Mathematical Aspects of Word Embeddings

Rachel Carrington, MMath

Thesis submitted to The University of Nottingham

for the degree of Doctor of Philosophy

April 2021

Contents

Abstract

Word embeddings are a popular way of modelling relationships between words. Words are represented as low-dimensional vectors, such that the distances between the vectors reflect relationships between the words: words which are more similar to each other should be closer together in the embedding space.

This thesis explores several different aspects of word embeddings. First, we look at the problem of non-identifiability: word embeddings are generated by optimizing an objective function, but the optimal embedding set is not unique. This has consequences for how embeddings are evaluated, and for making comparisons between different word embedding methods. We explain why this is the case and propose some solutions for dealing with it.

We then explore the potential for generating semi-supervised word embeddings, with the aim being to more accurately capture the relationships between words, compared to using standard unsupervised embedding methods. We introduce three semi-supervised objective functions, derive algorithms for optimizing them, and implement them on simulated and real data.

Finally, we look at the generation of time-dependent word embeddings, in particular the development of statistical tests for assessing whether certain words have changed in meaning or usage over a given time period. We introduce a time-dependent word embedding model and use it to test for change over time. However, we find that we are unable to distinguish between the presence of time dependence and a misspecified embedding dimension.

Chapter 1

Introduction

This thesis is concerned with the area of word embeddings. Word embeddings are a popular way of representing language mathematically, where each word is represented by a low-dimensional vector, or embedding. The aim is to generate a set of word embeddings in such a way that the distances between the embeddings (defined by some metric) reflect the linguistic relationships between words. For example, we would expect that words which have similar meanings (such as "car" and "automobile") should be close together in the embedding space. Word embeddings have a range of applications in areas such as sentiment analysis ([Liu, 2017, Acosta et al., 2017, Yu et al., 2017]), machine translation ([Mikolov et al., 2013b, Jansen, 2017]), sentence classification [Kim, 2014], story generation [Purdy et al., 2018], search engine indexing [Deerwester et al., 1990], and document summarization [Ng and Abrecht, 2015].

In the past few years, there has also been a growing interest in the development of time-dependent word embeddings, which are able to encapsulate changes in word relationships across time. This can be used, for example, to evaluate how certain words have changed in usage as well as how attitudes towards certain topics have changed.

Recent well-cited papers in the area of word embeddings include [Mikolov et al., 2013c], [Mikolov et al., 2013a] and [Pennington et al., 2014], which introduce models

which are used in a wide variety of applications, and [Hamilton et al., 2016], which looks at trends across time. In this thesis we will consider several aspects of word embeddings, including using them to test for change in word usage across time.

1.1 Definitions & notation

There are a number of different methods for generating word embeddings. We will discuss some of the most popular of these in more detail in Chapter 2, but here we give a general idea of the process for generating word embeddings. In doing so we will also introduce some of the main definitions and notation which will be used in this thesis.

We start with a text corpus, consisting of n documents – such as newspaper articles, books, or other forms of text – and containing p unique words. In order to represent the corpus mathematically, we convert it into a matrix representation. There are two main ways of doing this:

Definition 1. The **document-term matrix** is an $n \times p$ matrix, where each row of the matrix corresponds to a document, and the entries of the row are the number of times each word occurs in that document. The ijth element of the document-term matrix is the number of times word j occurs in document i.

Definition 2. The **co-occurrence matrix** is a $p \times p$ matrix which contains, for a given L, counts of how many times each pair of words occur within L places of each other in the corpus. (When words i and j occur within L places of each other, we say that word i occurs in context j, and vice versa.) The ijth element of the co-occurrence matrix is the number of times word j occurs in context i.

For example, in the following sentence, with $L = 2$, the words which co-occur with the word **example** are those which occur within $L = 2$ places of it: *is, an, of, a*:

`This` *is an* **example** *of a* `sentence`.

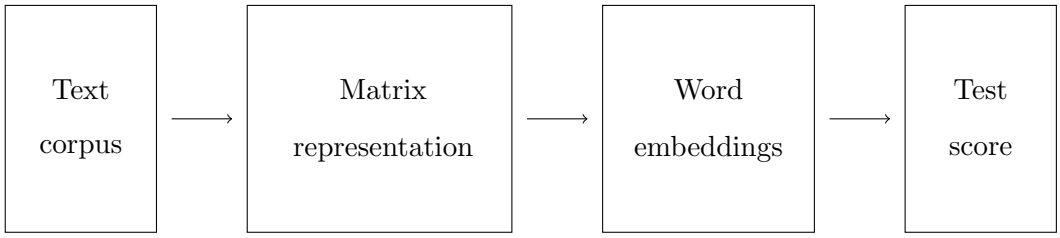

Figure 1.1: Figure illustrating the process of generating word embeddings. We start with a text corpus which is converted to a matrix representation $\boldsymbol{X}$. The word embeddings are generated using some method (examples are given in Chapter 2), and their performance is assessed on test data $\boldsymbol{D}$ (see Section 2.2).

We denote the data matrix – which may be either the document-term matrix or co-occurrence matrix, or a transformation of one of these (see below) – by $\boldsymbol{X}$. From $\boldsymbol{X}$ we generate an $r \times p$ matrix of word embeddings, $\boldsymbol{B}$, where $r < \min\{n, p\}$ is the embedding dimension, and the jth column of $\boldsymbol{B}$ (denoted $\boldsymbol{b}_j$) corresponds to the embedding associated with word j. For many word embedding methods we will also generate a matrix of context embeddings $\boldsymbol{A}$, which will be either $n \times r$ or $p \times r$ depending on the dimension of $\boldsymbol{X}$. The ith row of $\boldsymbol{A}$ is denoted $\boldsymbol{a}_i$, and is the embedding corresponding to document i (if $\boldsymbol{X}$ is the (possibly transformed) document-term matrix) or context i (if $\boldsymbol{X}$ is the co-occurrence matrix). In order to do this we optimize some objective function f, which is a function of the data matrix $\boldsymbol{X}$, word embedding matrix $\boldsymbol{B}$ and auxiliary matrix $\boldsymbol{A}$:

$$(\boldsymbol{A}, \boldsymbol{B}) = \arg\min_{\boldsymbol{A},\boldsymbol{B}} f(\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{B}). \tag{1.1}$$

The interpretation of $\boldsymbol{A}$ depends on the data matrix used. For example, if $\boldsymbol{X}$ is the document-term matrix, then there is an r-dimensional embedding $\boldsymbol{a}_i$ associated with each document. These can be used in applications such as document classification or clustering. However, in this thesis the main focus will be on the word embeddings $\boldsymbol{B}$.

Once the embeddings have been generated, they are assessed using a test function $g(\boldsymbol{D}, \boldsymbol{B})$, where $\boldsymbol{D}$ is a test dataset. Examples of functions used for both f and g

are given in Chapter 2.

Usually, the data matrix $\boldsymbol{X}$ used to generate word embeddings is a transformation of the document-term matrix or co-occurrence matrix. The main reason for using such a transformation is to reduce the impact of large documents or frequent words, as these may have a disproportionate effect on the results. For example, normalizing the rows of $\boldsymbol{X}$ so that they sum to 1 gives equal weight to all documents or contexts. Another option is to take the log of $\boldsymbol{X}$ (or more usually of $1 + \boldsymbol{X}$, to avoid infinite entries); this means that the difference between the largest and smallest entries is not so great. A more complex function which combines these two approaches is the Pointwise Mutual Information (PMI) (introduced by [Church and Hanks, 1990]) or Positive Pointwise Mutual Information (PPMI) (defined below). The idea is that we divide the observed x_{ij}'s by the expected value each of them would take under the assumption that all words and contexts were independent. We then take the log of this, so that a positive value indicates that a word-context pair occurs more frequently than we would expect if they were independent, and hence indicates an association between them.

Definition 3. The **Pointwise Mutual Information (PMI)** matrix of an $m \times n$ matrix $\boldsymbol{X}$ is a matrix whose ijth element is

$$PMI\left(i,j\right) = \log\left(\frac{x_{ij}}{\sum_{k=1}^{n} x_{ik} \sum_{l=1}^{m} x_{lj}} \sum_{k=1}^{m}\sum_{l=1}^{n} x_{kl}\right) \tag{1.2}$$

Definition 4. The **Positive Pointwise Mutual Information (PPMI)** of a matrix is defined as

$$PPMI\left(i,j\right) = \max\{PPMI\left(i,j\right), 0\} \tag{1.3}$$

Using PPMI rather than PMI avoids the issue of PMI taking value $-\infty$ when $x_{ij} = 0$, so it is often preferred in practice [Levy et al., 2015].

Another possible transformation is to use *tf-idf* (term frequency-inverse document frequency) [Zhang et al., 2011]. This is where the columns associated with each word are scaled, with a scaling factor which is inversely proportional to the

number of documents in which a word appears. So a word which appears in all of the documents will have its entries made much smaller, as this is assumed to be a very common word which is not of much interest. A word which appears in only a small number of documents will be raised in importance.

1.2 Datasets

Most of the results on real data given in this thesis are based on the *Corpus of Historical American English (COHA)* ([Davies, 2010, 2012]). This consists of a collection of novels, non-fiction, newspaper and magazine articles published between the 1810s and 2000s, containing over 400 million words in total. The documents are not evenly distributed across this time period – there are fewer documents for the earlier years – but it is balanced so that the proportion of each genre remains the same. The data can be accessed, for a fee, at `https://www.english-corpora.org/coha/`.

We also make use of word embeddings trained on the *Google News* dataset, which contains about 100 billion words. The dataset is not publicly available, but the pre-trained embeddings are available at `https://code.google.com/archive/p/word2vec/`.

1.3 Contributions of the thesis

In this thesis we investigate several mathematical aspects of word embeddings: identifiability, semi-supervised embeddings, testing for time dependence/statistical inference.

First, we consider the issue of identifiability. In most word embedding algorithms, the solution is not identifiable; in other words, there is not a unique embedding set which minimizes the objective function. This has consequences for model assessment and comparison, as sets of embeddings which perform equally well with respect to

the model objective may not be considered equally good when used in applications. It is also a problem if we want to use the embeddings for statistical inference, where identifiability of the parameters is often a necessary assumption. In Chapter 3 we look at the issue of non-identifiability in several popular word embedding methods. We show why it occurs, and discuss the consequences of non-identifiability and possible solutions for dealing with it. In later chapters, which focus on other mathematical aspects of word embeddings, we consider how the issue of non-identifiability relates to these aspects.

Secondly, we explore the idea of generating semi-supervised word embeddings, which make use of both a large text corpus and some supervised information about the relations between words, for example, human-assigned similarity scores between pairs of words. The aim is that using semi-supervised word embedding algorithms will enable us to generate better embeddings, particularly in situations where we may only have relatively small datasets available.

Thirdly, we look at applying statistical inference for time-dependent embeddings. In particular, we investigate the issue of testing for whether particular words change across time in meaning or usage: this is important as it is necessary to determine whether any changes we observe are statistically significant. We also link this issue to the issue of identifiability, as parameter identifiability is important for statistical inference to be performed.

1.4 Overview of thesis

The thesis is outlined as follows.

Chapter 2 reviews the literature on word embeddings, covering the most popular methods used, the ways in which different sets of word embeddings are evaluated, the efforts at extending these methods to incorporate time dependence, and some of the open problems which still remain in this area.

Chapter 3 looks at the issue of non-identifiability in word embedding models

– that is, the existence of multiple optimal solutions with respect to the model objective functions – and the consequences of this for assessing the performance of the embeddings. We show that different sets of word embeddings which are equally good according to the model's objective function can perform quite differently on test data, and propose two possible solutions for addressing this. The last part of the chapter looks at the impact of this issue on time-dependent word embeddings.

Chapter 4 investigates the possibilities of using multidimensional scaling to generate semi-supervised word embeddings, by combining standard word embedding objectives with a supervised component. In particular, we look at how this relates to the non-identifiability issues investigated in Chapter 3, as well as how this can help to improve performance on test data.

Chapter 5 introduces a new model for time-dependent embeddings, which attempts to circumvent some of the statistical issues with present approaches, such as the problem of non-identifiability. We make use of theory from factor analysis to introduce a statistical test which may be used to indicate the presence of time dependence in a dataset, and investigate via a simulation study in which situations the test is useful. We also show some results on data from COHA.

