PREDICTIVE MODEL BUILDING FOR UTILIZING WORD EMBEDDING MODELS: APPLICATIONS IN INSURANCE DATA

By

Scott Manski

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Statistics – Doctor of Philosophy

2020

ABSTRACT

PREDICTIVE MODEL BUILDING FOR UTILIZING WORD EMBEDDING MODELS: APPLICATIONS IN INSURANCE DATA

By

Scott Manski

Textual data contains a vast amount of information, yet for many researchers it has not been clear how the information could be used for an empirical analysis. Often times textual data are ignored or discarded in statistical analyses because regression and other statistical methods require numeric covariates. This dissertation will demonstrate how cutting-edge text mining technologies can improve empirical analyses by transforming textual data into numeric explanatory variables, thus allowing textual data to be incorporated into a statistical analysis. By transforming the textual data, the number of explanatory variables often becomes larger than the number of observations. For this reason, we explore the application of generalized additive models in tandem with adaptive lasso. In addition, we construct an algorithm for fitting a Gamma double generalized linear model with a group lasso penalty. Through this, we show how useful information can be extracted from textual data. We show how our methods can be applied through several insurance claims examples. We believe that our work can be widely used for other observational researchers in economics, business, statistical science, and social science.

ACKNOWLEDGEMENTS

I would like to express the utmost gratitude to Dr. Taps Maiti and Dr. Gee Lee, for their assistance, support, guidance, and encouragement throughout my time at MSU. I would also like to thank my entire graduate committee, including Dr. Hyokyoung Hong and Dr. Chenhui Guo. Their service is greatly appreciated. Furthermore, I would like to thank the Department of Statistics and Probability for the resources and opportunities provided to me during my Ph.D. career.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

KEY TO ABBREVIATIONS

CBOW Continuous bag-of-words

CNN Convolutional Neural Networks

GAM Generalized Additive Model

GLM Generalized Linear Model

GloVe Global Vectors for Word Representation

IBNR Incurred but not Reported

LGPIF Local Government Property Insurance Fund

NOAA National Oceanic and Atmospheric Administration

RNN Recurrent Neural Networks

CHAPTER 1

INTRODUCTION

1.1 Using Text as Data

With the advance in technology, data are being collected on a much larger scale. While traditional data types such as numerical and categorical data are still being collected, there has been an increase in the collection of digital text data. In many cases, text data can provide more information than a series of traditional variables, but extracting this vital information is not trivial.

Text mining has been utilized in a variety of contexts, including but not limited to spam filtering, sentiment analysis, customer churn, stock returns, and politics. Sentiment analysis is the process of categorizing texts based on the message of the text, usually positive or negative. Pang et al. (2002) explored the use of Naive Bayes, maximum entropy classification, and support vector machines to predict the overall opinion, positive or negative, of a movie review. Similar work includes using discriminant analysis to categorize texts into pre-determined genres by Karlgren & Cutting (1994), and finding features indicated by the use of subjective language by Hatzivassiloglou & Wiebe (2000). Text mining has also been used substantially in the prediction of stock prices. Antweiler & Frank (2004) analyzed internet stock message boards and found that the messages help to predict market volatility. Moreover, Tetlock (2007) found that high media pessimism predicts downward pressure on the market, and unusually high or low pessimism predicts high trading volume.

Each of the aforementioned examples are primarily concerned with prediction as opposed to inference of relationships between the response and the explanatory variables. Some authors have found text analysis to be helpful in telling a story using the relationship between the response and the explanatory variables. Stephens-Davidowitz (2014) explored racially charged language in Google search queries to explain how much racial animus costs a black presidential candidate. Gentzkow et al. (2016) studied congressional speeches and found that partisanship is far greater in recent years, with a sharp increase in the early 1990s.

In almost all situations where text data are utilized, the analysis can be summarized in the following way:

1. Represent the raw text $\mathcal{D}$ as some numeric vector or array $\boldsymbol{C}$
2. Map $\boldsymbol{C}$ to predicted values $\hat{\boldsymbol{Y}}$ of unknown response $\boldsymbol{Y}$
3. Use $\hat{\boldsymbol{Y}}$ in subsequent analysis

In the following sections, we will discuss a variety of methods used to construct $\boldsymbol{C}$. We start by exploring the use of indicator variables and count data to represent text in Section 1.1.1. In Section 1.1.2, we introduce word embeddings along with several methods for obtaining word embedding matrices. Finally, in Section 1.1.3 we discuss how we will be using cosine similarities to measure the similarity between words.

1.1.1 Word Occurrence Methods

The most common and simplest way of constructing $\boldsymbol{C}$ is by using indicator variables for the occurrence of words. We will define $\mathcal{D}$ as the collection of documents typically in the form of a column vector. Furthermore, we will denote the set of words in the ith document as $\boldsymbol{D}_i$, for $i = 1, ..., |\mathcal{D}|$.. Finally, let w be the set of all unique words found in $\mathcal{D}$. Then we can define the entries of $\boldsymbol{C}_{ind}$ as

$$\boldsymbol{C}_{ij} = \begin{cases} 1 & \text{if } w_j \in \boldsymbol{D}_i \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i = 1, ..., |\mathcal{D}|, \quad j = 1, .., |w|.$$

Consider the following example where you have textual descriptions for 3 observations,

$$\mathcal{D} = \begin{bmatrix} \text{Lightning struck a building} \\ \text{A lightning strike hit a building} \\ \text{Lightning struck a tree} \end{bmatrix}$$

The response in our example could be the amount of damage, in dollars, caused by the event. By our definition, $\boldsymbol{D}_1 = \{\text{lighning, struck, a, building}\}$, for example. Furthermore,

$$w = \{\text{lighning, struck, a, building, strike, hit, tree}\}.$$

Then

$$\boldsymbol{C}_{ind} = \begin{array}{c} \begin{array}{ccccccc} lightning & struck & a & building & strike & hit & tree \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

An extension of the indicator method is to count the number of occurrences of word w_j in each textual description. Using the count method, since the word a is the only word that occurs more than once in a description, we have

$$\boldsymbol{C}_{count} = \begin{array}{c} \begin{array}{ccccccc} lightning & struck & a & building & strike & hit & tree \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

If we conduct a lexical analysis of the first and second textual descriptions, we see

$$\underset{noun}{\text{Lightning}} \quad \underset{verb}{\text{struck}} \quad \text{a} \quad \underset{noun}{\text{building}}$$

$$\text{A} \quad \underset{adj.}{\text{lightning}} \quad \underset{noun}{\text{strike}} \quad \underset{verb}{\text{hit}} \quad \text{a} \quad \underset{noun}{\text{building.}}$$

While the sentence structure of the first and second textual descriptions are different, it is clear that their meanings are essentially equivalent. Since the sentence structure is different, $\boldsymbol{C}_{ind}$ and $\boldsymbol{C}_{count}$ are different and they do not appropriately capture this similarity in meaning. Additionally, the only difference between $\boldsymbol{C}_{ind}$ and $\boldsymbol{C}_{count}$ in the example is the third column, the column associated with the word a. While the use of the word a is necessary in creating grammatically sound sentences in our example, it does not add any additional information in regards to our response. Common words

that typically add little information to the meaning of a sentence are called stop words. Examples of stop words include *a*, *the*, *and*, *at*, and *for*. In practice, stop words are typically removed from w and $\boldsymbol{D}_i$ $(i = 1, ..., n)$ to reduce the number of columns in $\boldsymbol{C}$ and to reduce the number of computations when constructing $\boldsymbol{C}$.

1.1.2 Word Embeddings

So far, we have only represented text through indicator variables or counts. While this is easy to understand and easy to work with, it is a very basic way of summarizing text. Instead, word embedding representations map words into some vector space. The vector space consists of a word embedding for each word in the vocabulary. The word embeddings are determined by optimizing some objective function, such as a likelihood for word occurrences, on a library of texts. There are several algorithms for determining the word embeddings. We will be introducing two of these algorithms, word2vec and Global Vectors for Word Representation (GloVe).

1.1.2.1 *Word2vec*

One approach to obtaining the word embeddings is to use the word2vec algorithm. The word2vec algorithm can be illustrated using a simplified example. Suppose we use the following seven words as our simplified vocabulary list:

$$V = \{\text{lighning, struck, a, building, strike, hit, tree}\} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_7\}$$

Consider the sentence:

lightning struck a building

with center word $\boldsymbol{w}_2$, and context words $C = \{\tilde{\boldsymbol{w}}_1, \tilde{\boldsymbol{w}}_3\}$. What we want is some vector representation $\boldsymbol{W}$ of center words and $\tilde{\boldsymbol{W}}$ of context words. These word embedding matrices are obtained by letting an algorithm read through billions of sentences, maximizing a log likelihood, treating $\boldsymbol{W}$ and $\tilde{\boldsymbol{W}}$ as parameters to be estimated. For this, we can specify the probability of observing a context word

$\tilde{\boldsymbol{w}}_j$ given a center word $\boldsymbol{w}_2$ by

$$Pr\left(\tilde{\boldsymbol{w}}_j|\boldsymbol{w}_2\right) = \frac{\exp\left(\tilde{\boldsymbol{w}}_j \cdot \boldsymbol{w}_2\right)}{\sum_{k=1}^{|V|} \exp\left(\tilde{\boldsymbol{w}}_k \cdot \boldsymbol{w}_2\right)}$$

Going back to the sentence, *lightning struck a building*, using a naive Bayes assumption (where we assume the conditional independence of the events of observing context words given a center word), the negative log likelihood becomes:

$$L = -\log Pr\left(\tilde{\boldsymbol{w}}_1|\boldsymbol{w}_2\right) - \log Pr\left(\tilde{\boldsymbol{w}}_3|\boldsymbol{w}_2\right) = -\sum_{\tilde{\boldsymbol{w}}_j \in C} \tilde{\boldsymbol{w}}_j \cdot \boldsymbol{w}_2 + |C| \log \sum_{\tilde{\boldsymbol{w}}_k \in V} \exp\left(\tilde{\boldsymbol{w}}_k \cdot \boldsymbol{w}_2\right)$$

The negative log likelihood is minimized by gradient descent. Note that analytical formulas for the gradient can be obtained based on the likelihood. In practice using the analytical forms of the gradient speeds up the convergence, and makes the algorithm more stable. Extensions of the gradient descent algorithm such as Adam optimizers may be used as well. The $\boldsymbol{W}$ and $\tilde{\boldsymbol{W}}$ matrices start from a random initial matrix, and is updated each step of the gradient descent iteration. Repeating this process for millions and billions of center words and context words, an input word matrix $\boldsymbol{W}$ and a context word matrix $\tilde{\boldsymbol{W}}$ is learned.

Word2vec may use either the continuous bag-of-words (CBOW) model or the continuous skip-gram model, depending on how the log likelihood is defined. In the CBOW model, the log likelihood represents the probability of observing the center word within a window of context words. In the continuous skip-gram model, the probability of observing the context word given a center word is used. Hence, the example shown above would correspond to a continuous skip-gram model.

Nowadays, word embedding matrices trained by word2vec can be downloaded from Google's website (`https://code.google.com/archive/p/word2vec/`). An extension of word2vec, which uses character n-grams, is fastText, which can be downloaded from (`https://fasttext.cc`). FastText is a library for learning word embeddings, created by Facebook's AI Research (FAIR) lab.

1.1.2.2 *Global Vectors for Word Representation (GloVe)*

In this paper, we use the pre-trained word embeddings obtained via an algorithm called GloVe, developed by Pennington et al. (2014). We note that other methods to create word embedding matrices exist. From our perspective, the end result of word2vec and GloVe is similar. We chose GloVe because it is a straightforward algorithm based on word counts, and the approach is well documented with an emphasis on reproducibility. In practice, the GloVe algorithm has additional benefits over word2vec in that the algorithm is more easily parallelized. GloVe word embedding matrices can be downloaded from `https://nlp.stanford.edu/projects/glove/`. From this website, the 300 dimension word vectors containing 400 thousand vocabularies, trained over 6 billion tokens appearing in the Wikipedia corpus, has been downloaded. The algorithm used for this particular word embedding is to minimize a cost function, which has the form

$$J = \sum_{s=1}^{|V|} \sum_{t=1}^{|V|} \Psi\left(M_{s,t}\right) \left(\boldsymbol{w}_s \cdot \tilde{\boldsymbol{w}}_t + b_s + \tilde{b}_t - \log(M_{s,t}+1)\right)^2$$

where $|V|$ is the size of the vocabulary, b_s, $\tilde{b}_t$ are bias terms, $M_{s,t}$ are entries of the co-occurrence matrix for all the words found in the corpus over which the algorithm is being applied, $\boldsymbol{w}_s$ and $\tilde{\boldsymbol{w}}_t$ are the word embeddings corresponding to the position in the co-occurrence matrix $M_{s,t}$, and Ψ is a weighting function:

$$\Psi(x) = \begin{cases} (x/x_{max})^{\xi} & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases}$$

with $x_{max} = 100$ and $\xi = 3/4$. The motivation for $\xi = 3/4$ is empirical, and with this choice of the parameter, the performance of the model happens to improve when compared to $\xi = 1$. In practice, we may have any $0 < \xi < 1$. The reader may understand this is a way to give more weight to rare word combinations found in the corpus. Intuitively, the co-occurrence matrix $M_{s,t}$ records how often two words occur together in the training corpus.

The algorithm attempts to find a word embedding for each word that gives a dot product that is close to the transformed co-occurrence matrix entry of the word combination. The resulting word embedding gives a large dot product for those combinations of words that have a high co-occurrence

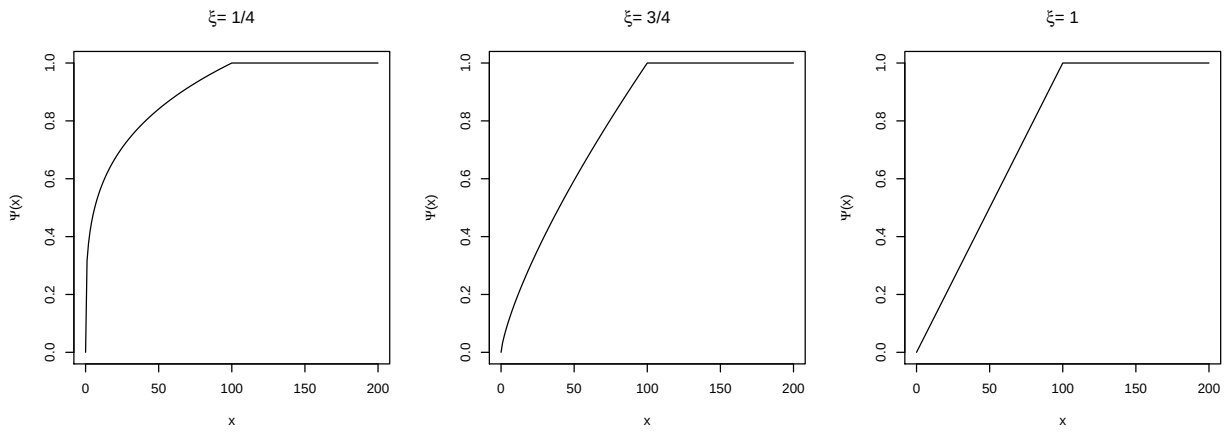

Figure 1.1: Plot of $\Psi(x)$ for different values of ξ, with $x_{max} = 100$

matrix entry, and a low dot product for those combinations of words that correspond to zero matrix entries. Note that most of the entries of the matrix would be zeros. For additional details on the GloVe algorithm, we refer to the original paper by Pennington et al. (2014).

Word embedding matrices obtained this way have numerous applications. In the natural language processing literature, word embeddings are used to construct neural networks that can predict missing words in sentences, or translate sentences in one language to another. Neural network based text models are powerful in terms of their prediction. Meanwhile, we are interested in models that allow us to interpret the relationship between explanatory variables and response variables. One way this can be achieved is to consider projecting the vector representation of words onto a set of axes, which we are able to understand. Given a vector representation of phrases, instead of using the vectors directly inside a prediction algorithm, one may consider projecting the vectors onto axes defined by a set of keywords.

1.1.3 Word similarities

Once the word vectors are obtained using approaches outlined in Section 1.1.2, explanatory variables can be formed using cosine similarity of words. The cosine similarity between two words with

nonzero vector representations $\boldsymbol{a}$ and $\boldsymbol{b}$ is given by

$$\text{sim}_{\cos}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{||\boldsymbol{a}||_2 \cdot ||\boldsymbol{b}||_2} \tag{1.1}$$

One may think geometrically, and interpret the cosine between two unit vectors as the dot product of the two vectors. The dot product ranges between -1 and 1, with 1 indicating identical vectors, and -1 indicating two vectors that point in the opposite direction. For example, the cosine similarity between the words *building* and *buildings* is 0.7946, while the cosine similarity between *building* and *hello* is -0.02810.

Figure 1.2 shows a projection of the word vectors in a two-dimensional space for an example set of words. In the figure, notice that *library* and *museum* appear close to each other, since they have similar functions, and hence may appear in similar contexts. Also, *graffiti*, *vandalism*, *theft*, *stolen* all appear at a similar location on the plot. Imagine drawing an arrow from the point $(0, 0)$ to the word, and the reader may see that the vector corresponding to each of these words are very similar to one another. The angle between the words is small, and hence the cosine of the angle between the words would be large (close to 1). Another way to say this is that the dot product between the words is large. Now consider the word *hail* and its corresponding vector, and compare it with the vector corresponding to *graffiti*. The two words are somewhat unrelated, and hence the angle between these two words is large. Another way to say this is that the cosine between the unit vectors corresponding to the two vectors is negative, or in other words the dot product is negative.

Now consider two documents $\boldsymbol{D}_1 = \{a_1, a_2, ..., a_S\}$ and $\boldsymbol{D}_2 = \{b_1, b_2, ..., b_T\}$. Let $\boldsymbol{a}_i$ ($i = 1, ..., S$) be the vector representation for word a_i, and let $\boldsymbol{b}_j$ ($j = 1, ..., T$) be the vector representation for word b_j. Goldberg (2017) suggests using the similarity metric

$$\text{sim}^*_{\cos}(\boldsymbol{D}_1, \boldsymbol{D}_2) = \sum_{s=1}^{S} \sum_{t=1}^{T} \text{sim}_{\cos}\left(\boldsymbol{a}_s, \boldsymbol{b}_t\right).$$

Essentially this metric assumes the vectors within a document can be added to form a vector representing the entire document. We tried using this metric, and discovered that the results could

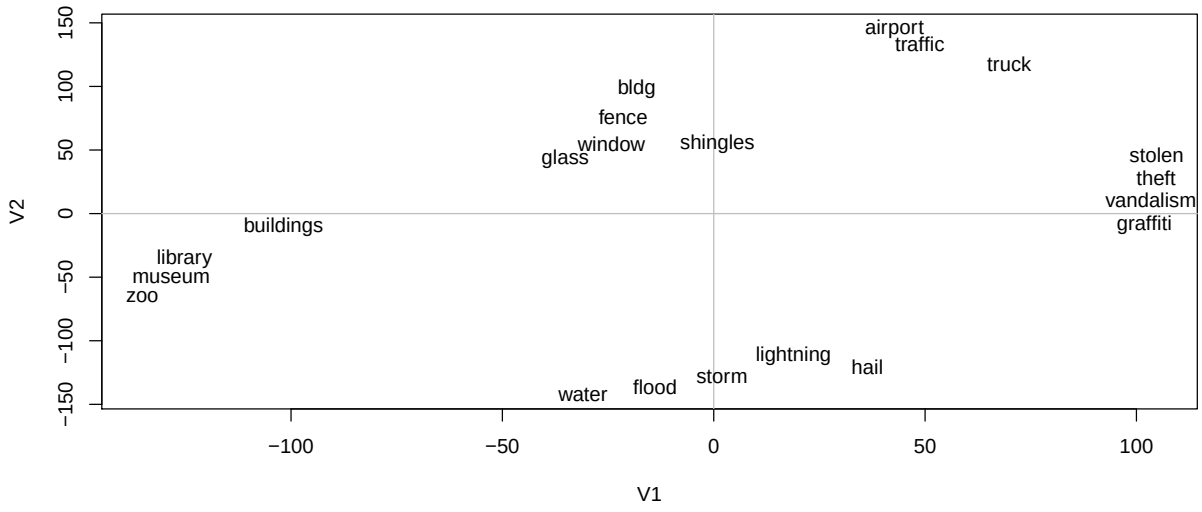

Figure 1.2: Example of a two dimensional projection of the word embeddings

be improved using a different metric. Hence, in this paper, we use the similarity metric

$$\text{sim}_{\cos}(\boldsymbol{D}_1, \boldsymbol{D}_2) = \max_{s=1,\ldots,S} \left(\max_{t=1,\ldots,T} (\text{sim}_{\cos}(\boldsymbol{a}_s, \boldsymbol{b}_t)) \right).$$

Thus the cosine similarity between two documents corresponds to the maximum cosine similarity between any two words found within the documents. In particular, the similarity between a single word a and $\boldsymbol{D} = \{b_1, b_2, ..., b_S\}$ is given by

$$\text{sim}_{\cos}(a, \boldsymbol{D}) = \max_{s=1,\ldots,S} (\text{sim}_{\cos}(\boldsymbol{a}, \boldsymbol{b}_s)) . \tag{1.2}$$

Defining the features this way is equivalent to max-pooling the features of a one-dimensional convolutional neural network. Max-pooling tends to work better than alternatives such as average pooling, as explained in page 120 of Chollet & Allaire (2018). When used with a single word a, the similarity will be 1 if the particular word appears in the claim description. Hence, in some sense, the similarity can be thought of as a detector of whether a particular word or concept appears in the claim description. This provides more interpretability, as it is a generalization of indicators for word appearance.

Continuing the example described in Section 1.1.1, we can calculate $\boldsymbol{C}$ using equation 1.2, the formula for cosine similarities,

$$\tilde{\boldsymbol{C}}_{cs} = \begin{array}{c} \begin{array}{ccccccc} \textit{lightning} & \textit{struck} & \textit{a} & \textit{building} & \textit{strike} & \textit{hit} & \textit{tree} \end{array} \\ \begin{bmatrix} 1.000 & 1.000 & 1.000 & 1.000 & 0.507 & 0.684 & 0.230 \\ 1.000 & 0.684 & 1.000 & 1.000 & 1.000 & 1.000 & 0.230 \\ 1.000 & 1.000 & 1.000 & 0.356 & 0.507 & 0.684 & 1.000 \end{bmatrix} \end{array}$$

Notice that the cosine similarities for the words *lightning* and *a* are 1 for all three cases because the words appear in each of the three documents. In addition, recall that $\boldsymbol{D}_1 = \{\textit{lightning}, \textit{struck}, \textit{a}, \textit{building}\}$. The cosine similarity between the word *tree* and $\boldsymbol{D}_1$ is 0.230. The word within $\boldsymbol{D}_1$ that achieves this maximum cosine similarity is the word *a*. This further emphasizes the importance of removing stop words from the textual descriptions. By removing stop words from each description and from w, we construct $\boldsymbol{C}$ as

$$\boldsymbol{C}_{cs} = \begin{array}{c} \begin{array}{cccccc} \textit{lightning} & \textit{struck} & \textit{building} & \textit{strike} & \textit{hit} & \textit{tree} \end{array} \\ \begin{bmatrix} 1.000 & 1.000 & 1.000 & 0.507 & 0.684 & 0.228 \\ 1.000 & 0.684 & 1.000 & 1.000 & 1.000 & 0.228 \\ 1.000 & 1.000 & 0.238 & 0.507 & 0.684 & 1.000 \end{bmatrix} \end{array}$$

When using cosine similarities, we are often more interested in the similarities closer to 1, while smaller similarities provide less information about the word of interest. For this reason, we can calculate the components of $\boldsymbol{C}$ as

$$\boldsymbol{C}_{ij}(\varepsilon) = \text{sim}_{\cos}(\boldsymbol{D}_i, w_j)\, I\left(\text{sim}_{\cos}(\boldsymbol{D}_i, w_j) \geq \varepsilon\right) \quad \text{for } i = 1, ..., n, \;\; j = 1, ..., |w| \tag{1.3}$$

where $0 < \varepsilon \leq 1$. Continuing our example, $\boldsymbol{C}(0.3)$ is

$$\boldsymbol{C}_{cs}(0.3) = \begin{array}{c} \begin{array}{cccccc} \textit{lightning} & \textit{struck} & \textit{building} & \textit{strike} & \textit{hit} & \textit{tree} \end{array} \\ \begin{bmatrix} 1.000 & 1.000 & 1.000 & 0.507 & 0.684 & 0 \\ 1.000 & 0.684 & 1.000 & 1.000 & 1.000 & 0 \\ 1.000 & 1.000 & 0 & 0.507 & 0.684 & 1.000 \end{bmatrix} \end{array}$$

Moreover, note that as ε increases to 1, $\boldsymbol{C}_{cs}(\varepsilon)$ becomes closer to $\boldsymbol{C}_{ind}$. Specifically, $\boldsymbol{C}_{cs}(1) = \boldsymbol{C}_{ind}$.

1.1.4 Data Size Considerations

The accuracy and efficiency of any statistical method applied to $\boldsymbol{C}$ is heavily influenced by the choice of w. Recall that the number of words in w is the number of columns that will be in $\boldsymbol{C}$. As we did with the example in Section 1.1.1, we could let w be the set of all words that appear in $\mathcal{D}$. Using the National Oceanic and Atmospheric Administration (NOAA) dataset, Figure 1.3 illustrates how the number of unique words is a function of the number of documents, or observations. It is clear that the number of unique words will plateau at some number of documents, which follows our intuition. Eventually the addition of another document in the dataset will not contain any words that have not appeared in the previous documents.

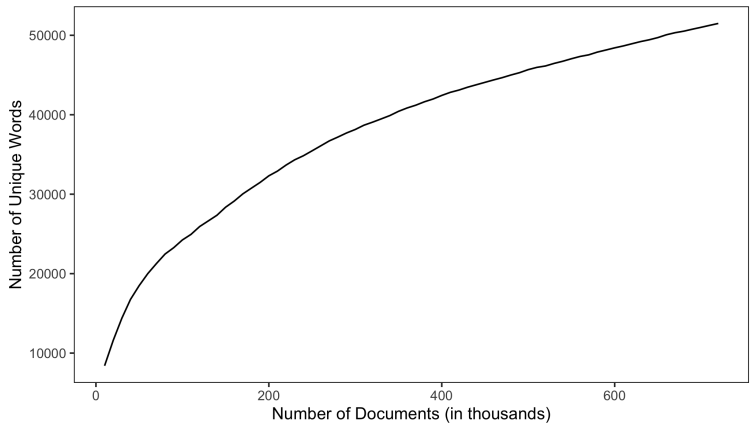

Figure 1.3: The number of unique words as a function of the number of documents in the NOAA dataset.

It is clear that choosing w to be the set of all words that appear in $\mathcal{D}$ will be difficult in any statistical analysis due to the large number of covariates. For this reason, we explore a variety of other options for choosing w.

The first method for determining w is having a subject matter expert manually choose words they believe to be the most related to the variable of interest. This method has the advantage of allowing the expert to choose as many or as few important words as they see fit. However, several possible issues may arise. To start, the subject matter expert will need to be familiar enough with cosine similarities to make appropriate choices. In addition, the expert will also have their own

biases that may influence the analysis. In Chapter 2 we will show how this method can be effective in claims classification and risk mitigation.

We could also consider using a data driven method for choosing w. Since the majority of words will most likely not be strong predictors of the response variable, we can assume that the true model will be sparse. Therefore, we choose to use Lasso to select our set of words w. In Section 1.2 we discuss the Lasso estimator along with several generalizations. In Chapters 3 and 4 we show how Lasso type penalties can be used to select w.

1.2 Lasso and Generalizations

In this section we describe the Lasso estimator along with generalizations of the Lasso method. Tibshirani (1996) originally proposed the lasso, a method combining the least squares loss with an ℓ_1-constraint. Suppose we have a response vector $\boldsymbol{y}$ of length n, an $n \times p$ covariate matrix $\boldsymbol{X}$, unknown intercept β_0, and unknown parameter vector $\boldsymbol{\beta}$ of length p. Then the lasso estimator is expressed

$$\underset{\beta_0,\boldsymbol{\beta}}{\arg\min} \left\{ \frac{1}{2n} ||\boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 \right\} \quad \text{subject to } ||\boldsymbol{\beta}||_1 \leq t, \tag{1.4}$$

where $\mathbf{1}$ is a vector of n ones. The tuning parameter t determines the total size of the coefficients, and therefore shrinks the coefficients to zero as t increases. Often, after standardizing the covariates, the lasso problem is rewritten in the Lagrangian form

$$\underset{\beta_0,\boldsymbol{\beta}}{\arg\min} \left\{ \frac{1}{2n} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_1 \right\}, \tag{1.5}$$

for some tuning parameter $\lambda \geq 0$. The tuning parameter λ is directly related to the tuning parameter t from Equation 1.4. Now that we have introduced the lasso estimator, we will discuss several generalizations. While variable selection by lasso is consistent under some conditions, the lasso method was extended to improve the consistency of variable selection.

1.2.1 Adaptive Lasso

Zou (2006) established a necessary condition for variable selection to be consistent. By doing so, they showed the existence of certain scenarios where the lasso estimator is inconsistent. For this reason, the adaptive lasso was proposed. Choosing $\gamma > 0$, the adaptive lasso solves

$$\arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j| \right\}, \tag{1.6}$$

where $w_j = 1/|\tilde{\beta}_j|^\gamma$, and $\tilde{\boldsymbol{\beta}}$ is some initial estimate. Given the initial estimates, it can be shown that equation 1.6 is convex in $\boldsymbol{\beta}$. When $p < n$, the least squares estimates can be used as the initial estimates. When $p \geq n$, the least squares estimates are not defined, however Huang, Ma, and Zhang (2008) showed that, under some general conditions, the marginal least squares estimates can be used as initial estimates. Zhang, Jeng, and Liu (2008) showed that using the lasso estimates as the initial estimates in adaptive lasso improves the sparsity recovery rate. In addition, Furthermore, Zou (2006) showed that the adaptive lasso recovers the true model under more general conditions than the lasso when the initial estimates are $\sqrt{n}$ consistent.

