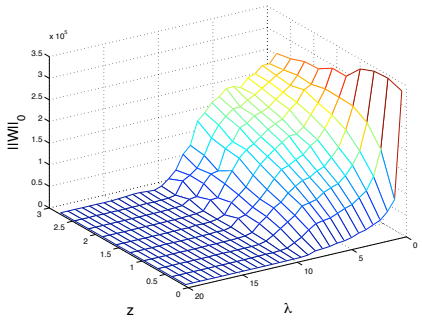
(a) 20ng



(b) bbc

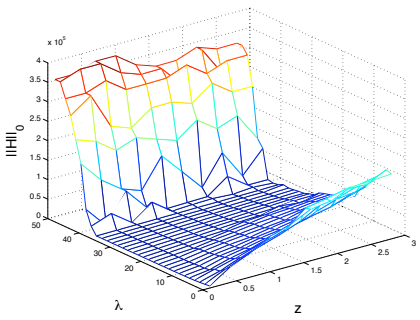


(c) reuters

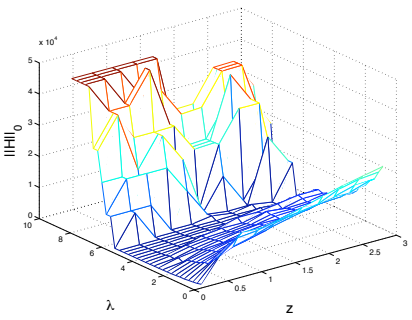


(d) tdt2top30

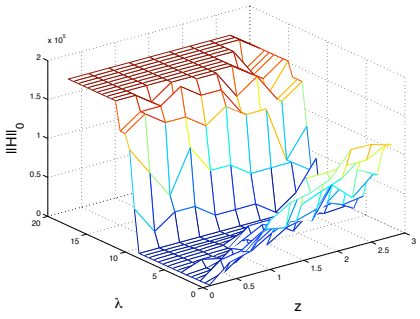
Figure 3.4: $||W||_0$ vs λ and z



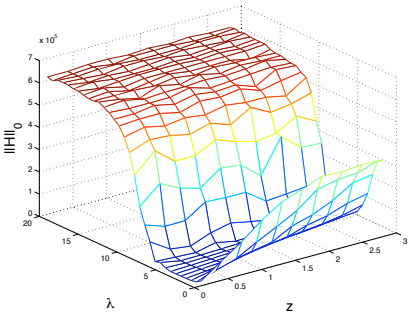
(a) 20ng



(b) bbc



(c) reuters



(d) tdt2top30

Figure 3.5: $||H||_0$ vs λ and z

algorithm, parametrized by choice of λ, fixing $z = 1$. The results are shown in Figure 3.6. For one data set, tdt2top30, we see a classic L-curve. That is because, as mentioned earlier, $||W||_0$ begins to fall before the reconstruction error blows up. For the other two data sets, however, we do not see an L-curve.

But recall that a higher λ leads to lower $||W||_0$ *and* lower $||H||_0$, and we care about both. If we plot the total zero norm $||W||_0 + ||H||_0$, we get more promising results (Figure 3.7). The most obvious difference is the spike in total 0-norm when λ gets very large, due to the expected explosion in $||H||_0$. But if we look at just the first half of the plot, we do see an L-curve (in fact in the tdt2top30 case we see an actual V, due to $||H||_0$ beginning to grow earlier). I will refer to this corner in the left-hand section of the curve as "corner 1", and the other corner which occurs where $||H||_0$ begins to grow rapidly as "corner 2".

We were unable to determine whether all matrices X would give rise to a curve with these properties. But in those cases where there is such a curve, corner-detection is a good method for model selection.

In order to find corner 1 without sampling too many choices of λ, we use a ternary search algorithm. The algorithm starts by choosing λ_-, parameterizing a point to the left of corner 1, and λ_+, parameterizing a point to the right of corner 1 but to the left of corner 2. Choosing such a λ_- is easy: clearly $\lambda_- = 0$ works, so that is what we do. As noted above (section **??**), if λ is greater than or equal to $z * \max_t ||X_t||_1$ then λ causes W to be set to 0, so $\lambda = z * \max_t ||X_t||_1$ parametrizes a point to the right of both corners. Next, we keep multiplying it by $a < 1$ until it no longer causes W to be zeroed out (meaning that is it to the left of corner 2). One must not choose a so small that λ_+ winds up to the left of corner 1, but if one takes a too small, this step in the algorithm will take a very long time. We have been using $a = \frac{3}{4}$.

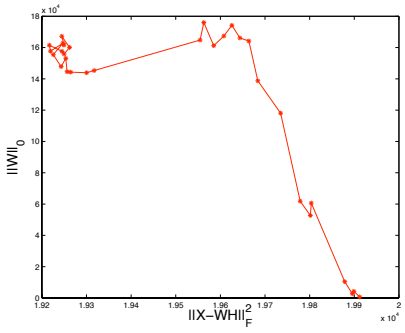
Next, the algorithm performs a ternary search. We choose $\lambda_- < \lambda_A < \lambda_B < \lambda_+$. These parametrize points q_-, q_A, q_B, q_+ in the $(||X - WH||_F^2, ||W||_0 + ||H||_0)$ plane. If the slope between q_- and q_+ is greater than the slope between q_A and q_B, then q_B must lie to the right of the corner, so we may update $\lambda_+ = \lambda_B$. Otherwise q_A is to the left of the corner, and we may update $\lambda_- = \lambda_A$. We repeat this process until a stopping condition is reached. At each step, the distance between λ_A and λ_B decreases by 33%, and at each step, the λ parameterizing the corner is in (λ_-, λ_+), so this allows us to zoom in on the corner.

Figure 3.8 reproduces the 0-norms versus error curve from before, with the 0-norm and error parametrized by the value of λ selected by this algorithm marked.

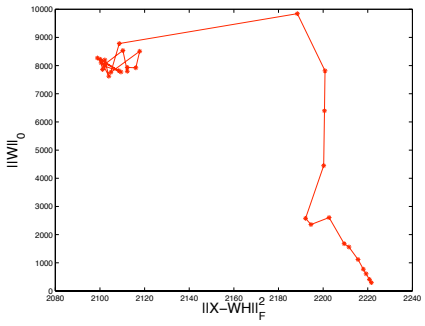
For λ, the error versus 0-norms curve parametrized by z also has nice corners, as shown in Figure 3.9. So holding λ fixed, we may use an identical algorithm to find the optimal z.

3.2.3 2-dimensional Corner Finding

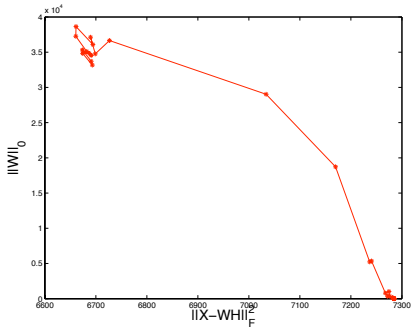
At this point, we have given algorithms for finding the optimal z with λ fixed and for finding the optimal λ with z fixed. One obvious way to select both would be to fix z and find the optimal λ, then fix that λ and find the optimal z, and alternate in this fashion until the algorithm converges to some (λ, z). However, this algorithm is not guaranteed to converge and frequently fails to find a good z or a good λ. Which λ appears optimal for a given z depends heavily on z, and vice versa, so if a bad choice of z is made in the beginning, the algorithm may fail entirely. In the following, we present an equally simple but somewhat more robust algorithm for finding z and λ. We were unable to empirically characterize the robustness of this algorithm because we have only 5 data sets, and we have not proven that it works. Future research might include characterizing its robustness or developing a hybrid algorithm which tries both the algorithm described below and the alternating one described



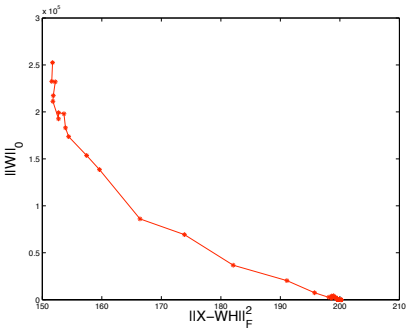
(a) 20ng



(b) bbc

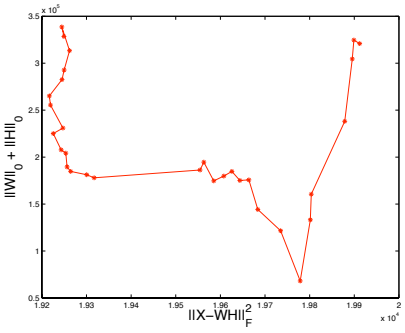


(c) reuters

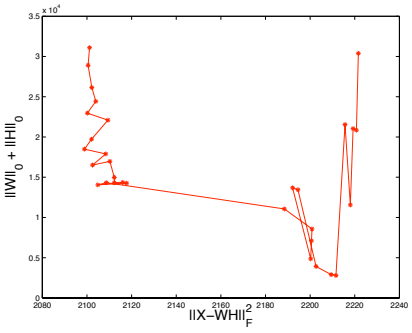


(d) tdt2top30

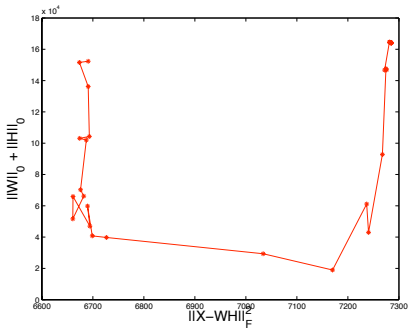
Figure 3.6: Plot of $||W||_0$ versus $||X - WH||_F^2$. Note that it is not typically an L-curve. All calculations were done with $z = 1$.



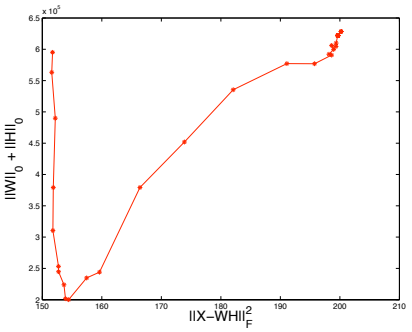
(a) 20ng



(b) bbc

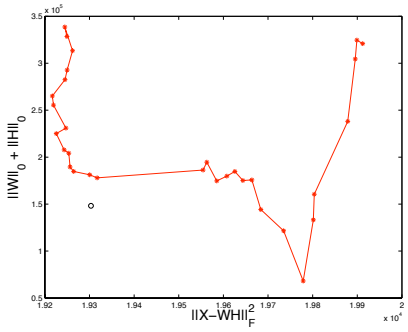


(c) reuters

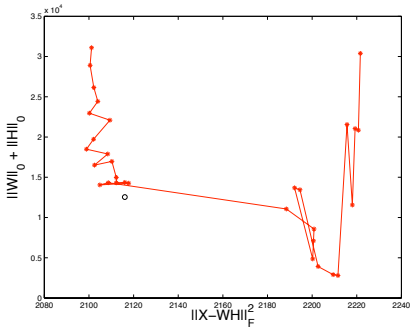


(d) tdt2top30

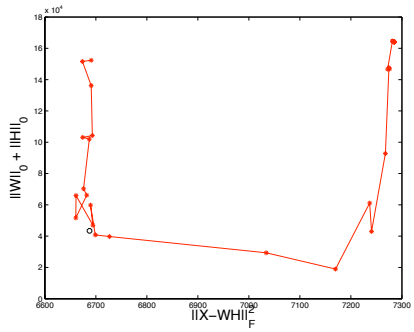
Figure 3.7: $||W||_0 + ||H||_0$ versus $||X - WH||_F^2$. Note that the left side approximaqtes an L-curve. All calculations were done with $z = 1$.



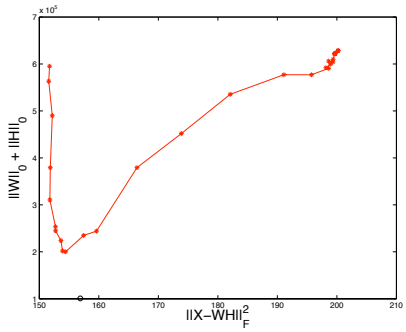
(a) 20ng



(b) bbc

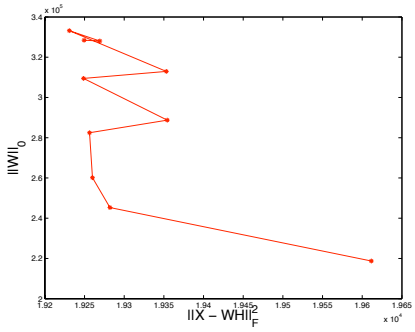


(c) reuters

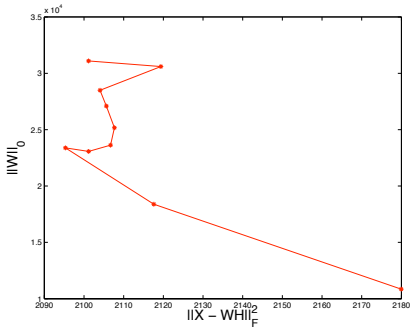


(d) tdt2top30

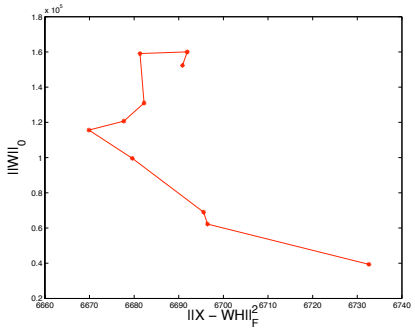
Figure 3.8: Results of our 1-dimensional corner-finding algorithm. The black circle represents the point parameterized by our algorithm's choice of λ. It consistently lies below the curve (sometimes far below) because we ran RRI on it with more iterations. All calculations were done with $z = 1$.



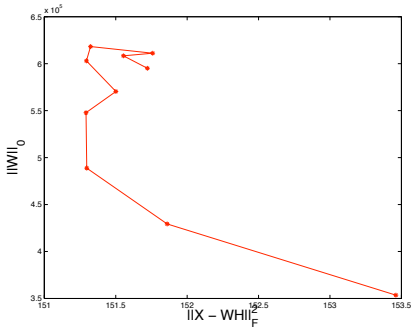
(a) 20ng



(b) bbc



(c) reuters



(d) tdt2top30

Figure 3.9: $||W||_0 + ||H||_0$ versus $||X - WH||_F^2$ for fixed λ and varying z. Note the (somewhat) L-curve-like shape, which allow for selection of z in the same way we select λ.

above and chooses the more reasonable of the two results (for example, choosing the one that gives a solution with lower l_2 norm in $(||X - WH||_F^2, ||W||_0, ||H||_0)$-space).

The idea for this algorithm comes from inspection of Figure 3.10. All four of these surfaces have two corners, one where $||X - WH||_F^2$ begins to increase rapidly and one where $||H||_0$ begins to increase rapidly. Analogously to the 1-dimensional L-curve, the former of these corners is the spot where inaccuracy flips from being primarily due to noise versus being primarily due to over-smoothing, and the tuple (λ, z) which parametrizes this point is the one we would like to choose.