Chapter 6 gives a summary of the work presented and some suggestions for further research.

Chapter 2

Literature review

Word embeddings have been an area of increasing interest over the last few decades. This is mainly due to large increases in computer power and in the amount of data available, which has allowed embeddings trained on very large datasets to capture semantic relationships between words more accurately than was previously possible. Methods such as Skip-Gram With Negative Sampling (SGNS) [Mikolov et al., 2013a,c] and Global Vectors (GloVe) [Pennington et al., 2014] have become very popular, with over 10,000 citations each, and word embeddings are used in a wide range of applications, such as machine translation [Mikolov et al., 2013b], story generation [Purdy et al., 2018], sentiment analysis (e.g. [Acosta et al., 2017, Liu, 2017]), search engine indexing (e.g. Latent Semantic Indexing, [Deerwester et al., 1990]), and document summarization [Ng and Abrecht, 2015]. Recently, word embedding methods have begun to be applied to other forms of data as well, for example to generate product recommendations (e.g. [Grbovic et al., 2016]) and to model networks ([Perozzi et al., 2014]), but we shall not focus on these in this thesis.

In this chapter we explain how word embeddings are generated and assessed, and discuss some of the recent work in this area. Section 2.1 illustrates the process of generating word embeddings using the example of one embedding method (the Singular Value Decomposition). Section 2.2 looks at how word embeddings are evaluated, in order to determine how well they capture the relationships between

words. In Section 2.3 we discuss some popular methods for generating word embeddings, including those regarded as the current state-of-the-art. Section 2.4 looks at practical considerations to be taken into account when implementing a word embedding method, such as how to choose the embedding dimension. Section 2.5 looks at how some of these methods have been extended to generate time-dependent word embeddings.

Sections 2.6 and 2.7 outline two dimension reduction methods, factor analysis and multidimensional scaling, which have not been applied to word embeddings but which will be made use of in Chapters 4 and 5. Section 2.8 summarizes the datasets used in this thesis.

2.1 Word embeddings via Singular Value Decomposition

The main idea behind most word embedding methods is to take a data matrix $\boldsymbol{X}$ (usually the document-term matrix or co-occurrence matrix, or some transform of one of these: see Section 1.1) and approximate it by the product of two low-rank matrices

$$\boldsymbol{X} \approx \boldsymbol{A}\boldsymbol{B},$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the matrices of context and word embeddings respectively. ("Context" usually refers to co-occurrences: the contexts of a word are the words with which it co-occurs. Here, it can also mean the document in which a word occurs.)

The Singular Value Decomposition (SVD) of an $n \times p$ matrix $\boldsymbol{X}$ is defined as

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T,$$

where $\boldsymbol{\Sigma}$ is a diagonal matrix containing the m singular values of $\boldsymbol{X}$ ($m = \min\{n, p\}$) arranged in descending order along its diagonal, and $\boldsymbol{U}$ and $\boldsymbol{V}$ are respectively $n \times m$

and $p \times m$ matrices containing the corresponding left and right singular vectors of $\boldsymbol{X}$. Both $\boldsymbol{U}$ and $\boldsymbol{V}$ have orthonormal columns, so $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_m$.

For $r < m$, the reduced-rank SVD of $\boldsymbol{X}$ is obtained by setting

$$\boldsymbol{X} \approx \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^T = \boldsymbol{X}_r^*,$$

where $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$ are $n \times r$ and $p \times r$ matrices containing only the first r columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively, and $\boldsymbol{\Sigma}_r$ is the $r \times r$ top left-hand corner of $\boldsymbol{\Sigma}$. This gives the best rank-r approximation of $\boldsymbol{X}$, as Lemma 1 states.

Lemma 1. *[Schmidt, 1907] The rank-r SVD of a matrix $\boldsymbol{X}$ gives the best approximation to $\boldsymbol{X}$ that has rank less than or equal to r, in the sense that it minimizes the difference between $\boldsymbol{X}$ and $\boldsymbol{X}_r$ when this is measured using the Frobenius norm:*

$$\boldsymbol{X}_r^* = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^T = \arg\min_{\boldsymbol{X}_r} ||\boldsymbol{X} - \boldsymbol{X}_r||_F,$$

where the Frobenius norm $||\ ||_F$ is defined as

$$||\boldsymbol{X}||_F = \left(\sum_{i,j} x_{ij}^2\right)^{1/2}.$$

To generate word embeddings using SVD, we first choose the data matrix $\boldsymbol{X}$. Latent Semantic Analysis (LSA) [Deerwester et al., 1990] takes $\boldsymbol{X}$ to be the document-term matrix (see Section 1.1); SVD has also been used to generate word embeddings using functions of the co-occurrence matrix, for example the PPMI of the co-occurrence matrix [Levy et al., 2015], and the square root and log of the co-occurrence matrix [Pennington et al., 2014]. (See Section 2.3.9 for more details.) Whichever matrix we decide to use, we choose an embedding dimension r, which is small compared to the dimensions of $\boldsymbol{X}$, and compute the rank-r SVD of $\boldsymbol{X}$:

$$\boldsymbol{X}_r = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^T.$$

We take $\boldsymbol{A} = \boldsymbol{U}_r$ to be the matrix of context embeddings and $\boldsymbol{B} = \boldsymbol{V}_r^T$ the matrix of word embeddings. Usually $\boldsymbol{\Sigma}_r$ is absorbed into either $\boldsymbol{A}$ or $\boldsymbol{B}$; it is also possible

to split it, taking the embedding matrices as $\boldsymbol{A} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r^{1-\alpha}$, $\boldsymbol{B} = \boldsymbol{\Sigma}_r^{\alpha}\boldsymbol{V}_r^T$ [Levy et al., 2015, Bullinaria and Levy, 2012]. We will explore this further in Chapter 3, so for now we will assume that the word embedding matrix is $\boldsymbol{B} = \boldsymbol{V}_r^T$.

2.2 Quantifying performance of word embeddings

The aim of word embedding methods (such as those described in the next section) is to generate representations that capture the meanings of and relations between words. In order to determine how good a particular method is at doing this, the performance of the embeddings is assessed on a set of test data, or tasks. There are two main types of tasks: word similarity and word analogy.

For *word similarity* tasks, the test data consists of a set of pairs of words, with human-assigned similarity scores between them, usually averaged over scores assigned by a number of people. For example, in the WordSim-353 test set [Finkelstein et al., 2001], each pair is rated by between 13 and 16 people. In the SimVerb-3500 test set [Gerz et al., 2016], there were 843 raters, each of whom rated a subset of the word pairs. We denote the test data by $\boldsymbol{D}$, the set of human-assigned scores by $\boldsymbol{y}(\boldsymbol{D})$, and the number of triples (w_{i_1}, w_{i_2}, y_i) by d. Table 2.1 shows an example of what the test data looks like.

Word 1	Word 2	Similarity score
love	sex	6.77
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	keyboard	7.62

Table 2.1: The first few rows of the WordSim-353 test set [Finkelstein et al., 2001]. Each word pair has a similarity score between 0 and 10.

To compute the test scores for the embeddings, we first calculate a similarity

score for each pair of words in the test set, by calculating the cosine similarity (defined below) between the corresponding embeddings for the two words in each pair. Then we take the correlation between the sets of human-assigned and word embedding similarity scores. This gives a value between -1 and 1: a score closer to 1 indicates that the word embeddings perform better with respect to the test set. Hence, we can write the word similarity test function as

$$g(\boldsymbol{D}, \boldsymbol{B}) = \rho\left(\boldsymbol{y}\left(\boldsymbol{D}\right), \boldsymbol{z}\left(\boldsymbol{D}, \boldsymbol{B}\right)\right), \tag{2.1}$$

where $\boldsymbol{y}$ is the vector of human-assigned similarity scores contained within the test data, and z_i is the cosine similarity between words w_{i_1} and w_{i_2}:

$$z_i = \cos\left(\langle \boldsymbol{b}(w_{i_1}), \boldsymbol{b}(w_{i_2})\rangle\right), \quad (i \in \{1, ..., d\}) \ ,$$

where $\boldsymbol{b}(w_k)$ denotes the embedding associated with word w_k. Table 2.6 lists some commonly used word similarity test sets with brief descriptions.

For *word analogy* tasks, we look for relationships of the form "w_{i_1} is to w_{i_2} as w_{i_3} is to w_{i_4}" (for example, " "king" is to "queen" as "man" is to w_{i_4}"), where the word embeddings are assessed on their ability to predict w_{i_4}, given the first three words. The most common way to solve an analogy [Levy and Goldberg, 2014b, Mikolov et al., 2013a, Pennington et al., 2014, Mnih and Kavukcuoglu, 2013] is to find

$$\hat{w}_i = \underset{w \in \{w_1, ..., w_p\}}{\arg\max} \ \cos\left(\boldsymbol{b}(w_{i_1}) - \boldsymbol{b}(w_{i_2}) + \boldsymbol{b}(w_{i_3}), \boldsymbol{b}(w)\right).$$

[Levy et al., 2015] uses 3CosMul, where we find

$$\hat{w}_i = \underset{w \in \{w_1, ..., w_p\}}{\arg\max} \ \frac{\cos\left(\langle \boldsymbol{b}(w), \boldsymbol{b}(w_{i_2})\rangle\right) \cos\left(\langle \boldsymbol{b}(w), \boldsymbol{b}(w_{i_3})\rangle\right)}{\cos\left(\langle \boldsymbol{b}(w), \boldsymbol{b}(w_{i_1})\rangle\right)}.$$

The test score is the rate of accuracy for the embedding set: i.e., the proportion of tasks for which $\hat{w}_i = w_{i_4}$. An example of an analogy test set is the Google Analogy test set [Mikolov et al., 2013a], which contains 19544 analogies to be evaluated. There are 14 different types of analogy (for example, country/capital, country/currency, male/female, etc.). A few examples are given in Table 2.2.

w_{i_1}	w_{i_2}	w_{i_3}	w_{i_4}
Athens	Greece	Bangkok	Iraq
Havana	Cuba	Minsk	Belarus
India	rupee	Japan	yen
son	daughter	boy	girl
amazing	amazingly	apparent	apparently
bad	worse	big	bigger

Table 2.2: Examples of analogies from the Google Analogy test set [Mikolov et al., 2013a]. The last word in each row is the one that must be predicted using the word embeddings.

2.3 Word embedding methods

In this section we discuss some popular methods for generating word embeddings. These methods can be divided into several categories:

Topic models. These are models for the document-term matrix, where the corpus is assumed to consist of a number of topics; for example, a corpus of news articles may include topics such as politics, the environment, business, sport, etc. Each dimension in the word and context embeddings (i.e. each column of $\boldsymbol{A}$ and row of $\boldsymbol{B}$) corresponds to a topic; a larger entry indicates that a word or document is more strongly associated with that topic.

Non-negative matrix factorization. Here we assume that all entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ are non-negative. This is essentially a subset of topic models (see above), which is employed because it is often easier to interpret non-negative results.

Machine learning methods. Here the method is defined by an objective function, which we are trying to optimize with respect to the word and context embeddings, rather than by a statistical model for generating the corpus.

In Chapter 3, we discuss identifiability with respect to Latent Semantic Analysis (LSA), which comes under topic models, and machine learning methods such as SGNS and GloVe. In Chapters 4 and 5, the novel approaches we develop are based

predominantly on Latent Semantic Analysis. Other methods, such as non-negative matrix factorization, are not discussed in any detail later on, but are included here for the sake of completeness.

Section 2.3.1 explains the topic model set-up, with Sections 2.3.2, 2.3.3, and 2.3.4 giving examples of topic models. Section 2.3.5 covers non-negative matrix factorization. Sections 2.3.6 to 2.3.9 cover machine learning methods. Table 2.3 summarises the main methods described in this section.

2.3.1 Topic modelling

In topic modelling we interpret the r dimensions of the word embeddings as corresponding to r different topics, of which the corpus is composed. These topics are assumed to correspond to meaningful entities; for example, in a corpus of news articles we may expect to see topics such as politics, health, sport, etc.; although in practice, it may sometimes be difficult to discern a meaningful interpretation of a topic. As well as word embeddings, we generate document embeddings, which quantify the association between each document and each topic. Hence, we approximate the document-term matrix by

$$\boldsymbol{X} \approx \boldsymbol{A}\boldsymbol{B},$$

where $\boldsymbol{A}$ is the $n \times r$ matrix of document embeddings, and $\boldsymbol{B}$ is the $r \times p$ matrix of word embeddings.