1.2.2 Group Lasso

In many applications of regression, the covariates may have some natural group or block structure, and it may be sensible to have all the coefficients within the group be estimated as zero or nonzero together. The most common example of this is the use of categorical variables. In practice, a categorical variable is typically coded as a series of dummy variables, and we want to either include or exclude the entire set of dummy variables from the model. This scenario provides motivation for the group lasso.

Consider a linear regression model with J groups of covariates, and each group contains p_j covariates for $j = 1, ..., J$. That is, we have design matrix $\boldsymbol{X} = (\boldsymbol{X}_1 \boldsymbol{X}_2 \cdots \boldsymbol{X}_J)$ where $\boldsymbol{X}$ is an $n \times \sum_{j=1}^{J} p_j$ matrix, and $\boldsymbol{\beta} = (\boldsymbol{\beta}_1 \boldsymbol{\beta}_2 \cdots \boldsymbol{\beta}_J)$ where $\boldsymbol{\beta}_j$ is a vector of length p_j for each $j = 1, ..., J$.

Yuan & Lin (2006) proposed the group lasso which solves

$$\arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2} \|\boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{J} \sqrt{p_j} \left\|\boldsymbol{\beta}_j\right\|_2 \right\}, \tag{1.7}$$

for some tuning parameter $\lambda \geq 0$. In addition to categorical data, group lasso can be used in the additive model framework. If we apply some basis expansion to a numeric explanatory variable, then the expansion could now be thought of having a group-like structure. This idea will be explored in Chapters 3 and 4.

1.2.3 Adaptive Group Lasso

Wang & Leng (2008) combined these extensions to formulate the adaptive group lasso, and showed the ability of the method to identify the true model consistently. For some tuning parameter $\lambda \geq 0$, the adaptive group lasso solves

$$\arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2} \|\boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{J} w_j \left\|\boldsymbol{\beta}_j\right\|_2 \right\}, \tag{1.8}$$

where $w_j = 1/\left\|\tilde{\beta}_j\right\|_2^{\gamma}$, and $\tilde{\boldsymbol{\beta}}$ is some initial estimate. In addition, Wang & Leng (2008) proved estimation consistency and selection consistency for the adaptive group lasso estimator. We will illustrate how adaptive group lasso can be utilized in Chapter 3.

CHAPTER 2

CLAIMS CLASSIFICATION AND RISK MITIGATION USING SHORT TEXTUAL DESCRIPTIONS

2.1 Introduction

When an insurance claim arises, one of the first things that is reported to a claims manager of an insurance company is a short textual description of the insurance claim, along with demographic information regarding the policyholder. Losses may be reported via a notice form, where one of the common forms used is the Association for Cooperative Operations Research and Development (ACORD) form. See page 7.16 of Kearney (2010). Upon this initial report, the claims department is responsible for a number of important tasks. The claims department would identify the policy and set adequate reserves, contact the insured, investigate the claim, document the claim, determine the precise cause of loss, liability, and the loss amount. In this process, the claims manager would coordinate with the actuarial department in order to predict the amount of insurance claim and set the adequate case reserve for each claim. In practice, parametric loss models are fit to data, and the resulting distribution is used for the prediction of the ultimate claim amount.

In this process, regression analysis helps us understand the relationship among variables. The goal of a standard regression model is to understand the relationship between the response variable $\boldsymbol{y}$, and explanatory variables $\boldsymbol{X}$, and predict future responses under a set of assumptions. The relationship $g\{\mathbb{E}[\boldsymbol{y}]\} = \boldsymbol{X}\boldsymbol{\beta}$, where typically a distributional assumption is imposed on the error, allows us to interpret the relationship between $\boldsymbol{y}$ and $\boldsymbol{X}$ in a systematic way. In case $\boldsymbol{y}$ is a binary response, logistic regression can be used. For a general overview of regression, see Frees (2009). When forming the $\boldsymbol{X}$ matrix for empirical research using traditional approaches, useful information is discarded from the analysis, because traditional regression analysis requires numeric descriptor variables. Textual information has been one example.

In this chapter, we demonstrate how we can implement the tools discussed in Section 1.1 so

that traditional analysis can be performed. The extracted information is used to build a regression model for the response variable of interest. Through this demonstration, we show how information can be extracted from textual data. We demonstrate two illustrative case studies in insurance claims classification, and insurance risk mitigation. We believe these approaches demonstrate how text processing methods can improve insurance analytics in the actuarial practice. In addition, understanding the factors that relate with large insurance losses may help us mitigate future insurance losses.

Generalized additive models (GAM) extend linear models to contain smooth functions for each of the covariates, while retaining inference about the functions. Applications of GAMs have been discussed in Hastie & Tibshirani (1990). These applications include studying kyphosis in laminectomy patients, atmospheric ozone concentration, and the intensity of ischaemic heart disease risk factors, among others.

Moreover, Hastie et al. (2009) describes an example of utilizing GAMs to classify emails as spam. While this example does analyze text, the method has several significant differences from that of our analysis. The spam example observes the number of occurrences of certain words, and fits a GAM with each word as a covariate. While the spam example in Hastie et al. (2009) also discusses the interpretability of the model, the primary goal is to predict the probability of an email being spam. In our analysis, we start by quantifying the similarity between a description and a series of selected words. The main interest in this analysis is the interpretation of the smoothing functions. This provides a much more complete explanation of the various factors that lead to high losses, and therefore, may provide for a more improved strategy of risk mitigation. A recent work, Wood (2017), provides a comprehensive overview of GAMs.

There is a vast literature in insurance claims modeling, where parametric models are employed to understand insurance claims distributions. To the best of our knowledge, combining text mining approaches with loss modeling is a new approach, which hasn't been attempted in the past. In particular, there seems to be no prior work utilizing text mining approaches to empirically understand and model insurance claims data. The rest of the Chapter proceeds in the following order:

In Section 2.2, the Wisconsin Local Government Property Insurance Fund (LGPIF) data are introduced. In Section 2.3, two applications of word similarity models are presented: insurance claims categorization, and risk mitigation. In Section 2.4, concluding remarks are provided.

2.2 Data

In this section we provide some summary statistics for the dataset. For our case study, we utilize a unique dataset of claim descriptions and loss amounts from the Wisconsin Local Government Property Insurance Fund (LGPIF). The dataset is obtained from the Office of the Commissioner of Insurance (OCI) of Wisconsin. The Wisconsin LGPIF has been established to make property insurance available for local government units. The property fund essentially acts as an insurance company in the area, providing property coverage for thousands of government entities.

In Table 2.1, it is interesting to observe that claim categories with high frequency tend to have low severity, while claim categories with low frequency tend to have high severity. In order to illustrate this effect, the table has been sorted in decreasing order of the number of observations N. Modelers have studied insurance claim frequency and severity models, and empirically it has been discovered that claim frequencies and severities are often correlated; see Frees et al. (2016).

2.2.1 Data Generation

The number of claims observations is 4991 in the training sample, and 1039 in the validation sample, which totals to 6030 observations. The data used for this chapter is already in tabular form corresponding to the data generating processes in Section 2.3, and we consider the data cleaning process, including the metadata analysis, as a black-box process that has already been performed by the provider of the data. We assume all claims are closed, and the claim amounts are fixed.

Descriptions for the observed insurance claims are recorded in the dataset. These claim descriptions are human generated, and there are 2797 unique words found in the training sample and validation sample all together. Figure 2.1 shows a projection of the word vectors in a two-dimensional space, for common words found in the dataset. Word vectors are explained in Section

1.1.2. For now, imagine that there exists a framework, where every word corresponds to some two-dimensional vector, with related words having similar vector representations. A plot of common words found in the claim descriptions file may look like Figure 2.1.

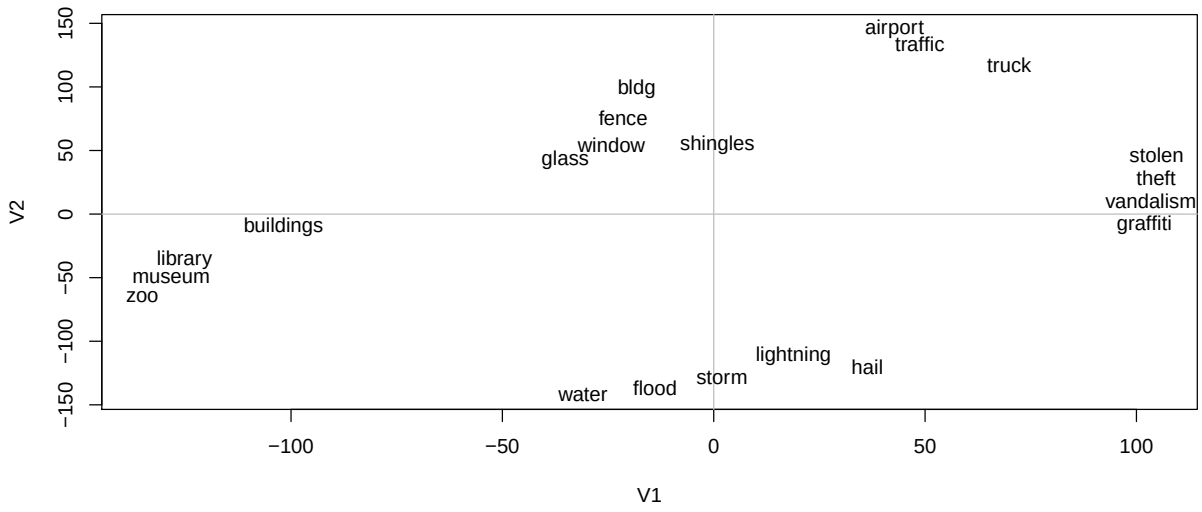

Figure 2.1: Two dimensional projection of the word embeddings for common words

In Figure 2.1, notice that *library* and *museum* appear close to each other, since they have similar functions, and hence may appear in similar contexts. Also, *graffiti*, *vandalism*, *theft*, *stolen* all appear at a similar location on the plot. Imagine drawing an arrow from the point $(0, 0)$ to the word, and the reader may see that the vector corresponding to each of these words are very similar to one another. The angle between the words is small, and hence the cosine of the angle between the words would be large (close to 1). Another way to say this is that the dot product between the words is large. Now consider the word *hail* and its corresponding vector, and compare it with the vector corresponding to *graffiti*. The two words are somewhat unrelated, and hence the angle between these two words is large. Another way to say this is that the cosine between the unit vectors corresponding to the two vectors is negative, or in other words the dot product is negative.

2.2.2 Textual Data Preprocessing

Figure 2.2 shows the most common words in the dataset. Stop-words such as *a, the, and*, etc. have been removed. No other pre-processing of the words have been performed. Notice that the word *damage* is most frequent in the descriptions. The word *vandalism* is also frequent, as vandalism turns out to be one of the most frequent claim causes in the dataset. Abbreviations such as *hs* (high school), *ms* (middle school), *es* (elementary school), *dmg* (damage), *bldg* (building) also appear in the dataset.

Note that both *bldg* and *building* are valid words. Imagine we search for the word *building* in a search engine. In this case, phrases with the word *bldg* and *building* should both appear in the search. We also note that *building* and *buildings* both appear in the vocabulary. The reason why both *bldg* and *buildings* appear when *building* is already in the vocabulary list, is because they are distinct words. How much these words relate to claim occurrence is determined by the cosine similarities of the words with selected key words. The advantage of our approach is that minimal data preprocessing is needed, so that such similar words can all be kept in the data.

Note, the claim descriptions are short phrases, such as *lightning damage to building*, or *vandalism damage at recycle center*. A total of 4991 such claim descriptions are in the training dataset. Each claim description is associated with the loss amount corresponding to the description. Each claim is categorized into one of the following claim categories: vandalism, fire, lightning, wind, hail, vehicle, water (weather related), water (non-weather related), and miscellaneous losses.

Table 2.1 shows a summary of the loss amounts for each of the nine claim categories. According to the table, vandalism has the lowest average loss amount, while the frequency of vandalism is the highest. Hail damages are the largest in terms of median and mean value, while the frequency is the lowest. The largest hail damage with a loss of 6.6 million is a hail damage to multiple buildings insured by the property fund. It is interesting to observe that the maximum loss amount happened within the weather related water damages category, which is a loss of 12.9 million. This largest loss corresponds to a water damage to a school. The second most frequent claim category is the vehicles category. This category of claim happens when a vehicle (car, plow, truck, etc.) runs into

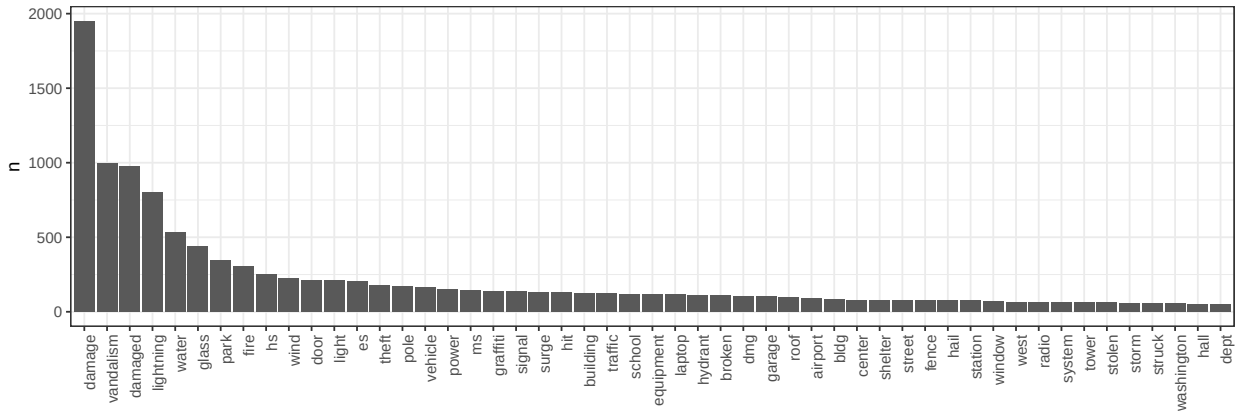

Figure 2.2: Distribution of common words

a government property building or structure, such as a light pole.

Table 2.1: Summary statistics of losses by claim category

	Validation sample					*Training sample*				
Peril	min	median	mean	max	N	min	median	mean	max	N
Vandalism	1	500	6,190	981,599	310	6	587	2,084	207,565	1774
Vehicle	1	3,000	5,662	135,268	227	37	2,500	3,905	111,740	852
Lightning	500	5,000	11,623	88,603	123	1	4,431	11,087	655,092	832
Water (weather)	55	19,337	51,608	411,641	38	1	8,898	80,432	12,922,218	426
Miscellaneous	70	3,025	9,723	242,918	103	1	3,929	29,150	2,633,822	362
Wind	325	9,010	46,304	1,048,683	107	1	4,960	18,125	492,478	296
Water (non-weather)	544	6,739	60,538	2,672,184	67	1	6,306	23,974	1,114,595	202
Fire	125	11,355	83,767	964,150	46	100	8,964	81,762	1,570,619	171
Hail	7,886	49,184	103,674	332,412	18	124	17,819	145,488	6,615,117	76

To conclude the data description, we provide some ground-truth information regarding the quality of the data. The data has no missing values, and the categories for each claim are all observed. We inspected the claim categorizations and believe that the data quality is mostly dependable, although some errors may exist because the categorization has been performed by human experts.

2.3 Applications

In this section, we demonstrate how the explanatory variables extracted from textual data can be used in specific models. We provide two examples: an example in claims classification, and an example in risk mitigation. In both applications, the Generalized Additive Models (GAM) framework is used as an underlying theme. We selected the GAM framework for its flexibility in capturing potential nonlinear effects of the cosine similarities. Word vectors are used in order to improve the classification results in all examples. These applications specifically use short textual descriptions of insurance claims, which may be found in initial reports of the claims to an insurance claims department. For long textual descriptions or claim adjuster notes, other methods such as recurrent neural networks (RNNs) using LSTM cells, or convolutional neural networks (CNNs) with multiple layers, may be preferred. For more details regarding neural network approaches to textual data analyses, see Goodfellow et al. (2016) and Goldberg (2017). Yet, the simplicity of our approach may make it attractive for actuaries working with simple textual descriptions of claims.

2.3.1 Claims Classification

2.3.1.1 *Model*

In practice, an insurance claims manager would have to classify given claims based on their properties. This task may be supported with a claims classification engine, which would take the claim description as its input, and output the correct claim category. This motivates our first application of word embedding models, which is the classification of insurance claims into discrete categories.

The engine would be trained over a training dataset, and validated over a test dataset. Within the training dataset, we assume category J_i and description $\boldsymbol{D}_i$ are observed for each claim i. Thus, the sample consists of observations $\{(J_1, \boldsymbol{D}_1), \ldots (J_n, \boldsymbol{D}_n)\}$. Note that $\boldsymbol{D}_i = (a_{i1}, a_{i2}, \ldots, a_{iq(i)})$, a description consisting of $q(i)$ words, where $q(i)$ is the number of words consisting the i-th description. We assume that $n = n_0 + n_1 + \ldots + n_{jmax}$, thus we imagine that nine samples each of size $n_0, n_1, \ldots, n_{jmax}$ are stacked to form the given sample of size n. In this paper, $j_{max} = 8$. The claim categories for the training sample and test sample are shown in Table 2.2. Thus, the response variable J_i takes on the values $0, \ldots, j_{max}$.

Table 2.2: Claim Categories for Training and Validation Datasets

	Misc. ($J_i = 0$)	Vandalism ($J_i = 1$)	Fire ($J_i = 2$)	Lightning ($J_i = 3$)	Wind ($J_i = 4$)	Hail ($J_i = 5$)	Vehicle ($J_i = 6$)	Water(NW) ($J_i = 7$)	Water(W) ($J_i = 8$)	Total
Training	362	1774	171	832	296	76	852	202	426	4991
Test	103	310	46	123	107	18	227	67	38	1039

A generalized additive model (GAM) framework can be thought of as a generalized linear model with a linear predictor involving smooth functions of covariates. See Hastie & Tibshirani (1990) and Wood (2017). For the analysis, we construct design matrix using Equation 1.3 as described in Section 1.1.3. That is, the explanatory variables, $\boldsymbol{u}_i$ are defined by

$$u_{i,k} = \text{sim}_{\cos}(w_k, \boldsymbol{D}_i) \cdot I(\text{sim}_{\cos}(w_k, \boldsymbol{D}_i) \geq \varepsilon), \quad k = 1, \ldots, K$$

where $\varepsilon = 0.2$ is used, and w_k is the k-th word used in the model, and $\boldsymbol{D}_i$ is the description of the i-th claim. For the model, we use a multinomial specification. We made this choice (as opposed to other classification methods), based on the fact that the multinomial model provides a framework that is easily generalizable to a GAM model. In this case, the probability of observing a specific peril type j is given by

$$\text{f}_i(j) = \begin{cases} 1 \Big/ \left(1 + \sum_{j'=0}^{jmax} \exp\left(\psi_{j',i}\right)\right) & \text{for base peril type } j = 0, \\ \exp\left(\psi_{j,i}\right) \Big/ \left(1 + \sum_{j'=0}^{jmax} \exp\left(\psi_{j',i}\right)\right) & \text{for peril type } 1 \leq j \leq j_{max} \end{cases}$$

with

$$\psi_{j,i} = \alpha_j + \sum_{k=1}^{K} \phi_{j,k}(u_{i,k})$$

where α_j is an intercept, and $\phi_{j,k}(k = 1, \ldots K)$ may be smooth functions of the covariate, and K is the number of words used in the model. We denote the base peril type as miscellaneous claims, and call it $j = 0$. If we assume the functions are linear so that estimation time could be saved, then we have

$$\phi_{j,k}(u_{i,k}) = u_{i,k}\beta_{j,k}, \quad k = 1, \ldots, K$$

and hence

$$\mathrm{f}_i(j) = \begin{cases} 1 \Big/ \left(1 + \sum_{j'=0}^{jmax} \exp\left(\alpha_{j'} + \boldsymbol{u}_i'\boldsymbol{\beta}_{j'}\right) \right) & \text{for base peril type } j = 0, \\ \exp\left(\alpha_j + \boldsymbol{u}_i'\boldsymbol{\beta}_j\right) \Big/ \left(1 + \sum_{j'=0}^{jmax} \exp\left(\alpha_{j'} + \boldsymbol{u}_i'\boldsymbol{\beta}_{j'}\right) \right) & \text{for peril type } 1 \le j \le jmax \end{cases}$$

Here, $\boldsymbol{\beta}_j$ are K dimensional coefficients. Note that with this choice of $\phi_{j,k}$, the GAM model becomes the GLM model. This simplification is useful for reducing the computation time, when a large number of explanatory variables are included in the model.

Table 2.3 shows $K = 7$ words defining the set w used in the model. The reader may imagine projecting each claim description $\boldsymbol{D}_i$ onto a space represented by $K = 7$ axes. In this paper, the feature words have been selected by a human expert with a good understanding of the dataset. In practice, the most frequent words found in the claim descriptions may be used as the key words. Abbreviated words are valid choices. In the claims classification problem, our goal is to construct an engine that gives the best possible classification result, and the focus is less on the interpretability of the coefficients resulting from the estimation. For applications such as that found in Section 2.3.2, the interpretability is more important, and hence the feature words should be selected more carefully by a human expert using the engine.

Table 2.3: Words used for Classification

vandalism	*fire*	*lightning*	*wind*	*hail*	*vehicle*	*water*

The reader may be curious why a censoring of the cosine similarities is needed. The reason is that cosine similarities smaller than the threshold is basically noise. One way to understand this phenomenon is to imagine a search engine returning results on the internet. The first few results are highly related to the search string that has been entered, but as one goes down the list, more and more irrelevant results may be observed. These junk results tend to add noise to the regression result, and hence we censor the cosine similarities with a threshold $\varepsilon = 0.2$, where the choice of ε is an empirical question. Figure 2.3 in Section 2.3.1.2 shows the classification accuracy as a function of the threshold ε.

2.3.1.2 *Result*

Table 2.4: Confusion matrix for multinomial model

Actual	Predicted Category									
Category	Misc.	Vandalism	Fire	Lightning	Wind	Hail	Vehicle	Water(NW)	Water(W)	Total
Misc.	24	33	1	1	0	0	39	0	5	103
Vandalism	17	267	0	0	2	0	23	0	1	310
Fire	0	2	18	3	0	0	20	2	1	46
Lightning	3	1	0	114	0	1	3	0	1	123
Wind	4	4	2	3	88	2	1	0	3	107
Hail	0	0	0	0	0	17	1	0	0	18
Vehicle	31	5	4	0	0	0	182	2	3	227
Water(NW)	2	4	0	0	0	0	5	4	52	67
Water(W)	5	1	0	0	4	0	0	1	27	38
Total	86	317	25	121	94	20	274	9	93	1039

The analysis of classification accuracy is often performed by the Receiver Operating Characteristic (ROC) curve in the binary classification problem. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various thresholds. The Area Under the Curve (AUC) is often used as a measure of how well the classification engine is performing. In the multiple class problem, numeric measures such as the average accuracy, error rate, and precision are better suited for analyzing the classification accuracy. These quantities are defined by

$$\text{Average Accuracy} = \frac{1}{j_{max}+1} \sum_{j=0}^{j_{max}} \frac{tp_j + tn_j}{tp_j + fn_j + fp_j + tn_j},$$

$$\text{Error Rate} = \frac{1}{j_{max}+1} \sum_{j=0}^{j_{max}} \frac{fp_j + fn_j}{tp_j + fn_j + fp_j + tn_j},$$

$$\text{Precision} = \frac{1}{j_{max}+1} \sum_{j=0}^{j_{max}} \frac{tp_j}{tp_j + fp_j},$$

where tp_j is the number of true positives, tn_j is the number of true negatives, fp_j is the number of false positives, fn_j is the number of false negatives; see Sokolova & Lapalme (2009). Figure 2.3 shows the average accuracy, error rate, and precision. With $\varepsilon = 0.2$, the average accuracy is 93.62%, the error rate is 6.37%, and the precision is 63.9%. We tested if these numbers changed as the threshold ε changed.

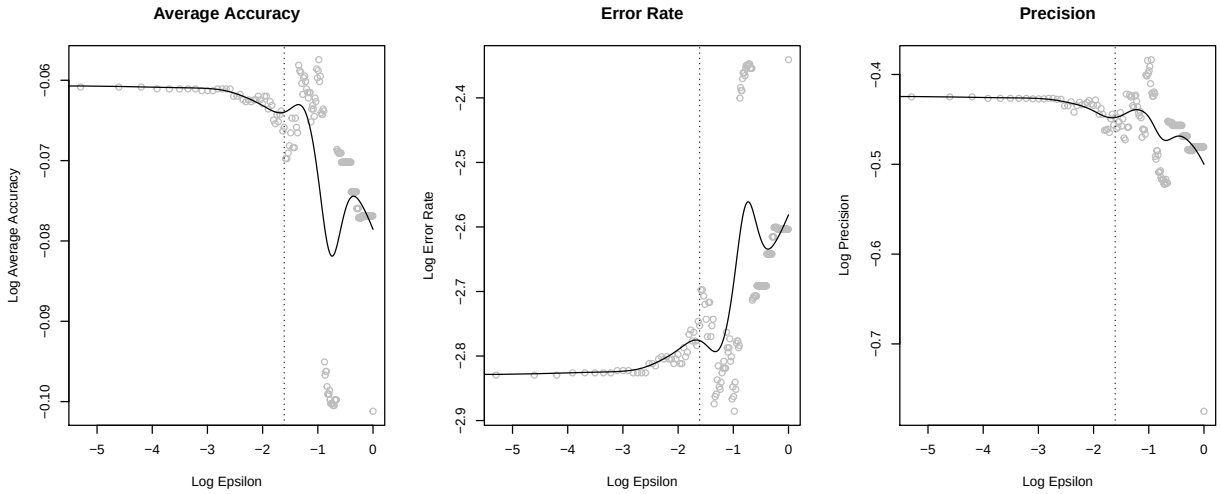

Figure 2.3: Average accuracy, error rate, and precision in log scale

Figure 2.3 shows that the average accuracy, error rate, and precision changes as ε is altered by 0.005. The solid lines show degree 10 splines fit to the experiment data. Notice that when $\varepsilon = 1$ (or in other words when $\log \varepsilon = 0$) the accuracy and precision reduces, and the error rate increases significantly. This is precisely the case when the cosine similarities boil down to indicator variables of whether the particular words are found in the claim descriptions (in other words, when there exists an embedding for at least one word in the description that is a constant multiple of the keyword). Figure 2.3 is evidence that the model matrix based on cosine similarities is outperforming that based on indicators. Our choice $\varepsilon = 0.2$ is shown as a vertical dotted line at $\log(0.2) = -1.609$. We emphasize once more that the choice of ε is an empirical question, as the classification accuracy is not influenced much by the threshold. Moreover, since the graph is stable on the left-hand side,

$\varepsilon = 0$ may also be a reasonable choice, if prediction alone is the concern. Yet, for interpretation of the coefficients, we have chosen a non-zero threshold.

2.3.2 Risk Mitigation

2.3.2.1 *Model*

In order to understand the factors that relate with high losses, we assume a sampling frame in the following form: We assume that the loss amount, the category of the loss, and the description of the loss are observed. While J_i was a random variable in section 2.3.1, here we assume $J_i = j$ is fixed. Also, we assume for each category, n_j losses are observed.

Let j be the category of the loss, and let $Y^*_{j,i}$ be the underlying loss amount, and $\boldsymbol{D}_{j,i}$ the description of the i-th loss in the j-th category. Thus, the dataset consists of distinct samples with observations of the form

$$\{(Y^*_{0,1}, \boldsymbol{D}_{0,1}), \ldots, (Y^*_{0,n_0}, \boldsymbol{D}_{0,n_0})\}$$
$$\vdots$$
$$\{(Y^*_{jmax,1}, \boldsymbol{D}_{jmax,1}), \ldots, (Y^*_{jmax,n_{jmax}}, \boldsymbol{D}_{jmax,n_{jmax}})\}$$

For a given $0 < \gamma < 1$, responses $Y_{j,i}$ are formed by

$$Y_{j,i} = I(Y^*_{j,i} > q_j(\gamma))$$
$$q_j(\gamma) = \inf\{y : P(Y^*_j \leq y) \geq \gamma\}$$

where $\gamma = 0.5$ is used to obtain the median for each category. For our analysis, the empirical quantile has been used for $q_j(\gamma)$. In other words, for any given loss, $Y_{j,i}$ is an indicator of whether the observed loss $Y^*_{j,i}$ is above the 50th percentile of losses in that category of loss. Any other quantile could have been used. For instance, the 95th percentile could have been used. Different quantiles would give different stories, because the definition of a large claim would be different for each case. The 50th percentile has been arbitrarily selected for demonstration.

We analyzed the losses for vandalism, fire, wind, vehicle, and the two water damage categories, omitting lightning, hail, and miscellaneous losses from the analysis because for the latter three it was difficult to identify keywords that correspond to large claims. Detailed results are shown for the vandalism peril type only, in order to limit the number of pages of the paper. The question is, whether we can understand the factors that are related to response values $Y_{j,i} = 1$, corresponding to high losses, through text analysis.

Using the word similarity metric described in Section 1.1.3, we can create variables for risk measures of interest. Consider a specific description of a claim in a given category. Suppose a modeler is interested in creating a risk metric corresponding to a word $w_{j,k}$, $k = 1, \ldots, K_j$ (K_j: number of risk metrics in category j). Suppose an insurance claim is described by the phrase $\boldsymbol{D}_i$, for the i-th observation, $i = 1, \ldots, n$, where n is the number of claims found in the dataset. We create a variable by

$$u_{j,i,k} = \text{sim}_{\cos}(w_{j,k}, \boldsymbol{D}_i) \cdot I(\text{sim}_{\cos}(w_{j,k}, \boldsymbol{D}_i) \geq \varepsilon_j)$$

for a threshold ε_j. In this paper, $\varepsilon_j = 0.2$ is chosen for each j. Thus, $u_{j,i,k}$ is the cosine similarity between a search word $w_{j,k}$ and the description of the claim D_i, with a censoring below a certain similarity level. For example, the modeler may be interested in the risk metric *graffiti*. The modeler may be interested in the relationship between this risk metric, and a response variable of interest, say the magnitude of loss. If the metric has a high correlation with large losses, then attention should be given to the particular risk metric in order to mitigate the risk inherent in this metric. In this case, the modeler may create a column vector for *graffiti*. For a particular response, say the likelihood of a high vandalism claim, a modeler may have a set of risk metrics of interest, say: *graffiti, laptop, window, shelter, pool* (In this case, $K_j = 5$). This gives a matrix of K_j explanatory variables, which can be used in standard regression models. Table 2.5 summarizes the explanatory variables used for the vandalism model.