The second of the two corners is sometimes absent, as occurs with the Twitter data set shown in Figure 3.11.

In order to find the former of the two corners, we begin by slightly generalizing the algorithm given above for finding λ. Let $\gamma : [0, 1] \to \mathbb{R}_+^2$ be any curve in the (λ, z)-plane, and let $a : \mathbb{R}_+^2 \to \mathbb{R}^2$ be the map from (z, λ) to $(||X - WH||_F^2, ||w||_0 + ||H||_0)$ where W and H are the solutions of the optimization problem for the given choice of λ and z. Then $a \circ \gamma$ is a curve somewhere on the $(||X - WH||_F^2, ||w||_0 + ||H||_0)$-plane. We can use the ternary search procedure described above to look for a corner along this curve. In the case where γ is the straight path from (λ_1, z) to (λ_2, z), this reduces to the algorithm for finding the optimal λ presented above.

In order to find σ, we first choose three points p_1, p_2, p_3 in the (λ, z)-plane such that the point parameterizing σ lies somewhere in the triangle with corner p_1, p_2, p_3. For this step we use guesswork, choosing points $p_1 = (\lambda_+, z_-)$, $p_2 = (\lambda_+, z_+)$, $p_3 = (\lambda_-, z_+)$, where λ_- and z_- are very small (0 and 10^{-12}, for example), z_+ is very large ($z_+ = 4$, for example), and λ_+ is chosen as in 1-dimensional corner finding. Next, we take each of the three sides in turn. For each side with vertices p_i, p_j, we check two points p_a and p_b intermediate between p_i and p_j. These parametrize points q_i, q_a, q_b, q_j on the $(||X - WH||_F^2, ||W||_0 + ||H||_0)$ plane. As before, if the slope between q_a and q_b is more negative than the slope between q_i and q_j, then the corner, if one exists, is parametrized by a point between pi and p_b, so we shrink the side of the triangle, resetting $p_j = p_b$. Otherwise, the corner is parametrized by a point between p_a and p_j, so we set $p_i = p_a$.

While we have not proven that this algorithm works, it is justified by the following intuition: if there is a pronounced corner somewhere along the curve parametrized by the line between p_i and p_j, the algorithm will find that corner with good accuracy. On the other hand, if there is not pronounced corner, the algorithm will not find an un-pronounced corner with any great accuracy, but it probably does not matter much if the algorithm messes up since there is no good corner there anyway.

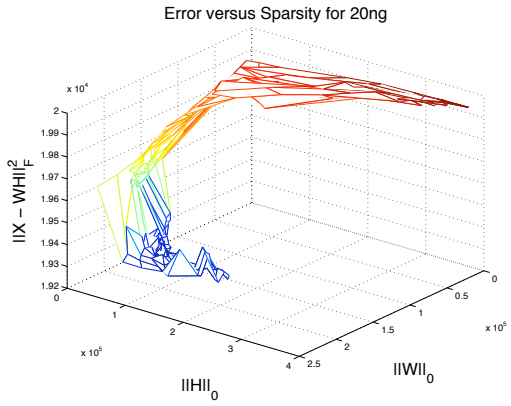
In Figure 3.12, we show the error and zero-norms parametrized by the duple (λ, z) chosen by this algorithm for each of the data sets. This shows that the algorithm is reasonably accurate.

Ultimately, we care more about the NMI than about the reconstruction error. Figure 3.13 shows the surface in NMI, $||W||_0$, and $||H||_0$ parametrized by the choice of λ and z. Again, the point parametrized by the λ and z chosen by our corner-finder is marked. These plots show that the algorithm makes a reasonable choice.

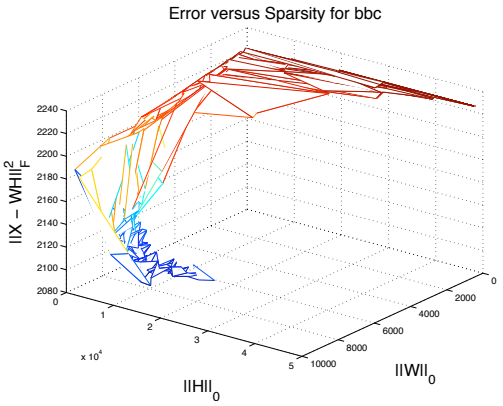
3.2.4 Model Selection via Cross-Validation

Cross-validation is another common method of model selection. In this section, we discuss how cross-validation might be used to choose λ and z in this context, but we ultimately conclude that it is not the right method.

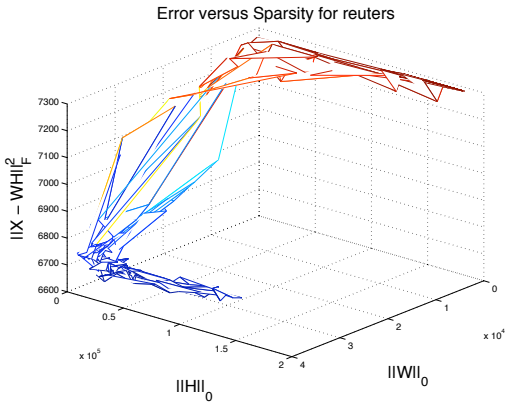
In cross-validation, we separate the data into two data sets: one to train the algorithm,



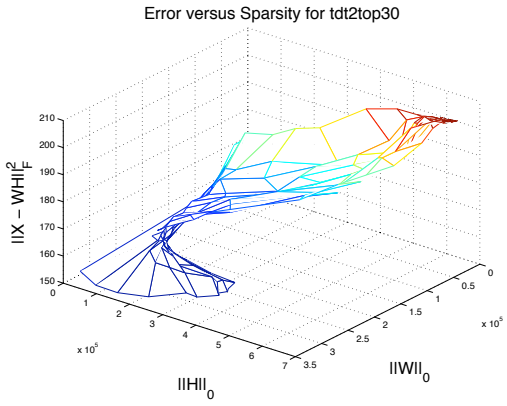
(a) 20ng



(b) bbc



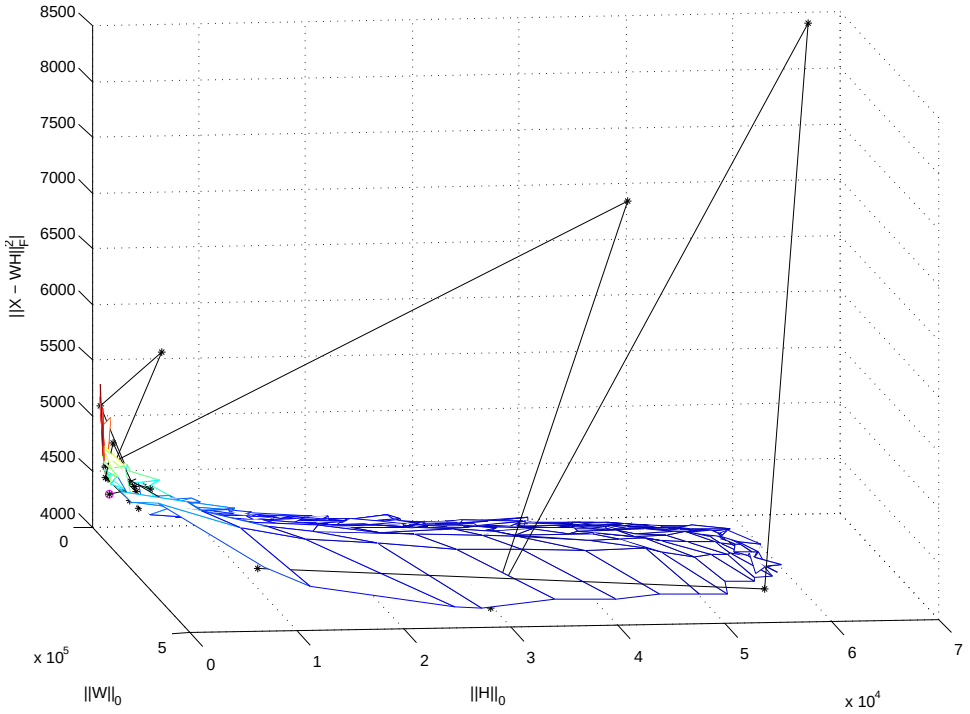
(c) reuters

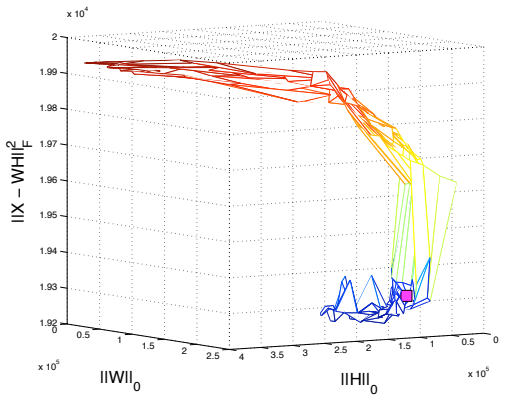


(d) tdt2top30

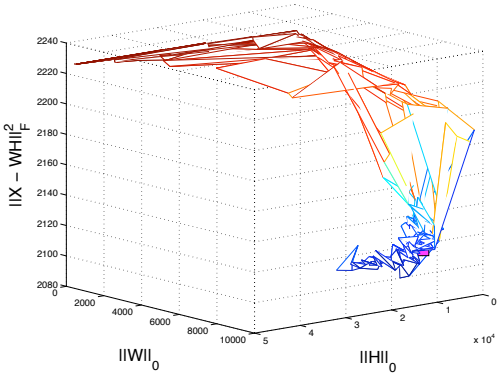
Figure 3.10: The surface parametrized by our choice of z and λ. Note the two corners.

Figure 3.11: For the Twitter.mat dataset the second corner (where $||H||_0$ grows rapidly) seems to be absent. The black dots are the points sampled by our corner-finder, and the magenta dot is the point ultimately selected.

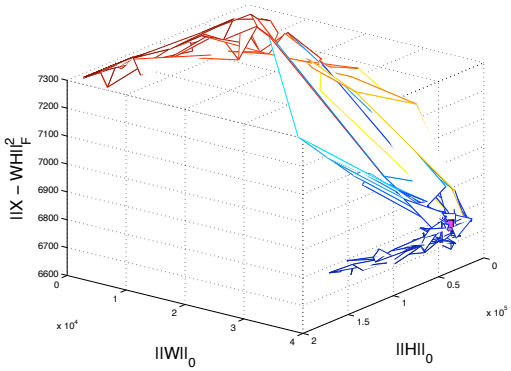




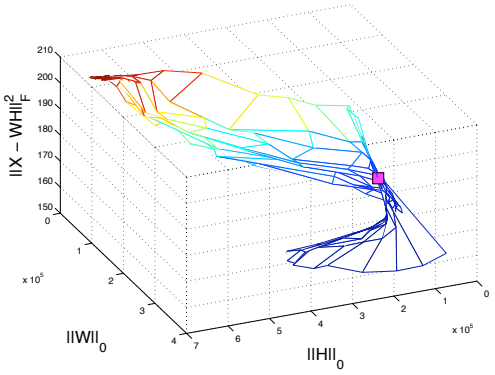
(a) 20ng



(b) bbc

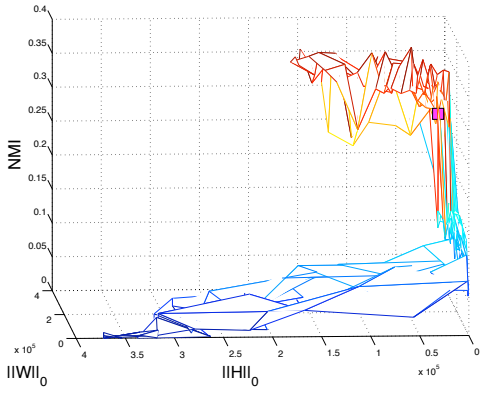


(c) reuters

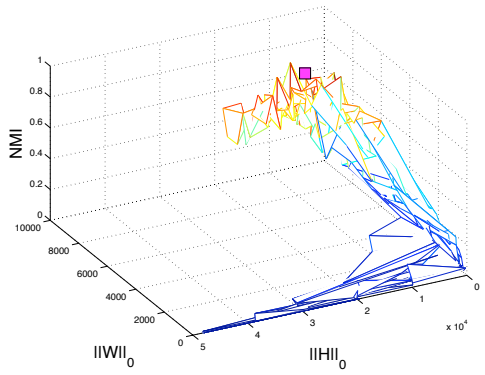


(d) tdt2top30

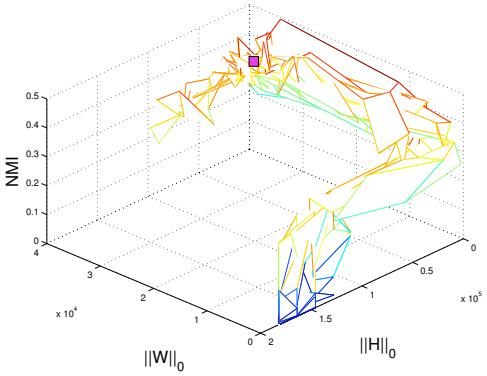
Figure 3.12: Results of our 2-dimensional corner-finding algorithm. The magenta square represents the point parameterized by our algorithm's choice of λ and z.



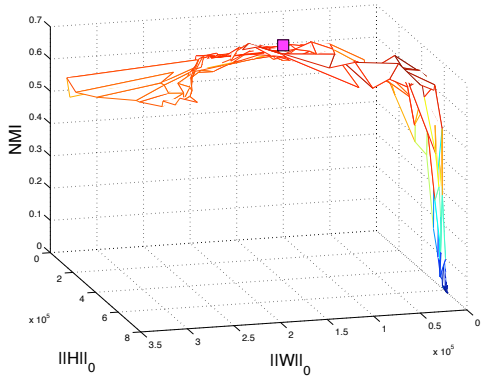
(a) 20ng



(b) bbc



(c) reuters



(d) tdt2top30

Figure 3.13: Results of our 2-dimensional corner-finding algorithm. The magenta square represents the point parameterized by our algorithm's choice of λ and z.

and one to test the trained algorithm. If the algorithm performs well on the test data set, then it has "learned the right things" and the model selected was a good one [9].

In our context, this works as follows: let X_1, X_2 be two matrices formed by removing some of the rows of X. Factor $X_1 \approx W_1 H_1$ and $X_2 \approx W_2 H_2$. Then factor $X_2 \approx \widehat{W}_2 H_1$, holding H_1 fixed and minimizing the objective function over all choices of $\widehat{W}_2$. H_1 is the matrix of topic-word associations for the documents represented by X_1. If X_1 was big enough, and if we made a good choice of λ and z, H_1 ought to be close to the true matrix of topic-word associations. That matrix of topic-word associations should still be correct when we pass to factorizing X_2. Thus, $||X_2 - W_2 H_2||_F^2$ should not be much smaller than $||X_2 - \widehat{W}_2 H_1||_F^2$.