The simplest such method is Latent Semantic Analysis (LSA), which we discussed in Section 2.1. LSA approximates the document-term matrix by its rank-r SVD, thus minimizing the objective function

$$||\boldsymbol{X} - \boldsymbol{A}\boldsymbol{B}||_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{p}\left(x_{ij} - \boldsymbol{a}_i^T\boldsymbol{b}_j\right)^2.$$

Minimizing this is equivalent to maximizing the likelihood of the model

$$x_{ij} \overset{ind}{\sim} N(\boldsymbol{a}_i^T\boldsymbol{b}_j, \sigma^2).$$

The log-likelihood of the model is

$$\begin{aligned} l &= \sum_{i=1}^{n}\sum_{j=1}^{p}\left(-\frac{1}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\left(x_{ij} - \boldsymbol{a}_i^T\boldsymbol{b}_j\right)^2\right) \\ &= -\frac{np}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\sum_{j=1}^{p}\left(x_{ij} - \boldsymbol{a}_i^T\boldsymbol{b}_j\right)^2 \\ &= -\frac{1}{2\sigma^2}||\boldsymbol{X} - \boldsymbol{AB}||_F^2 + constant. \end{aligned}$$

Hence, we can link the LSA method with a Normal model for the data. This will be useful when we consider extending LSA to generate time-dependent word embeddings in Chapter 5.

2.3.2 Probabilistic Latent Semantic Indexing

Probabilistic Latent Semantic Indexing (PLSI) [Hofmann, 1999] is a similar method to LSA. However, in PLSI the elements of $\boldsymbol{A}$ and $\boldsymbol{B}$ are interpreted as probabilities. We assume that in each document, each word in the document belongs to one topic. The ith row of $\boldsymbol{A}$, $\boldsymbol{a}_i$, is the topic probability distribution for document i; in other words, a_{ik} is the probability that a word selected at random from document i, belongs to topic k. The kth row of $\boldsymbol{B}$, $\boldsymbol{b}_k$, is the word probability distribution for topic k: if we pick a random word from the corpus which belongs to topic k, the probability that it is word j is b_{kj}. Hence, we assume that the rows of $\boldsymbol{A}$ and $\boldsymbol{B}$ contain non-negative entries which sum to 1. We obtain this by scaling the rows of the document-term matrix so that they sum to 1, meaning that the entries of $\boldsymbol{X}$ are now proportions, rather than counts; this means that if we have the rows of $\boldsymbol{B}$ summing to 1, the rows of $\boldsymbol{A}$ will do so also:

$$1 = \sum_{j=1}^{p} x_{ij} = \sum_{j=1}^{p}\sum_{k=1}^{r} a_{ik}b_{kj} = \sum_{k=1}^{r} a_{ik}\sum_{j=1}^{p} b_{kj} = \sum_{k=1}^{r} a_{ik}\ 1 = \sum_{k=1}^{r} a_{ik}.$$

The probability model for word w_j, document d_i, and topic z_k is

$$P\left(w_j|d_i\right) = \sum_{k=1}^{r} P\left(w_j|z_k\right)P\left(z_k|d_i\right) = \sum_{k=1}^{r} b_{kj}a_{ik} = \boldsymbol{a}_i^T\boldsymbol{b}_j.$$

Hence, if N_i is the number of words in document i, $\boldsymbol{x}_i \sim Multinomial\left(\boldsymbol{B}^T\boldsymbol{a}_i\right)$, with N_i trials. This has mean $N_i\boldsymbol{B}^T\boldsymbol{a}_i$ and variance $N_i\boldsymbol{\Sigma}_i$ where

$$\begin{aligned}\boldsymbol{\Sigma}_i &= \text{diag}\left(\boldsymbol{B}^T\boldsymbol{a}_i\right) - \boldsymbol{B}^T\boldsymbol{a}_i\left(\boldsymbol{B}^T\boldsymbol{a}_i\right)^T \\ &= \text{diag}\left(\boldsymbol{B}^T\boldsymbol{a}_i\right) - \boldsymbol{B}^T\boldsymbol{a}_i\boldsymbol{a}_i^T\boldsymbol{B},\end{aligned}$$

where $\text{diag}\left(\boldsymbol{B}^T\boldsymbol{a}_i\right)$ is a diagonal matrix with the elements of the vector $\boldsymbol{B}^T\boldsymbol{a}_i$ along its diagonal. Hence, when we row-normalize the document-term matrix, the normalized vector $\tilde{\boldsymbol{x}}_i = \frac{1}{N_i}\boldsymbol{x}_i$ will have mean $\boldsymbol{B}^T\boldsymbol{a}_i$ and variance $\frac{1}{N_i}\boldsymbol{\Sigma}_i$.

If the document length N_i is large, and $\boldsymbol{B}^T\boldsymbol{a}_i$ is not near the boundary of the parameter space, then the Multinomial distribution can be approximated by a multivariate Normal distribution

$$N\left(N\boldsymbol{B}^T\boldsymbol{a}_i, N\boldsymbol{\Sigma}_i\right).$$

(As $\boldsymbol{B}^T\boldsymbol{a}_i$ is required to be non-negative, the boundary of the parameter space is the set of non-negative r-dimensional vectors which contain at least one 0.) Thus the vector of proportions would approximately follow the distribution

$$\tilde{\boldsymbol{x}}_i \sim N\left(\boldsymbol{B}^T\boldsymbol{a}_i, \frac{1}{N}\boldsymbol{\Sigma}_i\right),$$

which means we can write

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B} + \boldsymbol{Z},$$

where $\boldsymbol{z}_i \sim N_p\left(\boldsymbol{0}, \frac{1}{N_i}\boldsymbol{\Sigma}_i\right)$, where $\boldsymbol{\Sigma}_i$ is a function of $\boldsymbol{B}^T\boldsymbol{a}_i$. However, in practice $\boldsymbol{D}$ will contain a large proportion of zero entries, so if we assume non-negativity then it is unlikely that we would not get values of $\boldsymbol{a}_i^T\boldsymbol{b}_j$ on or near the boundary.

2.3.3 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [Blei et al., 2003] is another topic model for the document-term matrix. As in PLSI, the distribution of the rows of the document-term matrix is given by

$$\boldsymbol{x}_i \sim Multinomial\left(\boldsymbol{B}^T\boldsymbol{a}_i, N\right),$$

where $\boldsymbol{B}$ is an $r \times p$ matrix, $\boldsymbol{a}_i$ an $r \times 1$ vector, and N the number of words in the document.

However, in LDA we also specify a distribution for the document embeddings $\boldsymbol{a}_i$

$$\boldsymbol{a}_i \sim Dir\left(\boldsymbol{\alpha}\right).$$

The Dirichlet distribution is defined on the $(r-1)$-dimensional simplex. A random variable following this distribution will be an r-dimensional vector where all elements are non-negative and sum to 1. The topics are assumed to be independent.

The advantages of using this method over PLSI are: (1) that there are fewer parameters to estimate, as we only have to estimate $\boldsymbol{\alpha}$ (r parameters) rather than all the entries of $\boldsymbol{A}$ (nr parameters), and the number does not grow with the number of documents; and (2) that it can be used for generating a corpus of any size (in PLSI, each document has a separate topic vector associated with it, which means that if more documents are added to the corpus, they cannot be generated using the existing model).

The model is computationally expensive to fit for large datasets [Mikolov et al., 2013a].

2.3.4 Correlated Topic Model

The Correlated Topic Model (CTM) [Blei and Lafferty, 2005] is similar to LDA, but uses the Logistic Normal distribution for $\boldsymbol{a}_i$ rather than the Dirichlet distribution. This is defined on the same parameter space as the Dirichlet distribution, so it is still true that the elements of $\boldsymbol{a}_i$ are non-negative and sum to 1. However, unlike LDA, which assumes that topics are independent, in CTM the topics are allowed to be correlated with each other, thus giving it added flexibility. In practice, it might be expected that some topics would be correlated, especially if a relatively large number of topics is used. The model for the document embeddings is now

$$\boldsymbol{a}_i \sim LogisticNormal\left(\boldsymbol{\alpha}, \boldsymbol{\Sigma}\right),$$

with the rest of the model as for LDA.

2.3.5 Non-Negative Matrix Factorization

Non-Negative Matrix Factorization (NMF) [Xu et al., 2003, Pauca et al., 2004] attempts to find the best approximation to $\boldsymbol{X}$ of the form

$$\boldsymbol{X} \approx \boldsymbol{A}\boldsymbol{B},$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are non-negative matrices of size $n \times r$ and $r \times p$ respectively, where $r << \min\{n, p\}$. NMF is sometimes preferred to SVD in certain applications, such as topic modelling, because the results are easier to interpret. The entries of the matrices can be viewed as relative proportions or strengths of association between a word and a topic (and between a topic and a document), but it is not clear what negative entries would mean in this context. The matrices can be scaled so that their rows sum to 1, in which case their elements can be interpreted as probabilities or proportions.

2.3.6 Skip-Gram with Negative Sampling

Skip-Gram with Negative Sampling (SGNS) is a machine learning method that generates word vectors using co-occurrence counts. It was developed in [Mikolov et al., 2013c] and [Mikolov et al., 2013a]. The objective function (to be maximized) is

$$\sum_{i=1}^{p}\sum_{j=1}^{p} x_{ij} \log\left(\sigma\left(\boldsymbol{a}_i^T \boldsymbol{b}_j\right)\right) + k \, \frac{\sum_{l=1}^{p} x_{il} \sum_{m=1}^{p} x_{mj}}{\sum_{k=1}^{p}\sum_{l=1}^{p} x_{kl}} \log\left(\sigma\left(-\boldsymbol{a}_i^T \boldsymbol{b}_j\right)\right),$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ and k is a hyperparameter that can take positive integer values. The objective function is maximized with respect to the context embeddings $\boldsymbol{a}_i$ and word embeddings $\boldsymbol{b}_j$. The idea is that we are trying to maximize $\sigma\left(\boldsymbol{a}_i^T \boldsymbol{b}_j\right)$ for observed word pairs (i, j) $(x_{ij} > 0)$, and minimize $1 - \sigma\left(\boldsymbol{a}_i^T \boldsymbol{b}_j\right)$ (equivalent to maximizing $\sigma\left(-\boldsymbol{a}_i^T \boldsymbol{b}_j\right)$) for unobserved pairs. The term $\frac{\sum_{l=1}^{p} x_{il} \sum_{m=1}^{p} x_{mj}}{\sum_{k=1}^{p}\sum_{l=1}^{p} x_{kl}}$ is the expected value of x_{ij} if the words in the corpus were ordered at random.

Topic models		
Method	Model	
LSA	$\boldsymbol{x}_i \sim N\left(\boldsymbol{B}^T\boldsymbol{a}_i, \sigma^2\boldsymbol{I}\right)$	
PLSI	$\boldsymbol{x}_i \sim Multinomial\left(\boldsymbol{B}^T\boldsymbol{a}_i\right)$	
LDA	$\boldsymbol{x}_i \sim Multinomial\left(\boldsymbol{B}^T\boldsymbol{a}_i\right)$	$\boldsymbol{a}_i \sim Dirichlet\left(\boldsymbol{\alpha}\right)$
CTM	$\boldsymbol{x}_i \sim Multinomial\left(\boldsymbol{B}^T\boldsymbol{a}_i\right)$	$\boldsymbol{a}_i \sim Logistic\ Normal\left(\boldsymbol{\alpha}, \boldsymbol{\Sigma}\right)$

Word embedding models		
Method	Target matrix	Objective function
PPMI	$\boldsymbol{X}^{PPMI}$	None
SVD	$\boldsymbol{F}\left(\boldsymbol{X}\right)$	$\left\|\left\|\boldsymbol{F}\left(\boldsymbol{X}\right) - \boldsymbol{AB}\right\|\right\|_F$
SGNS	$\boldsymbol{X}^{PMI}$	$\sum_{i=1}^{p}\sum_{j=1}^{p} x_{ij}\log\left(\sigma\left(\boldsymbol{a}_i^T\boldsymbol{b}_j\right)\right) + k\frac{\sum_{l=1}^{p} x_{il}\sum_{m=1}^{p} x_{mj}}{\sum_{i=1}^{p}\sum_{j=1}^{p} x_{ij}}\log\left(\sigma\left(-\boldsymbol{a}_i^T\boldsymbol{b}_j\right)\right)$
GloVe	$\boldsymbol{X}$	$\sum_i\sum_j f\left(x_{ij}\right)\left(\boldsymbol{a}_i^T\boldsymbol{b}_j - \left(\log\left(x_{ij}\right) - \boldsymbol{c}_i - \tilde{\boldsymbol{c}}_j\right)\right)$

Table 2.3: Table summarising the main word embedding models discussed in this literature review. (For SVD, $\boldsymbol{F}(\boldsymbol{X})$ denotes an element-wise function of $\boldsymbol{X}$; various examples are used in the literature (see Section 2.3.9).)