For category j (this section will focus on the vandalism category in particular), given a claim

Table 2.5: Summary statistics of explanatory variables (vandalism model)

	Min.	*Median*	*Mean*	*Max.*
laptop	0.000	0.000	0.130	1.000
graffiti	0.000	0.520	0.424	1.000
window	0.000	0.246	0.348	1.000
shelter	0.000	0.221	0.187	1.000
pool	0.000	0.260	0.209	1.000

i, the GAM model for each peril type can be specified as

$$g\left\{\mathbb{E}(Y_{j,i})\right\} = \alpha_j + \sum_{k=1}^{K_j} \phi_{j,k}(u_{j,i,k})$$

where

$$g(\mu_{j,i}) = \log\left(\frac{\mu_{j,i}}{1-\mu_{j,i}}\right),$$

and α_j is the coefficient for the intercept, and $\phi_{j,k}$ are smooth functions of the covariates $u_{j,i,k}$, subject to the constraint such that $\sum_{i=1}^{n} \phi_{j,k}(u_{j,i,k}) = 0$. In other words, a logit link is used since $Y_{j,i}$ is a binary variable taking on the value of 1 or 0. We have

$$\mu_{j,i} = \frac{\exp(\boldsymbol{X}'_{j,i}\boldsymbol{\beta}_j)}{1+\exp(\boldsymbol{X}'_{j,i}\boldsymbol{\beta}_j)},\ V(\mu_{j,i}) = \mu_{j,i}\left(1-\mu_{j,i}\right)$$

Let the matrix $\boldsymbol{X}_j$ include transformed columns representing the spline bases for the $\phi_{j,k}$. For this study, a second order B-spline basis with three terms is used. In this case, the design matrix $\boldsymbol{X}_j$ can be constructed by first forming the matrix

$$\boldsymbol{\Phi}_{j,k} = \begin{bmatrix} B_1^1(u_{j,1,k}) & B_2^1(u_{j,1,k}) & B_3^1(u_{j,1,k}) \\ B_1^1(u_{j,2,k}) & B_2^1(u_{j,2,k}) & B_3^1(u_{j,2,k}) \\ \vdots & \vdots & \vdots \\ B_1^1(u_{j,n,k}) & B_2^1(u_{j,n,k}) & B_3^1(u_{j,n,k}) \end{bmatrix}, \quad k = 1, \ldots K_j$$

where $B_l^m(u)$ are the B-spline basis functions, presented in Wood (2017). Then the columns are transformed using QR factorization, in order to impose the identifiability constraint. This involves decomposing each vector $\boldsymbol{\Phi}_{j,k}^T\mathbf{1}$ into the form

$$\boldsymbol{\Phi}_{j,k}^T\mathbf{1} = \begin{bmatrix} \boldsymbol{Q}_{j,k,1} & \boldsymbol{Q}_{j,k,2} \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_{j,k} \\ \mathbf{0} \end{bmatrix}$$

and then taking $\boldsymbol{Q}_{j,k,2}$ for $k = 1, \ldots, K_j$ to form the design matrix

$$\boldsymbol{X}_j = \left[\mathbf{1};\ \ \boldsymbol{\Phi}_{j,1}\boldsymbol{Q}_{j,1,2};\ \ \boldsymbol{\Phi}_{j,2}\boldsymbol{Q}_{j,2,2};\ \ \ldots\ \ \boldsymbol{\Phi}_{j,K_j}\boldsymbol{Q}_{j,K_j,2}\right].$$

This ensures that an intercept is included in the design, and also allows the basis functions $\phi_{j,k}$ to satisfy $\sum_{i=1}^{n} \phi_{j,k}(u_{j,i,k}) = 0$. The idea of P-IRLS is that a weight matrix is adjusted each time the algorithm iterates until convergence. The algorithm follows the following steps.

1. Given the current $\boldsymbol{X}_j\boldsymbol{\beta}_j^{[h]}$, calculate the diagonal matrix $\boldsymbol{W}_j^{[h]}$

   $$W_{j,ii}^{[h]} = \left[G_{j,ii}^{[h]2} V\left(\mu_{j,i}^{[h]}\right)\right]^{-1}$$

   and

   $$\boldsymbol{z}_j^{[h]} = \boldsymbol{G}_j^{[h]}\left(\boldsymbol{y}_j - \boldsymbol{\mu}_j^{[h]}\right) + \boldsymbol{X}_j\boldsymbol{\beta}_j^{[h]}$$

   where $\boldsymbol{G}_j^{[h]}$ is a diagonal matrix satisfying $G_{j,ii}^{[h]} = g'\left(\mu_{j,i}^{[h]}\right)$, and $\mu_{j,i}^{[h]} = g^{-1}\left(\boldsymbol{X}_j\boldsymbol{\beta}_j^{[h]}\right)$.

2. Then minimize

   $$\left\|\sqrt{\boldsymbol{W}_j^{[h]}}\left(\boldsymbol{z}_j^{[h]} - \boldsymbol{X}_j\boldsymbol{\beta}_j\right)\right\|^2 + \sum_{k=1}^{K_j} \lambda_{j,k}\boldsymbol{\beta}_j^T\boldsymbol{Q}_{j,k,2}^T\boldsymbol{S}_{j,k}\boldsymbol{Q}_{j,k,2}\boldsymbol{\beta}_j$$

   with respect to $\boldsymbol{\beta}_j$.

3. Repeat steps 1 and 2 until convergence.

Here, h is the iteration index, and $\boldsymbol{S}_{j,k}$ are matrices designed to penalize the roughness of the smooth functions. In this paper, we used the difference penalty

$$\boldsymbol{S}_{j,k} = \boldsymbol{D}^T\boldsymbol{D}, \quad \text{where} \quad \boldsymbol{D} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$$

The $\lambda_{j,k}$ are selected by generalized cross validation. Cross validation is applied to a subset of the training sample corresponding to each peril type. This involves minimizing the generalized cross validation score

$$\mathcal{V}_j = \frac{n_j \sum_{i=1}^{n_j}(y_{j,i} - \hat{y}_{j,i})^2}{[n_j - tr(\boldsymbol{A}_j)]^2}$$

where $\boldsymbol{A}_j$ is the influence matrix for the j-th category. For details of the theory behind cross validation, we reference Wood (2017). The above procedure is performed for each j, where in our work each of the following peril types are considered: vandalism, fire, wind, vehicle, water (non-weather), water (weather). Generalized additive models are implemented in the R programming language via packages such as `gam` and `mgcv`. In this paper, parameters are estimated using the `mgcv` R package, implemented by the author of Wood (2017). We chose this package because it provides a convenient interface regarding the choice of basis functions and graphical outputs.

2.3.2.2 *Result*

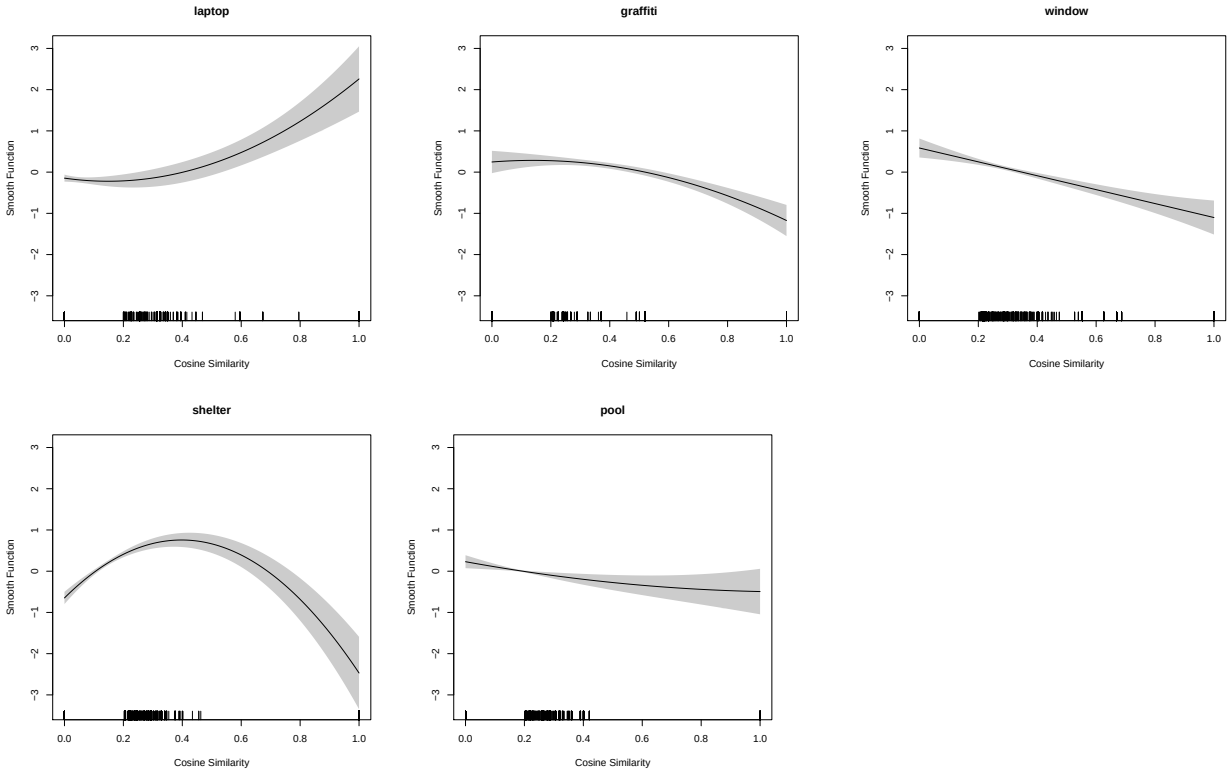

Figure 2.4: GAM model plots for vandalism

In this section, we present the analysis results for the vandalism category. For the vandalism category, explanatory variables *laptop*, *graffiti*, *window*, *shelter*, *pool* were included in the GAM model. Among these, *laptop* turns out to have a positive relationship with high losses. Figure 2.4 shows the shape of the smooth functions $\phi_{j,k}, j = 1, k = 1, \ldots, 5$, as the explanatory variable

varies from 0 to 1. The shaded regions in Figure 2.4 illustrate the 95% credible intervals of the smooth functions at each point on the curve.

Notice that data for large values of cosines are scarce, hence the credible interval widens for large values of the explanatory variables. Figure 2.5 shows the words with highest cosine similarity with *laptop*, which turns out to be positively related with high losses. Presumably, vandalisms and thefts to laptops, computers, portables turn out to result in relatively high losses within the vandalism category. Note that vandalisms are small frequent losses. Although the loss amounts in this category are small, the frequent nature of the losses may make it worthwhile to mitigate thefts to laptops. The words in Figure 2.5 are not necessarily found in the training data. They are words found in the word embedding matrices. What we are saying here is that since the word *laptop* has a high correlation with large losses, related words such as *laptops*, *computers*, and *phones* are also suspects for potential high losses, due to their relationship to the word *laptop*. This type of chart helps an insurance claims department to learn which type of property to focus on, when mitigating risk.

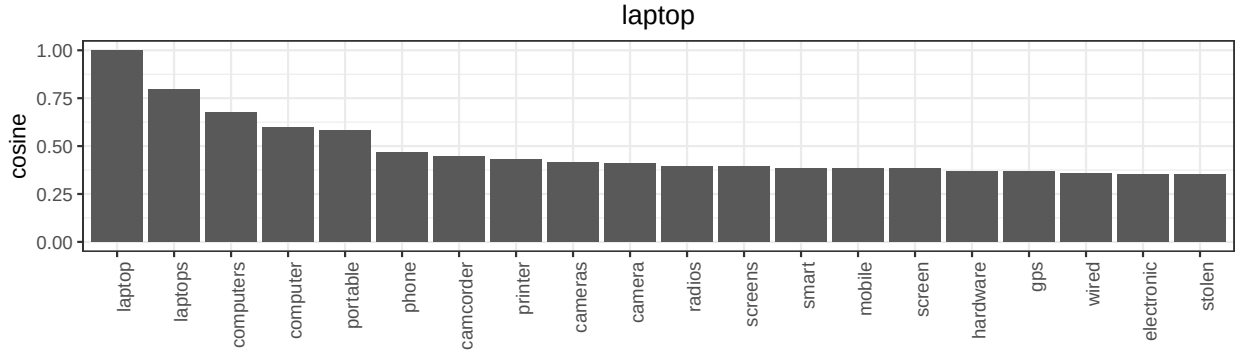

Figure 2.5: Words that relate with high vandalism losses

2.3.2.3 *Model diagnostic*

Table 2.6 provides a summary of the GAM models. The χ^2 statistic reported for each smooth function is based on Wood (2013), or page 305–308 of Wood (2017). Essentially, this tests the hypothesis

$$H_0 : \phi_{j,k}(u) = 0$$

for all u in the range of the cosine similarities for category j. A high p-value would indicate the function $\phi_{j,k}$ is not needed in the model. The smooth functions for vandalism all turn out to be significant, according to Table 2.6.

Table 2.6: GAM model summary (vandalism model)

	Chi.sq	*p.value*
s(laptop)	32.560	0.000
s(graffiti)	39.250	0.000
s(window)	47.430	0.000
s(shelter)	132.930	0.000
s(pool)	9.020	0.001
R-sq.(adj)	0.206	

2.4 Concluding Remarks

In this chapter, we introduced a framework for incorporating textual data into insurance claims modeling, and considered its applications in claims management processes. An insurance claim representative is responsible for investigating the claim, in order to determine the handling process. In this chapter, we explored the use of word similarities as a tool for modeling insurance claims and mitigating insurance risks. Our results demonstrate how text mining technology can be incorporated into a traditional regression analysis. The methodology is applicable in many different areas of applications, where textual data arises. Possible applications of our approach for an insurance risk manager may include:

- Classification of claims based on textual descriptions of the claims
- Classification of policyholders based on textual descriptions of the policyholders
- Prediction of insurance claims at the claim level
- Prediction of insurance claims at the policyholder level
- Analysis of insurance claims and risk mitigation

We explored the LGPIF data in the form of case studies to understand the factors that relate with high insurance losses, classify insurance claims, and model the loss amounts using parametric

distributions involving covariates derived from textual information. We make some remarks on the current limitations of our framework, where potential improvements can be made.

- Under the current framework, words not found in the word embedding matrix cannot be used in the modeling.
- The threshold ε is selected using heuristics by a human expert, under the current framework.
- Because pre-determined word embedding matrices are limited to one-grams (single words) at the time the paper is being written, the incorporation of n-grams (use of phrases longer than one word as a search key) remains an open question.
- Further linguistic barriers may exist, if the textual descriptions are longer than those appearing in the dataset used for this paper. Examples may be polysemy, false friends, compound words.
- In order to use the proposed method, insurers that focus on specific insurance segments may be constrained to build its own word embedding matrices, as the terms appearing in the claim descriptions may be specific to the field. For example, a medical insurer may find GloVe insufficient, and may need a word embedding matrix trained on medical terms in order to use our proposed approach.

Economic losses due to property damage caused by perils including fire, lightning, wind, hail, or vandalism have vast implications to our society. Understanding the nature of property damage can improve our readiness and contribute to minimizing the losses. We have illustrated a way to help realize this goal using a new analysis method, which, to the best of our knowledge, has not been attempted before in the actuarial literature. We believe our methodology may help broaden the horizon of empirical research, and contribute to the advancement of the understanding of our world and the risks residing within it. In addition, we believe that our approach will improve the claim handling procedures of insurance claims departments.

CHAPTER 3

A THREE STAGE APPROACH TO MODEL BUILDING

3.1 Introduction

In practice, an important task of an insurance analyst is to assist the claims manager in setting the case reserves for reported claims. The case reserve for a given claim can be understood as the difference between the reported claim amount and the paid amount for an individual claim. Note that the case reserve excludes incurred but unreported claims, for which a separate incurred but not reported (IBNR) reserve should be prepared. For the purpose of this chapter, the reader should understand that the case reserve is an approximation to the ultimate amount of the claim, given information available at the time of the report of the claim. Sometimes the case reserve is set by the claims department of an insurance company, while in other cases the task is outsourced to an outside adjustor. Part of the information available to the claims department at the report time of the claim is a textual description of the claim. In this chapter, we are interested in approaches that use the textual information regarding an insurance claim to predict the ultimate loss amount, by regressing the loss amount on a set of covariates derived from the textual description. Given a dataset of historic loss descriptions and ultimate loss amounts, an actuary may use the approach to improve the case reserving procedure.

Part of the problem in this prediction task is that if we use a large number of keywords in forming the design matrix extracted from the textual descriptions of claims, the resulting problem is high-dimensional in nature. In this case study, we use the framework of Lee et al. (2019) and analyze a dataset of loss descriptions and amounts, downloaded from the National Oceanic and Atmospheric Administration (NOAA).

For this analysis, a generalized linear model (GLM) may not be appropriate because the linearity assumption may not appropriately fit the data. To solve such a problem, we may consider using a nonparametric regression technique. A variety of nonparametric regression techniques have been

developed, including but not limited to regression splines, kernel smoothing, neural networks, and generalized additive models (GAM). Nonparametric regression has been applied in many areas, from modeling daily pollution in the U.K. (Wood et al., 2017), to estimating relative risk for disease mapping of lung cancer (Dreassi et al., 2014). See Simonoff (1996) for more details and examples of nonparametric regression. In this chapter, we consider the GAM.

Hastie & Tibshirani (1986) proposed the generalized additive model that consists of the summation of smooth functions, allowing for the ability to capture the true, not necessarily, nonlinear relationship. In the generalized additive model setup, more information is needed to estimate each function as compared to the generalized linear model setup. Therefore, the data must have many more observations than the number of covariates. In addition, when working with high dimensional data, the scalability of the algorithm is also extremely important when considering a method. Our approach is motivated by these characteristics.

Considerable work has been done in efficiently estimating larger datasets using generalized additive models. Most recently, Wood et al. (2017) developed a method for estimating GAM with the number of coefficients of order 10^4, and observations up to 10^8. This method reduces the number of matrix operations, utilizes parallelization, and reduces the memory necessary by marginal discretization of the model covariates. Li & Wood (2019) extended this work by proposing an alternative method of calculating $\mathbf{X}'\mathbf{W}\mathbf{X}$ where $\mathbf{X}$ is a model matrix and $\mathbf{W}$ a diagonal or tri-diagonal matrix, which results in a 30 fold reduction in computational time. Previous works include Marra & Wood (2011) and Wood et al. (2015). Code for these methods are found in the `R` package `mgcv`, Wood (2019).

While the aforementioned GAM results provide for a scalable algorithm, a hindrance of GAM is the restriction on the number of covariates. Considering the GLM, there are several methods for combating the high dimensionality issue, with most notably being lasso by Tibshirani (1996). Similar to the GLM, the lasso maximizes the likelihood, but instead has an additional L_1 penalty term. This term is typically referred to as the shrinkage term. Extensions of the lasso have also been developed, including but certainly not limited to, group lasso from Yuan & Lin (2006) and

adaptive lasso from Zou (2006). The group lasso is applied to variables with group-like structure, and it uses a slightly altered penalty term where each variable in a group is penalized equally. This is particularly important due to the group-like structure induced by the basis expansion used in the estimation of the generalized additive model. The adaptive lasso simply applies a weight to each coefficient in the penalty term, with these weights typically estimated through ordinary least squares or lasso. Wang & Leng (2008) combined these extensions to formulate the adaptive group lasso, and showed the ability of the method to identify the true model consistently.

Our proposed method is a three step approach consisting of (1) weight calculation by group lasso, (2) the shrinkage step by adaptive group lasso, and (3) the smoothing step. This approach combines the adaptive group lasso dimension reduction technique with the scalable GAM algorithm.

3.2 Data and Preprocessing

For our analysis, we utilize the publicly available NOAA Storm Events Database. The analysis is performed on property loss amounts at the event level, using storm event observations involving textual descriptions of the events. The data are collected over time, however we use a cross-sectional model in this paper, in order to focus on the relationship between the textual information and the response. Only *Thunderstorm Wind* events taking place in Michigan and from 2000 to 2018 are considered for the analysis. For validations, the dataset is divided into training and validation datasets. The reason we use this dataset is because it contains relatively clean, lengthy descriptions of losses from storm events in the United States each year, along with the property and crop damage amount estimates. These damage amounts are initial estimates of the losses, and hence are different from the ultimate loss amounts. Yet, the structure of the data is identical to that available to a claims adjuster, and hence is a good test dataset for the analytical framework explained in this chapter. Another advantage of this dataset is that it is publicly available, allowing dissemination and reproducibility to be easy.

Each event is recorded with an event narrative. An example of an observation with an estimated property damage of $10,000 has an event narrative that reads: *Roof damage was incurred to a barn*

Table 3.1: Summary statistics for the log(loss) for the training and validation datasets.

	N	Min	Mean	SD	Max
Training	2353	2.30	8.97	1.44	17.03
Validation	126	6.21	8.78	1.56	14.00

six miles northwest of Mason due to a severe thunderstorm wind gust and a large tree limb was blown down in South Lansing.

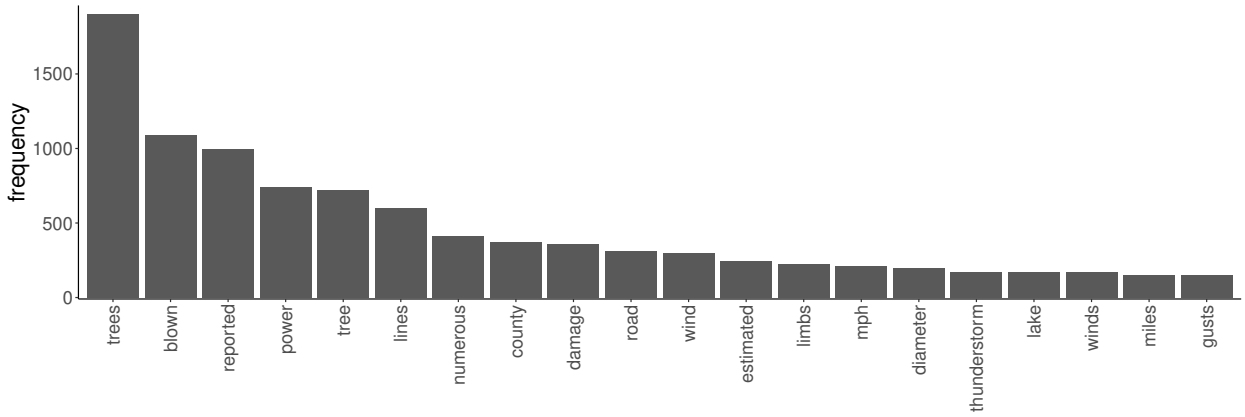

Figure 3.1: Frequency for the most common words.

Figure 3.1 shows the most common words in the descriptions of the losses. Stop-words such as *a, the, and*, etc. have been removed. Notice that the word *trees* is most frequent in the descriptions. A few of the most common words are typically used to describe what is happening to *trees*, such as *blown* and *wind*. In addition, several of the other most common words like *power, lines, damage,* and *outages* are used to describe the results of downed trees.

There are a total of $2,353$ observations in the training set, with 126 observations in the validation set. As previously mentioned, the claim descriptions are quite lengthy, with an average of 16.8 words per description. There are a total of $2,642$ unique words used in the dataset. To capture only relevant words, stop-words, numbers, and words that only occurred once were removed, resulting in $1,998$ words. Table 3.1 provides summary statistics for the log(loss) for the training and validation datasets.

In order to better understand the relationship between the words in the claim description and the property loss amount, each word is represented by a vector. Recent advancements in word

embedding models have made it possible to obtain these representations easily. We utilize the 300 dimensional word embeddings developed by the authors of Pennington et al. (2014). To form the design matrix, we follow the framework described in Section 1.1.3. That is, we will choose our design matrix, $\boldsymbol{X}_{n\times p_n}$, to be $\boldsymbol{C}_{cs}$ where w is the set of all unique words found in $\mathcal{D}$, n is the number of descriptions in $\mathcal{D}$, and p_n is the number of words in w.

Each value in the matrix is now continuous and restricted to $[-1, 1]$. Figure 3.2 shows the relationship between cosine similarity and property loss for *house* and *thunderstorm*. From the figure, we see that the relationship between cosine similarity and property loss is nonlinear in nature and therefore a generalized additive model is appropriate.

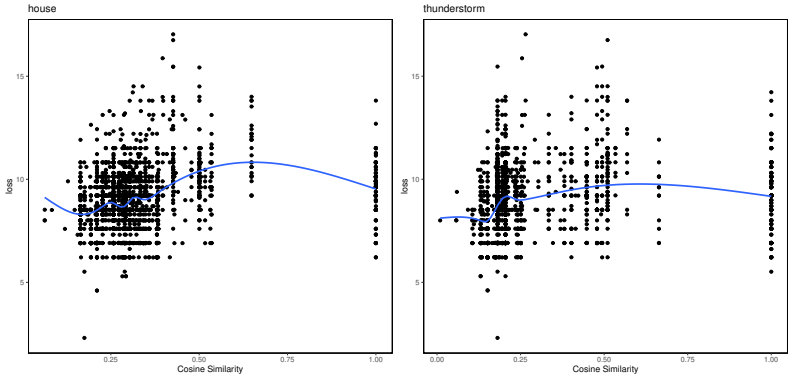

Figure 3.2: Cosine similarity against property loss for *house* and *thunderstorm*.