So a procedure for model selection via cross validation would consist of performing the factorizations described above for various choices of λ and z, and choosing the λ and z that produced the smallest difference $diff = ||X_2 - \widehat{W}_2 H_1||_F^2 - ||X_2 - W_2 H_2||_F^2$. In Figure **??**, we plot $||X_2 - \widehat{W}_2 H_1||_F^2$ for various choices of λ using the Twitter data set for X where X_1 is 80% of the rows randomly selected and X_2 the remaining 20%. This result is typical: $diff$ grows monotonically with λ. Therefore, model selection via cross-validation will always choose $\lambda = 0$. Since our intent in this project is to find sparse solutions, cross-validation is not useful to us for model selection.

3.3 NMI Versus Reconstruction Error

One drawback of the approach to model selection presented above is that it chooses a model based on the sparsity and reconstruction error of the solutions it obtains, whereas what we are really interested in is the sparsity and NMI of the solutions obtained. Our goal is primarily to compute topic classifications which agree with what a human would decide, not just to find a factorization with low reconstruction error. Faced with a data set of blog content without annotations, we cannot use NMI to help us choose a good model and must use the reconstruction error instead. Thus, it is important to know how well reconstruction error is correlated with NMI. In particular, it is possible that sparser solutions tend to be more intuitive (higher NMI) than less sparse solutions *ceteris paribus*. If this is so, then choosing our model based on reconstruction error rather than NMI might result in choices of λ and z that are too small.

In order to resolve this question, we used the data computed earlier (reconstruction errors and NMI's) for various choices of λ and z and computed the correlation between NMI and $||X - WH||_F^2$ on this data for each of the four annotated datasets. There are some methodological issues with doing this. If we find that $W = 0$ for λ very large, then NMI $= 0$ and $||X - WH||_F^2$ attains its maximum. Thus, the more large values of λ we try, the stronger the (negative) correlation between NMI and reconstruction error will appear to be. In order to get rid of this arbitrariness, we sampled z evenly from 0.1 to 3 and sampled λ evenly from zero to a very high number. We then threw out all the data points for which W was set to 0 by the algorithm. This choice is justified by the following considerations: since what is at issue here is the validity of our approach to model selection, we are only interested in the correlation between NMI and reconstruction error over plausible model parameters, and it is implausible to choose λ so large that W is set to 0. On the other hand, all values of λ below λ_+ are plausible for certain applications (if one has a sufficiently large data set and sufficiently little memory at one's disposal).

The results are shown in Figure 3.15. NMI and $||X - WH||_F^2$ were linearly correlated in the data sets with R^2 values x,x,x, and x. tdt2top30, for which there were many points with

Figure 3.14: $||X_2 - \widehat{W}_2 H_1||_F^2$ versus λ

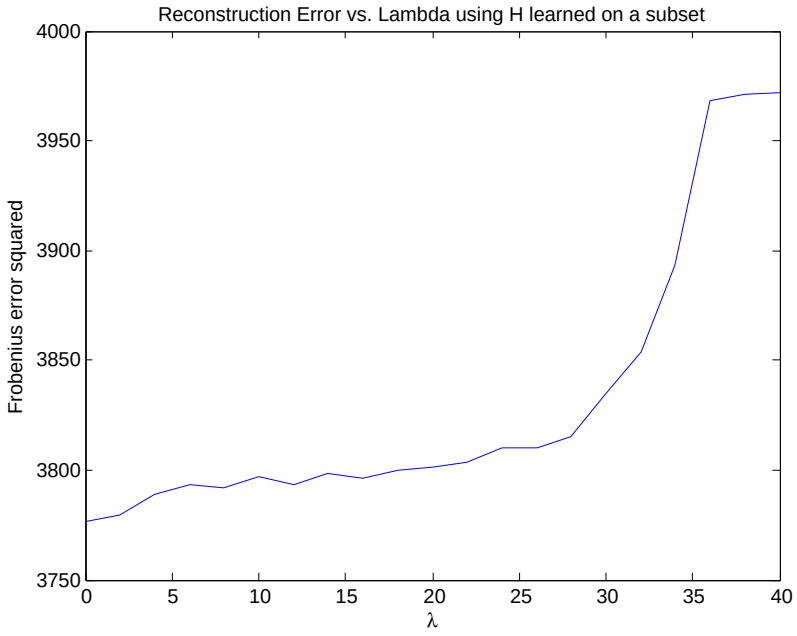


Table 3.1: R^2 values for the correlation between NMI' and $||H||_0$, $||W||_0$ and $||H||_0 + ||W||_0$ for each of the four datasets

bfdataset	NMI' versus $||W||_0$	NMI' versus $||H||_0$	NMI' versus$||W||_0 + ||H|_0$
20ng	0.0104	0.000477	0.0027
bbc	0.0014	0.0348	0.0195
reuters	0.00130	0.0888	0.0615
tdt2top30	0.0157	0.00100	0.00350

Table 3.2: R^2 corrleations bewteen NMI and sparsity, correcting for error

high error but also high NMI, is a clear outlier. The relatively high correlation between NMI and reconstruction error may be due more to a clustering effect than to a linear relationship: For λ below a threshold, NMI is more or less constant and high and error is more or less constant and low, while above that threshold, NMI is more or less constant and low and error is more or less constant and high. Thus, the data points are clustered around two centers, which makes it easy to fit a line to them. Nonetheless, this data provides some justification for using reconstruction error to determine model selection rather than searching for another proxy for NMI.

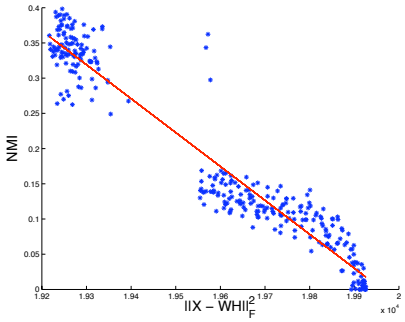
At the beginning of the study, it was hypothesized that $||H||_0$ and $||W||_0$ might be negatively correlated with NMI after controlling for reconstruction error, since sparser solutions to the factorization problem might be more intuitive. In order to test this, we went through each data point and subtracted the NMI that we would expect from the linear relationship with the reconstruction error from the actual NMI of that datapoint. We call this quantity NMI'. We then checked for a linear correlation between NMI' and $||H||_0$, $||W||_0$, and $||W||_0 + ||H||_0$. Table 3.1 shows the results of this calculation. The correlations were quite low, which suggests that sparse solutions do not tend to agree more with human annotations. However, this result is weakened by the fact that only linear relationships were checked.

3.4 Memory Usage Considerations

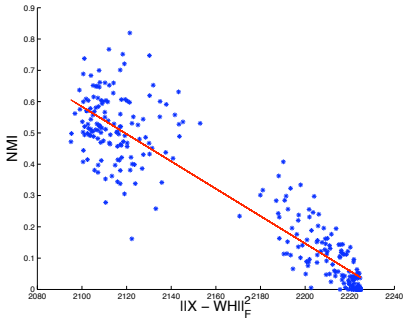
One other consideration which might be relevant to model selection is the amount of memory required to run the algorithm. One of the main points of enforcing sparsity is to decrease memory usage. But we are not only interested in the memory required to store and perform calculations with the factorization ultimately found by the algorithm; we also care about how much memory is required to run the algorithm in the first place.

Larger values of λ cause the sparsity of W and H to drop faster and stay down during the optimization. So if memory usage during the computation is very important, we might expect to want a higher λ and a lower z than the ones chosen using corner-finding.

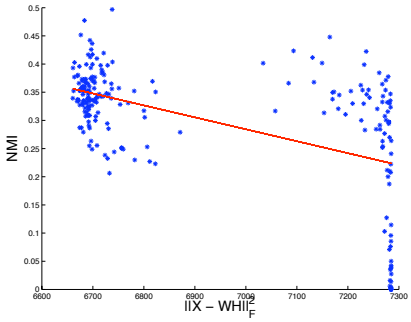
Unfortunately, it requires no less memory to run our sparse RRI algorithm than is required to perform RRI without 1-norm regularization and with H unconstrained. The reason for this is the initialization of W. Our current initialization procedure initializes W as an extremely dense matrix. The amount of memory required to run the algorithm is therefore dominated by the amount of memory required to store and manipulate this dense W during the first iteration of RRI, during which the values of W and H are no different than if we were using no 1-norm regularization and no constraint on H. Initializing W as a sparse matrix resulted in the algorithm failing to find a factorization, giving nearly as low



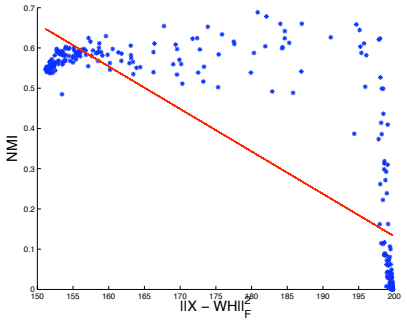
(a) 20ng: NMI $\approx (-4.81 \times 10^{-4})||X - WH||_F^2 + 9.61$, $R^2 = 0.931$



(b) bbc: NMI $\approx (-4.37 \times 10^{-3})||X - WH||_F^2 + 9.76$, $R^2 = 0.859$



(c) reuters: NMI $\approx (-2.13 \times 10^{-4})||X - WH||_F^2 + 1.77$, $R^2 = 0.280$



(d) tdt2top30: NMI $\approx (-1.06 \times 10^{-2})||X - WH||_F^2 + 2.24$, $R^2 = 0.673$

Figure 3.15: Correlation between NMI and reconstruction error.

a value of the objective function as was found when W was initialized densely (initializing H densely, on the other hand, did not confer significant benefits).

Therefore, the most promising avenue for reducing the memory usage of the algorithm is to find a better initialization procedure. We believe that using model selection to reduce memory usage would be premature; significant gains in memory usage will not be attained until a better initialization procedure has been found, and a different initialization procedure might greatly alter the effects of model selection on memory usage.

Chapter 4

Evaluation of Overall Performance

In this chapter we evaluate the overall performance of our sparse NMF algorithm together with model selection, by comparing it with the sparse NMF algorithm of Patrik Hoyer and by running some cross validation tests.

4.1 Comparison with Hoyer's Algorithm

Patrik Hoyer developed a sparse NMF algorithm in 2004 [11] that builds on top of Lee & Seung's 2001 NMF algorithm by enforcing sparsity constraints onto each column of W, H after Lee & Seung's update steps have been computed. The sparsity constraint is enforced by a projection algorithm which projects each constrained vector w onto the hyperplane $\sum w_i = l_1$, and then projects this point onto a hypersphere with some desired l_2 norm. It is not clear whether this projection guarantees that the projected vector minimizes the objective function over all vectors in the constraint set.

Hoyer's algorithm is similar to Sparse RRI in that it also requires a sparsity parameter. His algorithm takes a sparsity parameter $s(w)$ and $s(h)$ that determines what Hoyer sparsity each column of W or column of H, respectively. Where we define the Hoyer Sparsity of a vector x as

$$\mathrm{s}(x) = \frac{\sqrt{n} - \frac{||x||_1}{||x||_2}}{\sqrt{n} - 1}$$

where $n = \dim(x)$.

Hoyer originally developed his algorithm to generate parts-based basis images for datasets where non-sparse NMF did not perform well, however we compare it to Sparse RRI on textual data. We summarize these results in Table 4.1. There are several observations we can make:

1. For Hoyer's algorithm as well as ours there is an optimal sparsity parameter, which can be found using slope-based methods.

2. Sparse RRI achieves better reconstruction error than Hoyer's algorithm for a given sparsity for small data sets with relatively high initial density. (TDT, BBC)

3. Hoyer's algorithm seems to perform slightly more consistently for larger and sparser datasets (20NG, Reuters). Sparse RRI seems to behave strangely for particular values of λ.

4. Neither algorithm consistently achieves a better NMI than the other, for certain sparsity parameter values both algorithms achieve similar NMI values.

These results give us the confidence to say that sparse Sparse RRI is a good solution to the sparse NMF problem, but is not significantly better than existing sparse NMF algorithms. The encouraging result that can be seen in Table 4.2 is that Sparse RRI seems to use less memory and run faster than Hoyer's RRI algorithm.

4.2 Cross Validation

As a final check that the algorithm is giving plausible results, we do a cross-validation check on each of the data sets. For each one we randomly shuffle the rows of X (so that all permutations are equally likely), and let X_j, $j = 0, ..., 20$, be the first $80 + j\%$ of the rows. We choose optimal z and λ for X_0 and factor $X_0 \approx W_0 H_0$. Keeping the same λ we then factor $X_j \approx W_j H_0$ for each J and let $e_j = ||X_j - W_j H_0||_F^2$. do this for 3 shufflings. Figure 4.1 shows the results of this test. If $e_j - e_0$ were very large, especially if this were so for small j, this would suuggest that our algorithm's choice of H is overly sensitive to small changes in X, which would suggest that H is an implausible topic-word associataion matrix. But such troubling behavior does not occur.

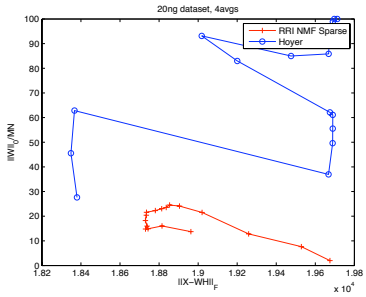
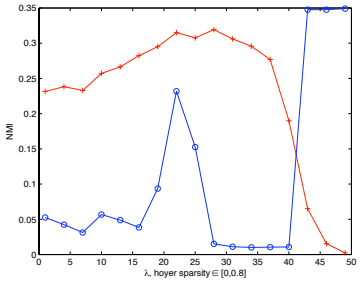
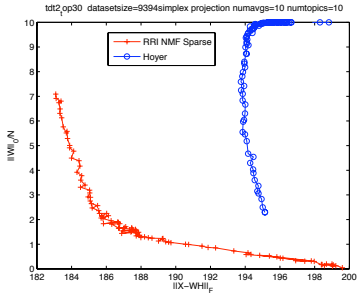
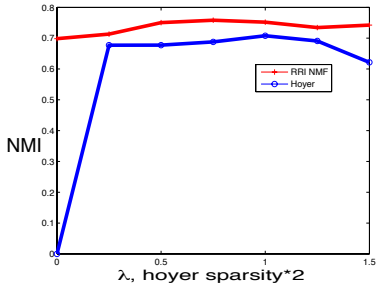
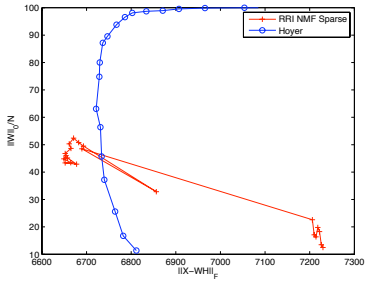
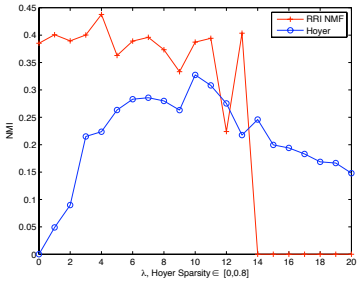
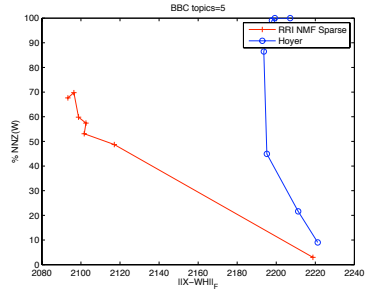
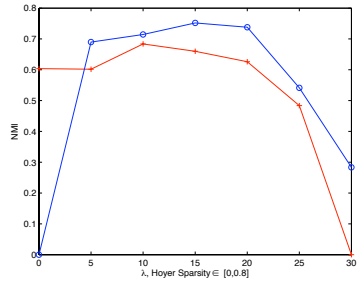
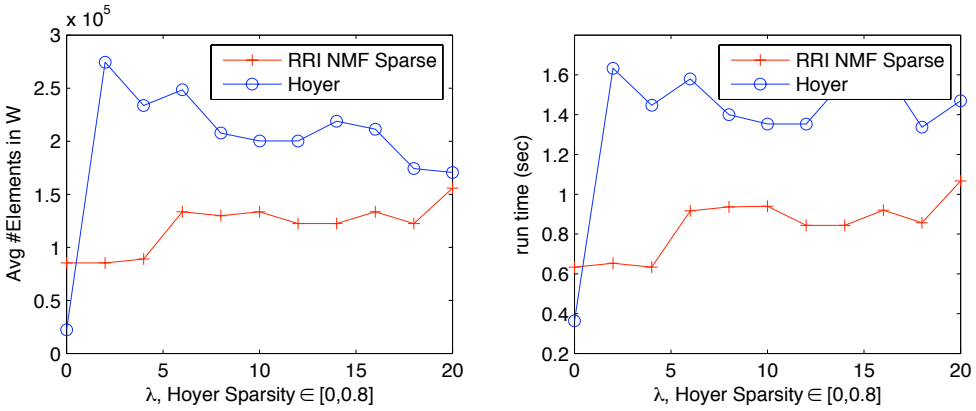
	Reconstruction Error	**NMI**
20NG dataset (89 topics, 19928 docs, 18607 words, 0.39% density)		
TDT dataset (30 topics, 9394 docs, 6545 words, 1.66% density)		
Reuters dataset (10 topics, 7285 docs, 18221 words, 0.25% density)		
BBC dataset (5 topics, 2225 docs, 9635 words, 1.34% density)		