Although SGNS does not directly factorize a matrix, [Levy and Goldberg, 2014b] shows that it is implicitly factorizing a matrix $\boldsymbol{M}$, where

$$m_{ij} = PMI\left(i,j\right) - \log k = \log\left(\frac{x_{ij}\sum_{k=1}^{p}\sum_{l=1}^{p} x_{kl}}{\sum_{k=1}^{p} x_{ik}\sum_{l=1}^{p} x_{lj}}\right) - \log k,$$

where $PMI\left(i,j\right)$ is the Pointwise Mutual Information of i and j, defined in Section 1.1. This is equal to negative infinity if $x_{ij} = 0$, but the SGNS objective function ignores such values.

2.3.7 Global Vectors

Global Vectors (GloVe) [Pennington et al., 2014] also uses co-occurrence counts. The objective function (to be minimized) is

$$\sum_i\sum_j f\left(x_{ij}\right)\left(\boldsymbol{a}_i^T\boldsymbol{b}_j - \left(\log\left(x_{ij}\right) - \boldsymbol{c}_i - \tilde{\boldsymbol{c}}_j\right)\right),$$

where $\boldsymbol{c}_i$ and $\tilde{\boldsymbol{c}}_j$ are bias terms that must be estimated during optimization, along with $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$, and f is a weighting function

$$f(x) = \min\left(\left(\frac{x}{x_{max}}\right)^{\alpha}, 1\right),$$

for hyperparameters α and x_{max}. The paper states that the choice of weighting function is fairly arbitrary and that it is possible to choose a different function.

In [Pennington et al., 2014] this method was found to perform better than SVD and SGNS on most sets of tasks. However, in [Levy et al., 2015] it performed less well. (These two papers used different embedding dimensions and hyperparameter settings.)

2.3.8 Positive Pointwise Mutual Information

In [Levy et al., 2015], one method used is to take the rows of the PPMI of the co-occurrence matrix (see Section 1.1) to be the word embeddings. Although the dimension of the embeddings is much larger than usual, the embeddings generated using this method were found to perform competitively on some word similarity tasks, although they did less well on analogy tasks.

2.3.9 Singular Value Decomposition

We have already looked at Latent Semantic Indexing (LSA), which uses the SVD of the document-term matrix, but there are also several papers that generate word embeddings using the SVD of a transformation of the co-occurrence matrix. For example, [Pennington et al., 2014] investigated taking the SVD of $\sqrt{\boldsymbol{X}}$ and $\log(1 + \boldsymbol{X})$ (where each transformation is applied element-wise), where $\boldsymbol{X}$ is a truncated version of the co-occurrence matrix that contains the number of co-occurrences of each word with each of the top 10,000 words. [Levy et al., 2015] takes the SVD of the PPMI matrix to generate word embeddings.

2.4 Practical considerations

There are several practical issues that must be taken into account when generating word embeddings, regardless of which method is used to do this. For example, we have to choose the dimension of the embedding, and whether to discard words which occur with frequency below a certain threshold. We could also consider using transformations of the data matrix such as row-normalization or PPMI.

2.4.1 Choosing the embedding dimension

One thing we must determine when generating embeddings is deciding what dimension the embeddings should have. Increasing the embedding dimension gives more degrees of freedom, so we can do better with respect to the objective function. However, if the embedding dimension is too high then we risk overfitting, and the embeddings will perform less well on test data. This will also be important when we consider generating time-dependent word embedding models in Chapter 5, particularly with regard to carrying out statistical tests for whether words have changed over time.

There is no general method for determining a good embedding dimension, although $r = 300$ is a common choice (e.g. [Mikolov et al., 2013c, Levy and Goldberg, 2014a, Suzuki and Nagata, 2015]). There are several papers that compare empirically the performance of sets of word embeddings with different embedding dimensions. In [Suzuki and Nagata, 2016], embeddings are trained with dimension 32, 64, 128, 256, 512, and 1024. They find that the performance of the embeddings increases up to $r = 512$, but the performance decreases or stays the same when r is increased again to 1024. The perceived superiority of SVD over PPMI in [Levy et al., 2015] also shows that increasing r too much leads to a negative effect on performance (PPMI is equivalent to SVD with $r = p$).

Figure 2.1 shows how performance on the WordSim-353 test set changes as the embedding dimension increases, for LSA and SGNS embeddings trained on COHA.

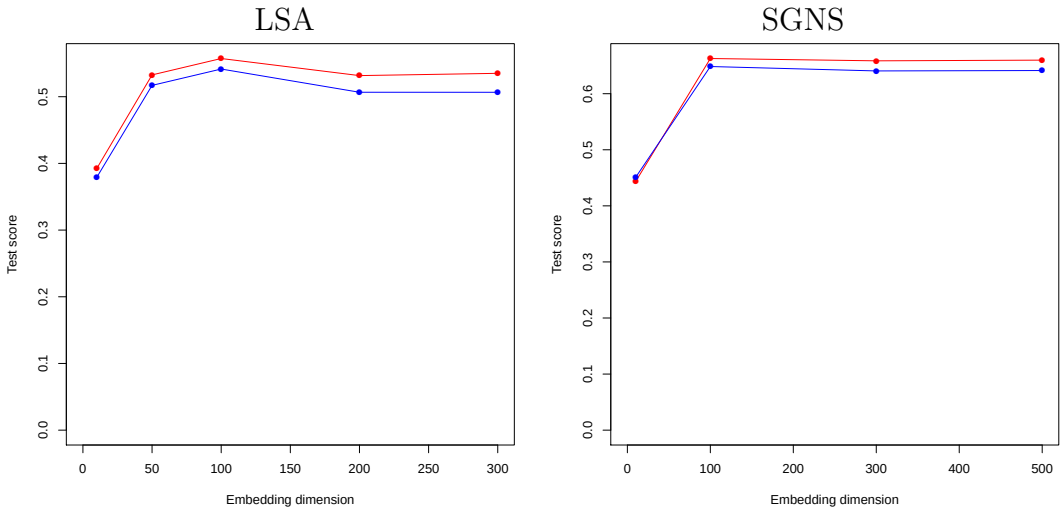

Figure 2.1: Graphs showing how performance on the WordSim-353 test set changes with embedding dimension r. The left-hand graph is for LSA embeddings and the right-hand graph is for SGNS embeddings. In both cases embeddings are trained on COHA.

The left-hand graph is for LSA embeddings and the right-hand graph is for SGNS embeddings. In both cases the performance increases up to $r = 100$, then it levels off (and even decreases slightly). For this dataset, there does not seem to be much to gain from increasing r above 100.

Figure 2.2 shows how the performance of SGNS embeddings trained on COHA changes with r. Each line represents one of the seven test sets listed in Table 2.6. The values of r used are $10, 100, 300, 500$. We can see that in each case, performance increases significantly between $r = 10$ and $r = 100$, and in some cases there is a slight increase in performance between $r = 100$ and $r = 300$, but there is not much difference between $r = 300$ and $r = 500$.

Tables 2.4 and 2.5 compare the word similarity test scores (see Section 2.2) of word embeddings trained on COHA, for different embedding dimensions and pre-processing choices. The test data is the WordSim-353 test set of similarity tasks [Finkelstein et al., 2001]. As with Figures 2.1 and 2.2, we see that increasing the embedding dimension tends to improve the performance of the embeddings up to a

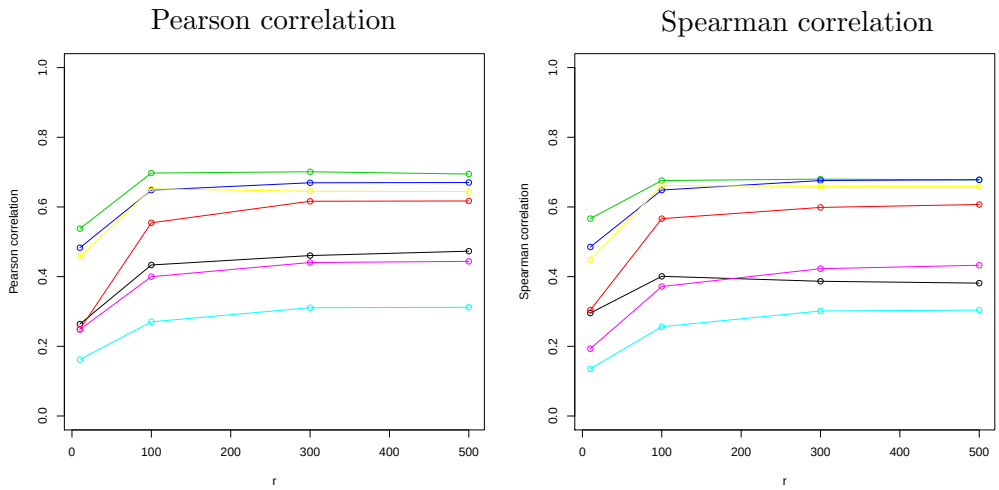

Figure 2.2: Performance of SGNS embeddings on seven different test sets when the embedding dimension r takes different values. The embeddings were trained on COHA, with values of $r = 10, 100, 300, 500$.

point, but continuing to increase the embedding dimension above this point does not increase performance further, and is sometimes detrimental. The optimal embedding dimension is usually around 100, although in a few cases we can increase the test score by increasing the dimensionality to 300 or 500. In the literature, the most common choice of embedding dimension is $r = 300$; however, the COHA dataset is relatively small compared to other datasets used to train word embeddings (e.g. Google News), so it is likely that overfitting happens more easily. Thus we need to take the size of the dataset into account when choosing the embedding dimension. In particular, it is important to be aware that increasing the dimension of the embeddings will not necessarily lead to an increase in performance.

We also see that normalizing the rows of the document-term matrix leads to an improvement in performance of the embeddings, compared to just using the raw counts, and using the PPMI of the document-term matrix produces even better results, even outperforming SGNS.

Method	$r = 10$	50	100	200	300	500
SGNS	0.451		**0.648**		0.641	0.641
SVD, plain	0.306	0.446	0.480	**0.489**	0.477	0.470
SVD, row-normalized	0.379	0.517	**0.541**	0.507	0.507	0.485
SVD, PPMI	0.464	0.617	**0.635**	0.627	0.622	0.605

Table 2.4: Pearson correlations between human-assigned similarity scores and cosine similarity between embeddings, using word embeddings trained on COHA with different preprocessing steps. SVD refers to SVD of the document-term matrix. The test set used is the WordSim-353 similarity test set [Finkelstein et al., 2001]. For each row, the highest score is in bold.

2.4.2 Word choice and preprocessing

There are other preprocessing choices that have to be made when training a set of word embeddings. These include the context window size and weighting of contexts (when working with the co-occurrence matrix), whether to discard infrequent words (and what threshold to choose for this), and whether to downsample frequent words.

Context window size. For methods based on co-occurrence counts, we must choose the size of the context window. Common choices are 2 [Levy et al., 2015], 5 [Levy et al., 2015, Suzuki and Nagata, 2016], and 10 [Mikolov et al., 2013a, Pennington et al., 2014, Levy et al., 2015, Suzuki and Nagata, 2015].