3.3 The Model and Basis Expansion

In this section, we describe our methodology in specific terms. We consider the generalized additive model

$$\mu_i = E[y_i|\boldsymbol{X}_i] = g^{-1}\left(\sum_{j=1}^{p_n} f_j(X_{ij})\right), \tag{3.1}$$

where the link function corresponds to that of the corresponding exponential family distribution. For each of the n independent observations, the density function is given by

$$f_{y_i} = c(y)\exp\left[\frac{y\theta_i - b(\theta_i)}{\phi}\right], \quad 1 \le i \le n, \quad \theta_i \in \mathbb{R} \tag{3.2}$$

We assume that a matrix of explanatory variables is given. Let's call it $\boldsymbol{X}_{n\times p_n}$, and use the notation $\boldsymbol{X} = (\boldsymbol{X}_1^T, \boldsymbol{X}_2^T, \ldots, \boldsymbol{X}_n^T)^T$. We have

$$\boldsymbol{X}_{n\times p_n} = \begin{bmatrix} X_{11} & X_{12} & \ldots & X_{1p_n} \\ X_{21} & X_{22} & \ldots & X_{2p_n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \ldots & X_{np_n} \end{bmatrix} \tag{3.3}$$

Thus, n is the number of observations, and p_n is the number of explanatory variables available. We assume that the parameter $0 < \phi < \infty$ is known. Without loss of generality, let $\phi = 1$. Also, we assume that the density of y_i depends on $\boldsymbol{X}_i$ via the structure

$$\theta_i = \sum_{j=1}^{p_n} f_j(X_{ij}), \tag{3.4}$$

where θ_i are defined in equation 3.2. Let the values X_{ij} be defined over intervals $[a, b]$, and let $a = \xi_0 < \xi_1 < \ldots < \xi_K < \xi_{K+1} = b$ be a partition on $[a, b]$, where $K = K_n = n^v$, with $0 < v < 0.5$, such that

$$\max_{1\le k\le K+1} |\xi_k - \xi_{k-1}| = O(n^{-v}). \tag{3.5}$$

For smooth functions $f_1, \ldots, f_{p_n}$, there exist functions $f_{n1}, \ldots, f_{np_n} \in \mathcal{S}_n$, such that $f_{n1}, \ldots, f_{np_n}$ well approximate $f_1, \ldots, f_{p_n}$. Here, $\mathcal{S}_n$ is the space of polynomial splines of degree $l \ge 1$. In other words, $\mathcal{S}_n$ is the space of piecewise polynomials on each interval of the partition, which are $l' \le l - 2$ times continuously differentiable for $l \ge 2$. Smoothness is defined in the same way as Huang et al. (2010). According to Schumaker (1981), there exists a normalized B-spline basis $\{\phi_k, 1 \le k \le m_n\}$ for $\mathcal{S}_n$, where $m_n = K_n + l = n^v 1$. For a practical overview of the B-spline basis function, the reader may refer to Wood (2017), section 5.3.3, starting from page 204. We want to write

$$f_{nj}(X_{ij}) = \sum_{k=1}^{m_n} \Phi_{ik}^{[j]} \beta_{jk}, \tag{3.6}$$

for some value $\Phi_{ik}^{[j]}$. For this, our next goal is to construct a design matrix

$$\begin{aligned}\boldsymbol{\Phi}_{n\times q_n} &= \left(\mathbf{1} : \boldsymbol{\Phi}^{[1]} : \boldsymbol{\Phi}^{[2]} : \ldots : \boldsymbol{\Phi}^{[p_n]}\right) \\ &= \left(\mathbf{1} : \boldsymbol{\Theta}^{[1]}\boldsymbol{Q}_2^{[1]} : \boldsymbol{\Theta}^{[2]}\boldsymbol{Q}_2^{[2]} : \ldots : \boldsymbol{\Theta}^{[p_n]}\boldsymbol{Q}_2^{[p_n]}\right)\end{aligned} \tag{3.7}$$

from $\boldsymbol{X}_{n\times p_n}$, where $\boldsymbol{\Theta}^{[j]}$ are matrices of basis functions. Denote $q_n = m_n \times p_n$. Here, $\boldsymbol{\Phi}^{[j]}$ are each $n \times m_n$ matrices of transformed basis functions, where m_n is the number of parameters within each basis function consisting the design matrix. We consider the matrix of basis functions

$$\boldsymbol{\Theta}^{[j]} = \begin{bmatrix} \phi_1(X_{1j}) & \phi_2(X_{1j}) & \ldots & \phi_{m_n}(X_{1j}) \\ \phi_1(X_{2j}) & \phi_2(X_{2j}) & \ldots & \phi_{m_n}(X_{2j}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(X_{nj}) & \phi_2(X_{nj}) & \ldots & \phi_{m_n}(X_{nj}) \end{bmatrix}. \tag{3.8}$$

From here on, we assume $\phi_k(x)$ are m_n-parameter B-spline basis functions of order 3. In order to obtain the design matrix, we form the QR decomposition

$$\left(\boldsymbol{\Theta}^{[j]}\right)^T \mathbf{1} = \boldsymbol{Q}_1^{[j]}\boldsymbol{R}^{[j]} + \boldsymbol{Q}_2^{[j]}\mathbf{0} \tag{3.9}$$

and take the $\boldsymbol{Q}_2^{[j]}$ matrix corresponding to the zero part of the decomposition, and define

$$\boldsymbol{\Phi}^{[j]} = \boldsymbol{\Theta}^{[j]}\boldsymbol{Q}_2^{[j]} \tag{3.10}$$

Finally, let the $\boldsymbol{\Phi}_{n\times q_n}$ matrix be given by expression (3.7). We call $\boldsymbol{\Phi}$ our design matrix, and denote the elements of the design matrix ϕ_{it}, for $i = 1, \ldots, n$ and $t = 1, \ldots q_n$. We also denote

$$\boldsymbol{\Phi}_{ij} = \left(\Phi_{i1}^{[j]}, \Phi_{i2}^{[j]}, \ldots, \Phi_{im_n}^{[j]}\right)^T, \quad \text{for } i = 1, \ldots, n, \text{ and } j = 1, \ldots p_n. \tag{3.11}$$

Under this framework, the response variable is related to the covariate X_{ij} via

$$f_j(X_{ij}) = \boldsymbol{\Phi}_{ij}^T\boldsymbol{\beta}_j, \quad i = 1, \ldots, n, \quad j = 1, \ldots, q, \tag{3.12}$$

where $\boldsymbol{\beta}_j$ is the coefficient corresponding to the j-th explanatory variable. We may see that $\boldsymbol{\beta}_j$ must be a length m_n vector, since the j-th spline contains m_n parameters. The illustrated approach

to constructing the design matrix ensures that the basis functions satisfy

$$\sum_{i=1}^{n} f_{nj}(X_{ij}) = 0 \quad \text{for each } j = 1, \ldots, p_n. \tag{3.13}$$

In other words, selecting the model matrix this way imposes an identifiability constraint, which states that the smooth function defined by the basis functions must satisfy

$$\sum_{i=1}^{n} f_{nj}(X_{ij}) = \mathbf{1}^T \boldsymbol{\Phi}^{[j]} \boldsymbol{\beta}_j = 0 \tag{3.14}$$

Notice that equation (3.14) holds because

$$\mathbf{1}^T \boldsymbol{\Phi}^{[j]} \boldsymbol{\beta}_j = \mathbf{1}^T \boldsymbol{\Theta}^{[j]} \boldsymbol{Q}_2^{[j]} \boldsymbol{\beta}_j = \begin{bmatrix} \boldsymbol{R}^{[j]} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_1^{[j]} \\ \boldsymbol{Q}_2^{[j]} \end{bmatrix} \boldsymbol{Q}_2^{[j]} \boldsymbol{\beta}_j = \boldsymbol{R}^{[j]} \boldsymbol{Q}_1^{[j]} \boldsymbol{Q}_2^{[j]} \boldsymbol{\beta}_j = 0 \tag{3.15}$$

since $\boldsymbol{Q}_1^{[j]}$ and $\boldsymbol{Q}_2^{[j]}$ are orthogonal. Simply stated, if $f_{nj}(x)$ is a spline, the coefficients for the spline are now given by $\boldsymbol{Q}_2^{[j]} \boldsymbol{\beta}_j$ instead of $\boldsymbol{\beta}_j$ after the transformation. Our methodology is related to Chouldechova & Hastie (2015), yet we use a three step procedure. We are looking for the parameters for

$$g\left\{E[y_i | \boldsymbol{X}_i]\right\} = \beta_0 + \sum_{j=1}^{p_n} f_{nj}(X_{ij}) = \beta_0 + \sum_{j=1}^{p_n} \boldsymbol{\Phi}_{ij}^T \boldsymbol{\beta}_j = \beta_0 + \boldsymbol{\Phi}_i \boldsymbol{\beta} \tag{3.16}$$

where we have used the notation $\boldsymbol{\Phi}_i$ to denote the i-th row of $\boldsymbol{\Phi}$, and $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^T, \boldsymbol{\beta}_2^T, \ldots, \boldsymbol{\beta}_{p_n}^T)^T$, where some of the β_j's are zero, while others are non-zero. The approach in Chouldechova & Hastie (2015) is to minimize the penalized negative log-likelihood

$$-\frac{1}{n}\ell(\boldsymbol{\beta}) + \lambda_{n2} \sum_{j=1}^{p_n} \sqrt{\boldsymbol{\beta}'_j \boldsymbol{S}_j \boldsymbol{\beta}_j} + \frac{1}{2\phi} \sum_{j=1}^{p_n} \lambda_{n3j} \boldsymbol{\beta}' \boldsymbol{D}_j \boldsymbol{\beta} \tag{3.17}$$

where $\ell(\boldsymbol{\beta})$ is the log-likelihood for an exponential family distribution:

$$\begin{aligned} \ell(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \left[y_i \left(\sum_{j=1}^{p_n} \sum_{k=1}^{m_n} \Phi_{ik}^{[j]} \beta_{jk} \right) - b \left(\sum_{j=1}^{p_n} \sum_{k=1}^{m_n} \Phi_{ik}^{[j]} \beta_{jk} \right) \right] \\ &= \sum_{i=1}^{n} \left[y_i \left(\boldsymbol{\Phi}_i^T \boldsymbol{\beta} \right) - b \left(\boldsymbol{\Phi}_i^T \boldsymbol{\beta} \right) \right] \end{aligned} \tag{3.18}$$

The hope is that the second term in equation (3.17) induces zeros into groups of coefficients, while the last term imposes smoothness into the "surviving" coefficients. Here, $\boldsymbol{S}_j$ is an identity matrix of dimension m_n, and $\boldsymbol{D}_j$ is a constraint matrix to impose smoothness into the estimated functions f_j. There are several practical difficulties with this approach:

- When p_n is large, or in other words when the problem dimension is large, there are too many λ_{n3j} tuning parameters to estimate. Wood (2017) discusses algorithms for large n cases, but does not talk about cases where p_n is large.
- Theory behind selecting the tuning parameters λ_{n3j}, discussed in Wood (2017) is no longer directly applicable, because of the extra group lasso type penalty term.
- Implementing the coordinate descent algorithm, which brings in sparsity into $\boldsymbol{\beta}$, becomes tricky with the smoothing penalty. Usually fast algorithms for lasso type estimators with GLMs are implemented by locally approximating the likelihood with a Taylor's approximation at each iterative step, yet the extra penalty term makes this tricky.
- Estimating the coefficients may take a very long time, especially when the number of explanatory variables p_n is large, as in the application we consider in this paper.

Hence, in order to keep the estimation procedure scalable for large p_n (and hence large q_n), we propose a three step approach to the estimation problem for the model (3.16). The first step of the approach is to perform a group lasso estimation with the first and second terms of equation (3.17). The second step uses the resulting coefficient estimates to perform an adaptive group lasso estimation of the parameters. The third and final step uses the nonzero coefficients obtained from the second step to induce smoothness into the implied spline function $f_{nj}(\cdot)$, for each nonzero function f_{nj}. These steps are formalized in the following section.

Moreover, to provide a statistical validation, we present both the numerical results in section 3.6 and the theory for the estimated functions in Appendix A, which works as another support of our proposed 3-stage approach. We aim at validating two things: the variables selected are consistent;

the estimators are consistent with respect to the unknown true functions. Appendix B contains a short simulation study illustrating the effectiveness of the method on a simulated dataset.

3.4 The 3-Stage Approach

3.4.1 Stage 1 – Group lasso

Define the objective function to be

$$L(\boldsymbol{\beta}; \lambda_{n1}) = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \left(\boldsymbol{\Phi}_i^T \boldsymbol{\beta} \right) - b \left(\boldsymbol{\Phi}_i^T \boldsymbol{\beta} \right) \right] + \lambda_{n1} \sum_{i=1}^{p_n} \|\boldsymbol{\beta}_j\|_2 \tag{3.19}$$

Let $\hat{\boldsymbol{\beta}}$ be the optimizer for (3.19), or in other words

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p_n \cdot m_n}}{\arg\min} \; L(\boldsymbol{\beta}; \lambda_{n1}) \tag{3.20}$$

3.4.2 Stage 2 – Adaptive group lasso

Define the objective function to be

$$L_a(\boldsymbol{\beta}; \lambda_{n2}) = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \left(\boldsymbol{\Phi}_i^T \boldsymbol{\beta} \right) - b \left(\boldsymbol{\Phi}_i^T \boldsymbol{\beta} \right) \right] + \lambda_{n2} \sum_{j=1}^{p_n} w_{nj} \|\boldsymbol{\beta}_j\|_2 \tag{3.21}$$

where the weights depend on the screening stage group lasso estimator

$$w_{nj} = \begin{cases} \|\hat{\boldsymbol{\beta}}_j\|_2^{-1} & \text{if } \|\hat{\boldsymbol{\beta}}_j\|_2 \; > 0 \\ \infty & \text{if } \|\hat{\boldsymbol{\beta}}_j\|_2 \; = 0 \end{cases} \tag{3.22}$$

Numerically, the weights are set to a large number, for the case when $\|\hat{\boldsymbol{\beta}}_j\|_2 = 0$.

Let $\hat{\boldsymbol{\beta}}_{AGL}$ be the optimizer for (3.21). In other words,

$$\hat{\boldsymbol{\beta}}_{AGL} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p_n \cdot m_n}}{\arg\min} \; L_a(\boldsymbol{\beta}; \lambda_{n2}) \tag{3.23}$$

Let $\hat{S}_n$ be the subset of $\{1, \ldots, p\}$, such that the jth coefficient of $\boldsymbol{\beta}_{AGL}$ with $j \in \hat{S}_n$ are nonzero. Thus, the second stage estimates are sparse, meaning that the coefficients are zero for some j. This reduces the coefficient size in the third stage.

3.4.3 Stage 3 – The smoothness penalty

Let $\boldsymbol{\Phi}^{\hat{S}_n}$ be the matrix consisting of columns from $\boldsymbol{\Phi}$ corresponding to the set $\hat{S}_n$. Let $\boldsymbol{\beta}_{\hat{S}_n}$ be in $\mathbb{R}^{\hat{s}_n \cdot m_n}$, where $\hat{s}_n = |\hat{S}_n|$. Define the objective function to be

$$L_{sm}(\boldsymbol{\beta}; \boldsymbol{\lambda}_{n3}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\left(\boldsymbol{\beta}^T\boldsymbol{\Phi}_i^{\hat{S}_n}\right) - b\left(\boldsymbol{\beta}^T\boldsymbol{\Phi}_i^{\hat{S}_n}\right)\right] + \frac{1}{2\phi}\sum_{j\in\hat{S}_n}\lambda_{n3j}\boldsymbol{\beta}_j^T\boldsymbol{D}_j\boldsymbol{\beta}_j \tag{3.24}$$

where $\boldsymbol{\lambda}_{n3} = (\lambda_{n31}, \lambda_{n32}, \ldots, \lambda_{n3p_n})$. Let $\hat{\boldsymbol{\beta}}_{sm}$ be the optimizer for (3.24). In other words

$$\hat{\boldsymbol{\beta}}_{sm} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{m_n\hat{s}_n}}{\arg\min}\ L_{sm}(\boldsymbol{\beta}; \boldsymbol{\lambda}_{n3}) \tag{3.25}$$

Since the problem of dimension has been reduced, the third step estimation may be performed using existing generalized additive models routines, using $\hat{\boldsymbol{\beta}}_{AGL}$ as the initial guess for the P-IRLS procedure. The tuning parameters $\boldsymbol{\lambda}_{n3}$ may be obtained by generalized cross validation or REML as described in Wood (2017).

3.4.4 Tuning Parameters

Each stage has a tuning parameter, λ_{n1}, λ_{n2}, and $\boldsymbol{\lambda}_{n3}$, respectively. The selection of λ_{n1} and λ_{n2} can greatly influence the performance of the model and the efficiency of the algorithm. Larger values of λ_{n1} and λ_{n2} will lead to an over-simplified model with faster computation time, while smaller values will lead to an over-fitted model with slower computation time. To find the "sweet spot", cross validation is used to determine λ_{n1} and λ_{n2}. The tuning parameters $\boldsymbol{\lambda}_{n3}$ is obtained by generalized cross validation or REML as described in Wood (2017).

3.5 Algorithm

We now discuss the implementation of the method using `R`. For stage 1 and stage 2, we utilize functions from the `gglasso` package (Yang & Zou, 2017) and for stage 3 we utilize functions from the `mgcv` package (Wood, 2019). The code is provided in an R package at `github.com/scottmanski/TAGAM`.

3.5.1 Stage 1 – Group lasso

The `gglasso` function is modified such that we loop through the grid of λ_{n1} values, but once the number of nonzero coefficients is greater than n, the algorithm is stopped. By doing so, we ensure that we will be able to execute stage 3.

3.5.2 Stage 2 – Adaptive group lasso

The implementation of stage 2 is very similar to that of stage 1, except for the addition of the weights. In order to incorporate the weights, let $\beta_j^{'} = w_{nj}\beta_j$ for each $j \in \{1, ..., p_n\}$. Then equation 3.21 can be written as

$$L_a(\boldsymbol{\beta}^{'}; \lambda_{n2}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\left(\sum_{j=1}^{p_n}\frac{1}{w_{nj}}\boldsymbol{\Phi}_i^{[j]T}\boldsymbol{\beta}_j^{'}\right) - b\left(\sum_{j=1}^{p_n}\frac{1}{w_{nj}}\boldsymbol{\Phi}_i^{[j]T}\boldsymbol{\beta}_j^{'}\right)\right] + \lambda_{n2}\sum_{j=1}^{p_n}\|\boldsymbol{\beta}_j^{'}\|_2 \quad (3.26)$$

3.5.3 Stage 3 – The smoothness penalty

The `mgcv` package is used to implement stage 3. In the `mgcv` package, there is a `gam` function and a `bam` function, with the former designed for smaller datasets and the latter designed for much larger datasets. In this analysis, we utilize `bam`. To increase the computational efficiency we also choose to have the function discretize the data following the method described in Wood et al. (2017).

3.6 Results

In this section, we discuss the results of our model. Table 3.2 provides information for the final model. As previously mentioned, 1,998 words appeared in the dataset, and were considered as possible covariates. For the model, we chose to use the penalized regression spline. Stage 1 effectively reduced the number of covariates to 261, and stage 2 further reduced the number of words to 149. While the number of functions to interpret may seem cumbersome, the final model is relatively simple compared to the number of possible covariates that could have been in the model.

Figure 3.3 shows the estimated functions for several covariates. All of the function estimates have a few characteristics in common. For smaller cosine similarity values, the estimated functions

Table 3.2: Summary statistics for the final model. The Residual DF comes from the estimated degrees of freedom from the GAM, and the MSPE is the out of sample mean squared prediction error.

K_n	l	λ_{n1}	λ_{n2}	$\hat{s}_n$	Residual DF	Deviance	MSPE
4	2	0.0005255074	0.0001063902	149	2167.387	70.7%	1.016

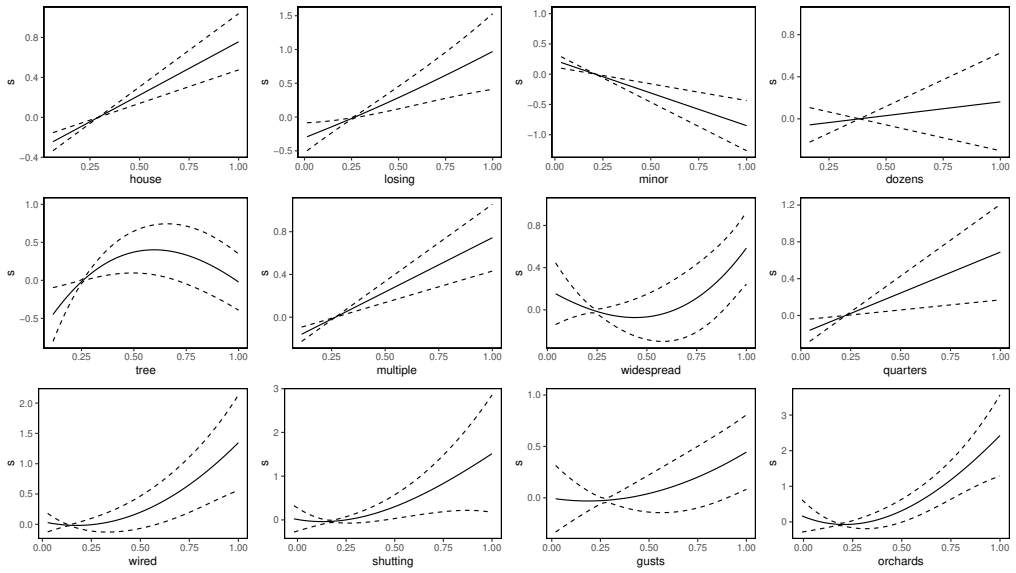

Figure 3.3: Function estimates for several covariates.

are approximately zero. We expect this because smaller cosine similarities between a word and a phrase indicates that the word has very little meaning in common with the phrase. For large cosine similarity values, the 95% credible interval for the functions becomes wider as compared to cosine similarity values around 0.2. This is also expected simply due to the lack of observations for higher cosine similarities.

The function estimates help us understand the relationship between a word, its related words, and the property loss amount. Many of these estimated functions seem to follow our intuition. For example, *house*, *losing*, *widespread*, *gusts*, and *tree* are all words that would typically be associated with property loss. Words with the highest cosine similarity to *house* are shown in figure 3.4. Most of these related words are types of homes. From the function estimate, we see that an incident involving a house results in higher property loss than that of an incident involving offices or apartments.

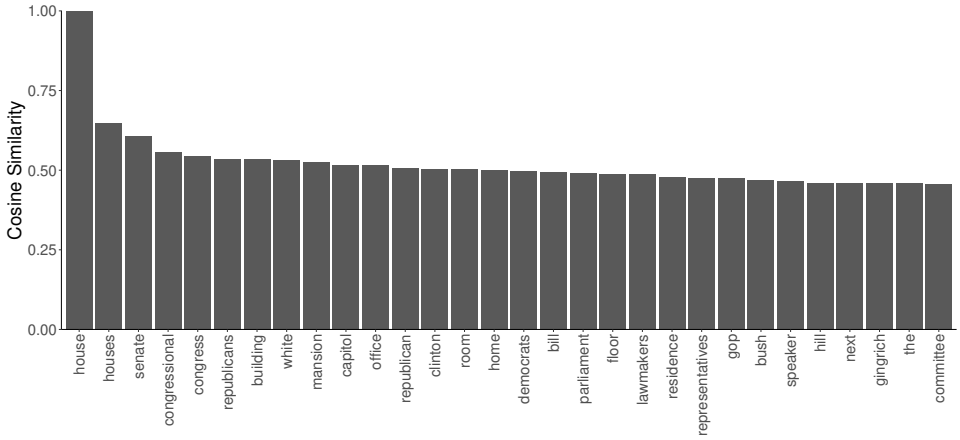

Figure 3.4: Words with the highest cosine similarity with *house*.

While many function estimates obviously follow our intuition, there are some that seem harder to interpret. Words like *quarters*, *shutting*, and *orchards* all seem unrelated to property loss. To shed some light on this issue, we look at a sentence from a description that includes *quarters*; *Two eyewitnesses in Covington reported hail greater than the size of quarters during the peak of the storm.* The use of *quarters* here is related to the size of hail. It is expected that larger hail will lead to larger property loss. Words related to *quarters* include *nickel* and *dime*, which are also used to describe hail size. In a similar way, we find out that *shutting* is referring to the closure of major roadways. In the case of *orchards*, several observations involved damage to apple orchards. With Michigan producing the third most apples of any state, it is clear why damage to apple orchards results in large property loss.

The model also performed well with out of sample prediction. Figure 3.5 shows the predicted property loss amounts against the true loss amounts for the validation sample. The Spearman correlation for the validation set is 76.06%, while the Spearman correlation for the training dataset is 80.30%.

To measure the stability of the method, for a selected year, the model was trained using the previous years, and tested on data from the selected year. This was completed for each year from 2001 to 2018. This resulted in an average mean squared prediction error of 1.34 with a standard error of 0.123. Using a lasso model increases each of these values by about 8% respectively. The

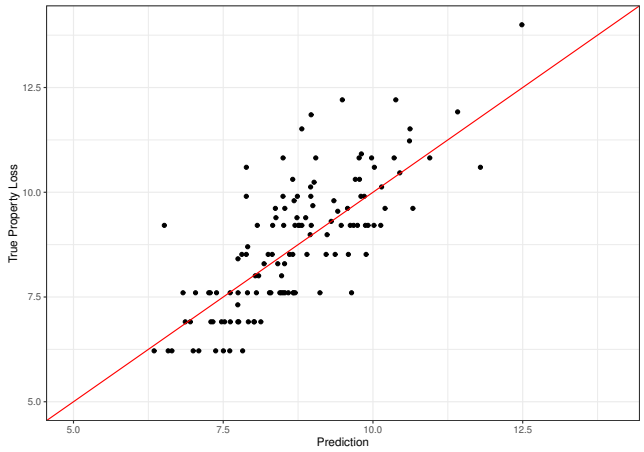

Figure 3.5: Predicted property loss amounts against the true property loss amounts for the validation sample. The Spearman correlation is 76.06%.

3 stage method selected a more parsimonious model as compared to the single step lasso model, resulting in greater model stability.

3.7 Implications

We have presented an analytical method for analyzing losses due to storm events in relation to their textual descriptions. The fact that losses may be predicted more accurately with textual information implies that the case reserving procedure may be improved significantly. The traditional approach to case reserving is to take the average amount of the reported losses, yet this does not take advantage of the heterogeneity of information contained within the initial report of a loss to an insurance company. The new method allows for a more accurate prediction of the ultimate loss to be indemnified for a specific reported loss.

Being able to explain the factors that contribute to higher or lower severity of losses by selecting the relevant keywords from a set of words allows the actuarial analyst to avoid manually selecting the keywords needed for the textual risk analysis. This technique may be useful especially when the number of words describing the loss is large, or statistically the problem is high-dimensional. The analyst may also be able to understand the factors that relate to high losses using the selected covariates, and this may help mitigate future losses.

In addition, these factors that contribute to higher severity of property loss can indicate areas

needing improvement in the way they protect against various weather events. For example, events involving *orchards* resulted in high property loss, illustrating the need for additional preventative measures to protect the apple trees during a thunderstorm.

The fact that a simple three-step approach allows for the regression selection problem to be solved easily using existing routines in the R programming language. The approach may be applied in general to problems where nonlinear effects of a large number of continuous explanatory variables must be understood in relation to the response. We have focused on the log-normal case of the response, yet the method is general enough to be applied to non-normal responses, including responses following a gamma distribution, or Poisson distribution. Future work may focus on these specific cases.

3.8 Concluding Remarks

In this chapter, we consider a general high dimensional text analysis problem and propose a 3 stage approach by adopting modern statistical methods. Stage 1 and 2 effectively reduced the high dimensional problem to one that `mgcv` can handle. The use of stage 1 and 2 to reduce the problem instead of utilizing a subject matter expert allows for simple replicability of the process. We showed how the use of cosine similarities from textual descriptions can provide interpretable results when predicting property loss. While there are many other possible applications in risk analysis, our framework could also be applied in:

- classification of users on a social networking site based on their posts,
- prediction of a company's change in stock price from related articles,
- caller scam classification based on call transcripts.

CHAPTER 4

GAMMA DOUBLE GENERALIZED LINEAR MODEL WITH GROUP LASSO

4.1 Introduction

4.1.1 Overview

The recent advancement of coordinate descent approaches to the Lasso estimation problem has allowed the variable selection problem to be solved in a systematic way. Lasso is an approach to induce sparsity into the coefficients estimated from a regression analysis, which has been introduced by Tibshirani (1996). Since its introduction, the method has gained popularity in the statistics literature due to its elegancy and ability to solve high dimensional regression problems. In a highly cited paper by Yuan & Lin (2006), the authors have introduced a method to group the coefficients in a regression for group-wise variable selection.

In spite of the popularity and the extensive research performed on the Lasso method, applications in the actuarial science literature has been limited until now. The reason is in part because datasets found in the actuarial literature are typically not high dimensional in nature, and in part also because generalizing the fast coordinate descent methods to non-normal responses remained a challenging task. Although works by authors such as Friedman et al. (2010) have extended the Lasso method to logistic regression and multinomial regression, problems involving other non-normal responses such as the Tweedie distributed case were left as open problems.

Yet, in a recent strand of works by Yang & Zou (2015) and Qian et al. (2016), an approach to solve the Tweedie distributed case with high speed has been introduced. The approach approximates the likelihood function with a second order Taylor series expansion around the current estimate of the coefficients, and uses the approximation to construct a closed form update to the coefficient estimates. This paper extends the works by Yang & Zou (2015) and Qian et al. (2016) and applies the technique to textual data analysis in the actuarial science literature using the gamma distribution,

which is a special case of the Tweedie distribution.

Insurance claims prediction using textual data analysis methods is a natural application of the group Lasso technique for non-normal responses. The use of the technique in conjunction with dispersion modeling is new to the literature to the best of our knowledge. In the actuarial science literature, double GLM approaches have been used by authors such as Smyth (1989) and Smyth & Jørgensen (2002). In a dispersion modeling framework, one uses an additional link function for the dispersion. The gamma distribution double GLM model has been explored in Smyth (1989), and the Tweedie model has been explored in Smyth & Jørgensen (2002). Our idea in this paper is to incorporate a group Lasso type penalty in both the dispersion parameterization and the mean parameterization of a gamma model.

The rest of the chapter proceeds in the following order: In Section 4.2 the model is explained along with a fast algorithm for obtaining the parameters. In Section 4.3 we discuss the asymptotic convexity of the negative log-likelihood. In Section 4.4 two simulations are conducted to investigate the performance of the models explained in Section 4.2. Section 4.5 describes the application of the method to insurance claim predictions using textual data analysis. Finally, Section 4.6 concludes the chapter with closing remarks.