Table 4.1: Comparison of Sparse RRI and Hoyer's sparse NMF algorithm on various text datasets.

Average (over iterations) Number of Entries in W, Run time

BBC dataset



TDT dataset (The Average for RRI remains around 280,000)

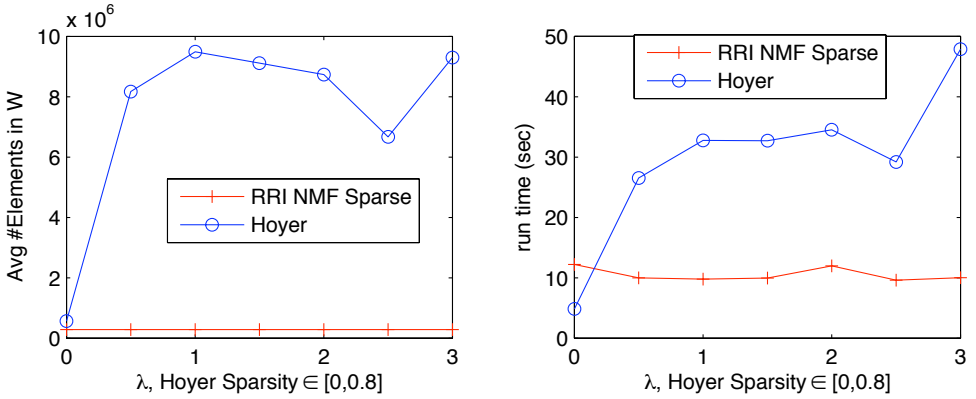
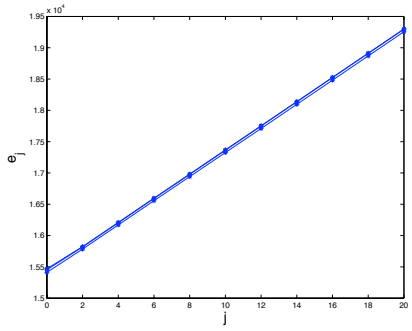
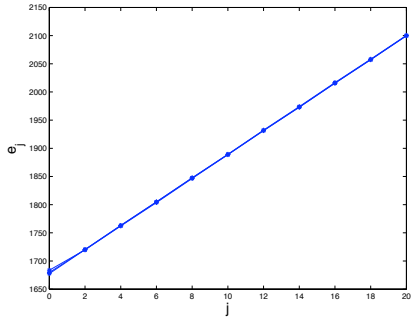


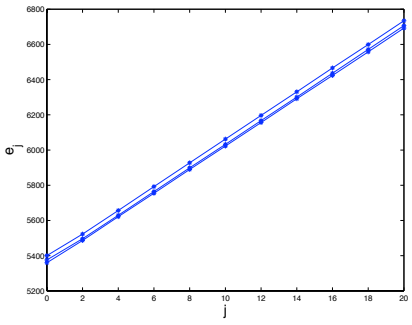
Table 4.2: Comparison of space & time requirements of Sparse RRI and Hoyer's sparse NMF algorithm.



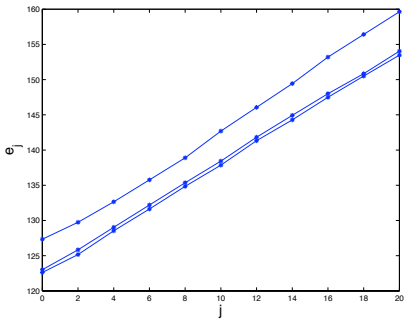
(a) 20ng



(b) bbc



(c) reuters



(d) tdt2top30

Figure 4.1: Results of the cross-validation test applied to the four datasets. The three lines correspond to three different shufflings of the data. Error rises only slightly, and very regularly, as we include more and more data.

Chapter 5

Introduction to MGGCM

In this chapter we introduce the algorithm we use to determine who is important online.

5.1 Mathematical Background

One key aspect of online communities that businesses would like to understand is *influence*. Who is most influential in a community? Who originates ideas and who propagates them? By determining the most influential people in an online community, businesses can identify who they should target for costly marketing interventions (such as distribution of free products to generate buzz). This knowledge would make marketing strategies more effective and efficient.

5.1.1 Granger Causality

To approach this problem we must first determine what we mean by causal influence. How can we characterize influence in an online community to identify who is most influential? Past efforts have focused on hyperlinks to illustrate influence: if person A links to person B's blog or website, person B is expected to exert an influence on person A. But what happens if we believe that people can be influenced more subtly?

Consider the following example of a more subtle form of influence. I discover a new restaurant that I want to try online. I read a review of the restaurant on a food blog, and the author of this blog raves about the duck confit at the restaurant. When I later go to the restaurant, I naturally want to try the duck confit. Afterward, I myself write a review of the restaurant on my blog and write about the duck confit. I do not link specifically to the food blog that I originally read, but I was clearly influenced by the other blogger. In this case a comparison of the actual content of each blog would suggest that the content of our blogs is correlated. The fact that content about the duck confit appeared on the other blog before it appeared on mine suggests that I was influenced by the other blogger.

5.1.2 Methodology

The sort of reasoning above was made explicit by the Nobel prize winner Clive Granger in order to form a mathematical definition of influence. Applied to our case it would be defined as follows:

Granger Causality: A group of bloggers is said to influence blogger B_i if their past collective content (i.e. their past blog posts) is predictive of the future content of blogger B_i more so than the past content of blogger B_i alone.

In other words, we determine causal influence by examining the content of blogs while considering the temporal order of events. Referring back to the past example, we would see that I was influenced by the other blog because my current content reflects the other blog's past content. This definition allows us to capture influence that is not immediately apparent. We can thus determine who is most influential in guiding or shaping a discussion, rather than merely determining who is most cited in a discussion as past efforts at characterizing influence have done.

To quantify the notion of one blogger's content being *predictive* of another's we must model blog content over time mathematically. Recall that the content of a blog post may be represented as a vector in $\mathbb{R}_+^d$, where d is the size of our dictionary. Thus to any blogger B_i we may associate a time series $\{\mathbf{B}_i^t\} \subseteq \mathbb{R}_+^d$, where $\mathbf{B}_i^t$ is the vector representation of all content produced by blogger B_i during time-step t, treated as a single document. *Predicting* the content of B_i in terms of the past content of B_j consists of solving a multivariate regression with response variable $\mathbf{B}_i^t$ and explanatory variables $\{\mathbf{B}_j^1, ..., \mathbf{B}_j^{t-1}\}$.

5.1.3 Our Approach

Given time-series $\{\mathbf{B}_i^t\}$ corresponding to a community of bloggers $B_1, ..., B_l$ we wish to output a weighted directed graph whose vertices are the bloggers, with an edge from blogger B_j to blogger B_i if and only if B_j influences B_i, and with a weight on each such edge quantifying the degree of influence which B_j has over B_i.

To construct this output graph, we process each blogger one at a time. For blogger B_i, we attempt to predict his current content at time t, $\mathbf{B_i^t}$, in terms of the past content of the other bloggers. We use a variable selection method that determines which variable groups or bloggers are relevant in predicting the response variable $\mathbf{B_i^t}$, and the selected groups will tell us what edges or causal relationships should be included in the output graph. The regression coefficients for each pair of bloggers will determine the weight to be put on the edge between them.

There are two key features that we would like our variable selection procedure to have:

1. The selection process should select groups instead of individual content variables. If blogger B_j influences blogger B_i, we want to say that the entirety of the past content of blogger B_j predicts the content of blogger B_i, not just the occurrence of a particular word in the past content of blogger B_j. We thus treat the past content of bloggers as groups of variables that must be selected together.

2. The selection process should be made simultaneously for each of the K word frequencies of $\mathbf{B_i^t}$. In other words, if we select blogger B_j as influencing blogger B_i, then we expect the past content of blogger B_j to predict the entirety of the content of blogger B_i and not just predict the frequency of a certain word in blogger B_i's current content.

With these two features, our method should faithfully reproduce the Granger causal influence relations amongst. We will refer to the algorithmic instantiation of the above as Multivariate Group Granger Causal Modeling (MGGCM). In order to implement MGGCM, however, we still need to detail a simultaneous group variable selection procedure.

5.1.4 Multivariate Group Orthogonal Matching Pursuit (MGOMP)

MGOMP is a simultaneous group variable selection procedure devised by Aurélie Lozano, Grzegorz Świrszcz and Naoki Abe [2] which performs multivariate regression with variable group selection.

For our case, we will run the MGOMP algorithm for each blogger B_j to determine which other bloggers have a causal influence on blogger B_j. We let Y be the content of blogger B_j that we want to predict. The variable groups are sets $\mathcal{G}_i = \{\mathbf{B_i^1}, \mathbf{B_i^2}, ...\mathbf{B_i^{\tau-1}}\}$. X_G is a matrix of explanatory variables in group G. $\mathcal{A}$ will be the set of groups selected to have an influence on blogger B_j.

The Algorithm

1. Set $Y = \begin{bmatrix} B_j^{\tau} \\ B_j^{\tau-1} \\ \vdots \\ B_j^{d} \end{bmatrix}$
2. Initialize variables: Set $\mathcal{A} = \emptyset$, $W_{\mathcal{A}} = 0$
3. Calculate initial residual: $R = Y - X_{\mathcal{A}} W_{\mathcal{A}}$
4. While $||R||$ > tolerance
   (a) Variable selection:
   $$i = \arg\min_k \min_{W_{\mathcal{G}_k}} ||R - X_{\mathcal{G}_k} W_{\mathcal{G}_k}||_{fro}^2$$
   (b) Update variable set: $\mathcal{A} = \mathcal{A} \cup \mathcal{G}_i$
   (c) Update residual: $R = Y - X_{\mathcal{A}} W_{\mathcal{A}}$

   end while

Initially, we say that we have selected no groups and the regression coefficient matrix is the zero matrix. Then, for as long as the residual is greater than some tolerance that we have set, we determine which group of variables explains the most of the remaining residual. We then select that group, update the selected variable set and residual, and then repeat the selection step if the residual is still greater than the tolerance. Since we select the optimal group to reduce the residual at every iteration, we say that MGOMP is a greedy algorithm.

Stopping the algorithm before all groups have been included in $\mathcal{A}$ ensures that not too many variable groups are selected (we might expect that any explanatory variable, so long as it varies somewhat, no matter how unrelated it actually is to the predicted variable, might be able to decrease the residue infinitesimally. Including the stopping criterion prevents this). The greediness of the algorithm avoids problems with correlations between explanatory variables. Suppose that B_a and B_b are both influenced by B_c, and that B_b does not influence B_a, and that we are trying to predict B_a. Then the explanatory variables B_b and B_c are correlated, so that both B_b and B_c will be correlated with B_a. If we try to predict B_a, MGOMP will select B_c first. On the next iteration it will *not* select B_b, since B_c has already been selected, and B_b will not predict any content of B_a that is not already predicted by B_c. Thus MGOMP ought to ameliorate the problems due to correlations between explanatory variables.

5.1.5 Kernelization

To implement MGOMP, we actually use a kernelized method which is more efficient than the algorithm detailed above. For very high-dimensional problems, kernelization can be beneficial to reduce the problem to a lower-dimensional one.

In our case, $\mathbf{X} \in \mathbf{R}^{(\tau-\mathbf{d})*\mathbf{dKG}}$ and $\mathbf{Y} \in \mathbf{R}^{(\tau-\mathbf{d})*\mathbf{K}}$, where τ is the number of observations or time steps, d is the lag, K is the number of words in our dictionary, and G is the number of bloggers in our network. Since we expect that $\tau - d << dKG$, kernelization is a good choice since we can shift our problem from dKG dimensions to $\tau - d$ dimensions.

The Kernelized Algorithm

1. Set $Y = \begin{bmatrix} B_j^{\tau} \\ B_j^{\tau-1} \\ \vdots \\ B_j^{d} \end{bmatrix}$
2. Precompute: $K_{\mathcal{G}_i} = X_{\mathcal{G}_i} X_{\mathcal{G}_i}^T$ for $i = 1, \ldots, G$
3. Precompute: $F_{\mathcal{G}_i} = I - K_{\mathcal{G}_i}(K_{\mathcal{G}_i} + \lambda I)^{-1}$ for $i = 1, \ldots, G$
4. Initialize variables: Set $\mathcal{A} = \emptyset$, $K = 0 \in \mathbf{R}^{(\tau-\mathbf{d})*(\tau-\mathbf{d})}$
5. Calculate initial residual: $R = Y$
6. While $||R|| >$ tolerance
   (a) Variable selection:
   $$i = \arg\min_k ||F_{\mathcal{G}_k} R||_{fro}^2$$
   (b) Update variable set: $\mathcal{A} = \mathcal{A} \cup \mathcal{G}_i$
   (c) Incorporate chosen kernel: $K = K + K_{\mathcal{G}_i}$
   (d) Update residual: $R = I - K(K + \lambda I)^{-1} Y$

   end while

Because we are working in a much lower dimension, the kernelized algorithm should be much faster than the original proposed algorithm. In addition, we no longer have to find the optimal W for each blogger during each iteration of the selection process, which can be a difficult calculation. Thus, for our implementations of MGOMP when applied to large data sets such as real Twitter data, we use the kernelized version of MGOMP.