Weighting of contexts. It is generally assumed that contexts that are closer to a given word will be more important than those that are further away. Therefore, it is common to weight contexts to give greater importance to those closer to a given word (compared to contexts that are further away, but are still within the context window). This is done in [Pennington et al., 2014], for example, by assigning contexts a weight $\frac{1}{d}$, where d is the distance from the given word. Other papers ([Levy et al., 2015, Mikolov et al., 2013a]) use a dynamic context window, where the context window for each word in the corpus is sampled uniformly from $1, 2, ..., L$. This has the effect of down-weighting co-occurrences that occur further apart, but is faster

Method	$r = 10$	50	100	200	300	500
SGNS	0.443		**0.663**		0.659	0.660
SVD, plain	0.317	0.439	0.492	0.507	0.507	**0.535**
SVD, row-normalized	0.392	0.533	**0.557**	0.532	0.535	0.547
SVD, PPMI	0.466	0.638	0.687	0.704	**0.711**	0.701

Table 2.5: Spearman rank correlations between human-assigned similarity scores and cosine similarity between embeddings, using word embeddings trained on COHA with different preprocessing steps. SVD refers to SVD of the document-term matrix. The test set used is the WordSim-353 similarity test set [Finkelstein et al., 2001]. For each row, the highest score is in bold.

to implement as not all the co-occurrences need to be observed.

Frequency threshold. Words that occur below a certain number of times in the corpus are discarded. This helps to reduce the size of the dataset, and results in cleaner data.

Dealing with frequent words. Very high-frequency words (such as "the", "a", "and") are usually not of much interest [Mikolov et al., 2013c], but will generally have large values associated with them (in either the document-term matrix or co-occurrence matrix). Hence, it is often useful to implement some preprocessing steps to downgrade their importance, to prevent these words from dominating when embeddings are generated. Some common approaches are:

- Subsampling. In [Mikolov et al., 2013a], frequent words are subsampled in order that they do not dominate the training process. Words are removed with probability

$$1 - \sqrt{\frac{t}{f_i}},$$

where f_i is the frequency of word i. t is a hyperparameter, which in [Mikolov et al., 2013a] and [Levy et al., 2015] is taken to be 10^{-5}. We also have to decide whether to remove these words before or after fixing context windows (i.e., whether to delete or just ignore them).

- Term frequency-inverse document frequency (tf-idf) [Ramos, 2003]. This is a weighting scheme where entries of the document-term matrix are weighted by the inverse of the proportion of documents. The entries of the tf-idf weighted document-term matrix are

$$x_{ij} = x_{ij}^0 \ \log\left(\frac{n}{f_j}\right),$$

where x_{ij}^0 is the raw count for the number of times word j occurs in document i, n is the number of documents in the corpus, and f_j is the number of documents word j occurs in. The weighting results in x_{ij} being smaller for words that occur in most or all of the documents, and larger for words that occur in only a few documents.

Normalization. Whether we are working with the document-term matrix or co-occurrence matrix, there is an option to normalize the rows (and/or columns). For the document-term matrix, dividing each row by its sum results in entries that are equal to the proportion of each document made up by each word, rather than the counts. This prevents the results from being dominated by a large document. Alternatively, we could use a different weighting system such as PPMI.

Many word embedding methods also have hyperparameters that need to be chosen, such as the number of negative samples in SGNS.

2.5 Time-dependent word embeddings

2.5.1 Time-dependent topic modelling

Several papers, such as [Blei and Lafferty, 2006] and [Jacobi et al., 2016], use an LDA framework 2.3.3 to look at topic change across time. This requires producing a document-term matrix for each time stage (for example a year) and analysing them separately, then seeing how the results change across time.

Topics-Over-Time [Wang and McCallum, 2006] presents a time-dependent extension of LDA, where the topics are assumed to follow a Beta distribution, with the aim being to infer the posterior distribution of the topics over a time period. The paper is concerned with trying to infer how the prevalence of each topic changes over time within the corpus, rather than trying to make inferences about the words.

This method is used by [Wijaya and Yeniterzi, 2011]. Their approach is to take a word of interest and extract all its co-occurrences for each year (using Google Ngrams). These sets of co-occurrences are then treated as documents. Topics-Over-Time is used to generate document embeddings and these are then clustered using K-means. It is assumed that a topic change occurs when two consecutive years are assigned to different clusters.

2.5.2 Other time-dependent word embedding methods

The extension of word embedding methods to incorporate a temporal component has been a growing area of interest in the past few years. The aim is to construct time-dependent embeddings that encode changes in word meaning and usage over time, enabling us to uncover trends in how words change. Several papers (e.g. [Hamilton et al., 2016, Dubossarsky et al., 2015]) attempt to predict how much a word will change (where 'change' is defined as movement in the embedding space) based on certain of its features, such as its frequency or polysemy (how many different meanings or 'senses' a word has). As well as semantic change – where a word has either changed its meaning, or gained an additional meaning, we also detect change in usage, where the meaning of a word remains the same, but changes in discourse lead to it being used in different contexts. For example, one of the words that is found to have changed in [Hamilton et al., 2016] is the word 'male'; this is assumed to be due to changes in the way gender is perceived and talked about in society.

The most common approach to generating time-dependent word embeddings (e.g. [Hamilton et al., 2016, Kim et al., 2014, Dubossarsky et al., 2015]) is to divide

the dataset into discrete time periods, such as years or decades, and generate separate sets of embeddings for each period using static word embedding methods such as SGNS. The embeddings must then be aligned so that cross-period comparisons can be made. There are several different ways of doing this: [Hamilton et al., 2016] finds the rotation of the set of word embeddings for each time period that is closest to the set of embeddings for the previous period, using an orthogonal Procrustes algorithm, whilst [Kim et al., 2014] and [Dubossarsky et al., 2015] find the embeddings for each period sequentially, using the embeddings from one period as initial values for the optimization algorithm for the next period.

[Bamler and Mandt, 2017] criticises this approach; dividing up the data in this way and generating multiple sets of embeddings leads to a large increase in the number of parameters, although the number of data points is the same, which increases the risk of overfitting. In addition, word embedding objective functions are usually non-convex so there is no guarantee that the algorithm used will converge to the same set of values each time. This may be problematic when trying to compare word embeddings across times.

A different approach is to optimize the objective over all time periods at once. This is the approach taken in [Yao et al., 2018], whose objective function includes a penalty term that is related to sum of the norms of the differences in the matrices of word embeddings for each pair of consecutive time periods. This has the effect of forcing embedding sets for subsequent time periods to be closer together.

Other ways of generating time-dependent embeddings include those of [Bamler and Mandt, 2017] and [Rosenfeld and Erk, 2018]. [Bamler and Mandt, 2017] implements a time-dependent Bayesian skip-gram model (the static version is described in [Barkan, 2017]; it has the same objective function as SGNS with a Gaussian prior). The corpus is divided into m time periods. The distribution of the matrices of word

and context vectors for time τ_{t+1}, conditional on times τ_t, are

$$\boldsymbol{W}_{t+1}|\boldsymbol{W}_t \sim N\left(\frac{\boldsymbol{W}_t}{1+\sigma_t^2/\sigma_0^2}, \frac{1}{\sigma_t^{-2}+\sigma_0^{-2}}\boldsymbol{I}\right)$$
$$\boldsymbol{C}_{t+1}|\boldsymbol{C}_t \sim N\left(\frac{\boldsymbol{C}_t}{1+\sigma_t^2/\sigma_0^2}, \frac{1}{\sigma_t^{-2}+\sigma_0^{-2}}\boldsymbol{I}\right),$$

where $\sigma_t^2 = D\left(\tau_{t+1} - \tau_t\right)$; D is a global diffusion constant.

[Rosenfeld and Erk, 2018] attempts to model words as a continuous function of time, using a neural network model that is based on SGNS.

Determination of how much a word has changed is usually done by finding the cosine distance between embeddings associated with that word at different time periods. The words that have changed the most between times t_i and t_j are those for which the cosine distance between the respective embeddings of that word at those times is the greatest.

[Hamilton et al., 2016] attempt to detect a set of 28 known shifts in word meaning, by looking at whether each of the words known to have changed meaning are moving towards or away from other similar words (for example, "gay" moving away from "happy" and becoming closer to "homosexual"). They find that in most cases the direction of movement was correct but the distance moved was not always statistically significant. They also posit two "Laws of Semantic Change" that predict how much words change: the law of conformity (more frequent words change more slowly) and the law of innovation (polysemous words change more quickly).

[Dubossarsky et al., 2015] use a clustering algorithm on the word embeddings, and find that there is a correlation between the distance of a word from its cluster centre and the speed at which it changes. [Dubossarsky et al., 2016] propose that verbs change more quickly than nouns, and nouns change more than adjectives.

However, [Dubossarsky et al., 2017] claims that the significance of the results found in [Hamilton et al., 2016] and [Dubossarsky et al., 2015] may be overstated. They suggest that many findings in this area may be artefacts of the text models and comparison methods used. They also show that the average cosine distance between two i.i.d. count vectors is negatively correlated with frequency and posi-

tively correlated with polysemy score, which suggests that the two laws of semantic change claimed by [Hamilton et al., 2016] may arise merely as a result of using cosine distance to measure similarity, rather than from patterns in the data. There is no attempt to control for false positives in [Hamilton et al., 2016], so it is difficult to tell whether their findings are genuinely significant or not.

Open problems

[Kutuzov et al., 2018] lists some open problems in the field of diachronic text analysis. These include:

1. Analysis of languages other than English. A little work has been done in this area, but not much. It may be that different conclusions can be drawn about other languages (for example, [Hellrich and Hahn, 2016] draws different conclusions when analysing German text than when they analyse English text).

2. Creation of methods that work on small datasets, which is necessary for some applications, such as biomedical applications.

3. Creation of gold standard test sets of semantic shifts, similar to those used for static word embeddings. ([Hamilton et al., 2016] uses a set of 28 known shifts for evaluation, but this is very small.)

4. Development of more rigorous mathematical models, for example accounting for false positives.

5. Analysis of types of shift: e.g. broadening vs. narrowing (or positive/negative)

In Chapter 5 of this thesis, which looks at time-dependent word embeddings, we focus mostly on addressing problem 4, the development of more rigorous mathematical models.

2.6 Factor analysis

We now briefly look at two other dimension reduction methods: factor analysis and multidimensional scaling. These are not used to generate word embeddings, but we will make use of them in Chapters 4 and 5, and hence we provide a summary of these methods here. This section deals with factor analysis, and Section 2.7 with multidimensional scaling.

Factor analysis is a more general approach for analysing data matrices that are assumed to have an underlying low-rank structure. It is widely used in areas such as psychology and economics (e.g. [Ford et al., 1986, Bai et al., 2015]). It does not appear to have been applied to the generation of word embeddings, although it is related to the topic modelling approaches described in Section 2.3. We include an outline of it here as we will make use of it in Chapter 5.

In factor analysis, we have an observed data matrix $\boldsymbol{X}$, consisting of n observations (rows) of p variables (columns). As in the topic modelling scenario (Section 2.3.1), each row of $\boldsymbol{X}$ is interpreted as being composed of a set of r latent factors, where $r << \{n, p\}$. The factor model for each row of $\boldsymbol{X}$, $\boldsymbol{x}_i$, is defined as

$$\boldsymbol{x}_i = \boldsymbol{\mu} + \boldsymbol{B}^T \boldsymbol{a}_i + \boldsymbol{z}_i \qquad (i = 1, ..., n), \tag{2.2}$$

where $\boldsymbol{\mu}$ is a constant mean vector, $\boldsymbol{a}_i$ is an $r \times 1$ random vector with mean 0, $\boldsymbol{B}$ is an $r \times p$ matrix, and $\boldsymbol{z}_i$ is a $p \times 1$ vector of errors. $\boldsymbol{z}_i$ has mean 0 and is usually assumed to be normally distributed. The elements of $\boldsymbol{a}_i$ can either be assumed to be fixed parameters, or random variables from some distribution. In the latter case, they are usually taken to be i.i.d. Normal random variables. If we relax the assumption that $\boldsymbol{a}_i$ has mean 0, then the $\boldsymbol{\mu}$ term can be absorbed into $\boldsymbol{B}^T \boldsymbol{a}_i$, and the model is written as

$$\boldsymbol{x}_i = \boldsymbol{B}^T \boldsymbol{a}_i + \boldsymbol{z}_i,$$

or, in matrix form,

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B} + \boldsymbol{Z},$$

which, under the assumption that $\boldsymbol{z}_i \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$, is the same as the Normal model for LSA (see Section 2.3.1). In factor analysis, however, it is more common to use a slightly more general model where $\boldsymbol{z}_i \sim N(\boldsymbol{0}, \boldsymbol{\Sigma}_z)$, with $\boldsymbol{\Sigma}_z$ restricted to be diagonal.