4.2 Model

4.2.1 One Dimensional Unpenalized Problem

In order to explain our approach, we demonstrate the algorithm in Yang & Zou (2015) and Qian et al. (2016) for a one-dimensional problem. For simplicity, suppose we have the one dimensional negative log-likelihood

$$\ell(\beta) = \sum_{i=1}^{n} y_i e^{-\beta} + \beta, \tag{4.1}$$

which arises from the gamma regression model with only an intercept term β. The Taylor series approximation around a point $\tilde{\beta}$ is given by

$$\ell_Q(\beta) = \ell(\tilde{\beta}) + \sum_{i=1}^{n} \left(-y_i e^{-\tilde{\beta}} + 1\right)(\beta - \tilde{\beta}) + \frac{1}{2}\sum_{i=1}^{n} y_i e^{-\tilde{\beta}}(\beta - \tilde{\beta})^2 \tag{4.2}$$

This can be simplified to

$$\ell_Q(\beta) = \frac{1}{2}\sum_{i=1}^{n} \tilde{\nu}_i(\tilde{y}_i - \beta)^2 + c(\tilde{\beta}) \tag{4.3}$$

where

$$\tilde{\nu}_i = y_i e^{-\tilde{\beta}} \quad \text{and} \quad \tilde{y}_i = \tilde{\beta} + (1 - y_i^{-1} e^{\tilde{\beta}}) \tag{4.4}$$

and $c(\tilde{\beta})$ is constant given $\tilde{\beta}$. Fixing the $\ell_Q(\beta)$ function, which is an approximation to the $\ell(\beta)$ function, we try to minimize $\ell_Q(\beta)$ by solving the problem

$$\arg\min_{\beta} \ell_Q(\breve{\beta}) + \breve{U}(\beta - \breve{\beta}) + \frac{1}{2}\tilde{H}(\beta - \breve{\beta})^2 \tag{4.5}$$

assuming a current estimate for $\breve{\beta}$ is given. The idea is, we update $\breve{\beta}$ until it converges to the minimum of a given $\ell_Q(\beta)$ fixed for a given $\tilde{\beta}$. Once this minimization is achieved, we update $\tilde{\beta}$ and continue. Here,

$$\breve{U} = \frac{\partial \ell_Q}{\partial \beta} = -\sum_{i=1}^{n} (\tilde{y}_i - \breve{\beta})\tilde{\nu}_i \quad \text{and} \quad \tilde{H} = \frac{\partial^2 \ell_Q}{\partial \beta^2} = \sum_{i=1}^{n} \tilde{\nu}_i$$

The first order condition for equation (4.5) is given by

$$\breve{\beta}_{(new)} = \breve{\beta} - \tilde{H}^{-1}\breve{U} \tag{4.6}$$

The resulting $\breve{\beta}_{(new)}$ satisfies $\ell_Q(\breve{\beta}_{(new)}) \leq \ell_Q(\breve{\beta})$, because

$$\begin{aligned}
\ell_Q(\breve{\beta}_{(new)}) &= \frac{1}{2}\sum_{i=1}^{n} \tilde{\nu}_i(\tilde{y}_i - \breve{\beta}_{(new)})^2 + c(\tilde{\beta}) \\
&= \frac{1}{2}\sum_{i=1}^{n} \tilde{\nu}_i(\tilde{y}_i - \breve{\beta} - (\breve{\beta}_{(new)} - \breve{\beta}))^2 + c(\tilde{\beta}) \\
&= \frac{1}{2}\sum_{i=1}^{n} \tilde{\nu}_i(\tilde{y}_i - \breve{\beta})^2 - \sum_{i=1}^{n} \tilde{\nu}_i(\tilde{y}_i - \breve{\beta})(\breve{\beta}_{(new)} - \breve{\beta}) + \frac{1}{2}\sum_{i=1}^{n} \tilde{\nu}_i(\breve{\beta}_{(new)} - \breve{\beta})^2 + c(\tilde{\beta}) \\
&= \ell_Q(\breve{\beta}) - \sum_{i=1}^{n} \tilde{\nu}_i(\tilde{y}_i - \breve{\beta})(\breve{\beta}_{(new)} - \breve{\beta}) + \frac{1}{2}\sum_{i=1}^{n} \tilde{\nu}_i(\breve{\beta}_{(new)} - \breve{\beta})^2 \\
&= \ell_Q(\breve{\beta}) + \breve{U}(\breve{\beta}_{(new)} - \breve{\beta}) + \frac{1}{2}\tilde{H}(\breve{\beta}_{(new)} - \breve{\beta})^2 \\
&\leq \ell_Q(\breve{\beta}) + \breve{U}(\breve{\beta} - \breve{\beta}) + \frac{1}{2}\tilde{H}(\breve{\beta} - \breve{\beta})^2 = \ell_Q(\breve{\beta})
\end{aligned} \tag{4.7}$$

where, the inequality in equation (4.7) is due to the fact that $\breve{\beta}_{(new)}$ is the solution for the minimization problem (4.5). Hence, the iteration scheme eventually converges to the solution

$$\hat{\beta} = \arg\min_{\beta} \ell(\beta) \tag{4.8}$$

4.2.2 One Dimensional Penalized Problem

Now, consider the penalized problem

$$\hat{\beta}^* = \arg\min_{\beta} \ell(\beta) + \lambda|\beta| \tag{4.9}$$

where λ is a tuning parameter. In this subsection, we denote the solution to the penalized problem with an upper asterisk (*). We try to minimize $P_Q(\beta) = \ell_Q(\beta) + \lambda|\beta|$ by solving the minimization problem

$$\arg\min_{\beta} \ell_Q(\breve{\beta}^*) + \breve{U}(\beta - \breve{\beta}^*) + \frac{1}{2}\tilde{H}(\beta - \breve{\beta}^*)^2 + \lambda|\beta| \tag{4.10}$$

at each iterative step, for a given current estimate $\breve{\beta}^*$. Notice, we have

$$\begin{aligned}
P_Q(\breve{\beta}^*_{(new)}) &= \frac{1}{2}\sum_{i=1}^{n} \tilde{v}_i(\tilde{y}_i - \breve{\beta}^*_{(new)})^2 + c(\tilde{\beta}) + \lambda|\breve{\beta}^*_{(new)}| \\
&= \frac{1}{2}\sum_{i=1}^{n} \tilde{v}_i(\tilde{y}_i - \breve{\beta}^* - (\breve{\beta}^*_{(new)} - \breve{\beta}^*))^2 + c(\tilde{\beta}) + \lambda|\breve{\beta}^*_{(new)}| \\
&= P_Q(\breve{\beta}^*) - \sum_{i=1}^{n} \tilde{v}_i(\tilde{y}_i - \breve{\beta}^*)(\breve{\beta}_{(new)} - \breve{\beta}^*) \\
&\qquad + \frac{1}{2}\sum_{i=1}^{n} \tilde{v}_i(\breve{\beta}_{(new)} - \breve{\beta}^*)^2 + \lambda|\breve{\beta}^*_{(new)}| - \lambda|\breve{\beta}^*| \\
&= P_Q(\breve{\beta}^*) + \breve{U}^*(\breve{\beta}^*_{(new)} - \breve{\beta}^*) + \frac{1}{2}\tilde{H}(\breve{\beta}^*_{(new)} - \breve{\beta}^*)^2 + \lambda|\breve{\beta}^*_{(new)}| - \lambda|\breve{\beta}^*| \\
&\leq P_Q(\breve{\beta}^*) + \breve{U}^*(\breve{\beta}^* - \breve{\beta}^*) + \frac{1}{2}\tilde{H}(\breve{\beta}^* - \breve{\beta}^*)^2 + \lambda|\breve{\beta}^*| - \lambda|\breve{\beta}^*| = P_Q(\breve{\beta}^*)
\end{aligned} \tag{4.11}$$

Hence, the iteration scheme converges. Let the unpenalized solution be $\breve{\beta}_{(new)}$, in contrast to the penalized solution $\breve{\beta}^*_{(new)}$. According to the first order condition without the penalty, as in section 4.2.1, we would have $\breve{\beta}_{(new)} = \breve{\beta} - \tilde{H}^{-1}\breve{U}^*$. The solution for equation (4.10) can

be obtained using the first order conditions for three different cases. The first case is when $\breve{\beta}_{(new)} = \breve{\beta} - \tilde{H}^{-1}\breve{U}^* > \tilde{H}^{-1}\lambda > 0$. In this case, locally, the problem would be

$$\arg\min_{\beta} \ell_Q(\breve{\beta}^*) + \breve{U}^*(\beta - \breve{\beta}^*) + \frac{1}{2}\tilde{H}(\beta - \breve{\beta}^*)^2 + \lambda\beta \tag{4.12}$$

Differentiating this objective function with respect to β and setting the expression to zero results in the solution:

$$\breve{\beta}^* - \frac{\breve{U}^*}{\breve{H}} - \frac{\lambda}{\breve{H}} \tag{4.13}$$

The second case is when $\breve{\beta}_{(new)} = \breve{\beta} - \tilde{H}^{-1}\breve{U}^* < -\tilde{H}^{-1}\lambda < 0$. In this case, using a similar argument, the solution becomes

$$\breve{\beta}^* - \frac{\breve{U}^*}{\breve{H}} + \frac{\lambda}{\breve{H}} \tag{4.14}$$

The third case is when $|\breve{\beta}^*_{(new)}| = |\breve{\beta}^* - \tilde{H}^{-1}\breve{U}^*| \leq \tilde{H}^{-1}\lambda$. In this case, we have $\breve{\beta}^*_{(new)} = 0$. To see why, without loss of generality, assume $0 \leq \breve{\beta}^*_{(new)} = \breve{\beta}^* - \tilde{H}^{-1}\breve{U}^* \leq \tilde{H}^{-1}\lambda$. The objective function in equation (4.12) relevant to the solution (excluding constant terms) can be simplified to obtain the following expression:

$$(\breve{U}^* - \breve{\beta}^*\tilde{H} + \lambda)\beta + \frac{1}{2}\tilde{H}\beta^2 \tag{4.15}$$

which is minimized at zero, since the coefficient for β in the first term is positive, and we have assumed that $\breve{\beta}^*_{(new)}$ is positive. The three cases can be expressed compactly in one expression:

$$\breve{\beta}^*_{(new)} = \tilde{H}^{-1}(\tilde{H}\breve{\beta}^* - \breve{U}^*)\left(1 - \frac{\lambda}{|\tilde{H}\breve{\beta}^* - \breve{U}^*|}\right)_+ \tag{4.16}$$

Hence, we repeatedly update the estimate using expression (4.16), updating $\breve{U}^*$ and $\tilde{H}$ along the way. This gives an iteration scheme, which ultimately converges to $\hat{\beta}^*$.

4.2.3 Gamma Generalized Linear Model

The unpenalized negative log-likelihood for the gamma generalized linear model is

$$\ell(\beta_0, \boldsymbol{\beta}) = \sum_{i=1}^{n} v_i \left(y_i \exp\left(-\beta_0 - \boldsymbol{\beta}^T \boldsymbol{x}_i\right) + \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i\right), \tag{4.17}$$

where v_i are weights, y_i are responses, β_0 is an intercept, $\boldsymbol{\beta}$ are coefficients excluding the intercept, and $\boldsymbol{x}_i$ are the explanatory variables for observation i. We are interested in solutions of the form

$$\left(\hat{\beta}_0, \hat{\boldsymbol{\beta}}\right) = \underset{(\beta_0, \boldsymbol{\beta})}{\arg\min}\, \ell\left(\beta_0, \boldsymbol{\beta}\right) + \lambda \sum_{j=1}^{J} w_j \|\boldsymbol{\beta}_j\|_2 \tag{4.18}$$

where $\boldsymbol{\beta} = \left(\boldsymbol{\beta}_1^T, \boldsymbol{\beta}_2^T, \ldots, \boldsymbol{\beta}_J^T\right)^T$ treating the coefficients for each word $j = 1, \ldots, J$ as groups. We use the algorithm presented in Yang & Zou (2015) and Qian et al. (2016) to estimate the gamma regression coefficients with a group lasso penalty. The solution for the Tweedie model with group lasso penalty is discussed in Qian et al. (2016) for the case when the power parameter is less than two. When the power parameter equals two, the model results in the gamma model, yet the derivation for this special case is not explicitly discussed in Qian et al. (2016), so we present it below. The algorithm iteratively updates the estimate of the coefficients given a current estimate. Given current estimates $\tilde{\beta}_0$, $\tilde{\boldsymbol{\beta}}$, the second order Taylor series approximation of the negative log-likelihood around the current estimate is

$$\begin{aligned}
\ell_Q(\beta_0, \boldsymbol{\beta}) &= \ell(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) + \sum_{i=1}^{n} v_i \left(-y_i \exp\left(-\tilde{\beta}_0 - \tilde{\boldsymbol{\beta}}^T \boldsymbol{x}_i\right) + 1\right) \left(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i - \tilde{\beta}_0 - \tilde{\boldsymbol{\beta}}^T \boldsymbol{x}_i\right) \\
&\quad + \frac{1}{2} \sum_{i=1}^{n} v_i y_i \exp\left(-\tilde{\beta}_0 - \tilde{\boldsymbol{\beta}}^T \boldsymbol{x}_i\right) \left(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i - \tilde{\beta}_0 - \tilde{\boldsymbol{\beta}}^T \boldsymbol{x}_i\right)^2 \\
&= \frac{1}{2} \sum_{i=1}^{n} \tilde{v}_i (\tilde{y}_i - \beta_0 - \boldsymbol{\beta}^T \boldsymbol{x}_i)^2 + C(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}),
\end{aligned} \tag{4.19}$$

where

$$\tilde{v}_i = v_i y_i \exp\left(-\tilde{\beta}_0 - \tilde{\boldsymbol{\beta}}^T \boldsymbol{x}_i\right) \tag{4.20}$$

$$\tilde{y}_i = \tilde{\beta}_0 + \tilde{\boldsymbol{\beta}}^T \boldsymbol{x}_i + \frac{v_i}{\tilde{v}_i} (y_i) \tag{4.21}$$

and $C(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$ is a constant given the current estimate. Given $\ell_Q(\beta_0, \boldsymbol{\beta})$, consider updating the current estimates $\breve{\beta}_0, \breve{\boldsymbol{\beta}}$. Then

$$U_j\left(\beta_0, \boldsymbol{\beta}\right) = \frac{\partial \ell_Q\left(\beta_0, \boldsymbol{\beta}\right)}{\partial \boldsymbol{\beta}_j} = -\sum_{i=1}^{n} \tilde{\nu}_i \left(\tilde{y}_i - \beta_0 - \boldsymbol{\beta}^T \boldsymbol{x}_i\right) \boldsymbol{x}_{ij} \tag{4.22}$$

$$U_0\left(\beta_0, \boldsymbol{\beta}\right) = \frac{\partial \ell_Q\left(\beta_0, \boldsymbol{\beta}\right)}{\partial \beta_0} = -\sum_{i=1}^{n} \tilde{\nu}_i \left(\tilde{y}_i - \beta_0 - \boldsymbol{\beta}^T \boldsymbol{x}_i\right) \tag{4.23}$$

$$H_j\left(\beta_0, \boldsymbol{\beta}\right) = \frac{\partial^2 \ell_Q\left(\beta_0, \boldsymbol{\beta}\right)}{\partial \boldsymbol{\beta}_j \boldsymbol{\beta}_j^T} = \sum_{i=1}^{n} \tilde{\nu}_i \boldsymbol{x}_{ij} \boldsymbol{x}_{ij}^T \tag{4.24}$$

$$H_0\left(\beta_0, \boldsymbol{\beta}\right) = \frac{\partial^2 \ell_Q(\beta_0, \boldsymbol{\beta})}{\partial \beta_0 \partial \beta_0^T} = \sum_{i=1}^{n} \tilde{\nu}_i \tag{4.25}$$

With these values, the expression for the new coefficients for the penalized problem in equation (4.18), according to the algorithm in Qian et al. (2016) is

$$\breve{\boldsymbol{\beta}}_j(new) = \frac{\left(\tilde{\gamma}_j \breve{\boldsymbol{\beta}}_j - \breve{U}_j\right)\left(1 - \frac{\lambda w_j}{\|\tilde{\gamma}_j \breve{\boldsymbol{\beta}}_j - \breve{U}_j\|_2}\right)_+}{\tilde{\gamma}_j} \tag{4.26}$$

$$\breve{\boldsymbol{\beta}}_0(new) = \breve{\beta}_0 - \tilde{\gamma}_0^{-1} \breve{U}_0 \tag{4.27}$$

where $\tilde{\gamma}_j$ is the largest eigenvalue of $\breve{H}_j$, and $\tilde{\gamma}_0 = \sum_{i=1}^{n} \tilde{\nu}_i$.

4.2.4 Gamma Double Generalized Linear Model

$$Y_i \sim Gamma(\theta_i, k_i), \quad i = 1, ..., n \tag{4.28}$$

where

$$\mu_i = \theta_i k_i = e^{f_i} \qquad \text{and} \qquad k_i = e^{g_i}. \tag{4.29}$$

For simplicity of the derivation, in this section we assume the design matrix includes a column of ones, corresponding to the intercept. Then, the negative log-likelihood is

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell_i(f_i, g_i) = \sum_{i=1}^{n} \left\{ e^{g_i} \left(y_i e^{-f_i} + f_i\right) + \log \Gamma\left(e^{g_i}\right) - g_i e^{g_i} + \left(1 - e^{g_i}\right) \log y_i \right\} \tag{4.30}$$

where $f_i = \boldsymbol{\beta}^T \boldsymbol{x}_i$, and $g_i = \boldsymbol{\alpha}^T \boldsymbol{x}_i$. Let $\boldsymbol{\theta} = (\boldsymbol{\beta}^T, \boldsymbol{\alpha}^T)^T$. The goal is to solve the following minimization problem

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) + \lambda_\alpha \sum_{j=1}^{J} w_j \left\|\boldsymbol{\alpha}_j\right\|_2 + \lambda_\beta \sum_{j=1}^{J} w_j \left\|\boldsymbol{\beta}_j\right\|_2 \tag{4.31}$$

where $\lambda_\alpha > 0$ and $\lambda_\beta > 0$ are tuning parameters, and w_j are the positive group lasso weights for each group $j = 1, 2, ..., J$. We have $\partial f_i / \partial \boldsymbol{\beta} = \partial g_i / \partial \boldsymbol{\alpha} = \boldsymbol{x}_i$. Define

$$A_i = \frac{\partial^2 \ell_i}{\partial f_i^2} = e^{g_i} y_i e^{-f_i} \tag{4.32}$$

$$B_i = \frac{\partial \ell_i}{\partial f_i} = e^{g_i} \left(-y_i e^{-f_i} + 1\right) \tag{4.33}$$

$$C_i = \frac{\partial^2 \ell_i}{\partial g_i^2} = e^{g_i} \left(y_i e^{-f_i} + f_i\right) + \psi^{(1)}\left(e^{g_i}\right) e^{2g_i} + \psi^{(0)}\left(e^{g_i}\right) e^{g_i} - 2e^{g_i} - g_i e^{g_i} - e^{g_i} \log y_i \tag{4.34}$$

$$D_i = \frac{\partial \ell_i}{\partial g_i} = e^{g_i} \left(y_i e^{-f_i} + f_i\right) + \psi^{(0)}\left(e^{g_i}\right) e^{g_i} - e^{g_i} - g_i e^{g_i} - e^{g_i} \log y_i \tag{4.35}$$

Then, by the chain rule, we have

$$\nabla \ell_i = \begin{bmatrix} B_i \boldsymbol{x}_i \\ D_i \boldsymbol{x}_i \end{bmatrix} \quad \text{and} \quad \nabla^2 \ell_i = \begin{bmatrix} A_i \boldsymbol{x}_i \boldsymbol{x}_i^T & B_i \boldsymbol{x}_i \boldsymbol{x}_i^T \\ B_i \boldsymbol{x}_i \boldsymbol{x}_i^T & C_i \boldsymbol{x}_i \boldsymbol{x}_i^T \end{bmatrix} \tag{4.36}$$

We define $\ell_Q(\boldsymbol{\theta})$ to be the second order Taylor series expansion of $\ell(\boldsymbol{\theta})$ around the current estimate of the parameters $\tilde{\boldsymbol{\theta}}$:

$$\ell_Q(\boldsymbol{\theta}) = \ell(\tilde{\boldsymbol{\theta}}) + \nabla \ell(\tilde{\boldsymbol{\theta}})(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}) + \frac{1}{2}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^T \nabla^2 \ell(\tilde{\boldsymbol{\theta}})(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}) \tag{4.37}$$

Then plug in the expressions for $\nabla \ell(\tilde{\boldsymbol{\theta}})$ and $\nabla^2 \ell(\tilde{\boldsymbol{\theta}})$ in terms of A_i, B_i, C_i, and D_i into the expression for $\ell_Q(\boldsymbol{\theta})$ and simplify. The result is

$$\begin{aligned} \ell_Q(\boldsymbol{\theta}) = \xi(\tilde{\boldsymbol{\theta}}) &+ \frac{1}{2} \sum_{i=1}^{n} \tilde{A}_i (\boldsymbol{x}_i^T \boldsymbol{\beta})^2 + \frac{1}{2} \sum_{i=1}^{n} \tilde{C}_i (\boldsymbol{x}_i^T \boldsymbol{\alpha})^2 \\ &+ \sum_{i=1}^{n} \left\{ \tilde{B}_i \boldsymbol{x}_i^T \boldsymbol{\beta} - \tilde{A}_i (\boldsymbol{x}_i^T \tilde{\boldsymbol{\beta}})(\boldsymbol{x}_i^T \boldsymbol{\beta}) - \tilde{B}_i (\boldsymbol{x}_i^T \tilde{\boldsymbol{\alpha}})(\boldsymbol{x}_i^T \boldsymbol{\beta}) \right\} \\ &+ \sum_{i=1}^{n} \left\{ \tilde{D}_i \boldsymbol{x}_i^T \boldsymbol{\alpha} - \tilde{C}_i (\boldsymbol{x}_i^T \tilde{\boldsymbol{\alpha}})(\boldsymbol{x}_i^T \boldsymbol{\alpha}) - \tilde{B}_i (\boldsymbol{x}_i^T \tilde{\boldsymbol{\beta}})(\boldsymbol{x}_i^T \boldsymbol{\alpha}) \right\} + \sum_{i=1}^{n} \tilde{B}_i (\boldsymbol{x}_i^T \boldsymbol{\beta})(\boldsymbol{x}_i^T \boldsymbol{\alpha}) \end{aligned} \tag{4.38}$$

where $\xi(\tilde{\boldsymbol{\theta}})$ is a constant given $\tilde{\boldsymbol{\theta}}$. Then we would like to minimize the following penalized objective function

$$P_Q(\boldsymbol{\theta}) := \ell_Q(\boldsymbol{\theta}) + \lambda_\alpha \sum_{j=1}^{J} w_j \left\|\boldsymbol{\alpha}_j\right\|_2 + \lambda_\beta \sum_{j=1}^{J} w_j \left\|\boldsymbol{\beta}_j\right\|_2 . \tag{4.39}$$

To minimize 4.39, we utilize a blockwise majorization descent method. Fixing the $\ell_Q(\boldsymbol{\theta})$ function at the point $\tilde{\boldsymbol{\theta}}$, we have

$$\nabla \ell_Q(\boldsymbol{\theta}) = \begin{bmatrix} \partial \ell_Q / \partial \boldsymbol{\beta} \\ \partial \ell_Q / \partial \boldsymbol{\alpha} \end{bmatrix} \tag{4.40}$$

where

$$\begin{aligned} \partial \ell_Q / \partial \boldsymbol{\beta} = & \sum_{i=1}^{n} \tilde{A}_i(\boldsymbol{x}_i^T \boldsymbol{\beta}) \boldsymbol{x}_i + \sum_{i=1}^{n} \tilde{B}_i \boldsymbol{x}_i \\ & - \sum_{i=1}^{n} \tilde{A}_i(\boldsymbol{x}_i^T \tilde{\boldsymbol{\beta}}) \boldsymbol{x}_i - \sum_{i=1}^{n} \tilde{B}_i(\boldsymbol{x}_i^T \tilde{\boldsymbol{\alpha}}) \boldsymbol{x}_i + \sum_{i=1}^{n} \tilde{B}_i(\boldsymbol{x}_i^T \boldsymbol{\alpha}) \boldsymbol{x}_i \end{aligned} \tag{4.41}$$

$$\begin{aligned} \partial \ell_Q / \partial \boldsymbol{\alpha} = & \sum_{i=1}^{n} \tilde{C}_i(\boldsymbol{x}_i^T \boldsymbol{\alpha}) \boldsymbol{x}_i + \sum_{i=1}^{n} \tilde{D}_i \boldsymbol{x}_i \\ & - \sum_{i=1}^{n} \tilde{C}_i(\boldsymbol{x}_i^T \tilde{\boldsymbol{\alpha}}) \boldsymbol{x}_i - \sum_{i=1}^{n} \tilde{B}_i(\boldsymbol{x}_i^T \tilde{\boldsymbol{\beta}}) \boldsymbol{x}_i + \sum_{i=1}^{n} \tilde{B}_i(\boldsymbol{x}_i^T \boldsymbol{\beta}) \boldsymbol{x}_i \end{aligned} \tag{4.42}$$

Also, we have the hessian matrix

$$\nabla^2 \ell_Q(\boldsymbol{\theta}) = \begin{bmatrix} \sum_{i=1}^{n} \tilde{A}_i \boldsymbol{x}_i \boldsymbol{x}_i^T & \sum_{i=1}^{n} \tilde{B}_i \boldsymbol{x}_i \boldsymbol{x}_i^T \\ \sum_{i=1}^{n} \tilde{B}_i \boldsymbol{x}_i \boldsymbol{x}_i^T & \sum_{i=1}^{n} \tilde{C}_i \boldsymbol{x}_i \boldsymbol{x}_i^T \end{bmatrix} \tag{4.43}$$

For a specific group of the parameter, call it $\boldsymbol{\beta}_j$, or $\boldsymbol{\alpha}_j$, we have

$$\begin{aligned} U_j(\boldsymbol{\theta}) = \partial \ell_Q / \partial \boldsymbol{\beta}_j = & \Bigg[\sum_{i=1}^{n} \tilde{A}_i(\boldsymbol{x}_i^T \boldsymbol{\beta}) + \sum_{i=1}^{n} \tilde{B}_i \\ & - \sum_{i=1}^{n} \tilde{A}_i(\boldsymbol{x}_i^T \tilde{\boldsymbol{\beta}}) - \sum_{i=1}^{n} \tilde{B}_i(\boldsymbol{x}_i^T \tilde{\boldsymbol{\alpha}}) + \sum_{i=1}^{n} \tilde{B}_i(\boldsymbol{x}_i^T \boldsymbol{\alpha}) \Bigg] \boldsymbol{x}_{ij} \end{aligned} \tag{4.44}$$

$$\begin{aligned} V_j(\boldsymbol{\theta}) = \partial \ell_Q / \partial \boldsymbol{\alpha}_j = & \Bigg[\sum_{i=1}^{n} \tilde{C}_i(\boldsymbol{x}_i^T \boldsymbol{\alpha}) + \sum_{i=1}^{n} \tilde{D}_i \\ & - \sum_{i=1}^{n} \tilde{C}_i(\boldsymbol{x}_i^T \tilde{\boldsymbol{\alpha}}) - \sum_{i=1}^{n} \tilde{B}_i(\boldsymbol{x}_i^T \tilde{\boldsymbol{\beta}}) + \sum_{i=1}^{n} \tilde{B}_i(\boldsymbol{x}_i^T \boldsymbol{\beta}) \Bigg] \boldsymbol{x}_{ij} \end{aligned} \tag{4.45}$$

and we have the following hessian matrices:

$$\tilde{H}_{\boldsymbol{\beta}_j} = \nabla^2_{\boldsymbol{\beta}_j} \ell_Q = \sum_{i=1}^{n} \tilde{A}_i \boldsymbol{x}_{ij} \boldsymbol{x}_{ij}^T \tag{4.46}$$

$$\tilde{H}_{\boldsymbol{\alpha}_j} = \nabla^2_{\boldsymbol{\alpha}_j} \ell_Q = \sum_{i=1}^{n} \tilde{C}_i \boldsymbol{x}_{ij} \boldsymbol{x}_{ij}^T \tag{4.47}$$

Given $\ell_Q(\boldsymbol{\theta})$, consider updating the current estimates $\breve{\boldsymbol{\beta}}_j$ for a given group j ($1 \leq j \leq J$). Suppose that $\tilde{\boldsymbol{\alpha}}$ and $\breve{\boldsymbol{\beta}}$ are the current estimates, and define $\breve{U}_j = \tilde{U}_j(\breve{\boldsymbol{\beta}})$. Then $\breve{\boldsymbol{\beta}}_j(new)$ is updated by solving

$$\underset{\boldsymbol{\beta}_j}{\arg\min}\ \ell_Q(\breve{\boldsymbol{\beta}}, \tilde{\boldsymbol{\alpha}}) + \breve{U}_j^T(\boldsymbol{\beta_j} - \breve{\boldsymbol{\beta}}_j) + \frac{\tilde{\gamma}_{\beta_j}}{2}(\boldsymbol{\beta_j} - \breve{\boldsymbol{\beta}}_j)^T(\boldsymbol{\beta_j} - \breve{\boldsymbol{\beta}}_j) + \lambda_\beta w_j \left\|\boldsymbol{\beta}_j\right\|_2 . \tag{4.48}$$

Similarly, $\breve{\boldsymbol{\alpha}}_j(new)$ is updated by solving

$$\underset{\boldsymbol{\alpha}_j}{\arg\min}\ \ell_Q(\tilde{\boldsymbol{\beta}}, \breve{\boldsymbol{\alpha}}) + \breve{V}_j^T(\boldsymbol{\alpha_j} - \breve{\boldsymbol{\alpha}}_j) + \frac{\tilde{\gamma}_{\alpha_j}}{2}(\boldsymbol{\alpha_j} - \breve{\boldsymbol{\alpha}}_j)^T(\boldsymbol{\alpha_j} - \breve{\boldsymbol{\alpha}}_j) + \lambda_\alpha w_j \left\|\boldsymbol{\alpha}_j\right\|_2 . \tag{4.49}$$

The expression for the new coefficients for the penalized problem is

$$\breve{\boldsymbol{\beta}}_j(new) = \frac{\left(\tilde{\gamma}_{\boldsymbol{\beta}_j}\breve{\boldsymbol{\beta}}_j - \breve{U}_j\right)\left(1 - \frac{\lambda w_j}{\|\tilde{\gamma}_{\boldsymbol{\beta}_j}\breve{\boldsymbol{\beta}}_j - \breve{U}_j\|_2}\right)_+}{\tilde{\gamma}_{\boldsymbol{\beta}_j}} \tag{4.50}$$

$$\breve{\boldsymbol{\alpha}}_j(new) = \frac{\left(\tilde{\gamma}_{\boldsymbol{\alpha}_j}\breve{\boldsymbol{\alpha}}_j - \breve{V}_j\right)\left(1 - \frac{\lambda w_j}{\|\tilde{\gamma}_{\boldsymbol{\alpha}_j}\breve{\boldsymbol{\alpha}}_j - \breve{V}_j\|_2}\right)_+}{\tilde{\gamma}_{\boldsymbol{\alpha}_j}} \tag{4.51}$$

where $\tilde{\gamma}_{\boldsymbol{\beta}_j}$ is the largest eigenvalue of $\breve{H}_{\boldsymbol{\beta}_j}$, and $\tilde{\gamma}_{\boldsymbol{\alpha}_j}$ is the largest eigenvalue of $\breve{H}_{\boldsymbol{\alpha}_j}$. Algorithm 4.1 summarizes the process of estimating $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by minimizing $P_Q(\boldsymbol{\theta})$.

Algorithm 4.1: Algorithm for solving the Gamma Double GLM

1. Initialize $\tilde{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{\alpha}}$.
2. (Outer Loop) Update the penalized objective function 4.39.
   a) (Inner Loop: $\boldsymbol{\beta}$). Obtain the minimizer of the objective function 4.39.
      - Compute $\tilde{A}_i$, $\tilde{B}_i$, $\tilde{C}_i$, $\tilde{D}_i$ for each $i = 1, 2, ..., n$.
      - Compute $\tilde{H}_{\boldsymbol{\beta}_j}$ and the maximum eigenvalue $\tilde{\gamma}_{\boldsymbol{\beta}_j}$ for $j = 1, 2, ..., J$.
      - Initialize $\breve{\boldsymbol{\beta}} = \tilde{\boldsymbol{\beta}}$.
      - Repeat the following until $\breve{\boldsymbol{\beta}}$ converges.
        - Update $\breve{\boldsymbol{\beta}}$. For $j = 1, 2, ..., J$,
          * Compute $\breve{U}_j = \tilde{U}_j(\breve{\boldsymbol{\beta}})$ by 4.44.
          * Compute $\breve{\boldsymbol{\beta}}_j(new)$ by 4.50.
          * Set $\breve{\boldsymbol{\beta}}_j = \breve{\boldsymbol{\beta}}_j(new)$.
      - Set $\tilde{\boldsymbol{\beta}} = \breve{\boldsymbol{\beta}}$.
   b) (Inner Loop: $\boldsymbol{\alpha}$). Obtain the minimizer of the objective function 4.39.
      - Compute $\tilde{A}_i$, $\tilde{B}_i$, $\tilde{C}_i$, $\tilde{D}_i$ for each $i = 1, 2, ..., n$.
      - Compute $\tilde{H}_{\boldsymbol{\alpha}_j}$ and the maximum eigenvalue $\tilde{\gamma}_{\boldsymbol{\alpha}_j}$ for $j = 1, 2, ..., J$.
      - Initialize $\breve{\boldsymbol{\alpha}} = \tilde{\boldsymbol{\alpha}}$.
      - Repeat the following until $\breve{\boldsymbol{\alpha}}$ converges.
        - Update $\breve{\boldsymbol{\alpha}}$. For $j = 1, 2, ..., J$,
          * Compute $\breve{V}_j = \tilde{V}_j(\breve{\boldsymbol{\alpha}})$ by 4.45.
          * Compute $\breve{\boldsymbol{\alpha}}_j(new)$ by 4.51.
          * Set $\breve{\boldsymbol{\alpha}}_j = \breve{\boldsymbol{\alpha}}_j(new)$.
      - Set $\tilde{\boldsymbol{\alpha}} = \breve{\boldsymbol{\alpha}}$.
3. Repeat until $\tilde{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{\alpha}}$ converges.