Chapter 6

Stopping Criteria for MGOMP

6.1 Testing MGGCM on Simulated Data

Before applying our algorithms to real-world data, we want to test our methods on simulated data. To produce these simulations, we first generate an underlying true blogger network structure. This structure can be represented as a $G * G$ adjacency matrix where the (i, j) entry represents the influence that blogger B_i exerts on blogger B_j. This $G * G$ adjacency matrix is what we hope our MGGCM algorithm will recover as its output. From this, we use a Vector Auto-Regression (VAR) model [19] to generate the simulated data. If $\mathbf{B^t}$ is the vector of all bloggers' content at time t, then a VAR model is expressed as the following:

$$\mathbf{B^t} = \mathbf{A_1} * \mathbf{B^{t-1}} + \ldots + \mathbf{A_d} * \mathbf{B^{t-d}} + \text{noise}$$

A_l for lag $l = 1, \ldots, d$ is a coefficient matrix that explains the relationship between bloggers. A_l is only non-zero where the associated edge in the $G * G$ adjacency matrix is also non-zero, so our model will reflect the underlying true blogger structure. We randomly generate the initial content at time 0 and the noise values at each step, and we apply the model to the content at time step $t - 1$ to find the content at time step t.

We then run the MGGCM algorithm on the simulated data to see if we can recover the true structure of the blogger network. We compare the output of our algorithm to the underlying randomly generated structure and use three statistical measures of variable group selection accuracy:

1. Precision
$$P = \frac{|\{\text{relevantgroups}\} \cap \{\text{retrievedgroups}\}|}{|\{\text{retrievedgroups}\}|}$$

2. Recall
$$R = \frac{|\{\text{relevantgroups}\} \cap \{\text{retrievedgroups}\}|}{|\{\text{relevantgroups}\}|}$$

3. Harmonic Mean of Precision and Recall
$$F1 = \frac{2PR}{(P + R)}$$

In this series of tests, we set the number of bloggers $G = 20$, the number of words $K = 10$, the lag $d = 5$, and the number of time steps $\tau = 100$.

6.1.1 Preliminary Results

We ran the MGGCM code on 20 sets of simulated data and averaged the evaluations measures to come up with the following results.

Method	Precision	Recall	$F1$
MGGCM	.4088	.8957	.5609

While the algorithm works with rather high recall, the precision is relatively low, which in turn leads to a low $F1$ value. This implies that MGGCM does well in choosing groups that are relevant, but it also retrieves many groups that are irrelevant. If we are able to find a better stopping criterion to the selection process, then we should find that the precision and $F1$ will increase.

6.1.2 Oracle Estimate

In order to test the hypothesis that it suffices to find a good stopping condition in order to make the algorithm perform well, we use an oracle estimate. Because we know the in-degree of each blogger in the true randomly generated influence graph, we can alter the MGGCM code to include an oracle estimate with this information. Thus, the selection process will stop once it has chosen as many groups as the oracle estimate allows for each individual blogger. The oracle does not tell the algorithm which groups to select; the oracle merely tells the algorithm when to stop for each blogger. Adding an oracle estimate thus stops the selection process prematurely when compared to the normal MGGCM code. The results averaged over 20 sets of simulated data are shown below.

Method	Precision	Recall	$F1$
MGGCM	.4088	.8957	.5609
Oracle Estimate	.9656	.8957	.9292

Adding an oracle estimate to our algorithm greatly increases the precision and $F1$ value while leaving recall the same. This implies that our algorithm does a good job at choosing groups initially, but often chooses too many groups when the stopping criterion is the residual falling below the tolerance level. Thus, if we find an appropriate stopping strategy for our MGGCM algorithm, we will have a very good way to identify the underlying causal relationships in a blogger network.

Of course, we cannot use an oracle estimate in the real world since we have no way to determine how many bloggers should be chosen as influential. However, real data might be less in need of one. In our simulations, it is not uncommon for a blogger to have no influencers, while this should be rarer in Twitter data. Also, because our simulations are run on very small networks of bloggers with relatively few edges in the causal influence graph, a small number of errors in the selection process can have a large effect on precision and recall values. These problems would disappear when looking at larger networks of bloggers.

6.2 Stopping Conditions for MGOMP

For real world applications we must find a stopping criterion for MGOMP which uses no knowledge of an underlying structure.

6.2.1 Residual Plots

For inspiration, we plot the residuals that are calculated at each step of the selection process. Examples of these residual plots on simulation data are 6.1 and 6.2.

Heuristically, we can see which groups we hope that the algorithm selects for each figure. In Figure 6.1, we hope that the algorithm will select blogger 15 as an influencer as that is the primary and only group that lowers the residual significantly. In Figure 6.2, we hope that the algorithm selects bloggers 3 and 7 as these correspond to the two significant dips in the residual.

Because MGOMP is a greedy selection algorithm, it will select the group that has the least residual in a given iteration. While this works well for the examples with one or two influencers for the first iteration, this is not the behavior that we want for the example with no influencers, nor is it guaranteed to correctly select the second influencer in the case of two influencers during the second iteration.

6.2.2 Threshold Criterion

A group should be selected if and only if it explains a significant portion of the remaining residual; otherwise, we want the algorithm to terminate. This can be thought of as including a group if and only if the residual when selecting that group is lower than a *threshold* multiplied by the average residual of selecting any other group. Assuming that we have G bloggers, we modify the selection step as follows:

if

$$\text{if} \min_{W_{\mathcal{G}_j}} ||R - X_{\mathcal{G}_j} W_{\mathcal{G}_j}||_{fro}^2 < \text{threshold} * \frac{1}{G} \sum_{k=1}^{G} \min_{W_{\mathcal{G}_k}} ||R - X_{\mathcal{G}_k} W_{\mathcal{G}_k}||_{fro}^2$$

then

$$\mathcal{A} = \mathcal{A} \cup \mathcal{G}_j$$

This is equivalent to adding a group as an influencer if and only if it deviates substantially from the mean in terms of its explanatory power. When applied to the simulation data with the same parameters as above and letting the threshold = .825, we find that this method is very successful.

Method	Precision	Recall	$F1$
MGGCM	.4088	.8957	.5609
Oracle Estimate	.9656	.8957	.9292
Threshold	.9227	.8181	.8605

Adding a threshold criterion maintains a very high precision for the algorithm while not losing too much in recall. This seems like a good solution to our selection problem; however, we question its applicability to non-simulation data. Figures 6.1 and 6.2 show how nicely the simulation data behaves, and we have little reason to think that real-world data would so clearly reveal who is influencing a given person.

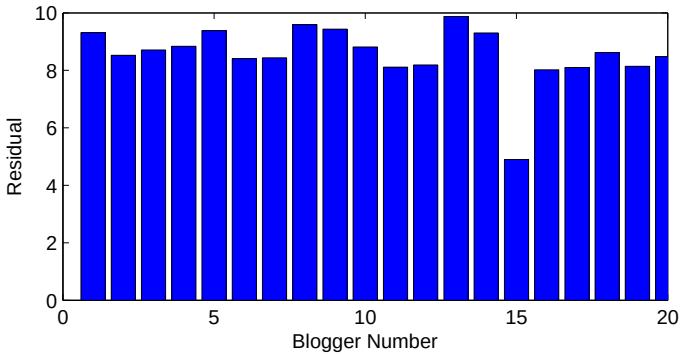


Figure 6.1: Residual Plot of Simulation Data for a Blogger with One Influencer

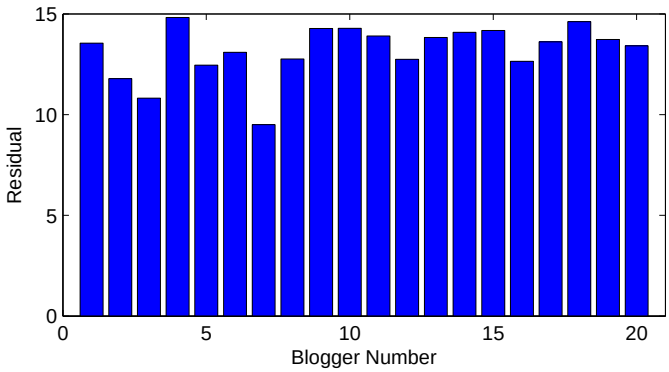


Figure 6.2: Residual Plot of Simulation Data for a Blogger with Two Influencers

6.2.3 Model Selection Criterion

An alternative to stopping the algorithm is to let the algorithm run for a given amount of time, examine the results at various time steps, and artificially pretend that we stopped the algorithm at an earlier step and use those results to continue. Since the algorithm gets to see what would happen if it keeps selecting more variables, it has access to more information when deciding which explanatory variables to ultimately include. To do this, we calculate a *model selection criterion*, *modsel*, at each iteration of the algorithm. We let the algorithm run for some maximum number of iterations, and then we find the iteration at which the modsel attains its minimum value and throw out all calculations performed at later iterations. We define modsel as the following:

$$\text{modsel}[i] = \log(\frac{1}{G} \min_k \min_{W_{\mathcal{G}_k}} ||R - X_{\mathcal{G}_k} W_{\mathcal{G}_k}||_{fro}^2) + \delta * \frac{\log(n)}{n} * \text{df}[i]$$

where δ is a parameter, n is the number of data points used, and df$[i]$ is the number of non-zero regression coefficients at iteration i divided by the number of words in our dictionary K.

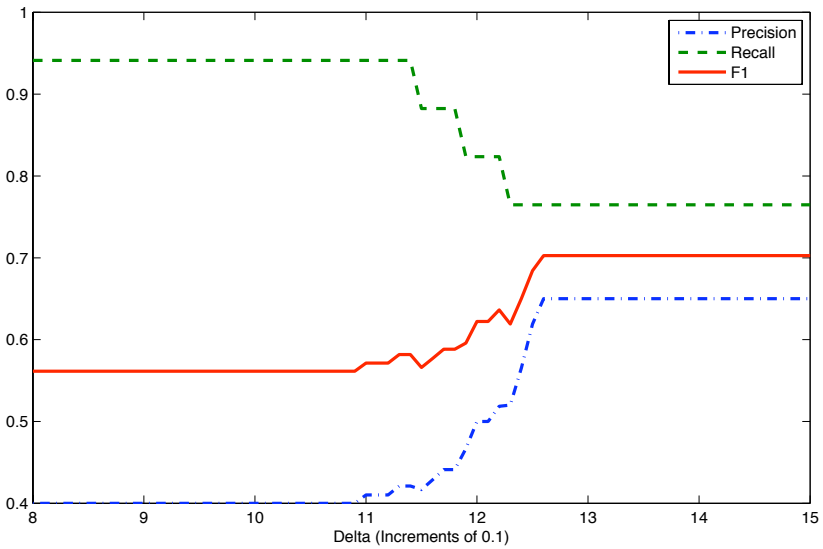
Intuitively, the first term of the model selection criterion weighs the strength of the approximation while the second term weighs the sparsity of the solution. Thus, when δ is small, we expect MGGCM with model selection to perform identically to MGGCM without model selection because the model selection criterion is minimized when the residual is smallest, which will happen when the residual is smaller than the stopping criterion, as occurs in the normal MGGCM algorithm. When δ is very large, we expect the algorithm to save information from only the first iteration since the sparsity of the regression coefficient matrix is the dominating factor. We experimented with different values of δ to determine how best to vary the weighting of these two conflicting priorities. 6.3 displays the results of varying δ for an algorithm implementing model selection applied to simulation data with the same parameters as above.

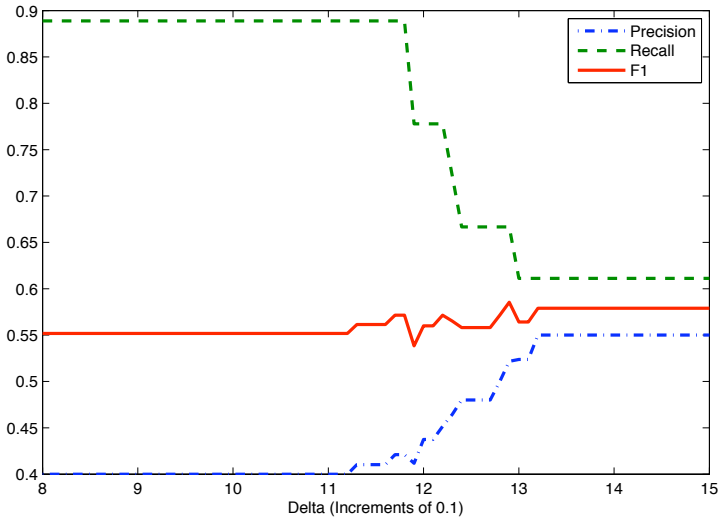
In all three tests, we see that as we increase δ, the precision of the algorithm increases. This is to be expected since the model selection criterion will stop the MGGCM algorithm before it normally terminates. Since the MGGCM algorithm already has high recall but low precision, we know that it is selecting the correct groups while also selecting extraneous, non-relevant groups after it has selected the correct groups. By stopping the selection step earlier, we expect that the algorithm will select fewer extraneous groups. Unfortunately, we also see a decrease in recall as we increase δ in all three graphs. Thus, even though the precision is increasing, the model selection criterion is not always stopping the algorithm correctly; sometimes the model selection criterion stops the selection process even if the algorithm has not yet selected the correct groups.

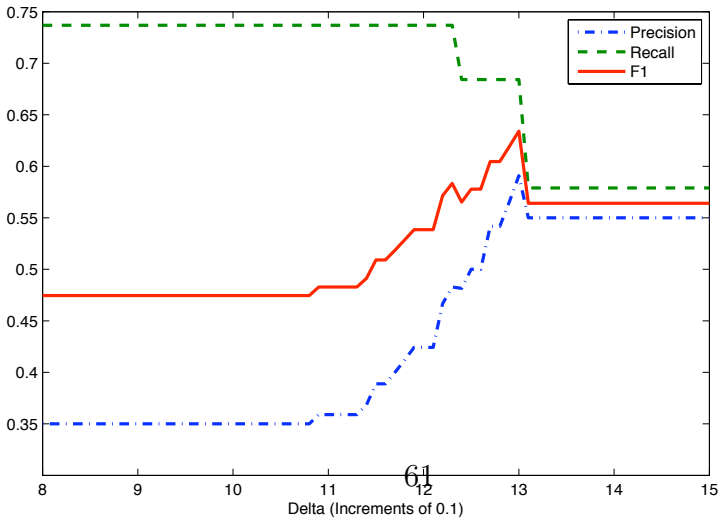
Ideally, we hope that the precision will continue to increase up to a point without any loss in recall. Figure 6.4(c) illustrates this desired behavior well: the precision can be raised over 10 points before the recall starts to suffer. Similarly, we see in 6.4(a) and 6.4(c) that the increase in precision that we can achieve outweighs the loss in recall since the $F1$ value steadily increases as we increase δ. Unfortunately, this behavior is not typical across all simulations. In Figure 6.4(b), we see that experimenting with δ has a marginal effect on the $F1$ value. 6.4(a) and 6.4(c) that the increase in precision that we can achieve outweighs the loss in recall since the $F1$ value steadily increases as we increase δ. Unfortunately, this behavior is not typical across all simulations. In Figure 6.4(b), we see that experimenting with δ has a marginal effect on the $F1$ value.