One necessary decision in factor analysis is how to make the parameters of the model identifiable, so that they can be uniquely identified. This is done by imposing constraints on $\boldsymbol{A}$ and $\boldsymbol{B}$. Some common choices are:

- Requiring $\boldsymbol{M}_a = \frac{1}{n-1} \sum_{i=1}^{n} (\boldsymbol{a}_i - \bar{\boldsymbol{a}}) (\boldsymbol{a}_i - \bar{\boldsymbol{a}})^T = \boldsymbol{I}$ and $\boldsymbol{B}^T \boldsymbol{\Sigma}_z^{-1} \boldsymbol{B}$ to be diagonal [Anderson and Amemiya, 1988, Anderson and Rubin, 1956].

- Requiring either the first or last $r \times r$ block of $\boldsymbol{B}$ to be the identity matrix ($\boldsymbol{B} = (\boldsymbol{I}_r, \boldsymbol{\beta})$ or $\boldsymbol{B} = (\boldsymbol{\beta}, \boldsymbol{I}_r)$, where $\boldsymbol{\beta}$ is an $r \times (p - r)$ matrix) [Amemiya and Fuller, 1987, Anderson and Amemiya, 1988, Anderson and Rubin, 1956].

- Requiring specified elements of $\boldsymbol{B}$ to be 0 [Anderson and Rubin, 1956]. This is usually done when we have prior information about the data that would lead us to believe that certain elements of $\boldsymbol{B}$ should be 0.

There are several ways of obtaining estimates for $\boldsymbol{A}$ and $\boldsymbol{B}$. One method is to use principal components analysis (PCA) [Dunteman, 1989]. PCA seeks to find a small number of principal components; that is, vectors $\boldsymbol{v}$ which maximize $\boldsymbol{v}^T \boldsymbol{M}_x \boldsymbol{v}$, where $\boldsymbol{M}_x$ is the covariance matrix of $\boldsymbol{X}$ ($\boldsymbol{v}^T \boldsymbol{M}_x \boldsymbol{v}$ is the variance of $\boldsymbol{v}^T \boldsymbol{x} = \sum_i v_i x_i$). The principal components are the eigenvectors of $\boldsymbol{M}_x$, which means that if $\boldsymbol{X}$ is centred, then PCA gives the same estimates as SVD.

Another common method is to find the maximum likelihood estimates of $\boldsymbol{B}$ and $\boldsymbol{\Sigma}_z$, usually under the assumption that $\boldsymbol{a}_i$ and $\boldsymbol{z}_i$ are independently normally distributed. In this case the model is

$$\boldsymbol{x}_i \sim N\left(\boldsymbol{\mu}, \boldsymbol{B}^T \boldsymbol{\Sigma}_a \boldsymbol{B} + \boldsymbol{\Sigma}_z\right),$$

where $\boldsymbol{\Sigma}_a$ and $\boldsymbol{\Sigma}_z$ are the covariance matrices of $\boldsymbol{a}_i$ and $\boldsymbol{z}_i$ respectively. $\boldsymbol{\Sigma}_z$ is assumed to be diagonal, so the elements of $\boldsymbol{z}_i$ are independent; if the variances are assumed

to be equal ($\boldsymbol{\Sigma}_z = \sigma^2 \boldsymbol{I}_p$), then the maximum likelihood estimates are the same as the PCA estimates.

The asymptotic behaviour of parameter estimates for the factor model, which will be of particular interest in Chapter 5, is well studied. [Amemiya and Fuller, 1987] derives the limiting distribution for the maximum likelihood estimates of $\boldsymbol{B}$, $\boldsymbol{\Sigma}_a$ and $\boldsymbol{\Sigma}_z$ under the assumption that the $\boldsymbol{a}_i$'s are Normally distributed, as well as the maximum likelihood estimates of $\boldsymbol{B}$, $\boldsymbol{M}_a = \frac{1}{n-1} \sum_{i=1}^{n} (\boldsymbol{a}_i - \bar{\boldsymbol{a}}) (\boldsymbol{a}_i - \bar{\boldsymbol{a}})^T$, and $\boldsymbol{\Sigma}_z$ under the assumption that the $\boldsymbol{a}_i$'s are fixed. In both cases the estimates converge to the true parameter values with rate $n^{-1/2}$. However, this result only holds when n is large compared to p.

There also exists a time-dependent version of factor analysis, dynamic factor analysis, (e.g. [Bai, 2003]) where $\boldsymbol{a}_i$ is modelled as a multivariate time series.

2.7 Multidimensional scaling

Multidimensional scaling (MDS) is a method of projecting high-dimensional data into a lower dimensional space. We are presented with data in the form of a set of pairwise dissimilarities or distances between objects, and the aim is to construct a set of low-dimensional embeddings (one for each object) such that the distances between embeddings are as close as possible to the given dissimilarities.

MDS tries to minimize the objective function

$$L_{MDS} = \sum_{i<j} w_{ij} \left(\delta \left(x_{ij} \right) - ||\boldsymbol{b}_i - \boldsymbol{b}_j|| \right)^2,$$

where the x_{ij}'s are the given dissimilarities, $\boldsymbol{b}_i$ is the low-dimensional embedding associated with object i, and the w_{ij}'s are non-negative weights that control the relative importance of each (i, j) pair (for example, $w_{ij} = 0$ if x_{ij} is unobserved). δ is a function of the dissimilarities that may be either selected in advance or estimated during optimization (more details below): we write $\delta_{ij} = \delta(x_{ij})$. The minimization

is with respect to $\boldsymbol{B}$: we try to find

$$\boldsymbol{B} = \arg\min_{\boldsymbol{B}} \sum_{i<j} w_{ij} \left(\delta\left(x_{ij}\right) - ||\boldsymbol{b}_i - \boldsymbol{b}_j||\right)^2 .$$

There are two main types of MDS: metric and non-metric. In metric MDS, δ is specified to be a particular function of x_{ij}: the simplest case is $\delta(x_{ij}) = x_{ij}$. In non-metric MDS, we are interested only in the rank-order of the x_{ij}'s, rather than their precise values. In this case, rather than specifying δ in advance, we estimate it along with $\boldsymbol{B}$ in the minimization of L_{MDS}. We require only that the order of the x_{ij}'s is preserved: i.e., if $x_{ij} \leq x_{kl}$, then $\delta_{ij} \leq \delta_{kl}$. To minimize the objective, we alternate between solving for $\boldsymbol{\Delta}$ and solving for $\boldsymbol{B}$, whilst keeping the other fixed.

2.7.1 The majorization algorithm

A standard method for minimizing the MDS objective function is to use a majorization algorithm (given in e.g. Chapter 8 of [Borg and Groenen, 1997]). The idea of majorization is that for a function $f(x)$, we find a function $m(x, \tilde{x})$ that is simpler to minimize than $f(x)$, and for which $m(x, \tilde{x}) \geq f(x)$ for all $\tilde{x}$, with equality if $x = \tilde{x}$. The value of x is updated at each stage of the algorithm, by finding the updated value x_{k+1} such that $m(x_{k+1}, x_k) \leq m(x_k, x_k)$. This gives the series of inequalities

$$f(x_{k+1}) \leq m(x_{k+1}, x_k) \leq m(x_k, x_k) = f(x_k).$$

Thus, the sequence $\{f(x_k)\}$ is non-increasing, and so, as long as f is bounded below, it will converge to a local minimum of f (although it is not guaranteed to converge to the true minimum).

A majorizing function for the MDS objective function

$$L_{MDS}(\boldsymbol{B}) = \sum_{i<j} w_{ij}(\delta_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||)^2$$

is [Borg and Groenen, 1997]

$$m(\boldsymbol{B}, \tilde{\boldsymbol{B}}) = \eta_\delta^2 + \operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{Y} \boldsymbol{B}\right) - 2\operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{H}(\tilde{\boldsymbol{B}})\tilde{\boldsymbol{B}}\right), \tag{2.3}$$

where

$$\eta_\delta^2 = \sum_{i<j} w_{ij}\delta_{ij}^2;$$

$\boldsymbol{Y}$ is a matrix with

$$y_{ij} = \begin{cases} \sum_{k=1,k\neq i}^{n} w_{ik}, & \text{if } i = j \\ -w_{ij}, & \text{if } i \neq j; \end{cases}$$

and $\boldsymbol{H}(\tilde{\boldsymbol{B}})$ has elements h_{ij}, where

$$h_{ij} = \begin{cases} -\frac{w_{ij}\delta_{ij}}{||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j||} & \text{if } i \neq j \text{ and } ||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| \neq 0 \\ 0 & \text{if } i \neq j \text{ and } ||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| = 0 \\ -\sum_{k=1,k\neq i}^{n} h_{ik} & \text{if } i = j. \end{cases}$$

A derivation of this is given in the Appendix (Appendix B.1), since it will be useful in Chapter 4.

Algorithm 1 gives the majorization algorithm for metric MDS.

Algorithm 1 Majorization: Metric MDS

Require: $\tilde{\boldsymbol{B}} = \boldsymbol{B}^{[0]}$ and $k = 0$.

$objective^{[0]} = L_{MDS}\left(\boldsymbol{B}^{[0]}\right)$.

while $convergence = 0$ **do**

$k = k + 1$.

$\boldsymbol{B}^{[k]} = \boldsymbol{Y}^{+}\boldsymbol{H}(\tilde{\boldsymbol{B}})\tilde{\boldsymbol{B}}$, where $\boldsymbol{Y}^{+}$ is the Moore-Penrose inverse of $\boldsymbol{Y}$.

$objective^{[k]} = L_{MDS}\left(\boldsymbol{B}^{[k]}\right)$.

if $objective^{[k-1]} - objective^{[k]} < \epsilon$ **then**

$convergence = 1$.

else if $k = maxiter$ **then**

$convergence = 1$.

end if

$\tilde{\boldsymbol{B}} = \boldsymbol{B}^{[k]}$.

end while

2.7.2 Stochastic gradient descent

In this section we give an outline of stochastic gradient descent, which we will also use to optimize the MDS objective in Chapter 4.

Gradient descent (Algorithm 2) is an optimization algorithm for minimizing a function $f(x)$. At each iteration, we calculate the gradient at the current value of the parameters, and update the parameter values by taking a small step in the opposite direction from the gradient, until we reach a point where the gradient is 0.

Algorithm 2 Gradient Descent

Require: $\boldsymbol{B}^{[0]}$, γ.

$k = 0$.

while $|\frac{\partial f}{\partial \boldsymbol{B}^{[k]}}| > 0$ **do**

$k = k + 1$.

Calculate $\frac{\partial f}{\partial \boldsymbol{B}^{[k-1]}}$.

$\boldsymbol{B}^{[k]} = \boldsymbol{B}^{[k-1]} - \gamma \frac{\partial f}{\partial \boldsymbol{B}^{[k]}}$.

end while

The parameter γ is the learning rate, which we must tune to achieve a good rate of convergence. If γ is too small, then convergence will be very slow; if γ is too large, then the value of the gradient may increase instead of decreasing, and the algorithm may not converge at all. It is also possible that the algorithm may converge to a local minimum, rather than the overall minimum.

Stochastic gradient descent is used when the gradient at a point $\boldsymbol{B}$ can be written as a sum:

$$\frac{\partial f}{\partial \boldsymbol{B}} = \sum_{i=1}^{m} g_i(\boldsymbol{B}),$$

for some functions $g_1, ..., g_m$. Instead of calculating the full gradient, we approximate it by taking the sum over a subset of the g_i's:

$$\frac{\partial f}{\partial \boldsymbol{B}} \approx \sum_{i \in \mathcal{M}} g_i(\boldsymbol{B}),$$

where $\mathcal{M}$ is a random sample from $\{1, ..., m\}$. If the batch size $|\mathcal{M}|$ is small, then this is much quicker to compute than the full gradient, so although it is less accurate, it is often preferred in practice.

The algorithm for stochastic gradient descent is given in Algorithm 3.

Algorithm 3 Stochastic Gradient Descent

Require: $\boldsymbol{B}^{[0]}$, γ, m_b.