4.3 Convexity

In this section, we show the convexity of the problem described in the previous section. We can use an alternative formulation to express the model in equation 4.29. Model the response as

$$Y_i \sim Gamma(\theta_i, k_i), \quad i = 1, ..., n$$

where

$$\theta_i = e^{\tilde{f}_i} \qquad and \qquad k_i = e^{\tilde{g}_i}$$

and

$$\tilde{f}_i = \boldsymbol{b}^T \boldsymbol{x}_i \qquad and \qquad \tilde{g}_i = \boldsymbol{a}^T \boldsymbol{x}_i$$

We have

$$\begin{cases} e^{f_i} = e^{\tilde{f}_i + \tilde{g}_i} \\ e^{g_i} = e^{\tilde{g}_i} \end{cases}$$

and thus

$$\begin{cases} \boldsymbol{\alpha} = \boldsymbol{a} \\ \boldsymbol{\beta} = \boldsymbol{a} + \boldsymbol{b} \end{cases}$$

i.e.,

$$\begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}$$

The negative log-likelihood function of the equivalent formulation is

$$\ell(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} l_i(\tilde{f}_i, \tilde{g}_i) = \frac{1}{n} \sum_{i=1}^{n} \left\{ \log \Gamma(e^{\tilde{g}_i}) + \tilde{f}_i e^{\tilde{g}_i} - (e^{\tilde{g}_i} - 1) \log y_i + y_i e^{-\tilde{f}_i} \right\} \tag{4.52}$$

where $\boldsymbol{\theta} = (\boldsymbol{a}^T, \boldsymbol{b}^T)^T$. For the gradient descent algorithm to reach to the global optimum, the negative log-likelihood function needs to have a global minimum. Therefore, $\ell(\boldsymbol{\theta})$ needs to be a convex function. This function may not be convex in general, because an extreme case of the observations may violate the definition of convexity. However, it is convex with high probability as we increase the sample size.

Definition 4.1 (Asymptotically convex)**.** *We say that a sequence of random functions $f_n(x)$ is* ***asymptotically convex*** *if*

$$\lim_{n\to\infty} \mathbb{P}\left(f_n(x) \text{ is convex}\right) = 1 \tag{4.53}$$

We have, for the loss function above, the following theorem. The theorem guarantees that when we have a large sample size, it's very likely that we observe a convex loss function.

Theorem 4.1. *The loss function 4.52 is asymptotically convex.*

Proof. First, we quantify the variation of y_i and $\log(y_i)$ using law of large numbers. Observe that the gamma distribution has finite expectation and variance and that

$$\mathbb{E}(y_i) = \theta_i k_i, \quad i = 1, ..., n$$

Therefore, the variables $y_i - \theta_i k_i$ have mean zero. By the strong law of large numbers, we have

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \theta_i k_i) \xrightarrow{a.s.} 0 \quad as\ n \to \infty \tag{4.54}$$

Also observe that

$$\mathbb{E}(\log y_i) = \log\theta_i + \psi^{(0)}(k_i)$$

where $\psi^{(0)}(x)$ is the digamma function. Similarly, we have

$$\frac{1}{n}\sum_{i=1}^{n}(\log y_i - \log\theta_i - \psi^{(0)}(k_i)) \xrightarrow{a.s.} 0 \quad as\ n \to \infty \tag{4.55}$$

Equations 4.54 and 4.55 guarantee that the following quantities can be arbitrarily close when n is large

$$\frac{1}{n}\sum_{i=1}^{n} y_i \sim \frac{1}{n}\sum_{i=1}^{n}\theta_i k_i \qquad and \qquad \frac{1}{n}\sum_{i=1}^{n}\log y_i \sim \frac{1}{n}\sum_{i=1}^{n}\log\theta_i + \psi(k_i)$$

Therefore, when n is large, we have

$$\ell(\boldsymbol{\theta}) \approx \frac{1}{n}\sum_{i=1}^{n}\left\{\log\Gamma(e^{\tilde{g}_i}) - \psi^{(0)}(e^{\tilde{g}_i})e^{\tilde{g}_i} + \log y_i + y_i e^{-\tilde{f}_i}\right\} \tag{4.56}$$

By definition, it's easy to show that

- the sum of two convex functions

- the composition of two convex functions

are both convex. By the first rule, it suffices to show that

$$l_i(\boldsymbol{\theta}) = \log\Gamma(e^{\tilde{g}_i}) - \psi^{(0)}(e^{\tilde{g}_i})e^{\tilde{g}_i} + \log y_i + y_i e^{-\tilde{f}_i} \tag{4.57}$$

is convex. Again observe that $\log y_i$ is a constant which is automatically convex. Then observe that $y_i > 0$ since it follows a gamma distribution, thus we have $y_i e^{-\tilde{f}_i}$ is also convex in $\boldsymbol{\theta}$. Therefore, it suffices to show that

$$l_i(\boldsymbol{\theta}) = \log\Gamma(e^{\tilde{g}_i}) - \psi^{(0)}(e^{\tilde{g}_i})e^{\tilde{g}_i} \tag{4.58}$$

is convex. By the second rule, the composition of convex functions is convex. Since $e^{\tilde{g}_i}$ is convex in $\boldsymbol{\theta}$, we only need to show that the function

$$h(u) = \log\Gamma(u) - u\psi^{(0)}(u) \tag{4.59}$$

is convex on $u \in (0, \infty)$, since $u = e^{\tilde{g}_i} > 0$. Denote with $\psi^k(x)$ the $(k+1)^{th}$ derivative of $\log\Gamma(x)$, i.e. the polygamma function or order k, we have

$$h'(u) = -u\psi^{(0)}(u)$$

and

$$h''(u) = -\psi^{(0)}(u) - u\psi^{(1)}(u)$$

Observe that $h''(u)$ is continuous and decreasing, we have

$$h''(u) \geq \lim_{x\to\infty} h''(x) = 0, \quad \forall u \in (0, \infty)$$

Therefore, the function $h(u)$ in equation 4.59 is convex. We have with probability converging to one that the loss function is convex, as $n \to \infty$. □

Remark 4.1. *Since the parameters in the classic formulation is a linear transformation of the parameters in the second formulation, the negative log-likelihood function of the classic formulation is also asymptotically convex. It doesn't matter which formulation we use, we may easily obtain the parameters of another formulation by performing the linear transformation.*

4.3.1 The Asymptotic Convexity

We draw a random design matrix $\boldsymbol{x}$ of size $n \times p$ from a standard multivariate normal distribution with independent entries. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are also drawn from standard multivariate normal distributions. The $\boldsymbol{y}$ is generated according to the gamma distribution. Then we randomly draw $N = 10000$ pairs of $\boldsymbol{\theta}_1 = (\boldsymbol{\alpha}_1, \boldsymbol{\beta}_1)$ and $\boldsymbol{\theta}_2 = (\boldsymbol{\alpha}_2, \boldsymbol{\beta}_2)$ and check the convexity by definition

$$\lambda \ell(\boldsymbol{\theta}_1) + (1 - \lambda)\ell(\boldsymbol{\theta}_2) \geq \ell(\lambda \boldsymbol{\theta}_1 + (1 - \lambda)\boldsymbol{\theta}_2) \tag{4.60}$$

where λ is randomly drawn from uniform distribution between 0 and 1. We fix $p = 10$ and repeat the simulation for $n = 1, 5, 10, 20, 50, 100$. The proportion of pairs of points that violates the convexity definition is given in Table 4.1. The simulation shows that as the sample size increases, there's less chance to see a non-convex region. Moreover, the sample size needs not be too big to reach this good property.

Table 4.1: Proportion of randomly sampled pairs of points that violate the convexity definition for different sample sizes, each over 5 repetitions. The standard error of the 5 repetitions are given in parenthesis.

n	1	5	10	20	100
Proportion	0.1210	0.0231	0.0048	0.0004	0.0000
Standard Error	(0.0012)	(0.0009)	(0.0003)	(0.0001)	(0.0000)

4.4 Simulation

In the simulation study, we investigate the performance of the method using two examples. The following models are investigated in the simulation study,

- *Double Group Lasso* - The mean and dispersion are modeled by group lasso
- *Group Lasso* - The mean is modeled by group lasso with only an intercept term for modeling the dispersion. This is achieved by choosing λ_α to be large.
- *Double Lasso* - The mean and dispersion are modeled by lasso, no group structure is assumed in the model.

- *Lasso* - The mean is modeled by lasso with only an intercept term for modeling the dispersion, no group structure is assumed in the model.

All models assume the response follows a Gamma distribution. In both examples, λ_α and λ_β are selected using the Bayesian information criterion (BIC).

4.4.1 Example 1

In each run of the simulation, 500 observations and 200 observations are used for the training and testing datasets, respectively. To create the design matrix, we will construct four blocks, each with three covariates. The lth block, $\boldsymbol{B}_l$ ($l = 1, ..., 4$), is sampled from a multivariate normal distribution with mean $\mathbf{0}$ and variance $\boldsymbol{\Sigma}$. For $k, j = 1, 2, 3$, we set $\boldsymbol{\Sigma}_{kj} = 1$ when $k = j$ and $\boldsymbol{\Sigma}_{kj} = \omega$ when $k \neq j$, where $\omega = 0$ or 0.5. The design matrix is constructed as $\boldsymbol{X} = (\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{B}_3, \boldsymbol{B}_4)$. For $i = 1, ..., n$, y_i is sampled from a Gamma distribution with scale parameter θ_i and shape parameter k_i, where

$$\log(\mu_i) = \log(\theta_i k_i) = 1 + \sum_{j=1}^{3} (-1)^{j+1} (\boldsymbol{B}_{1ij} + \boldsymbol{B}_{2ij}) \tag{4.61}$$

$$\log(k_i) = -1 + \sum_{j=1}^{3} (-1)^{j} (0.5\boldsymbol{B}_{1ij} + 0.5\boldsymbol{B}_{3ij}) \tag{4.62}$$

where $\boldsymbol{B}_{lij}$ is the ith observation for the jth covariate from the lth block. In the construction of the response, $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are the only active blocks in calculating the mean, and $\boldsymbol{B}_1$ and $\boldsymbol{B}_3$ are the only active blocks in calculating the dispersion. In the simulation study, we are interested in how well the method selects the true nonzero blocks. We will consider a block active in the model if at least one of the coefficients in the block are nonzero. We will quantify the performance by counting the number of correctly active blocks (C) along with the number of incorrectly active blocks (IC).

Table 4.2 summarizes the average simulation results for the Lasso model and the Double Lasso model. The Double Lasso model performs better than the Lasso model with respect to the log-likelihood and the Gini index calculated on the testing dataset. While the Double Lasso model and

Table 4.2: Average simulation results for 100 runs for the Lasso and Double Lasso models. The standard error is provided in the parentheses.

	Dispersion		Mean		BIC	log-likelihood	Gini Index
	C	IC	C	IC			
$\omega = 0$							
Double Lasso	2	1.17	2	1.25	-119.86 (28.92)	12.61 (6.12)	0.863 (0.005)
Lasso	—	—	2	1.6	260.76 (26.96)	-28.34 (6.04)	0.852 (0.005)
$\omega = 0.5$							
Double Lasso	2	1.52	2	1.44	398.54 (35.63)	-27.08 (8.46)	0.791 (0.009)
Lasso	—	—	2	1.76	659.07 (32.29)	-57.77 (7.55)	0.787 (0.008)

Table 4.3: Average simulation results for 100 runs for the Grouped Lasso and Double Grouped Lasso models. The standard error is provided in the parentheses.

	Dispersion		Mean		BIC	log-likelihood	Gini Index
	C	IC	C	IC			
$\omega = 0$							
Double Grouped Lasso	2	0.58	2	0.18	-115.23 (28.84)	11.62 (6.15)	0.853 (0.005)
Grouped Lasso	—	—	2	1.07	263.39 (27.06)	-28.1 (6.06)	0.854 (0.006)
$\omega = 0.5$							
Double Grouped Lasso	2	0.8	2	1.04	400.07 (37.65)	-28.09 (8.5)	0.792 (0.009)
Grouped Lasso	—	—	2	1.18	665.39 (32.67)	-58.4 (7.61)	0.781 (0.011)

the Lasso model both appropriately estimate $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ as nonzero when modeling the mean, the Double Lasso model incorrectly estimates nonzero blocks less often than the Lasso model.

Table 4.3 summarizes the average simulation results for the Grouped Lasso model and the Double Grouped Lasso model. The Double Grouped Lasso model performs better than the Lasso group model with respect to the log-likelihood and the Gini index calculated on the testing dataset. While the Double Grouped Lasso model and the Grouped Lasso model both appropriately estimate $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ as nonzero when modeling the mean, the Double Grouped Lasso model incorrectly estimates nonzero blocks less often than the Lasso model.

In general, the Double Lasso and Double Grouped Lasso methods outperform the Lasso and Grouped Lasso methods, respectively. In addition, by modeling for the dispersion, the model for the mean more accurately reflects the true relationship. That is, it is more likely that $\boldsymbol{B}_3$ and $\boldsymbol{B}_4$ will be incorrectly included in the model for the mean by not modeling for dispersion.

4.4.2 Example 2

Consider a scenario where we have some variable X_1 that is binary, with 0 indicating a good driver and 1 indicating a bad driver. We want to estimate the premium Y_i for each driver $i = 1, ..., 1000$. Assume further that we have 9 additional highly correlated explanatory variables that do not affect our response. Formally,

$$y_i \sim Gamma(\theta_i, k_i), \quad i = 1, ..., n$$

where

$$\mu_i = \theta_i k_i = \exp\{\log(100) + \log(100) X_{1i}\} \qquad \text{and} \qquad k_i = \exp\{\log(1) + \log(10) X_{1i}\}.$$

We compare the results of two different models; Double Lasso and Lasso. The data are simulated 1000 times. Table 4.4 shows the proportion of replications where the α and β coefficients related to X_1 and X_2 (noise) are nonzero. In general, the Double Lasso model appropriately shrinks the coefficients related to X_2 while accurately estimating the coefficients related to X_1.

Table 4.4: Proportion of replications that the coefficient is nonzero in the Double Lasso model.

	X_1	X_2
α	0.9941 (0.0042)	0.0445 (0.0113)
β	0.9941 (0.0042)	0.0653 (0.0135)

The value at risk (VaR) is also estimated for each simulation and the resulting distributions are shown in figure 4.1 and figure 4.2. The plots on the left represent the 95% VaR for good drivers ($X_1 = 0$) while the 95% VaR for bad drivers ($X_1 = 1$) are shown in the plots on the right. The vertical lines represent the true 95% VaR in each case. Since the Double Lasso method models the dispersion as a function of X_1, the estimated 95% VaR is much closer to the true value as compared to the Lasso model. The Lasso model underestimates the 95% VaR for good drivers while overestimating the 95% VaR for bad drivers. Since the risk capital is underestimated for good drivers, there is a risk of insolvency. Since the risk captial is overestimated for bad drivers, it is likely that the company will lose business from bad drivers.

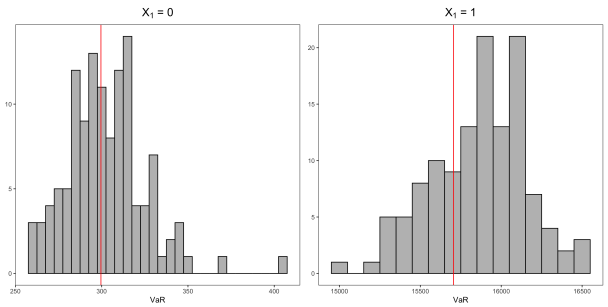

Figure 4.1: 95% VaR for Double Lasso model

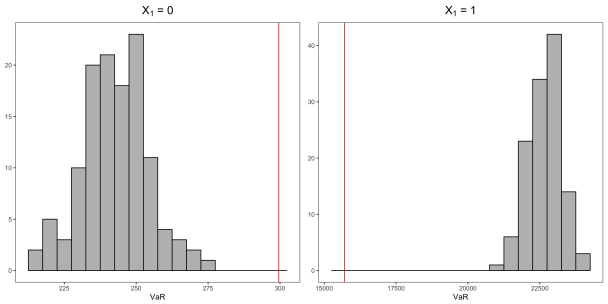

Figure 4.2: 95% VaR for Lasso model

4.5 Empirical Analysis

4.5.1 Data

For our case study, we obtained data from the Office of the Commissioner of Insurance (OCI) of Wisconsin. The dataset contains claim descriptions with loss amounts from the Wisconsin Local Government Property Insurance Fund (LGPIF). Table 4.5 is a summary of the frequency and severity of loss amounts for 6030 events recorded in the LGPIF dataset. For the analysis, we will focus only on claims in the Vandalism event type. The resulting 2084 events have an average property damage amount estimate of $2,695. The maximum loss amount in the sample was caused by vandalism to a pool that resulted in a loss of $981,598.

Figure 4.3 gives the reader a better idea of the distribution of words found in the event narratives of the events. The counts of each word is shown for the four most common event types. The reader may observe that the word vandalism and the word glass are common in descriptions for Vandalism events, whereas in the descriptions for Lightning, *light*, *pole*, and *hit* are common words. Given

Table 4.5: Summary of loss amounts by event type

Event type	Frequency	Severity
Vandalism	2084	2,695
Vehicle	1079	4,274
Lightning	955	11,156
Misc	465	24,846
WaterW	464	78,071
Wind	403	25,606
WaterNW	269	33,081
Fire	217	82,187
Hail	94	137,480
Total	6030	19,674

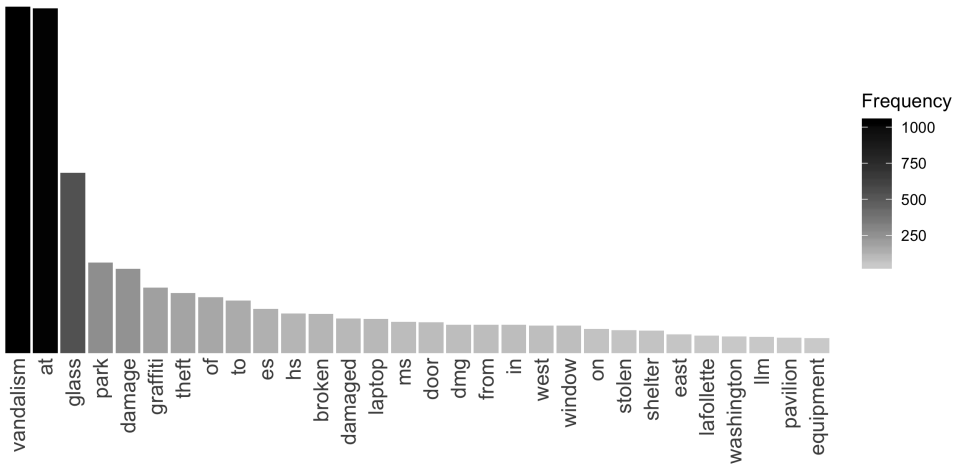

Figure 4.3: Frequency of words in descriptions of Vandalism events

that different event types have different average severities, as implied by Table 4.5, we may infer that certain keywords may be indicative of the magnitude of property losses.

4.5.2 Methods

By using cosine similarities, we can extract features from textual descriptions of losses. For example, if we consider the word *vandalism*, the cosine similarities will give a value between -1 and 1, increasing in the word's relationship with the observed sentence. If we consider multiple words, then we may obtain multiple explanatory variables, which can be used for a multivariate regression model. The question is, which words should be used for extracting the explanatory

variables? Previous work has focused on using words carefully selected by a human expert. This approach worked for short textual descriptions of insurance losses. However, as the descriptions become longer, we may require a method that is more clever. One possible approach would be to try every possible word in the English language, and then use variable selection methods to reduce the model to one that includes only significant variables. Another possible approach would be to try only the words observed in the descriptions within the dataset. We use the latter approach in this case study.

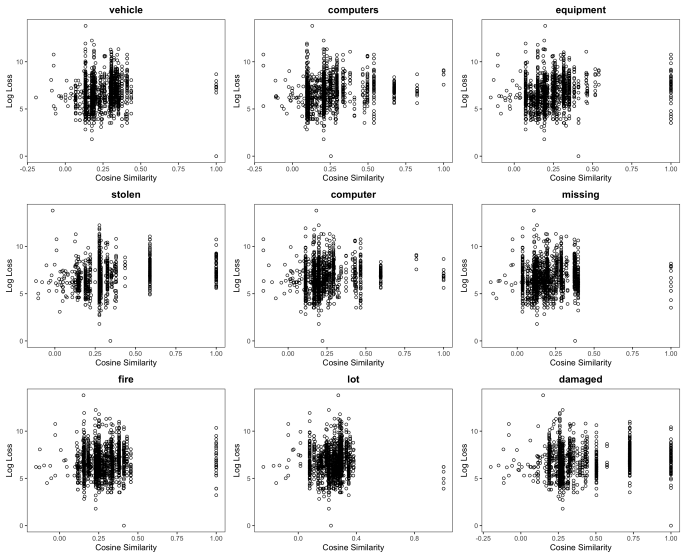

Figure 4.4: Cosine similarities of selected words with log loss amounts for Vandalism

Figure 4.4 shows a plot of the cosine similarities for selected words with the corresponding loss amounts in log scale. The spearman correlations with the loss amounts for the nine words are shown in Table 4.6. Keywords shown in Table 4.6 may be the words that are indicative of large losses. Similarly, those words with small correlation may be words indicative of small losses. Figure 4.5 illustrates the effect of the existence of selected words in the narrative. The plot illustrates that, for instance, the occurrence of the word *caused* increases the average loss amount of the event by a significant amount.

We construct the design matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^T$ using the cosine similarities. In order to do this, consider the matrix of basis functions $\boldsymbol{\Phi}_j = \left(\boldsymbol{\phi}_{1,j}, \boldsymbol{\phi}_{2,j}, \ldots, \boldsymbol{\phi}_{n,j}\right)^T$, where $\boldsymbol{\phi}_{i,j} = \left(P_1(u_{i,j}), P_2(u_{i,j}), P_3(u_{i,j})\right)^T$ with $u_{i,j} = \text{sim}_{\cos}(\boldsymbol{a}, d)$ for description d_i and unique words a_j.

Table 4.6: Spearman correlation of cosine similarities and loss amounts

vehicle	computers	equipment	stolen	computer	missing	fire	lot	damaged
0.334	0.321	0.29	0.285	0.282	0.214	0.148	0.142	0.137

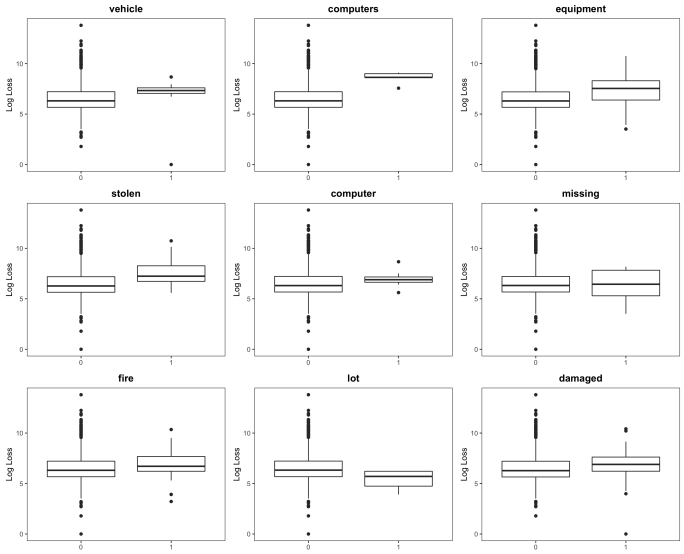

Figure 4.5: Effect of word indicator on loss amount

Here, $P_k(u)$ are cubic penalized regression spline (P-spline) basis functions with penalty order 1. For an overview of the P-spline basis function, the reader may refer to Wood (2017). Then, from the QR decomposition $\boldsymbol{\Phi}_j^T \mathbf{1} = \boldsymbol{Q}_{j,1}\boldsymbol{R}_j + \boldsymbol{Q}_{j,2}\mathbf{0}$ take $\boldsymbol{Q}_{j,2}$ and form the design matrix $\boldsymbol{X} = \left(\boldsymbol{\Phi}_1\boldsymbol{Q}_{1,2}, \boldsymbol{\Phi}_2\boldsymbol{Q}_{2,2}, \ldots, \boldsymbol{\Phi}_J\boldsymbol{Q}_{J,2}\right)$. Under this framework, the response variable is related to the covariate $u_{i,j}$ via a smooth function

$$f_j(u_{i,j}) = \boldsymbol{\beta}_j^T \boldsymbol{x}_{ij}, \tag{4.63}$$

where $\boldsymbol{\beta}_j$ is the coefficient corresponding to the j-th word describing observation i. This ensures that the basis functions satisfy $\sum_{i=1}^{n} f_j(u_{i,j}) = 0$ for each $j = 1, \ldots, J$.

4.5.3 Results

Figure 4.6 shows the cross-sectional solution path of the algorithm for three selected words. The reader may verify that group Lasso is applied to both the mean and the dispersion of the gamma

distribution. Notice that the three lines corresponding to a single word converge to zero at the same tuning parameter value.

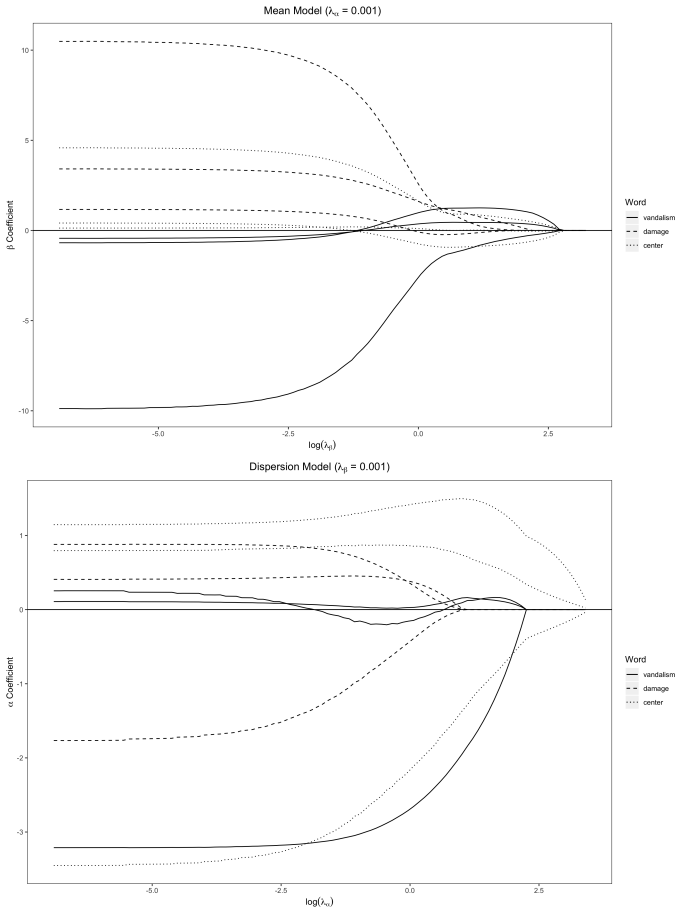

Figure 4.6: Solution path of the algorithm

Ten-fold cross validation using the Spearman correlation of the predicted loss amount is used in order to determine the tuning parameters for the model. The tuning parameters for the final model are $\lambda_\alpha = 0.7040$ and $\lambda_\beta = 1.0822$. The final model contained 172 words for modeling the mean and 130 words for modeling the dispersion. Figures 4.7 and 4.8 show the curve estimates for 9 selected words for modeling the mean and the dispersion, respectively. Note that the curve estimates for the dispersion associated with *computer* and *damaged* are zero. Looking back at figure 4.4, we can see how the model has captured the change in the mean and variance of the loss amount as the cosine similarity changes.