Regardless of the inconsistent behavior in our experiments varying δ, adding the model

Figure 6.3: MGOMP performance on three simulated data sets for varying δ ($G = 20$, $K = 10$, $d = 5$, $\tau = 100$)







selection criterion to the algorithm does improve its overall performance as measured by $F1$ value. The following table compares the results of using the model selection criterion to the other algorithms discussed across 20 simulations. For these tests δ was held constant at 12.8.

Method	Precision	Recall	$F1$
MGGCM	.4088	.8957	.5609
Oracle Estimate	.9656	.8957	.9292
Threshold	.9227	.8181	.8605
Model Selection	.6013	.7044	.6473

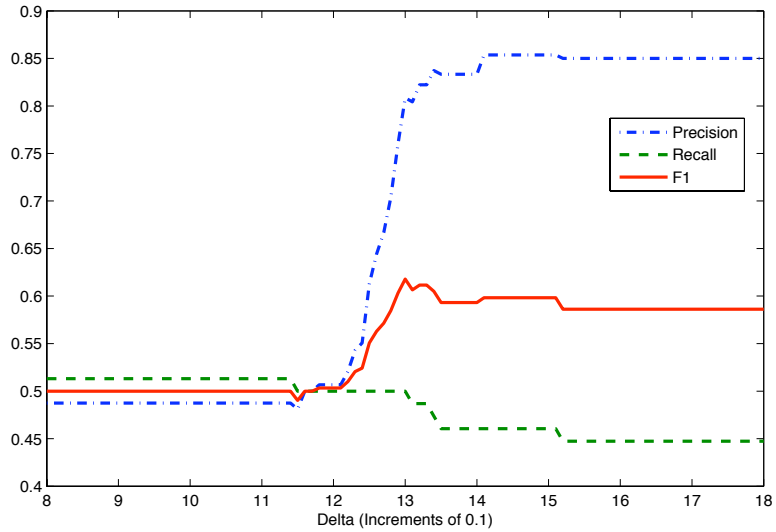
Even though adding the model selection criterion is not as effective as adding an oracle estimate or a threshold criterion, model selection still improves the MGGCM algorithm significantly on these simulations and appears to have the most applicability to more general cases, specifically real-world data, since it does not depend on the niceness of the data or any knowledge of an underlying structure.

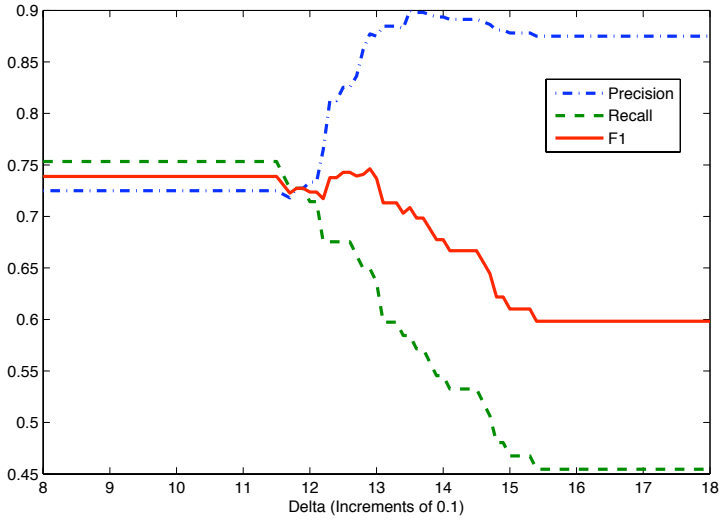
Applying the model selection criterion to simulations with other parameters, however, leads to more mixed results. We repeated our above experiments varying δ with the same parameters, except that we increased the number of bloggers G to 40. 6.4 shows the results.

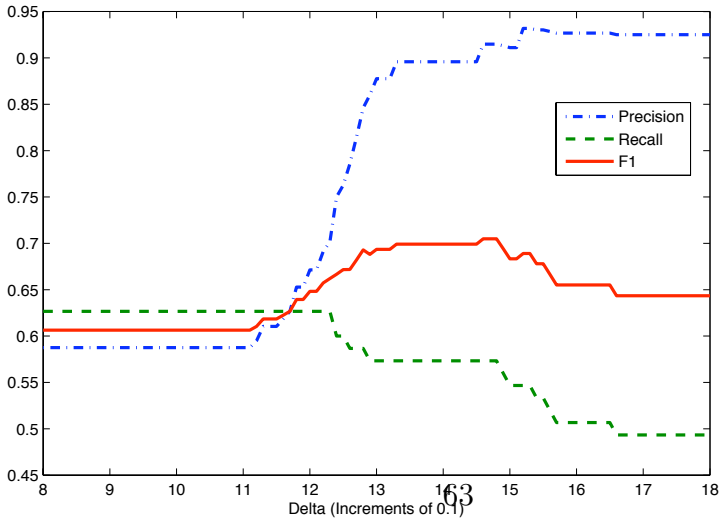
While Figure 6.5(a) displays behavior very similar to the simulations with $G = 20$, Figure 6.5(b) casts doubt on the effectiveness of the model selection criterion. This figure reveals that in some cases, the implementation of the model selection criterion can actually be detrimental to the performance of the algorithm since the $F1$ value decreases as we increase δ (remember that MGGCM with model selection performs identically to MGGCM without model selection when δ is small). On the other hand, we note that the $F1$ value in Figure 6.5(b) starts off high, and so we would expect it to be hard to make improvements. In Figures 6.5(a) and 6.5(c), we see that the $F1$ value starts low, and we can improve it with the appropriate choice of δ for the model selection criterion.

The model selection criterion shows promise in improving the stopping criterion during the selection process, but further research should be done. More tests are needed to verify the applicability of the model selection criterion to other data sets, and more investigation is needed to determine how best to optimize the choice of δ.

Figure 6.4: MGOMP performance on three simulated data sets for varying δ ($G = 40$, $K = 10$, $d = 5$, $\tau = 100$)







Chapter 7

Applying MGGCM to Twitter

7.1 Difficulties Working with Twitter Data

After testing our algorithm on simulation data, we want to see how MGGCM performs on real-world Twitter data. The Twitter data we used consists of tweets of 1000 bloggers spanning 2 months. The bloggers selected were those users on Twitter who mentioned the word IBM the most frequently from May 22, 2010 to July 22, 2010. This set of tweets contains more than 7500 unique words when parsed; if we were to attempt to apply MGGCM directly to this data set, the code would use over 1 terabyte of memory since each tweet would be represented as a vector over the entire dictionary of words. Thus, we need a way of compressing the information included in each document or tweet to make the application of MGGCM to Twitter data feasible.

7.1.1 Integration of Topic Modeling

Rather than attempt to use the full dictionary of words, we use the techniques of the topic modeling project to enable us to apply MGGCM to our Twitter data. The NMF algorithm factors our Twitter data X into matrices W and H, where W is the document-to-topic association matrix. When we use the rows of W instead of the corresponding rows of X in our algorithm to represent the content of our bloggers, we effectively compress the information from X and make the problem of determining influence between bloggers computationally feasible. For example, if we use non-negative matrix factorization to divide our dictionary of words into 50 topics, we represent each tweet as a vector over a set of 50 topics instead of representing each tweet as a vector over a dictionary with over 7500 words. Thus, we will use only 0.67% of the memory we would have needed if we had opted to process the data in the raw word-frequency format.

With this modification, we use topic-frequency chunks instead of word-frequency chunks as input for the MGGCM algorithm. We thus refer to this modification of the input to the algorithm as *Topical MGGCM* (TMGGCM). Because it is infeasible to apply MGGCM to Twitter directly, from here on we use TMGGCM. While we expect information to be lost with this transformation, for example we already know that the factorization of X into W and H is only an approximation, we allow ourselves to use TMGGCM as a tool to determine causal relationships when it would not be applicable otherwise.

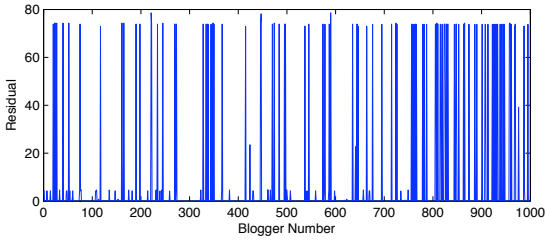


Figure 7.1: Residual Plot of Twitter Data 1

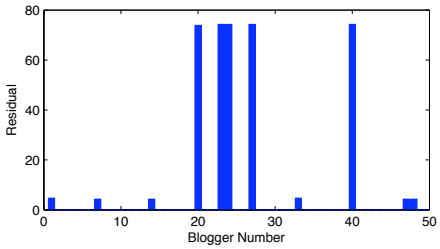


Figure 7.2: Residual Plot on Twitter Data 1 Zoomed on First 50 Bloggers

7.1.2 Sensitivity to Parameters

Before we can directly apply TMGGCM to Twitter data, there are numerous other parameters to consider that did not arise when looking at simulation data. For example, TMGGCM requires that we divide our data into time steps and requires us to choose the lag which indicates how far back we want to look for influence. Figures 7.1 and 7.2 are example residual plots from a naive parametrization of our Twitter data.

Figures 6.1 and 6.2 show very clearly how it is relatively easy to determine which groups are causally influential using our simulated data; on the other hand, Figures 7.1 and 7.2 indicate that there are numerous bloggers that completely explain the residual. This is an undesirable result and could be caused by a bad or naive parametrization of the discretization of time τ, the choice of lag d, or the number of topics K to use.

For Figures 7.3 and 7.4, we chose $\tau = 110$, $d = 5$, and $K = 30$. These choices for parameters provide a much more reasonable residual plot than our first tests; rather than having many bloggers apparently explain the entire residual, we see a more uniformly distributed residual plot. Comparing Figures 7.2 and 7.4, we can see how sensitive our algorithm is to the choices for parameters, and the comparison indicates that the parametrization is a crucial problem in learning how to apply TMGGCM to Twitter data.

7.1.3 Benchmarking

One problem with working on Twitter data is that we have no clear way to determine if we have correctly solved the problem of determining who is influential. While we could look

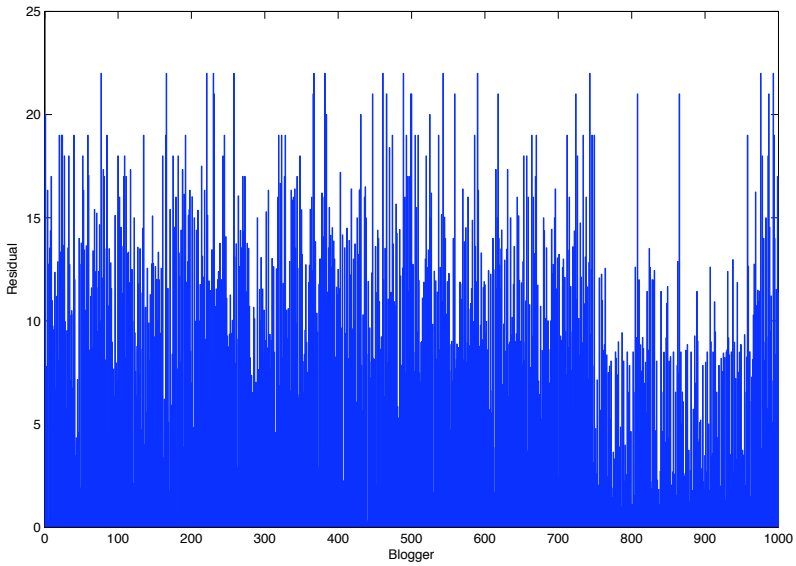


Figure 7.3: Residual Plot of Twitter Data 2

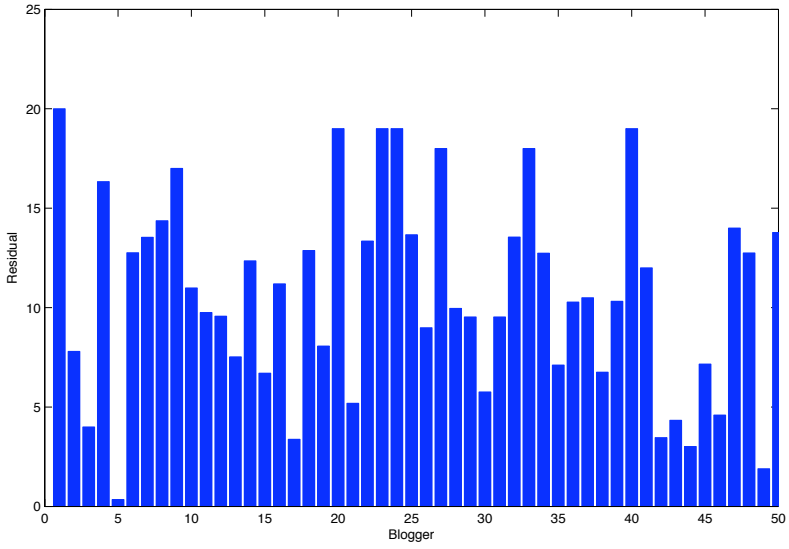


Figure 7.4: Residual Plot of Twitter Data 2 Zoomed on First 50 Bloggers

at measures such as the number of followers a user has or the number of times he or she is retweeted, neither of these measures captures exactly what we hope our algorithm measures since we are looking for more subtle forms of influence that are never explicitly stated. On the other hand, we would assume that if a given tweet is retweeted, then it is being read and digested and the blogger is exerting an influence. Thus, to test our algorithm's performance on Twitter data, we often look at the correlation between the Granger rank of a blogger as determined by the output causal graph of our algorithm and the blogger's retweet rate.

To calculate the correlation between the Granger rank of a blogger and the the blogger's retweet rate, there are several possibilities to consider. First, we must decide if we want to restrict the retweets to be those retweets that occur between two bloggers within our set, or if one of the bloggers can be from outside of our set. We will call the restriction of retweets to those that occur between two bloggers in our set *restricted retweet rate*; otherwise, we will be using *unrestricted retweet rate*, which takes into account all retweets. A third possibility is to binarize the retweet information we gain from the *restricted retweet rate*. This counts the number of people who have retweeted a given blogger. We will call this *binarized retweet rate*.

We also use three different correlational measures: Spearman, Pearson, and Kendall. For more information about each measure, please refer to [9] and [12] respectively.

7.1.4 Removing Zero-Content Entries

Another problem with working on real Twitter data is that not all Twitter users tweet at the same rate. Some are quite prolific and so they tweet more often, while others may tweet much less frequently. This makes it hard to decide how best to represent a user's content during a time step in which he or she does not tweet. There are several options to consider:

1. We could do nothing. Even if a user has no content during a time step, we will leave it be and represent it as all zeros. However, this could lead our algorithm to select other bloggers that also do nothing as predicting what the user will do. This seems like an unwanted consequence of our definition of causal influence, and so we want to avoid this situation.