$k = 0$.

while $|\frac{\partial f}{\partial \boldsymbol{B}^{[k]}}| > 0$ **do**

$k = k + 1$.

Take a $\mathcal{M}$ to be a random sample of size m_b from $\{1, ..., m\}$.

$\boldsymbol{B}^{[k]} = \boldsymbol{B}^{[k-1]} - \gamma \sum_{i \in \mathcal{M}} g_i(\boldsymbol{B})$.

end while

2.7.3 Non-metric MDS

In non-metric MDS we are only interested in the rank-order of the dissimilarities. In the objective function we use estimated disparities $\hat{\delta}_{ij}$:

$$L_{MDS}(\boldsymbol{B}) = \sum_{i<j} w_{ij} \left(\hat{\delta}_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j|| \right)^2 = \eta_\delta^2 + \eta^2(\boldsymbol{B}) - 2\rho(\hat{\boldsymbol{\Delta}}, \boldsymbol{B}).$$

It is usual to constrain the $\hat{\delta}_{ij}$'s so that $\eta_{\hat{\delta}}^2 = \frac{1}{2}n(n-1)$, as otherwise the objective would be minimized by the trivial solution $\hat{\boldsymbol{\Delta}} = \boldsymbol{0}, \boldsymbol{B} = \boldsymbol{0}$.

In non-metric MDS, we are only interested in the ordering of the dissimilarities, and δ is not directly specified. The requirement is:

$$\text{If } x_{ij} < x_{kl}, \text{ then } \hat{\delta}_{ij} \leq \hat{\delta}_{kl}.$$

The majorization algorithm for non-metric MDS is given in Algorithm 4.

In order to find

$$\hat{\delta}^{[0]} = \arg\min_{\hat{\delta}} \sum_{i<j} \left(\hat{\delta}_{ij} - ||\boldsymbol{b}_i^{[0]} - \boldsymbol{b}_j^{[0]}|| \right)^2,$$

we use Kruskal's up-and-down-blocks algorithm [Kruskal, 1964].

Algorithm 4 Majorization: Non-metric MDS

Require: $\tilde{\boldsymbol{B}} = \boldsymbol{B}^{[0]}$, $k = 0$, ϵ.

$\hat{\delta}_{ij}^{[0]} = \gamma \delta_{ij}^{[0]}$, where γ is chosen such that we get $\eta_{\hat{\delta}}^{[0]} = \frac{1}{2}n(n-1)$.

Compute $L_{MDS}\left(\boldsymbol{B}^{[0]}\right)$.

while $convergence = 0$ **do**

$k = k + 1$.

$\boldsymbol{B}^{[k]} = \boldsymbol{Y}^{+}\boldsymbol{H}\left(\tilde{\boldsymbol{B}}\right)\tilde{\boldsymbol{B}}$.

$\hat{\delta}_{ij}^{[k]} = \arg\min_{\hat{\delta}} \sum_{i<j}\left(\hat{\delta}_{ij} - ||\boldsymbol{b}_i^{[k]} - \boldsymbol{b}_j^{[k]}||\right)^2$.

$\hat{\delta}_{ij}^{[k]} = \gamma \hat{\delta}_{ij}^{[k]}$, where γ is chosen such that we get $\eta_{\hat{\delta}}^{[k]} = \frac{1}{2}n(n-1)$.

Compute $L_{MDS}\left(\boldsymbol{B}^{[k]}\right)$.

if $L_{MDS}\left(\boldsymbol{B}^{[k-1]}\right) - L_{MDS}\left(\boldsymbol{B}^{[k]}\right) < \epsilon$ **then**

break.

else

$\tilde{\boldsymbol{B}} = \boldsymbol{B}^{[k]}$.

end if

end while

2.7.4 Link between GloVe and MDS

The GloVe objective function [Pennington et al., 2014] is

$$L_{GloVe}(\boldsymbol{A}, \boldsymbol{B}) = \sum_i \sum_j f(x_{ij}) \left(\log x_{ij} - \boldsymbol{a}_i^T \boldsymbol{b}_j - c_i - \tilde{c}_j \right)^2 . \tag{2.4}$$

This is similar to Equation 2.7, if we ignore the bias terms c_i and c_j, and replace the Euclidean distance $|| \cdot ||$ with the inner product (as we will consider doing for MDS later, in Section 4.3.1). We have $w_{ij} = f(x_{ij})$ and $\delta_{ij} = \log x_{ij}$. The difference is that in Equation 2.7, we just have one set of word embeddings rather than separate sets of word embeddings ($\boldsymbol{B}$) and context embeddings ($\boldsymbol{A}$). However, if $\boldsymbol{X}$ is a symmetric matrix (for example the word-word co-occurrence matrix), then because of the non-identifiability of the solution set (see Chapter 3) we can achieve $\boldsymbol{A} = \boldsymbol{B}^T$ by multiplying both $\boldsymbol{A}$ and $\boldsymbol{B}$ by a non-singular $r \times r$ matrix. Hence, with the additional bias terms included, GloVe can be viewed as a generalization of MDS.

2.8 Datasets

In this section we outline the datasets used in this thesis.

Training data. Most of the results on real data given in this thesis are based on the Corpus of Historical American English (COHA) ([Davies, 2010, 2012]). This consists of a collection of novels, non-fiction, newspaper and magazine articles published between the 1810s and 2000s, containing over 400 million words in total. The documents are not evenly distributed across this time period – there are relatively few documents for the early years. However, it is balanced so that the proportion of documents in each genre remains the same across the time frame. The data can be accessed, for a fee, at `https://www.english-corpora.org/coha/`.

We also make use of word embeddings trained on the Google News dataset, which contains about 100 billion words. These are available from `https://code.google.com/archive/p/word2vec/`.

Test data. In Chapters 3 and 4 we make use of some test datasets for word embeddings. For word similarity tasks (see Section 2.2), we use several different test sets, which are summarised in Table 2.6. For word analogy tasks, we use the Google Analogy test set, which contains 19544 analogies to be evaluated. There are 14 different types of analogy, which are listed in Table 2.7.

Test set	Description	Reference
MTurk-287	287 word pairs assessed by similarity	[Radinsky et al., 2011]
MTurk-771	771 word pairs assessed by relatedness	[Halawi et al., 2012]
RW	rare words	[Luong et al., 2014]
SimLex-999	999 pairs of words assessed by similarity	[Hill et al., 2015]
SimVerb-3500	3500 pairs of verbs assessed by semantic similarity	[Gerz et al., 2016]
Verb-143	143 pairs of verbs	[Baker et al., 2014]
WordSim-353	353 pairs of words assessed by relatedness	[Finkelstein et al., 2001]
YP-130	130 pairs of verbs	[Yang and Powers, 2005]

Table 2.6: Examples of test sets used for word similarity tasks.

Type of analogy	Number of tasks
Country–Capital (common)	506
Country–Capital (less common)	4524
Country–Currency	866
US city–State	2467
Male–Female	586
Adjective–Adverb	992
Adjective–Opposite	812
Adjective–Comparative	1332
Adjective–Superlative	1122
Verb infinitive–Present participle	1056
Country–Demonym	1599
Present participle–Past tense	1560
Singular noun–Plural	1332
Verb infinitive–3rd person present form	870

Table 2.7: Categories of analogy tasks in the Google Analogy test set [Mikolov et al., 2013a], with the number of tasks in each category.

Chapter 3

Non-identifiability of word embeddings

In this chapter, we investigate the issue of non-identifiability in word embedding methods: that is, the existence of multiple sets of embeddings which optimize the objective function. We explain how this arises as a consequence of objective functions being based on inner products between word and context embeddings. We explore what the consequences are for the assessment of word embeddings on test data, and propose two solutions.

3.1 Introduction

In Chapter 2, we looked at several different word embedding methods, including LSA (Section 2.1), SGNS (Section 2.3.6), and GloVe (Section 2.3.7). Each of these methods is implemented by finding a matrix of word embeddings which optimizes some objective function. In this chapter we show that a set of embeddings generated from any one of these methods is non-identifiable: for any given embedding set that optimizes the objective function, we can find multiple (indeed, infinitely many) other solutions that do the same. These different solutions may perform very differently on test data (see Section 2.2), as we shall show later on (see Tables 3.4, 3.6, and

3.7). This makes it difficult to compare different embedding methods.

In this chapter, we first show (Section 3.2) that most popular word embedding methods can be expressed implicitly in terms of matrix factorization, and we explain how this leads to non-identifiability. We then investigate the implications of this for assessing performance on tasks (Section 3.3), both for word similarity and analogy tasks, and discuss some possible solutions (Section 3.4).

To illustrate the problem of non-identifiability, we take the general case of matrix factorization (on which most word embedding methods are based). Let $\boldsymbol{X}$ be an $n \times p$ matrix. To find a low-rank matrix factorization of $\boldsymbol{X}$, we choose a rank $r < \min\{n, p\}$, and solve

$$(\boldsymbol{A}^*, \boldsymbol{B}^*) = \underset{\boldsymbol{A}_{n\times r}, \boldsymbol{B}_{r\times p}}{\arg\min} ||\boldsymbol{X} - \boldsymbol{A}\boldsymbol{B}||, \tag{3.1}$$

where $||\cdot||$ denotes some norm. If we impose no constraints on the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ other than requiring them to be of dimension $n \times r$ and $r \times p$ respectively, then $\boldsymbol{A}$ and $\boldsymbol{B}$ cannot be uniquely determined. If $(\boldsymbol{A}^*, \boldsymbol{B}^*)$ is a minimum of Equation 3.1, then we can take any invertible $r \times r$ matrix $\boldsymbol{C}$, and set $\tilde{\boldsymbol{A}} = \boldsymbol{A}^*\boldsymbol{C}^{-1}$ and $\tilde{\boldsymbol{B}} = \boldsymbol{C}\boldsymbol{B}^*$. Then $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}} = \boldsymbol{A}^*\boldsymbol{B}^*$, so $\left(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}\right)$ is also a solution of the minimization problem. However, we can make $(\boldsymbol{A}^*, \boldsymbol{B}^*)$ unique by imposing further constraints; this will be discussed in Section 3.4.1.

3.2 Non-identifiability of word embeddings

We now consider some of the different word embedding methods discussed in Chapter 2: LSA (and other SVD-based methods), GloVe, and SGNS. We show that they all have the same non-identifiability issue as Equation 3.1.

First, we show that the objective function for each method can be written in the form

$$f\left(\boldsymbol{X}, \boldsymbol{A}\boldsymbol{B}\right),$$

where $\boldsymbol{X}$ is some representation of the data (e.g. the document-term matrix or co-occurrence matrix), and $\boldsymbol{A}$ and $\boldsymbol{B}$ are the matrices of context and word embeddings respectively. This means that the objective function is a function of the data $\boldsymbol{X}$ and the matrix product $\boldsymbol{AB}$ only. As in Equation 3.1, replacing $\boldsymbol{A}$ and $\boldsymbol{B}$ with $\tilde{\boldsymbol{A}} = \boldsymbol{AC}^{-1}$ and $\tilde{\boldsymbol{B}} = \boldsymbol{CB}$ does not change the value of the objective function, so non-identifiability arises in the same way.

3.2.1 LSA

The LSA objective function is

$$f(\boldsymbol{X}, \boldsymbol{AB}) := ||\boldsymbol{X} - \boldsymbol{AB}||_F,$$

where $||\cdot||_F$ denotes the Frobenius norm:

$$||\boldsymbol{M}||_F = \left(\sum_i \sum_j m_{ij}^2 \right)^{1/2}. \tag{3.2}$$

This is clearly a function only of $\boldsymbol{X}$ and $\boldsymbol{AB}$. The same applies to other ways of generating word embeddings using SVD, such as those discussed in Section 2.1, which use the same objective function with a different $\boldsymbol{X}$. (It is true that when we compute the SVD it gives us matrices with orthonormal columns, but we can still multiply $\boldsymbol{A}$ and $\boldsymbol{B}$ by any non-singular matrix without changing the value of the objective function.)