The Spearman correlation with the validation sample loss amounts and the predicted loss

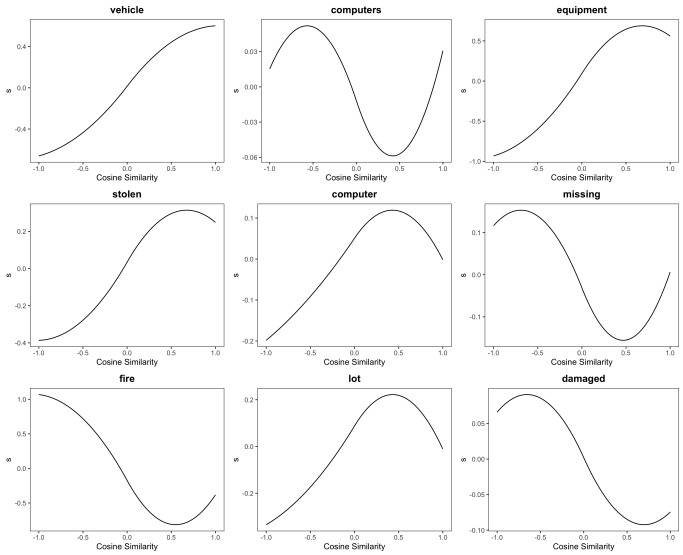

Figure 4.7: β curve estimates

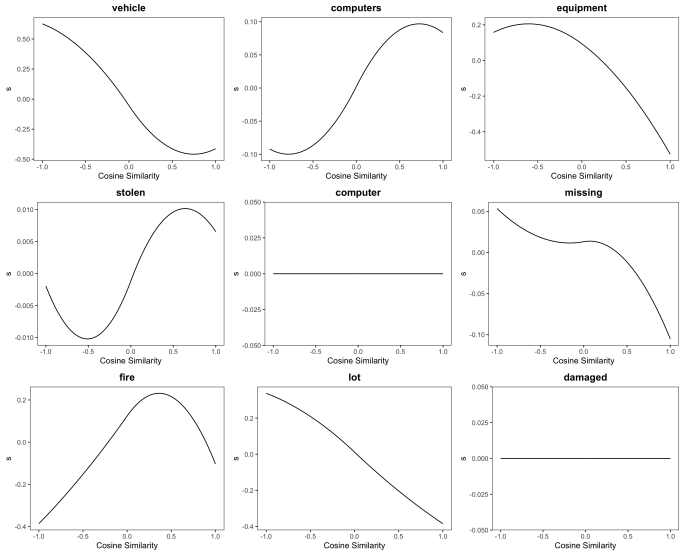

Figure 4.8: α curve estimates

amounts using the model turns out to be 48.68%, while the Spearman correlation for the training sample was 66.38%. A plot of the predicted losses versus the actual losses in the training and validation samples are shown in log scale in the figure 4.9.

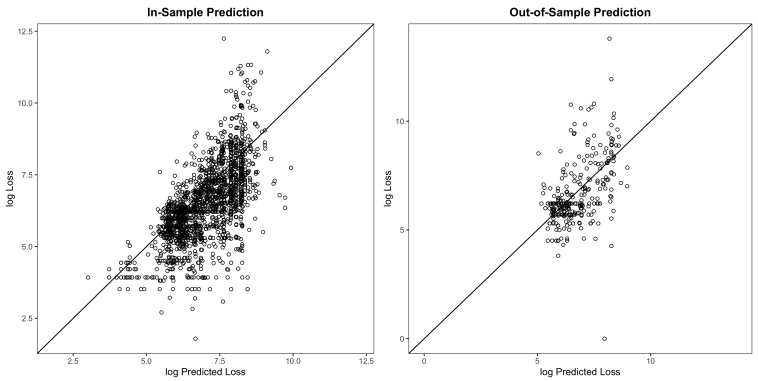

Figure 4.9: Prediction versus out-of-sample claims (log scale)

4.6 Concluding Remarks

In this paper, we explored the use of the group Lasso technique to predict non-normal loss amounts based on covariates derived from textual descriptions of the losses. Our contribution to the literature is the extension of the group Lasso technique to the case where the mean and the dispersion are both modeled with a penalty term, as well as the application of the methodology in the textual data analysis context. The approach has applications in the insurance case reserving problem, and has promising results according to the real data analysis performed using a training sample and a validation sample.

Limitations to our approach may be the fact that a vector representation of words should be available in order for the described approach to work. The methodology may be vulnerable to spelling errors in the descriptions of the losses. However, for the presented dataset, spelling errors were minimal and did not turn out to be a problem in terms of the prediction results. The methodology may also suffer from descriptions with a large number of abbreviations, as is sometimes the case with insurance claim adjuster notes. Dealing with such messy textual descriptions may be potential future work.

CHAPTER 5

CONCLUSION

In this paper, we introduced a framework for incorporating textual data into insurance claims modeling, and considered its applications in claims management processes. An insurance claim representative is responsible for investigating the claim, in order to determine the handling process. In this paper, we explored the use of word similarities as a tool for modeling insurance claims and mitigating insurance risks.

In Chapter 2, we illustrated how short textual descriptions can be leveraged in the GAM framework for insurance claim classification and risk mitigation. The method relies on the subject matter expert to choose the words most impactful to the response.

In Chapter 3, we generalize the previous work to longer textual descriptions without the need for an expert. When we consider all unique words found in the dataset as explanatory variables and impose a GAM, the resulting design matrix is high-dimensional. For this reason, we used a group Lasso penalty to reduce the number of coefficients in the model. The scalable, analytical framework proposed provides for a parsimonious and interpretable model. We discussed the implications of the analysis, including how the framework may be used by an insurance company.

In Chapter 4, we showed how we can incorporate a group Lasso type penalty in both the dispersion and the mean parameterization for a Gamma model, and illustrate its use in a predictive analytics application in actuarial science. In particular, we applied the method to an insurance claim prediction problem involving textual data analysis methods. Simulations illustrated the variable selection and model fitting performance of our method.

Our results demonstrate how text mining technology can be incorporated into a traditional regression analysis. The methodology is applicable in many different areas of applications, where textual data arises. Possible applications of our approach for an insurance risk manager may include:

- Classification of claims based on textual descriptions of the claims
- Classification of policyholders based on textual descriptions of the policyholders
- Prediction of insurance claims at the claim level
- Prediction of insurance claims at the policyholder level
- Analysis of insurance claims and risk mitigation

We make some remarks on the current limitations of our framework, where potential improvements can be made.

- Under the current framework, words not found in the word embedding matrix cannot be used in the modeling.
- The threshold ε is selected using heuristics by a human expert, under the current framework.
- Because pre-determined word embedding matrices are limited to one-grams (single words) at the time the paper is being written, the incorporation of n-grams (use of phrases longer than one word as a search key) remains an open question.
- Further linguistic barriers may exist, if the textual descriptions are longer than those appearing in the dataset used for this paper. Examples may be polysemy, false friends, compound words.
- In order to use the proposed method, insurers that focus on specific insurance segments may be constrained to build its own word embedding matrices, as the terms appearing in the claim descriptions may be specific to the field. For example, a medical insurer may find GloVe insufficient, and may need a word embedding matrix trained on medical terms in order to use our proposed approach.

APPENDICES

APPENDIX A

THREE STAGE APPROACH THEORETICAL RESULTS

In this section, we will provide statistical foundation for the proposed three stage approach. For this reason, we derive the convergence rate for our 3rd-stage estimator. This will establish statistical consistency of our procedure. In Yang & Maiti (2018), the following result for the second stage estimator has been established.

Lemma A.1 (K. Yang and T. Maiti, 2018)**.** *The adaptive group lasso consistently selects the true active predictors in probability, i.e., the estimator* $\hat{\boldsymbol{\beta}}_{AGL}$ *satisfies:*

$$\mathbb{P}\left(\|\hat{f}_{AGLj}(x)\|_2 > 0,\ j \in T \text{ and } \|\hat{f}_{AGLj}(x)\|_2 = 0,\ j \in T^c\right) \to 1. \tag{A.1}$$

The results states that with proper choices of λ_{n1} and λ_{n2}, the adaptive group lasso consistently selects the true nonzero predictors. This theorem guarantees the selection consistency of the 3-stage algorithm, since the variable selection is done in the second stage and the third stage does not do variable selection. It's important for an algorithm to select the correct subset of variables for the model built on them to work.

With similar assumptions, assume we have

Assumption 1. *The true functions* $f_1, ..., f_{s_n}$ *has smoothness order* o_n*, i.e.*

$$\int_a^b f_j''(x)^2 dx \asymp o_n$$

where $a_n \asymp b_n$ *means there exist constants* c *and* d *such that*

$$c \leq \frac{a_n}{b_n} \leq d$$

Then, we have

Theorem A.1. *Under assumptions 1 and assumptions in Yang & Maiti (2018), for tuning parameters* $\lambda_{n31}, ...\lambda_{n3s_n}$*, we have*

$$\|\hat{f}_{sm} - f^0_{sm}\|_2^2 = O_p\left(s_n\gamma_2^{-2s_n}m_n\frac{\log(s_nm_n)}{n}\right) + O_P(s_n^2\gamma_2^{-2s_n}m_n^{-2d}) + O_P\left(\sum_{j\in\hat{S}_n}\lambda_{n3j}o_n\right) \quad \text{(A.2)}$$

where γ_0 *and* γ_2 *are assumed bounds parameters in eigenvalues of* $\boldsymbol{X}$*, see Yang & Maiti (2018).*

Theorem A.1 shows the rate of convergence of the third stage estimator. There are three terms in the convergence rate: the estimation error, the spline approximation error and the regularization error. The greater o_n, the less λ_{n3}, thus the product won't change. This theorem guarantees that with proper choice of parameters, the estimated functions are consistent estimators of the true functions that describe the relationship between the variables and the response.

Proof. Consider the third step, where we have the smoothness penalty. Define the event

$$\mathcal{S}_n = \{\hat{S}_n = S\}$$

The previous lemma showed that

$$\mathbb{P}(\mathcal{S}_n) \to 1 \quad as\ n \to \infty$$

From now on, let's condition on the event $\mathcal{S}_n$. For convenience, we suppress the notations $\hat{\boldsymbol{\beta}}_{sm}$, $\boldsymbol{\beta}^0_{sm}$ and $\Phi^{\hat{S}_n}$ and denote them with $\hat{\boldsymbol{\beta}}$, $\boldsymbol{\beta}^0$ and Φ.

To study the characteristics of the smoothness term $\boldsymbol{S}_j$, where

$$\int_a^b f_j''(x)^2dx = \int_a^b \phi''(x)\phi''(x)^T dx = \boldsymbol{\beta}_j^T\boldsymbol{S}_j\boldsymbol{\beta}$$

without loss of generality, consider the case that the knots are evenly distributed on the interval $[a, b]$, since changing the length of the intervals does not change the shape of the B-splines but the span and height (Schumaker, 1981). In the following calculations, we normalize the interval $[a, b]$ to $[0, Kl_n]$, where each interval has length l_n. According to Huang et al. (2010), assume

the constant length of the interval satisfies $l_n = O(n^{-\nu})$ with $0 < \nu < 0.5$. The k^{th} cubic B-spline basis can be derived from definition

$$B_{k,4}(x) = \begin{cases} \dfrac{x^3}{6l_n^3} - \dfrac{kx^2}{2l_n^2} + \dfrac{k^2x}{2l_n} - \dfrac{k^3}{6}, & l_nk \le x \le l_n(k+1), \\ \dfrac{-3x^3}{6l_n^3} + \dfrac{(9k+10)x^2}{6l_n^2} - \dfrac{7k^2+16k+6}{6l_n} + \dfrac{k^3+2k^2-2k-2}{6}, & \\ & l_n(k+1) \le x \le l_n(k+2), \\ \dfrac{3x^3}{6l_n^3} - \dfrac{(9k+20)x^2}{6l_n^2} + \dfrac{9k^2+42k+34)x}{6l_n} - \dfrac{k^3+8k^2+14k+10}{6}, & \\ & l_n(k+2) \le x \le l_n(k+3), \\ \dfrac{-x^3}{6l_n^3} + \dfrac{(k+2)x^2}{6l_n^2} - \dfrac{3k^2+20k+32}{6l_n} + \dfrac{k^3+10k^2+32k+32}{6}, & \\ & l_n(k+3) \le x \le l_n(k+4), \\ 0, \quad o.w. \end{cases}$$

and we have $\phi(x) = \{B_{k,4}(x), k = 1, ..., m_n\}$. Taking derivative, we have the second derivative of the basis function satisfies

$$B''_{k,4}(x) = O(l_n^{-2}) = O(n^{2\nu})$$

Therefore, the elements

$$s_{j,ik} = O(n^{3\nu}) \quad for \quad j = 1, ..., p \quad and \quad i, k = 1, ..., m_n \quad where \quad s_{j,ik} \in \boldsymbol{S}_j$$

and equals exactly zero if $|i - k| > 3$. As a direct result, the eigenvalue of the matrix $\boldsymbol{S}_j$ is bounded from above by $O(n^{3\nu})$ and from below by some constant. Similarly, if we use a quadratic B-spline, the elements $s_{j,ik}$ are bounded from above by $O(n^{4\nu})$ and from below by some constant.

Then we begin the convergence rate part. For a converging sequence N_n such that $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2 \le N_n$, define $t = N_n/(N_n + \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2)$, then consider the convex combination $\boldsymbol{\beta}^* = t\hat{\boldsymbol{\beta}} + (1-t)\boldsymbol{\beta}^0$. We have $\boldsymbol{\beta}^* - \boldsymbol{\beta}^0 = t(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)$, which implies

$$\|\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\|_2 = t\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2 = \frac{N_n\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2}{N_n + \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2} \le N_n \tag{A.3}$$

This means $\boldsymbol{\beta}^*$ is within a small distance from $\boldsymbol{\beta}^0$ and we are safe to use Taylor expansion. Moreover, if we have

$$\|\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\|_2 \leq R_n,$$

then

$$\frac{N_n}{N_n + \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2} \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2 \leq R_n$$

Choosing N_n to be greater than R_n, we have

$$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2 \leq 2R_n$$

Therefore, it's sufficient to derive the convergence rate for $\boldsymbol{\beta}^*$.

Consider the Taylor expansion

$$\begin{aligned}
&-\frac{1}{n}\sum_{i=1}^{n}\left[y_i\left(\boldsymbol{\beta}^{*T}\Phi_i\right) - b\left(\boldsymbol{\beta}^{*T}\Phi_i\right)\right] \\
=&-\frac{1}{n}\sum_{i=1}^{n}\left[y_i\left(\boldsymbol{\beta}^{0T}\Phi_i\right) - b\left(\boldsymbol{\beta}^{0T}\Phi_i\right)\right] - \left(\frac{1}{n}\sum_{i=1}^{n}\left[y_i\Phi_i - b'\left(\boldsymbol{\beta}^{0T}\Phi_i\right)\Phi_i\right]\right)^T(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) \\
&+\frac{1}{2n}\sum_{i=1}^{n}(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T\Phi_i^T b''\left(\boldsymbol{\beta}^{**}\Phi_i\right)\Phi_i(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) \\
=:&-\frac{1}{n}\sum_{i=1}^{n}\left[y_i\left(\boldsymbol{\beta}^{0T}\Phi_i\right) - b\left(\boldsymbol{\beta}^{0T}\Phi_i\right)\right] - \frac{1}{n}(\boldsymbol{y} - \boldsymbol{\mu}^0)^T\Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) \\
&+\frac{1}{2n}(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T\Phi^T\Sigma(\boldsymbol{\beta}^{**})\Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)
\end{aligned}$$

where $\boldsymbol{\mu}^0$ is the expectation of $\boldsymbol{y}$ at $\boldsymbol{\beta}^0$ and $\Sigma(\boldsymbol{\beta}^{**})$ is the covariance matrix of $\boldsymbol{y}$ evaluated ast $\boldsymbol{\beta}^{**}$ which is located on the line segment joining $\boldsymbol{\beta}^0$ and $\boldsymbol{\beta}^*$.

By the definition of $\boldsymbol{\beta}^*$ and convexity, we have

$$\begin{aligned}
&-\frac{1}{n}\sum_{i=1}^{n}\left[y_i\left(\boldsymbol{\beta}^{*T}\Phi_i\right) - b\left(\boldsymbol{\beta}^{*T}\Phi_i\right)\right] + \sum_{j\in\hat{S}_n}\lambda_{n3j}\boldsymbol{\beta}_j^{*T}\boldsymbol{D}_j\boldsymbol{\beta}_j^* \\
\leq&-\frac{1}{n}\sum_{i=1}^{n}\left[y_i\left(\boldsymbol{\beta}^{0T}\Phi_i\right) - b\left(\boldsymbol{\beta}^{0T}\Phi_i\right)\right] + \sum_{j\in\hat{S}_n}\lambda_{n3j}\boldsymbol{\beta}_j^{0T}\boldsymbol{D}_j\boldsymbol{\beta}_j^0
\end{aligned}$$

Combine this with the Taylor expansion result, we have

$$
\begin{aligned}
&\frac{1}{2n}(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \Sigma(\boldsymbol{\beta}^{**}) \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) \\
\leq &\frac{1}{n}(\boldsymbol{y} - \boldsymbol{\mu}^0)^T \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) + \sum_{j \in \hat{S}_n} \lambda_{n3j} \left[\boldsymbol{\beta}_j^{0T} \boldsymbol{D}_j \boldsymbol{\beta}_j^0 - \boldsymbol{\beta}_j^{*T} \boldsymbol{D}_j \boldsymbol{\beta}_j^* \right] \\
\leq &\frac{1}{n} |(\boldsymbol{y} - \boldsymbol{\mu})^T \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)| + \frac{1}{n} |(\boldsymbol{\mu}^0 - \boldsymbol{\mu})^T \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)| + \sum_{j \in \hat{S}_n} \lambda_{n3j} \left[\boldsymbol{\beta}_j^{0T} \boldsymbol{D}_j \boldsymbol{\beta}_j^0 - \boldsymbol{\beta}_j^{*T} \boldsymbol{D}_j \boldsymbol{\beta}_j^* \right] \\
\leq &\frac{1}{n} |(\boldsymbol{y} - \boldsymbol{\mu})^T \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)| + \frac{1}{4n}(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \Sigma(\boldsymbol{\beta}^{**}) \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) + O(s_n^2 m_n^{-2d}) \\
&+ \sum_{j \in \hat{S}_n} \lambda_{n3j} \left[\boldsymbol{\beta}_j^{0T} \boldsymbol{D}_j \boldsymbol{\beta}_j^0 - \boldsymbol{\beta}_j^{*T} \boldsymbol{D}_j \boldsymbol{\beta}_j^* \right]
\end{aligned}
$$

where the second inequality comes from norm inequality, the third inequality comes from Cauchy-Swarchz inequality, and μ is the expectation of y given f^0. Rearrangling the inequality, we have

$$
\begin{aligned}
&\frac{1}{4n}(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \Sigma(\boldsymbol{\beta}^{**}) \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) \\
\leq &\frac{1}{n} |(\boldsymbol{y} - \boldsymbol{\mu})^T \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)| + O(s_n^2 m_n^{-2d}) + \sum_{j \in \hat{S}_n} \lambda_{n3j} \boldsymbol{\beta}_j^{0T} \boldsymbol{D}_j \boldsymbol{\beta}_j^0 \\
\leq &\frac{1}{8n}(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \Sigma(\boldsymbol{\beta}^{**}) \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) + \frac{2}{n} \|\Sigma^{1/2}(\boldsymbol{\beta}^{**})(\boldsymbol{y} - \mu)\|_2^2 \\
&+ \sum_{j \in \hat{S}_n} \lambda_{n3j} \boldsymbol{\beta}_j^{0T} \boldsymbol{D}_j \boldsymbol{\beta}_j^0 + O(s_n^2 m_n^{-2d})
\end{aligned}
$$

where the inequality is by Cauchy-Swarchz inequality. Consider the penalty matrix $\boldsymbol{D}_j$ who has entries $d_{j,ik} = 1$ if $i = k = 1$, $d_{j,ik} = 2$ if $i = k \neq 1$ and $d_{j,ik} = -1$ if $|i - k| = 1$. The matrix is a constant matrix, thus each $\boldsymbol{\beta}_j^{0T} \boldsymbol{D}_j \boldsymbol{\beta}_j^0$ is of the order $O(o_n)$. Rearranging the terms and by Concentration inequality, see for example Yang & Maiti (2018), we have

$$
\frac{1}{8n}(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \Sigma(\boldsymbol{\beta}^{**}) \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) = O\left(s_n m_n \frac{\log(s_n m_n)}{n} \right) + O\Big(\sum_{j \in \hat{S}_n} \lambda_{n3j} o_n \Big) + O(s_n^2 m_n^{-2d})
$$

By remark 2.1 in Yang & Maiti (2018), we have

$$
\frac{\gamma_0 c_1 \gamma_2^{2s_n}}{8 m_n} \|\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\|_2^2 \leq \frac{1}{8n}(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \Sigma(\boldsymbol{\beta}^{**}) \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)
$$

Combine the above result with the condition event $\mathcal{S}_n$, we have conditioning on $\mathcal{S}_n$

$$\|\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\|_2^2 = O_P\left(s_n \gamma_2^{-2s_n} m_n \frac{\log(s_n m_n)}{n}\right) + O(s_n^2 \gamma_2^{-2s_n} m_n^{-2d}) + O\left(\sum_{j \in \hat{S}_n} \lambda_{n3j} o_n\right)$$

By the inequality that

$$P(A) = P(A|B)P(B) + P(A|B^C)P(B^C) \leq P(A|B) + P(B^C)$$

consider $\mathcal{S}_n = B^C$, we have

$$\|\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\|_2^2 = O_p\left(s_n \gamma_2^{-2s_n} m_n \frac{\log(s_n m_n)}{n}\right) + O_P(s_n^2 \gamma_2^{-2s_n} m_n^{-2d}) + O_P\left(\sum_{j \in \hat{S}_n} \lambda_{n3j} o_n\right)$$

Combine this with the argument at the beginning of the proof, we have

$$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2^2 = O_p\left(s_n \gamma_2^{-2s_n} m_n \frac{\log(s_n m_n)}{n}\right) + O_P(s_n^2 \gamma_2^{-2s_n} m_n^{-2d}) + O_P\left(\sum_{j \in \hat{S}_n} \lambda_{n3j} o_n\right)$$

□

APPENDIX B

THREE STAGE APPROACH SIMULATION RESULTS

In this section, we outline the process of constructing the simulated datasets along with the results of our method.

B.1 Data

Several simulated datasets were constructed using various numbers of observations and covariates. For a prespecified number of observations, x_j is randomly generated from $U(0, 1)$ for each $j = 1, ..., q$. In addition, for each $j = 1, ..., q$, f_j is selected from a series of functions; zero, linear, polynomial, exponential, logarithmic, and sinusoidal. The functions are selected randomly based on chosen probabilities; 0.6, 0.05, 0.2, 0.05, 0.05, and 0.05. The coefficients used in each function are also randomly selected. The exact functions used in the simulation study can be found in the appendix.

B.2 Simulation Results

The performance of our method is summarized through three characteristics; out-of-sample prediction, function misclassification, and computational performance. In addition, performance results for other models are included for comparison. These models include results from intermediate steps such as *Initial Group Lasso* and *Adaptive Group Lasso*, as well as other models such as *True Selected*, the GAM model fit to only nonzero functions, and *mgcv*, the GAM model fit to all functions using the `mgcv` package.

Out-of-sample prediction is measured by mean squared prediction error (MSPE), with a training to testing set ratio of 80:20. Figure B.1 shows the comparison of the out-of-sample prediction performance for each method when $n = 800$ and $q = 200$. In addition to the aforementioned comparison models, *Null Model* and the true σ^2 were added. This plot is fairly representative of all plots constructed for each simulated sample. The MSPE from *Step 3* decreases as λ decreases, but

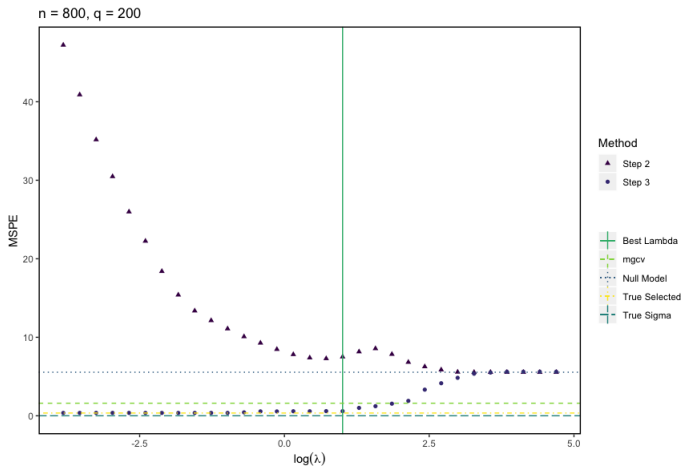

Figure B.1: Mean squared prediction error for each method when X has 800 observations and 200 covariates.

may slightly increases as λ becomes smallest. For the best λ, the MSPE of *step 3* is typically very similar to that of *True Selected* and *True Sigma*, with the MSPE of *mgcv* being relatively close all well. Note that the MSPE of *step 2* is higher than that of the null model, which is expected due to the overfitting nature of the model. This further emphasizes the importance of *step 3*. Figure B.2 shows the performance of the 3 step approach against the other methods, for a varying number of observations and covariates. These plots simply omit *step 2* for increased readability.

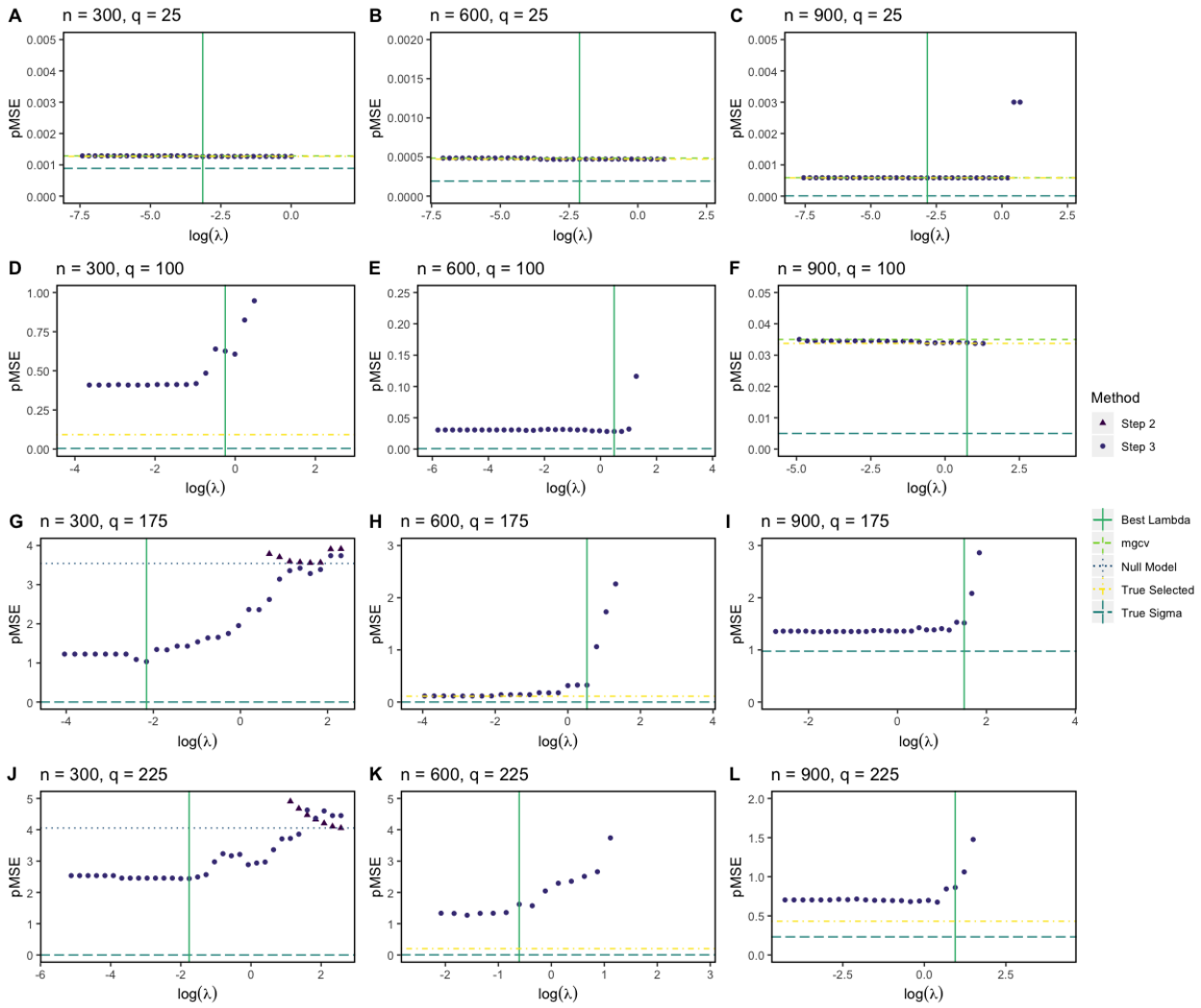

Figure B.2: Mean squared prediction error from the 3 step approach against other methods, at various dimensions of X.

We now consider the four misclassification types; *zero*, the percentage of truly zero functions estimated as nonzero functions, *nonzero*, the percentage of truly nonzero functions estimated as zero functions, *linear*, the percentage of truly linear functions estimated as nonlinear or zero functions, and *nonlinear*, the percentage of truly nonlinear functions estimated as linear or zero functions. Note nonlinear functions do not include zero functions. Figure B.3 shows the misclassification rates the for *step 2*, *step 3*, and the *gamsel* model.

Moreover, figures B.4 and B.5 show the estimated functions for the simulated dataset used in figure B.1 from the 3 step approach and the *mgcv* model, respectively. The 3 step approach provides

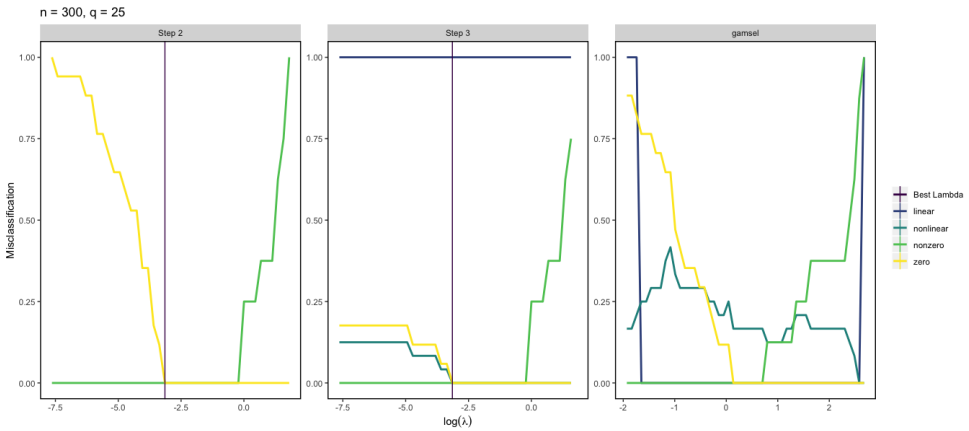

Figure B.3: Misclassification error rate for *step 2*, *step 3*, and *gamsel* model, for each misclassification type.

function estimates very similar to the true functions, with confidence bands that accurately capture the observations. The *mgcv* model does not perform as well, and has much wider confidence bands.