2. We could reduce the discretization of time. By decreasing the number of time steps that we use to divide our content, we increase the chance that any given user will have written a tweet in that time step since each time step will represent more real-world time. This could lead to a loss of information since multiple tweets could be collapsed in a single time step.

3. We could remove zero-content entries. Instead of trying to predict zero content, we could remove any entries that are all zeros. This would mean that we would only try to predict a user's content if there is non-zero content to predict. This eliminates the problem of using other bloggers to predict when a user does nothing.

From a cursory glance, the last option seems to be the most appealing in terms of desirable outcomes. To implement this option, when solving the multivariate regression $Y = XW$ that models our notion of influence, where Y is the content of the blogger we want to predict and X is the content of all the bloggers, we would remove any rows of Y that are all zeros. Because a row of Y is explained by a single corresponding row of X, we would also remove the corresponding row of X. Unfortunately, this means that we must also recalculate the kernels for each individual blogger when we run MGOMP to determine

the influential bloggers. In our original implementation of TMGGCM, we could reuse the kernels for all of the bloggers since they would not change. However, if we want to remove zero-content entries, then we must first find the zero-content entries for the blogger that we are trying to predict, and then modify the kernels to reflect this removal. Because we cannot reuse the kernels and must calculate them anew for each blogger, this process is computationally intensive and uses a lot of memory. From here on, we will refer to running TMGGCM with the zero content removed as *No Zero Topical MGGCM* (NZTMGGCM).

Due to the computational intensity of NZTMGGCM and other time constraints, we ran NZTMGGCM on a smaller subset of bloggers derived from our original Twitter data. We generated the smaller data set using the Twitter accounts from our original data set whose account names start with "ibm". There are a total of 23 such accounts, and there are 78 retweets amongst them. The table below shows a comparison of applying TMGGCM versus NZTMGGCM to this smaller data set. Applying the NZTMGGCM algorithm instead of the normal TMGGCM algorithm to our data leads to better results when comparing the Granger ranks of the corresponding output graphs to the *binarized retweet rate* of each blogger. Since these correlations are generally higher for the NZTMGGCM algorithm compared to the TMGGCM algorithm, in particular when $\tau = 50$, we are led to believe that NZTMGGCM gives us a better measure of influence than TMGGCM. All tests were run with $K = 50$ and $d = 5$.

	TMGGCM		NZTMGGCM	
	$\tau = 50$	$\tau = 110$	$\tau = 50$	$\tau = 110$
Spearman	0.243844	0.234563	0.466782	0.353553
Pearson	0.386650	0.602555	0.629276	0.594453
Kendall	0.182179	0.198020	0.393088	0.294884

7.2 Randomized Sub-Sampling of Projections

One drawback of MGOMP is the need to calculate the projection of every blogger onto the residual for each iteration of the selection process. If we have a network of 1000 bloggers, we must calculate 1000 projections for every iteration of MGOMP, and to run TMGGCM, we must run MGOMP on each of the 1000 bloggers. If it is possible to reduce the number of projections we must calculate, we should theoretically be able to make TMGGCM more computationally efficient and take less time.

If we assume that the projections have a uniform distribution, a sample of size $s = \lceil \frac{\log(\epsilon)}{\log(q)} \rceil$ has a maximum that is in the top $(1 - q)$ % of values with probability $1 - \epsilon$. For example, if we want to take a subset of the projections which has a maximum in the largest 5% of values with probability 95%, we need to sample only $s = \lceil \frac{\log(0.05)}{\log(0.95)} \rceil = 59$ elements and look at its maximum. Applying this idea to our algorithm, we could sample many fewer projections at each iteration and still be relatively confident that our algorithm will choose bloggers that predict a large amount of the content.

To test this idea, we applied both TMGGCM to a set of 1000 bloggers as well as a modified TMGGCM algorithm that utilizes random sub-sampling of the projections during MGOMP. We compared the results of the algorithm by looking at how well the TMGGCM with random sub-sampling algorithm reconstructs the causal graph that normal TMGGCM outputs and the correlation between the Granger ranks calculated from each causal graph.

Table 7.1: Granger Rank vs. Retweet Rate correlations

Type of Correlation	$\tau = 50$, $d = 5$	$\tau = 100$, $d = 5$
Spearman	0.259382	0.400011
Pearson	-0.006446	0.386752
Kendall	0.234703	0.375525

Parameters	Spearman Correlation	Pearson correlation	$F1$
$q = .95, \epsilon = .05$	.2827	.6502	.003766
$q = .97, \epsilon = .03$	.3431	.7364	.002722

As expected, the $F1$ value is very low for both runs of TMGGCM with random sub-sampling; we do not expect an algorithm with random sub-sampling to faithfully reconstruct the causal graph from TMGGCM because it cannot choose the same blogger for any given iteration if that blogger was not randomly included in the sample. On the other hand, we see that the Pearson correlations for the Granger ranks derived from the causal graphs are relatively high, implying that the TMGGCM algorithm with random sub-sampling does a fairly good job at reconstructing who is most influential from a Granger causality perspective. This suggests that randomized sub-sampling of projections might be very useful in this and other applications of MGOMP since we drastically reduce the number of projection computations calculated for each iteration.

For our situation, randomizing the sub-sampling is likely not going to help improve the efficiency of our code if we also want to remove zero-content entries. Using NZTMGGCM, the dominant computation time is spent removing zero-content entries and recalculating kernels rather than computing projections. Thus, the information lost is probably not worth the marginal gain in speed using NZTMGGCM, so this idea is only applicable to the normal TMGGCM algorithm.

7.3 Provisional Results

In this section, we include our provisional results of applying TMGGCM and NZTMGGCM to Twitter data. Due to time constraints, we did not run all of the tests and experiments that we would have liked and thus are hesitant to make claims about the results.

Applying NZTMGGCM to the small dataset and comparing the Granger ranks taken from the output of the NZTMGGCM algorithm with the *binarized retweet rate* for each blogger, we obtain the results shown in Figure 7.5.

Figure 7.5 illustrates how important the discretization of time is to obtaining a high correlation with retweet rate. When we choose the total number of time steps to be 110, we get an impressive correlation result of 0.52, but for some other choices of τ, the correlation is much weaker.

Table 7.1 gives a closer look at some points of the above graph, and Figure 7.6 shows analogous results for $d = 10$.

As mentioned previously, we have a full data set consisting of the content of 1000 bloggers. For those bloggers, we have two different sets of retweet data. One called *Retweet Dataset 1* consists of 5957 retweets, with 3392 occurring amongst our 1000 bloggers. The other called *Retweet Dataset 2* consists of 11584 retweets, of which 11133 include one of our 1000 bloggers as the retweeting one or the one being retweeted.

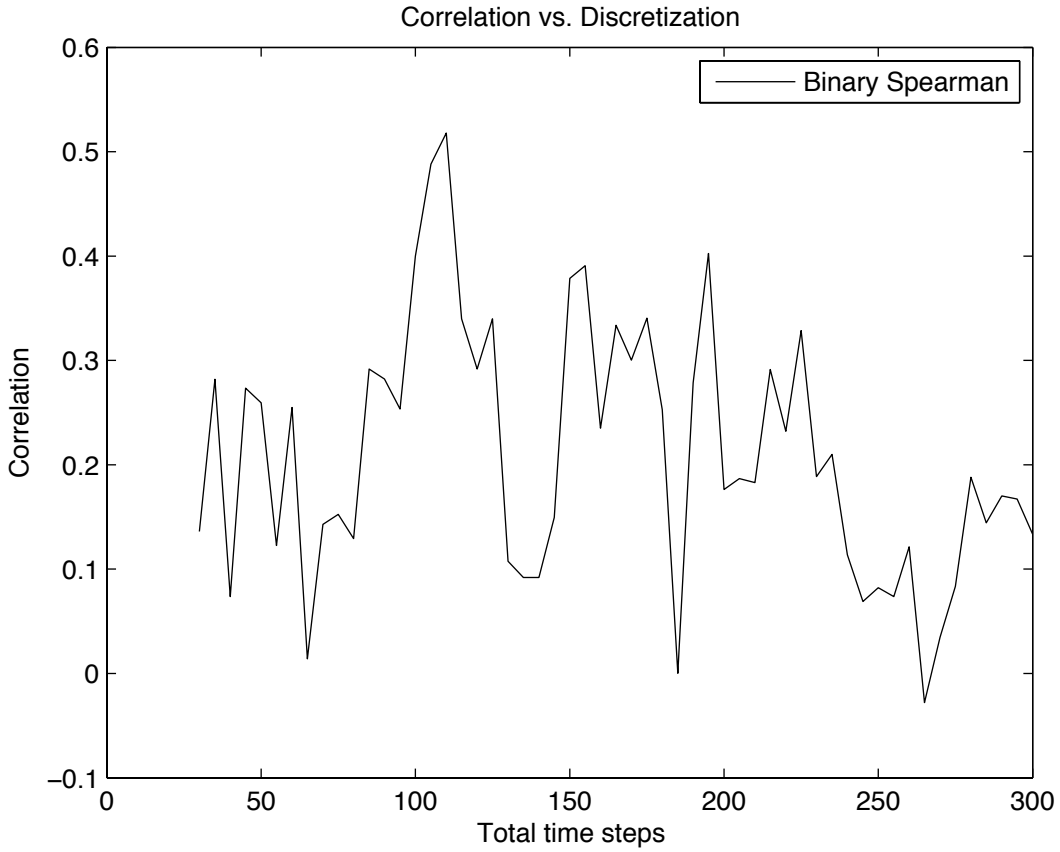


Figure 7.5: Spearman correlation between retweet rate and Granger Rank for various τ with $d = 5$

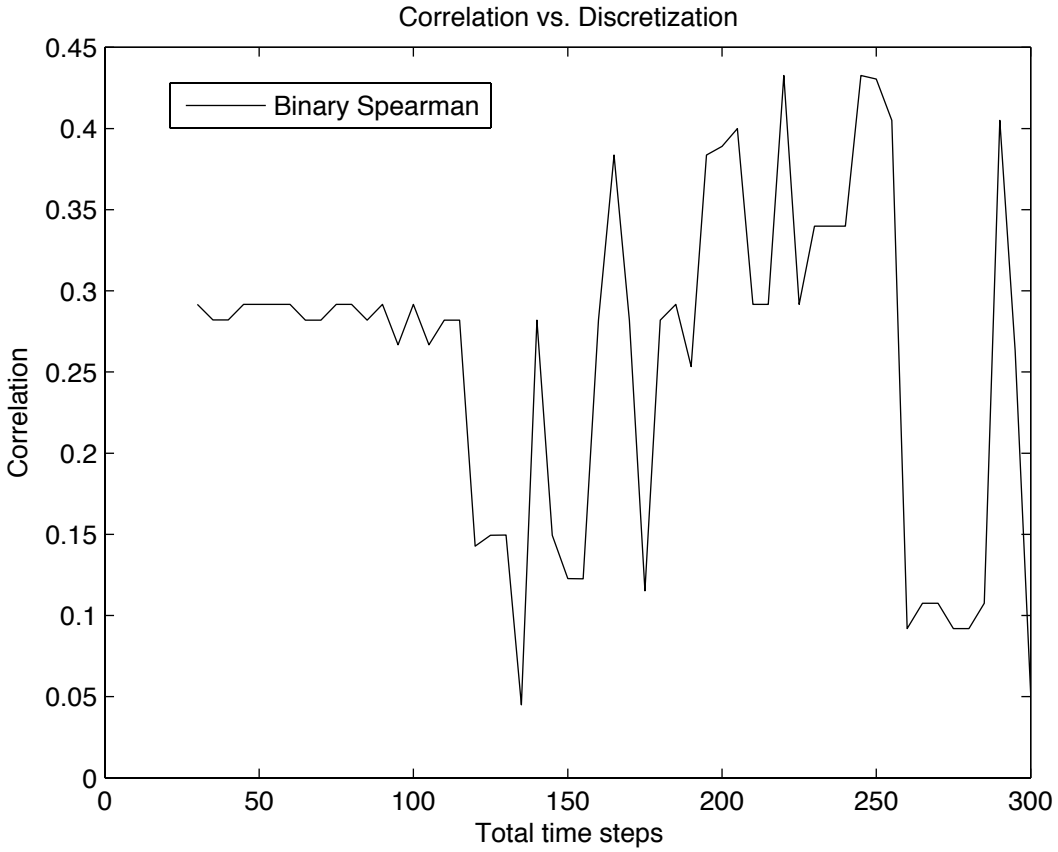


Figure 7.6: Spearman correlation between retweet rate and Granger Rank for various τ with $d = 10$

Running TMGGCM on 1000 bloggers with *Retweet Dataset 2* and using the number of topics $K = 75$, the number of time steps $\tau = 70$, and the lag $d = 5$, we get the following correlations with *unrestricted retweet rate*:

Spearman Correlation:	0.041110
Pearson Correlation:	-0.001355
Kendall Correlation:	0.036660

Below is a run of the same algorithm with the same parameters, but this time we use *Retweet Dataset 1* and *restricted retweet rate*. Pearson correlation results are especially promising in this case.

Non-Binarized Spearman Correlation:	0.129009
Non-Binarized Pearson Correlation:	0.395857
Non-Binarized Kendall Correlation:	0.119948

Below is the correlation between *unrestricted retweet rate* and *binarized retweet rate* for *Retweet Dataset 2*.

Non-Binarized Spearman Correlation:	0.431164
Non-Binarized Pearson Correlation:	0.285423
Non-Binarized Kendall Correlation:	0.327382

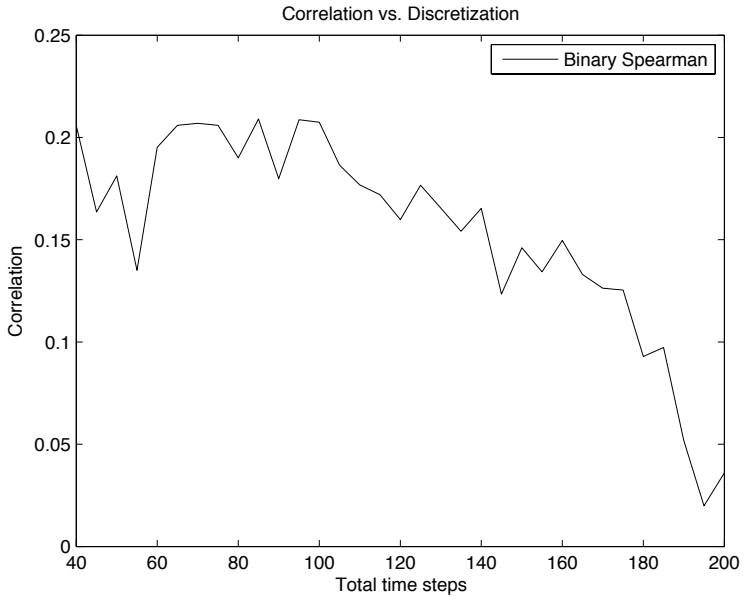
Finally, here is the same measurement using *Retweet Dataset 1*.

Non-Binarized Spearman Correlation:	0.120127
Non-Binarized Pearson Correlation:	0.239098
Non-Binarized Kendall Correlation:	0.093247

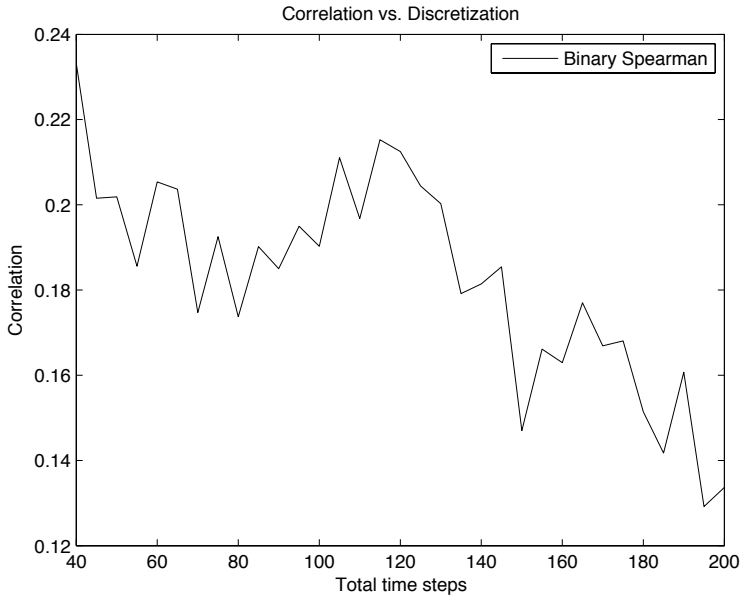
For Figures 7.3 and 7.3, we look at how changing the lag parameter d affects the graph over the discretization of time τ using *unrestricted retweet rate*.

Figure **??** shows the correlation results using *restricted retweet rate*, *Retweet Dataset 1*, and NZTMGGCM.

Figure 7.7: Correlation between Granger Rank and *unrestricted retweet rate* for 1000 Bloggers, 50 Topics and *Retweet Dataset 1* for 2 choices of lag

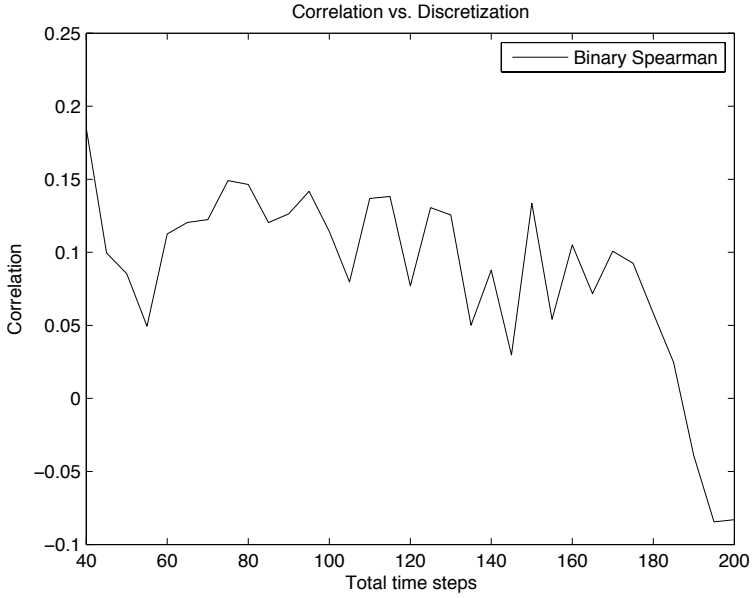


(a) $d = 5$

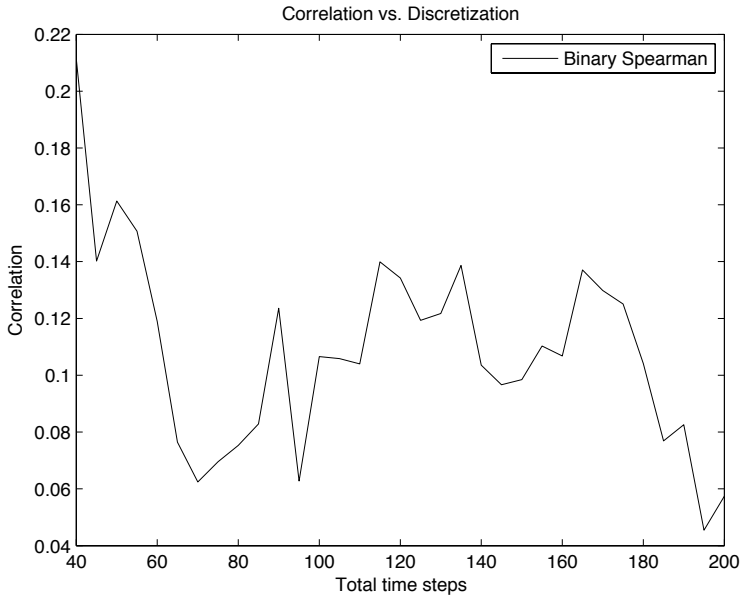


(b) $d = 10$

Figure 7.8: Correlation between Granger Rank and *unrestricted retweet rate* for 1000 Bloggers, 50 Topics and *Retweet Dataset 2* for 2 choices of lag

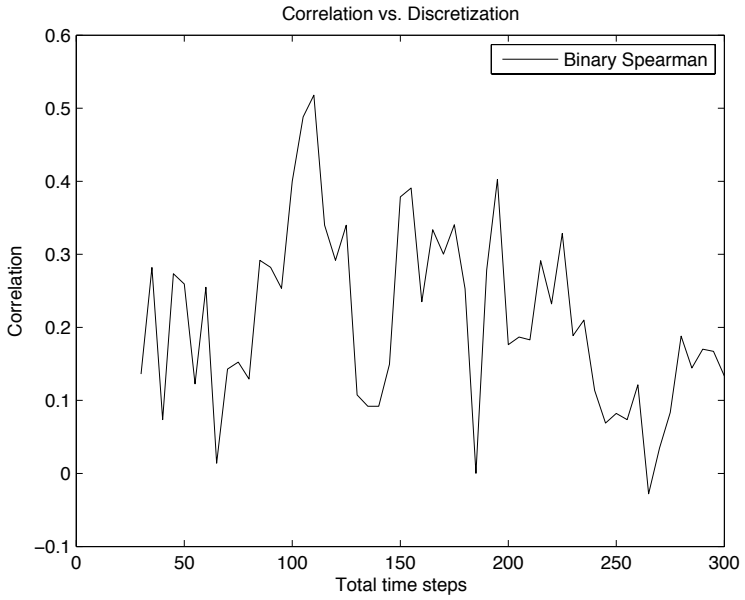


(a) $d = 5$

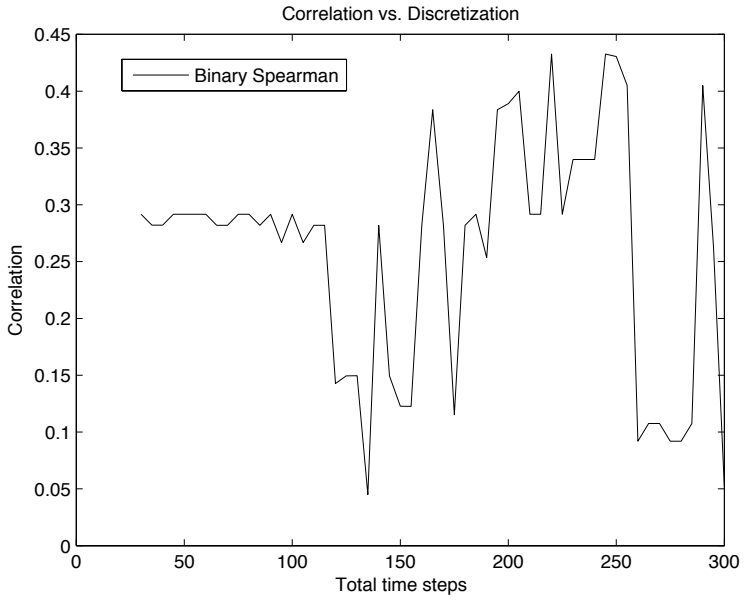


(b) $d = 10$

Figure 7.9: Correlation between Granger Rank and *restricted retweet rate* for 1000 Bloggers, 50 Topics and *Retweet Dataset 1* for two different lags, using NZTMGGCM



(a) $d = 5$



(b) $d = 10$

Chapter 8

Conclusion

8.1 Overview of Work Accomplished

We have produced an algorithm for sparse NMF which performed better than Hoyer's algorithm in most of our tests, and found algorithms for performing automatic selection of parameters. This NMF algorithm was then used to make the application of MGGCM to Twitter computationally feasible. Doing so, we were able to attain an impressive 0.5 correlation between Granger Rank and retweet rate for one set of parameters.

8.2 Future Avenues of Research

8.2.1 NMF

- The simplex-projection method is fast but not always accurate, especially for large values of z. A more accurate method of solving $\arg\min_{||h_t||_1=z} ||X - WH||_F^2$ would allow us to try a much wider range of z during model selection and could lead to faster convergence.

- We have never determined how well RRI performs in an absolute sense (i.e. in terms of finding the true global minimum. Let $f_X : \mathbb{R}^2 \to \mathbb{R}$ be defined such that $f_X(a, b) = \inf_{||W||_0=a,\, ||H||_0=b} ||X - WH||_F^2$. How well does the surface parametrized by our choice of λ and z approximate this function?

- Our model selection techniques rely heavily on unproven facts about the shape of the reconstruction error vs. $||W||_0$ vs. $||H||_0$ surface. Facts about the shape of the graph of the function f might be applicable to our model-selection techniques and might be of independent interest.

- Our initialization procedure requires a lot of memory - in fact the maximum memory usage of the algorithm occurs during the first iteration and depends primarily upon the density of the initialization of W. Can the initialization procedure be found which uses less memory without losing accuracy?

8.2.2 Causal Influence Modeling

- MGGCM has proven very effective on simulation data when it has an oracle estimate, but more research should be done on alternative methods to stop the selection process

in MGOMP.

- MGGCM is sensitive to numerous parameters such as the discretization of time τ, the lag d, the number of topics K, etc. Further studies are needed to determine how best to choose these parameters for a given set of data.
- MGGCM seems to perform better when we remove zero-content entries; however, the algorithm runs much slower and consumes more memory when we do this. More exploration on the effects of removing zero-content entries and how best to implement such an algorithm in a fast and efficient way is needed.

Appendix A

Matlab Code

In this appendix we present the main algorithms developed as implemented in Matlab code. This Matlab code as well as our diagnostic code will also be submitted electronically. The contents are as follows:

rri.m Performs Rank-1 Residue Iterations with ℓ_1 regularization on W and ℓ_1 constraint on H to factor $X \approx WH$

spinitialize_ nmf.m Initializes W and H for RRI algorithm

rri_ multiple_ iterations.m Performs RRI with a number of different iterations, chooses the best one

directionalCornerFinder.m Finds a corner on the curve in the $(||X - WH||_F^2, ||W||_0 + ||H||_0)$ plane parametrized by a line segment in the (λ, z) plane

optLambda.m Finds an optimal λ for fixed z using 1-dimensional corner-finding

tripletCornerFinder.m Finds an optimal (λ, z) using 2-dimensional corner-finding

optZ.m Finds an optimal z for fixed λ using 1-dimensional corner-finding

projfunc.m Projects vectors onto a zimplex of constant ℓ_1-norm in $\mathbb{R}^d$

optLambdaZAlternating.m Alternates optLambda and optZ to find optimal (λ, z)

measureReconstruction.m Calculates sparsities and reconstruction error for a factorization $X \approx WH$

ThresholdMGOMP.m Performs MGOMP with a threshold stopping criterion

MGGCMBlockSel.m Performs MGGCM using block-processing and a model selection criterion

MGGCMBlockNoZero Performs MGGCM with zero-removal

follower_retweet_corr_real.m Computes the unrestricted correlation results between the number of followers and retweets the bloggers have.

follower_retweet_corr.m Computes the correlation between the number of followers and retweeters/retweets the bloggers have. Both results are "restricted correlation results" i.e. retweets are restricted to be between the bloggers of our interest only.

get_B.m Discretizes the tweet data

rankComparator.m Computes the "restricted retweet correlation"

rankComparatorReal.m Computes "unrestricted retweet correlation"

tau_d_optimizer.m Runs the TMGGCM algorithm and saves the output causal graphs which will be used to calculate correlation results using different methods.

twitter_tester.m Performs the TMGGCM algorithm.

Selected Bibliography Including Cited Works

[1] Machine Learning Group at UCD. Bbc datasets, aug 2010. http://mlg.ucd.ie/datasets/bbc.html.

[2] Grzegorz Świrszcz Aurélie C. Lozano and Naoki Abe. Group orthogonal matching pursuit for variable selection and prediction. *Advances in Neural Information Processing Systems (NIPS)*, 22, 2009.

[3] Emmanuel Candes and Terrence Tao. Decoding by linear programming. *IEEE Trans. Info. Theory*, 51:4203–4215, 2004.

[4] The Linguistic Data Consortium. Description of the tdt2 data sources, aug 2010. http://projects.ldc.upenn.edu/TDT2/tdt2.corpusdoc.html.

[5] C.W.J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.

[6] Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press, 2006.

[7] Per Christian Hansen. Analysis of discrete ill-posed problems by means of the l-curve. *SIAM Review*, 34(4):561–580, dec 1992.

[8] Per Christian Hansen and Dianne Prost O'Leary. The use of the l-curve in the regularization of discrete ill-posed problems. *SIAM Journal of Scientific Computation*, 14(6):1487–1503, nov 1993.

[9] Tibshirani Hastie and Friedman. *Elements of Statistical Learning*. Springer-Verlag, 2008.

[10] Ngoc-Diep Ho. *Nonnegative Matrix Factorizations Algorithms and Applications*. PhD thesis, Académie universitaire Louvain, Boston, MA, USA, 2008.

[11] Patrik Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[12] M. Kendall. A new measure of rank correlation. *biometrica*, 30:81–89, 1938.

[13] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 1999.

[14] Aurélie Lozano and Vikas Sindhwani. Inferring key influencers in online communities using multivariate grouped granger causality. Submitted to the international Conference on Machine Learning, 2010.

[15] Jason Rennie. Home page for 20 newsgroups data set, aug 2010. http://people.csail.mit.edu/jrennie/20Newsgroups/.

[16] D. D. Lewis; Y. Yang; T. Rose and F. Li. F. rcv1: A new benchmark collection for text categorization research. *Jounal of Machine Learning Research*, 5:361–397, 2004. http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf.

[17] C. Spearman. The proof and measurement of association between two things. *Ameican. Journal of Psychology*, 15:72101, 1904.

[18] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Info. Theory*, 53(12):4655–4666, Dec 2007.

[19] Eric Zivot. Vector autoregressive models for multivariate time series (lecture notes). http://faculty.washington.edu/ezivot/econ584/notes/varModels.pdf, apr 2005.