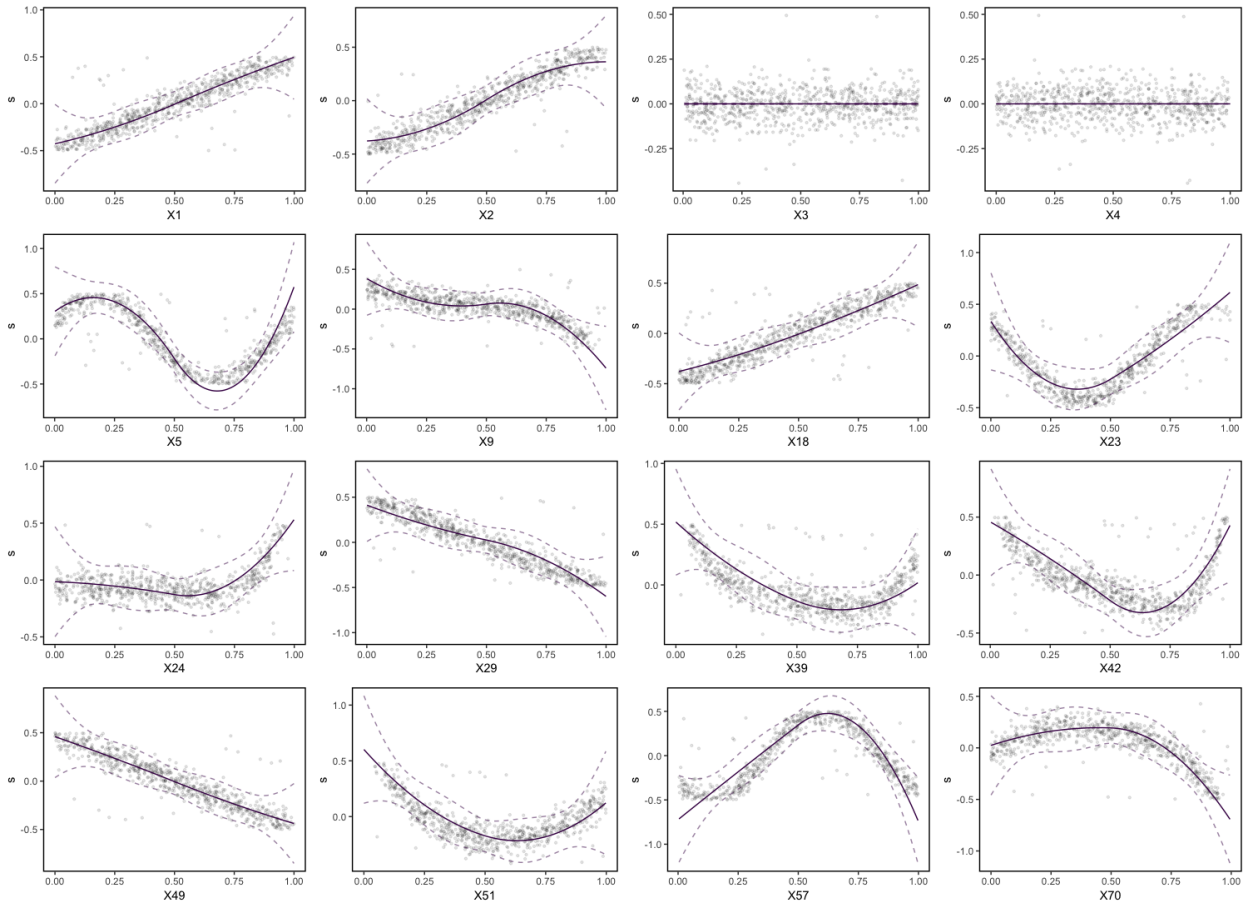

Figure B.4: 3 step approach function estimates for representative covariates from the model corresponding to figure B.1.

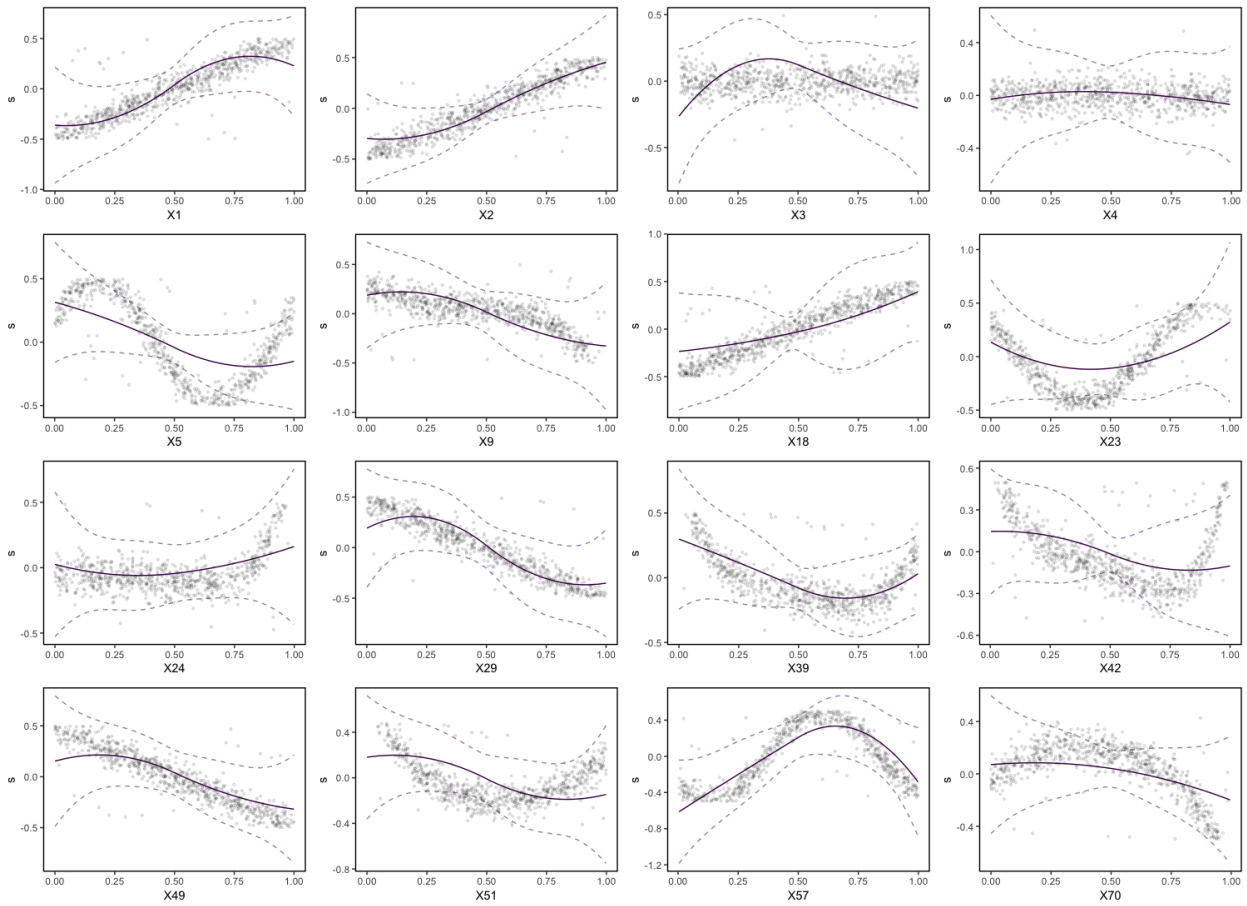

Figure B.5: *mgcv* function estimates for representative covariates from the model corresponding to figure B.1.

Finally, we consider method runtime as a measure of computational performance. Figure B.7 shows the runtime, in seconds, of each model for the various simulated datasets. The *mgcv* model has a shorter runtime than the *3 step method* at the best lambda in subplots A, B, C, and D. However, as the dimension of the dataset becomes larger, the *3 step method* at the best lambda runs faster. In figure B.7, the *mgcv* model had a runtime of about 12.55 hours compared to about 1.12 hours for the *3 step method* at the best lambda.

The performance of the final model with respect to the three characteristics discussed is directly affected by the choice of λ in the adaptive group lasso step. In figure B.1, smaller values of lambda provide for reduced mean squared prediction error. However, in figure B.7, smaller values of lambda generally lead to longer runtimes. These observations emphasize the importance of selecting and appropriate lambda in the adaptive lasso step.

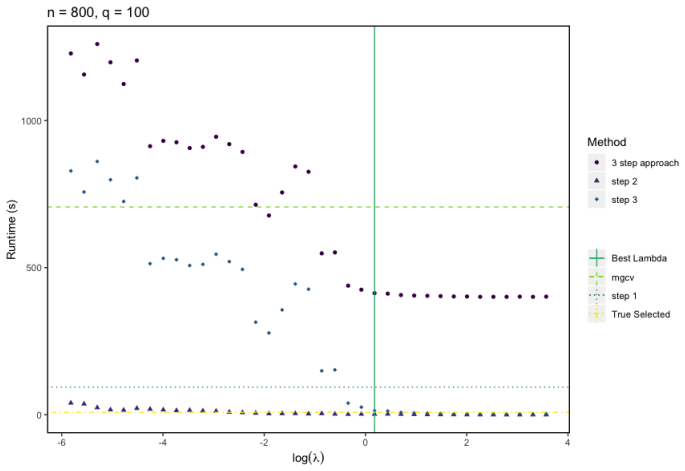

Figure B.6: Runtime, in seconds, for each method when X has 800 observations and 100 covariates.

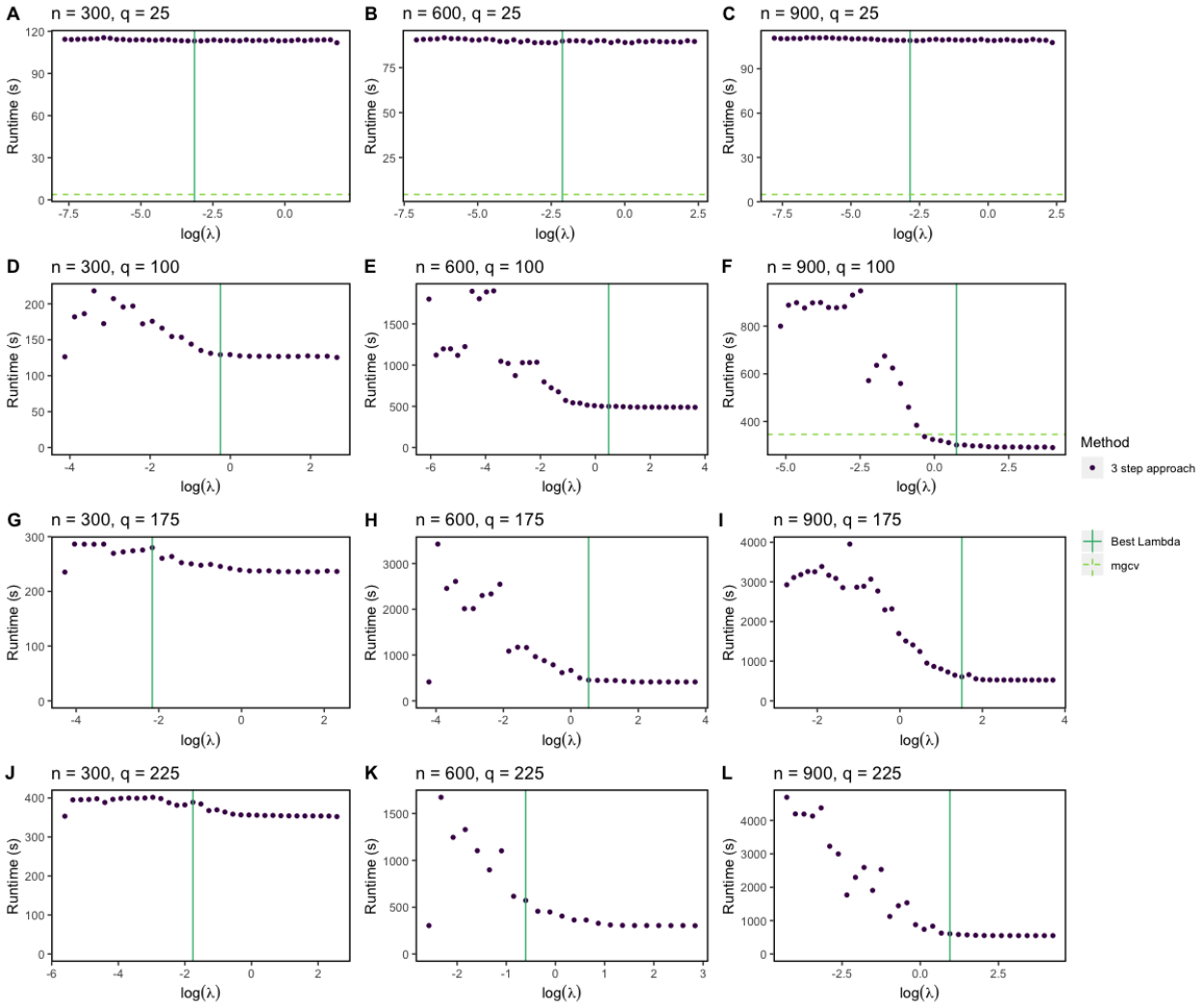

Figure B.7: Runtime, in seconds, from the 3 step approach the *mgcv* model, at various dimensions of X.

APPENDIX C

GAMMA DOUBLE GLM WITH LASSO ALGORITHM RESULTS

We want to justify that the update schemes in Equation 4.48 and Equation 4.49 by showing that the penalized objective function in Equation 4.39 decreases with each update.

Lemma C.1. *Let δ be a column vector of length n, let H a positive definite $n \times n$ matrix, and let γ be the largest eigenvalue of H. Then*

$$\delta^T H\delta \leq \gamma \|\delta\|_2^2 .$$

Proof. Letting $\nu = \delta/\|\delta\|_2$, it is equivalent to show

$$\nu^T H\nu \leq \gamma.$$

Let eigendecomposition

$$H = UDU^T,$$

where $U = (u_1^T \cdots u_n^T)^T$ is orthonormal and D is a diagonal matrix with $\lambda_1, ..., \lambda_n$ on the diagonals. The columns of U form an orthonormal basis and are the eigenvectors corresponding to $\lambda_1, ..., \lambda_n$. Therefore, we have $c_1, ..., c_n$ such that $\nu = \sum_{i=1}^n c_i u_i$. Since $\|\nu\|_2 = 1$, we have $\sum_{i=1}^n c_i^2 = 1$.

Moreover,

$$\begin{aligned}\nu^T H\nu &= \langle \nu, H\nu \rangle \\ &= \left\langle \sum_{i=1}^n c_i u_i, H \sum_{i=1}^n c_i u_i \right\rangle .\end{aligned}$$

By the definition of eigenvalue, $Hu_i = \lambda_i u_i$ for $i = 1, ..., n$. Then

$$\begin{aligned}\nu^T H\nu &= \left\langle \sum_{i=1}^n c_i u_i, H \sum_{i=1}^n c_i u_i \right\rangle \\ &= \sum_{i=1}^n c_i^2 \lambda_i \\ &\leq \left(\max_{i=1,...,n} \lambda_i \right) \sum_{i=1}^n c_i^2 = \gamma.\end{aligned}$$

The inequality holds since H is positive definite resulting in all λ_i, $i = 1, ..., n$ being positive. □

Result C.1.

$$\ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}(new)) \leq \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) + \breve{\boldsymbol{U}}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)^T + \frac{\tilde{\gamma}_j}{2}(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))$$

Proof.

$$\begin{aligned}
\ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}(new)) &= \xi(\tilde{\boldsymbol{\theta}}) + \frac{1}{2}\sum_{i=1}^{n}\tilde{A}_i(\boldsymbol{x}_i^T\breve{\boldsymbol{\beta}}(new))^2 + \frac{1}{2}\sum_{i=1}^{n}\tilde{C}_i(\boldsymbol{x}_i^T\breve{\boldsymbol{\alpha}})^2 \\
&\quad + \sum_{i=1}^{n}\left\{\tilde{B}_i\boldsymbol{x}_i^T\breve{\boldsymbol{\beta}}(new) - \tilde{A}_i(\boldsymbol{x}_i^T\tilde{\boldsymbol{\beta}})(\boldsymbol{x}_i^T\breve{\boldsymbol{\beta}}(new)) - \tilde{B}_i(\boldsymbol{x}_i^T\tilde{\boldsymbol{\alpha}})(\boldsymbol{x}_i^T\breve{\boldsymbol{\beta}}(new))\right\} \\
&\quad + \sum_{i=1}^{n}\left\{\tilde{D}_i\boldsymbol{x}_i^T\breve{\boldsymbol{\alpha}} - \tilde{C}_i(\boldsymbol{x}_i^T\tilde{\boldsymbol{\alpha}})(\boldsymbol{x}_i^T\breve{\boldsymbol{\alpha}}) - \tilde{B}_i(\boldsymbol{x}_i^T\tilde{\boldsymbol{\beta}})(\boldsymbol{x}_i^T\breve{\boldsymbol{\alpha}})\right\} \\
&\quad + \sum_{i=1}^{n}\tilde{B}_i(\boldsymbol{x}_i^T\breve{\boldsymbol{\beta}}(new))(\boldsymbol{x}_i^T\breve{\boldsymbol{\alpha}}) \\
&= \xi(\tilde{\boldsymbol{\theta}}) + \frac{1}{2}\sum_{i=1}^{n}\tilde{A}_i(\boldsymbol{x}_i^T\breve{\boldsymbol{\beta}} + x_{ij}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))^2 + \frac{1}{2}\sum_{i=1}^{n}\tilde{C}_i(\boldsymbol{x}_i^T\breve{\boldsymbol{\alpha}})^2 \\
&\quad + \sum_{i=1}^{n}\tilde{B}_i(\boldsymbol{x}_i^T\breve{\boldsymbol{\beta}} + x_{ij}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)) \\
&\quad - \tilde{A}_i(\boldsymbol{x}_i^T\tilde{\boldsymbol{\beta}})(\boldsymbol{x}_i^T\breve{\boldsymbol{\beta}} + x_{ij}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)) - \tilde{B}_i(\boldsymbol{x}_i^T\tilde{\boldsymbol{\alpha}})(\boldsymbol{x}_i^T\breve{\boldsymbol{\beta}} + x_{ij}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)) \\
&\quad + \sum_{i=1}^{n}\left\{\tilde{D}_i\boldsymbol{x}_i^T\breve{\boldsymbol{\alpha}} - \tilde{C}_i(\boldsymbol{x}_i^T\tilde{\boldsymbol{\alpha}})(\boldsymbol{x}_i^T\breve{\boldsymbol{\alpha}}) - \tilde{B}_i(\boldsymbol{x}_i^T\tilde{\boldsymbol{\beta}})(\boldsymbol{x}_i^T\breve{\boldsymbol{\alpha}})\right\} \\
&\quad + \sum_{i=1}^{n}\tilde{B}_i(\boldsymbol{x}_i^T\breve{\boldsymbol{\beta}} + x_{ij}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))(\boldsymbol{x}_i^T\breve{\boldsymbol{\alpha}}) \\
&= \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) + \sum_{i=1}^{n}\tilde{A}_i\left[x_{ij}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)\breve{\boldsymbol{\beta}}^T\boldsymbol{x}_i + \frac{1}{2}(x_{ij}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))^2\right] \\
&\quad + \sum_{i=1}^{n}\tilde{B}_i x_{ij}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j) \\
&\quad - \tilde{A}_i(\boldsymbol{x}_i^T\tilde{\boldsymbol{\beta}})(x_{ij}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)) - \tilde{B}_i(\boldsymbol{x}_i^T\tilde{\boldsymbol{\alpha}})(x_{ij}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)) \\
&\quad + \sum_{i=1}^{n}\tilde{B}_i(x_{ij}^T(\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))(\boldsymbol{x}_i^T\breve{\boldsymbol{\alpha}})
\end{aligned}$$

$$
\begin{aligned}
\ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}(new)) &= \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) + \sum_{i=1}^{n} \tilde{A}_i(\boldsymbol{x}_i^T \breve{\boldsymbol{\beta}}) x_{ij} (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)^T + \frac{1}{2} \tilde{A}_i (x_{ij}^T (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))^2 \\
&\quad + \sum_{i=1}^{n} \tilde{B}_i x_{ij} (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)^T \\
&\quad - \tilde{A}_i (\boldsymbol{x}_i^T \tilde{\boldsymbol{\beta}}) x_{ij} (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)^T - \tilde{B}_i (\boldsymbol{x}_i^T \tilde{\boldsymbol{\alpha}}) x_{ij} (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)^T \\
&\quad + \sum_{i=1}^{n} \tilde{B}_i (\boldsymbol{x}_i^T \breve{\boldsymbol{\alpha}}) x_{ij} (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)^T \\
&= \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) + \breve{\boldsymbol{U}}^T (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)^T + \frac{1}{2} (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))^T \tilde{H} (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)) \\
&\leq \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) + \breve{\boldsymbol{U}}^T (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)^T + \frac{\tilde{\gamma}_j}{2} (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))^T (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))
\end{aligned}
$$

The last inequality holds by Lemma C.1. □

Lemma C.2. *Each iteration of the inner loops in Algorithm 4.1 always decreases penalized objective function in Equation 4.39.*

Proof. It is sufficient to show that the update schemes in Equation 4.48 and Equation 4.49 always decreases the penalized objective function in Equation 4.39. Without loss of generality, fix $j \in \{1, ..., J\}$ and let us consider the update of $\boldsymbol{\beta}_j$. Then we simply need to show that

$$
P_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}(new)) - P_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) \leq 0.
$$

Then,

$$
\begin{aligned}
&P_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}(new)) - P_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) \\
&\quad = \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}(new)) + \lambda_\alpha \sum_{k=1}^{J} w_k \left\|\breve{\boldsymbol{\alpha}}_k\right\|_2 + \lambda_\beta \sum_{k=1}^{J} w_k \left\|\breve{\boldsymbol{\beta}}_k(new)\right\|_2 \\
&\qquad - \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) - \lambda_\alpha \sum_{k=1}^{J} w_k \left\|\breve{\boldsymbol{\alpha}}_k\right\|_2 - \lambda_\beta \sum_{k=1}^{J} w_k \left\|\breve{\boldsymbol{\beta}}_k\right\|_2 \\
&\quad = \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}(new)) + \lambda_\beta w_j \left\|\breve{\boldsymbol{\beta}}_j(new)\right\|_2 \\
&\qquad - \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) - \lambda_\beta w_j \left\|\breve{\boldsymbol{\beta}}_j\right\|_2 \\
&\quad \leq \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) + \breve{\boldsymbol{U}}^T (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)^T \\
&\qquad + \frac{\tilde{\gamma}_j}{2} (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))^T (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)) + \lambda_\beta w_j \left\|\breve{\boldsymbol{\beta}}_j(new)\right\|_2 \\
&\qquad - \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) - \lambda_\beta w_j \left\|\breve{\boldsymbol{\beta}}_j\right\|_2 \qquad \text{by Result C.1} \\
&\quad = \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) + \breve{\boldsymbol{U}}^T (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)^T \\
&\qquad + \frac{\tilde{\gamma}_j}{2} (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j))^T (\breve{\boldsymbol{\beta}}_j(new) - \breve{\boldsymbol{\beta}}_j)) + \lambda_\beta w_j \left\|\breve{\boldsymbol{\beta}}_j(new)\right\|_2 \\
&\qquad - \ell_Q(\breve{\boldsymbol{\alpha}}, \breve{\boldsymbol{\beta}}) - \breve{\boldsymbol{U}}^T (\breve{\boldsymbol{\beta}}_j - \breve{\boldsymbol{\beta}}_j)^T \\
&\qquad - \frac{\tilde{\gamma}_j}{2} (\breve{\boldsymbol{\beta}}_j - \breve{\boldsymbol{\beta}}_j))^T (\breve{\boldsymbol{\beta}}_j - \breve{\boldsymbol{\beta}}_j)) - \lambda_\beta w_j \left\|\breve{\boldsymbol{\beta}}_j\right\|_2 \\
&\quad \leq 0.
\end{aligned}
$$

The final inequality holds by Equation 4.48. □

BIBLIOGRAPHY

BIBLIOGRAPHY

Antweiler, Werner & Murray Frank. 2004. Is all that talk just noise? the information content of internet stock message boards. *The Journal of Finance* 59.

Chollet, Francois & J. J. Allaire. 2018. *Deep learning with r*. Manning Publications Co.

Chouldechova, Alexandra & Trevor Hastie. 2015. Generalized additive model selection. *arXiv preprint, arXiv:1506.03850v2 [stat.ML]* .

Dreassi, Emanuela, M Giovanna Ranalli & Nicola Salvati. 2014. Semiparametric m-quantile regression for count data. *Statistical Methods in Medical Research* 23(6). 591–610. doi:10.1177/0962280214536636. https://doi.org/10.1177/0962280214536636. PMID: 24847899.

Frees, Edward W. 2009. *Regression modeling with actuarial and financial applications*. Cambridge University Press.

Frees, Edward W., Gee Y. Lee & Lu Yang. 2016. Multivariate frequency-severity regression models in insurance. *Risks* 2016(4).

Friedman, Jerome, Trevor Hastie & Robert Tibshirani. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software* 33(1). 1–22.

Gentzkow, Matthew, Jesse Shapiro & Matt Taddy. 2016. Measuring polarization in high-dimensional data: method and application to congressional speech. *NBER Working Paper No. 22423* .

Goldberg, Yoav. 2017. *Neural network methods for natural language processing*. Morgan & Claypool Publishers.

Goodfellow, Ian, Yoshua Bengio & Aaron Courville. 2016. *Deep learning*. MIT Press.

Hastie, T. & R. Tibshirani. 1986. Generalized additive models. *Statistical science* 297–310.

Hastie, T. J. & R. J. Tibshirani. 1990. *Generalized additive models*. Chapman and Hall.

Hastie, Trevor, Robert Tibshirani & Jerome Friedman. 2009. *The elements of statistical learning: Data mining, inference, and prediction, second edition*. Springer Science & Business Media.

Hatzivassiloglou, Vasileios & Janyce Wiebe. 2000. Effects of adjective orientation and gradability on sentence subjectivity. *Proceedings of COLING* .

Huang, J., L. Horowitz & F. Wei. 2010. Variable selection in nonparametric additive models. *Annals of Statistics* 38(4). 2282 – 2313.

Karlgren, Jussi & Douglass Cutting. 1994. Recognizing text genres with simple metrics using discriminant analysis. *Proceedings of COLING* .

Kearney, Susan. 2010. *Insurance operations*. The Institutes.

Lee, Gee Y., Scott Manski & Tapabrata Maiti. 2019. Actuarial applications of word embedding models. *ASTIN Bulletin* DOI: https://doi.org/10.1017/asb.2019.28.

Li, Zheyuan & Simon N. Wood. 2019. Faster model matrix crossproducts for large generalized linear models with discretized covariates. *Statistics and Computing* doi:10.1007/s11222-019-09864-2. https://doi.org/10.1007/s11222-019-09864-2.

Marra, Giampiero & Simon Wood. 2011. Practical variable selection for generalized additive models. *Computational Statistics and Data Analysis* 55. 2372–2387.

Pang, Bo, Lillian Lee & Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)* .

Pennington, Jeffrey, Richard Socher & Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical methods in natural language processing (emnlp)*, 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Qian, Wei, Yi Yang & Hui Zou. 2016. Tweedie's compound poisson model with grouped elastic net. *Journal of Computational and Graphical Statistics* 25(2). 606–625.

Schumaker, L. 1981. *Spline functions: Basic theory*. John Wiley & Sonds, New York.

Simonoff, Jeffrey S. 1996. *Smoothing methods in statistics*. Springer, New York, NY.

Smyth, Gordon K. 1989. Generalized linear models with varying dispersion. *Journal of the Royal Statistical Society, Series B (Methodological)* 51(1). 47–60.

Smyth, Gordon K. & Bent Jørgensen. 2002. Fitting tweedie's compound poisson model to insurance claims data: dispersion modelling. *ASTIN Bulletin* 32(1). 143–157.

Sokolova, Marina & Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing and Management* 45. 427–437.

Stephens-Davidowitz, Seth. 2014. The cost of racial animus on a black candidate: Evidence using google search data. *Journal of Public Economics* 118. 26–40.

Tetlock, Paul. 2007. Giving content to investor sentiment: the role of media in the stock market. *The Journal of Finance* 62.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 267–288.

Wang, Hansheng & Chenlei Leng. 2008. A note on adaptive group lasso. *Computational Statistics & Data Analysis* 52(12). 5277 – 5286.

Wood, Simon. 2013. On p-values for smooth components of an extended generalized additive model. *Biometrika* 100. 221–228.

Wood, Simon. 2019. *mgcv: Mixed gam computation vehicle with automatic smoothness estimation*. `https://CRAN.R-project.org/package=mgcv`. R package version 1.8-31.

Wood, Simon N. 2017. *Generalized additive models: An introduction with r, second edition*. CRC Press.

Wood, Simon N., Yannig Goude & Simon Shaw. 2015. Generalized additive models for large data sets. *Journal of the Royal Statistical Society, Series C* 64(1). 139–155.

Wood, Simon N., Zheyuan Li, Gavin Shaddick & Nicole H. Augustin. 2017. Generalized additive models for gigadata: Modeling the u.k. black smoke network daily data. *Journal of the American Statistical Association* 112(519). 1199–1210.

Yang, Kaixu & Taps Maiti. 2018. Ultra high dimensional classification consisitency with generalized additive model. *Technical Reports, Michigan State University* .

Yang, Yi & Hui Zou. 2015. A fast unified algorithm for solving group-lasso penalize learning problems. *Statistics and Computing* 25(6). 1129–1141.

Yang, Yi & Hui Zou. 2017. *gglasso: Group lasso penalized learning using a unified bmd algorithm*. `https://CRAN.R-project.org/package=gglasso`. R package version 1.4.

Yuan, Ming & Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B* 68. 49–67.

Zou, Hui. 2006. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association* 101(476). 1418–1429.