3.2.2 GloVe

The objective function to be minimized for GloVe [Pennington et al., 2014] is

$$f(\boldsymbol{X}, \boldsymbol{AB}) := \sum_{i=1}^{p} \sum_{j=1}^{p} h\left(x_{ij}\right) \left(\boldsymbol{a}_i^T \boldsymbol{b}_j + c_i + \tilde{c}_j - \log x_{ij} \right)^2,$$

where h is an element-wise weighting function and c_i and $\tilde{c}_j$ are bias parameters estimated during optimization. As for SGNS, this depends on the word and context embeddings only through the elements of the matrix $\boldsymbol{AB}$, and so again can be written in the form $f\left(\boldsymbol{X}, \boldsymbol{AB}\right)$.

3.2.3 SGNS

The SGNS objective function [Levy and Goldberg, 2014b] is

$$f(\boldsymbol{X}, \boldsymbol{AB}) := \sum_{i=1}^{p}\sum_{j=1}^{p} x_{ij} \log\left(\sigma\left(\boldsymbol{a}_i^T \boldsymbol{b}_j\right)\right) + k\frac{\sum_{l=1}^{p} x_{il} \sum_{m=1}^{p} x_{mj}}{\sum_{i=1}^{p}\sum_{j=1}^{p} x_{ij}} \log\left(\sigma\left(-\boldsymbol{a}_i^T \boldsymbol{b}_j\right)\right),$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ and k is a hyperparameter that can take positive integer values.

This depends on the word and context embeddings only through the inner products $\boldsymbol{a}_i^T \boldsymbol{b}_j$, which are the elements of the matrix $\boldsymbol{AB}$. Therefore we can write the objective function as $f(\boldsymbol{X}, \boldsymbol{AB})$.

Remark. This problem does not occur with the PPMI method, where the word embeddings are taken to be the rows of the PPMI of the co-occurrence matrix (see Section 2.3.8). This is because there is no auxiliary matrix $\boldsymbol{A}$ for this method.

Figure 3.1 illustrates the possible result of a set of word embeddings $\boldsymbol{B}$ being multiplied by a non-singular matrix $\boldsymbol{C}$. In the left-hand graph the word embeddings are divided into two groups, and words tend to be closest to other words that have similar meanings. In the right-hand graph, it is difficult to distinguish the two groups, and in some cases unrelated words are closer together than words that are similar.

3.3 Assessing performance of word embeddings

We have seen that some popular word embedding methods are non-identifiable. We now consider the implications of this on how word embeddings are assessed. We show that different sets of embeddings that are jointly optimal with respect to the objective function may perform quite differently on test data, which makes it hard to compare the performance of different methods.

As discussed in Section 2.2, word embeddings are assessed in terms of how well they perform on a series of tasks. We consider two types of tasks: *word similarity*

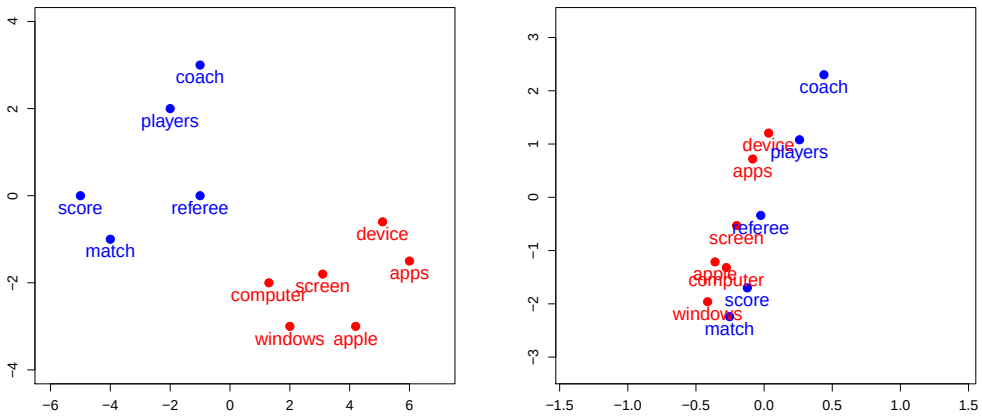

Figure 3.1: The left-hand graph shows a set of simulated word embeddings in two dimensions. The embeddings divide clearly into two main groups. The right-hand graph shows the result of the same set of embeddings having being multiplied by a non-singular transformation matrix $\boldsymbol{C} \in \mathbb{R}^{2\times 2}$. The different groups are no longer distinct, potentially leading to quite different inferences about the relationships between the words based on the embeddings.

and *word analogy*. Details are given in Chapter 2, but we summarise here how this is implemented.

First, we introduce some notation. Let $g(\boldsymbol{D}, \boldsymbol{B})$ denote the test score, where $\boldsymbol{D}$ is the test data, and $\boldsymbol{B}$ is the embedding set. To distinguish the two types of tasks, we write $g_S(\boldsymbol{D}_S, \boldsymbol{B})$ for word similarity tasks and $g_A(\boldsymbol{D}_A, \boldsymbol{B})$ for word analogy tasks. We use g and $\boldsymbol{D}$, without subscripts, to refer generically to the test function or test data for either type of task.

3.3.1 Word similarity tasks

For word similarity tasks, the test data $\boldsymbol{D}_S$ consists of a list of triples (w_{i_1}, w_{i_2}, y_i), where w_{i_1} and w_{i_2} are words, and y_i is a human-assigned similarity score for words w_{i_1} and w_{i_2}. Some examples of these are given in Table 3.1, which shows the first few rows of the WordSim-353 test set [Finkelstein et al., 2001]. We calculate $g_S(\boldsymbol{D}_S, \boldsymbol{B})$ as follows. For each i, we let $\boldsymbol{b}(w_{i_1})$ and $\boldsymbol{b}(w_{i_2})$ represent the word embeddings corresponding to words w_{i_1} and w_{i_2}. We denote the cosine similarity between these two embeddings by z_i:

$$z_i = \cos\left(\frac{\langle \boldsymbol{b}(w_{i_1}), \boldsymbol{b}(w_{i_2})\rangle}{|\boldsymbol{b}(w_{i_1})| \cdot |\boldsymbol{b}(w_{i_2})|}\right).$$

The test score is then given (as in Equation 2.1) by

$$g_S(\boldsymbol{D}_S, \boldsymbol{B}) = \rho(\boldsymbol{y}(\boldsymbol{D}_S), \boldsymbol{z}(\boldsymbol{D}_S, \boldsymbol{B})),$$

where ρ is the Spearman or Pearson correlation coefficient between $\boldsymbol{y}$ and $\boldsymbol{z}$. The idea is that a higher correlation between $\boldsymbol{y}$ and $\boldsymbol{z}$ indicates that word pairs that have high human-assigned similarity scores (y_i) also have high cosine similarity between their embeddings (z_i).

3.3.2 Word analogy tasks

For analogy tasks, the test data $\boldsymbol{D}_A$ is a list of sets of four words $(w_{i_1}, w_{i_2}, w_{i_3}, w_{i_4})$. Each set forms an analogy, "w_{i_1} is to w_{i_2} as w_{i_3} is to w_{i_4}." For example, "*king* is

Word 1	Word 2	Similarity score
love	sex	6.77
tiger	cat	7.35
tiger	tiger	10.00
book	paper	7.46
computer	keyboard	7.62

Table 3.1: The first few rows of the WordSim-353 test set [Finkelstein et al., 2001]

to *queen* as *man* is to *woman*." The aim is to use the set of word embeddings to predict the final word.

The usual way to solve an analogy is to find $\hat{w}$ that satisfies

$$\hat{w} = \underset{\{w_1,\ldots,w_p\}}{\arg\max} \cos\left(\langle \boldsymbol{b}(w_{i_1}) - \boldsymbol{b}(w_{i_2}) + \boldsymbol{b}(w_{i_3}), \boldsymbol{b}(w)\rangle\right), \tag{3.3}$$

where $\boldsymbol{b}(w_k)$ denotes the embedding corresponding to word w_k. An alternative method used in [Levy et al., 2015] is to solve

$$\hat{w}_i = \underset{\{w_1,\ldots,w_p\}}{\arg\max} \frac{\cos\left(\langle \boldsymbol{b}(w), \boldsymbol{b}(w_{i_2})\rangle\right)\cos\left(\langle \boldsymbol{b}(w), \boldsymbol{b}(w_{i_3})\rangle\right)}{\cos\left(\langle \boldsymbol{b}(w), \boldsymbol{b}(w_{i_1})\rangle\right)}. \tag{3.4}$$

In either case, $g_A(\boldsymbol{D}_A, \boldsymbol{B})$ is the proportion of analogies for which the predicted final word is correct:

$$g_A(\boldsymbol{D}_A, \boldsymbol{B}) = \frac{1}{d}\sum_{i=1}^{d} \mathbb{I}_{\hat{w}_i = w_{i_4}},$$

where d is the number of analogies in $\boldsymbol{D}_A$ and $\mathbb{I}$ is the indicator function:

$$\mathbb{I}_{\hat{w}_i = w_{i_4}} = \begin{cases} 1 & \text{if } \hat{w}_i = w_{i_4} \\ 0 & \text{otherwise.} \end{cases}$$

In both Equations 3.3 and 3.4, $\hat{w}_i$, and hence $g_A(\boldsymbol{D}_A, \boldsymbol{B})$, depends on the word embeddings only through their cosine similarities.

We have shown that for both word similarity and analogy tasks, g_S and g_A only depend on the word embedding set $\boldsymbol{B}$ through the cosine similarities between the

embeddings. Hence, for any transformation of the embeddings to which the cosine similarity function is invariant, the test functions g_S and g_A will also be invariant.

We now define some notation which will be used in this section.

Definition 5. $\mathsf{GL}(r) := \{\boldsymbol{C} \in \mathbb{R}^{r\times r} : |\boldsymbol{C}| \neq 0\}$

Definition 6. $\mathsf{O}(r) := \{\boldsymbol{Q} \in \mathbb{R}^{r\times r} : \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}_r\}$

Definition 7. $c\mathsf{O}(r) := \{c\boldsymbol{Q} : c > 0, \boldsymbol{Q} \in \mathsf{O}(r)\}$

Remark. We have $c > 0$ in the definition of $c\mathsf{O}(r)$, because a sign change can be absorbed into the matrix $\boldsymbol{Q} \in \mathsf{O}(r)$.

Lemma 2 states that the cosine similarity between two word embeddings is invariant to orthogonal and scale transformations (elements of $c\mathsf{O}(r)$), but not to general non-linear transformations (elements of $\mathsf{GL}(r)$).

Lemma 2. *Suppose we have context and word embedding matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, and define $\tilde{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{C}^{-1}$ and $\tilde{\boldsymbol{B}} = \boldsymbol{C}\boldsymbol{B}$, where $\boldsymbol{C} \in \mathsf{GL}(r)$. If $\boldsymbol{C} \in c\mathsf{O}(r)$, then the cosine similarity between columns i and j of $\boldsymbol{B}$ is the same as that between columns i and j of $\tilde{\boldsymbol{B}}$.*

Proof. If $\boldsymbol{C} \in c\mathsf{O}(r)$, then there exist $c > 0$ and $\boldsymbol{Q} \in \mathsf{O}(r)$ such that $\boldsymbol{C} = c\boldsymbol{Q}$. Thus,

$$\begin{aligned}\cos\left(\langle\tilde{\boldsymbol{b}}_i, \tilde{\boldsymbol{b}}_j\rangle\right) &= \frac{\boldsymbol{b}_i^T\boldsymbol{C}\boldsymbol{C}^T\boldsymbol{b}_j}{|\boldsymbol{C}|^2 \cdot |\boldsymbol{b}_i| \cdot |\boldsymbol{b}_j|} = \frac{\boldsymbol{b}_i^T c^2\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{b}_j}{|c\boldsymbol{Q}|^2 \cdot |\boldsymbol{b}_i| \cdot |\boldsymbol{b}_j|}|| \\ &= \frac{\boldsymbol{b}_i^T\left(c^2\boldsymbol{I}_r\right)\boldsymbol{b}_j}{c^2 \cdot |\boldsymbol{b}_i| \cdot |\boldsymbol{b}_j|} = \frac{\boldsymbol{b}_i^T\boldsymbol{b}_j}{|\boldsymbol{b}_i| \cdot |\boldsymbol{b}_j|} = \cos\left(\langle\boldsymbol{b}_i, \boldsymbol{b}_j\rangle\right).\end{aligned}$$

□

Thus, if we have two embedding sets $\boldsymbol{B}$ and $\tilde{\boldsymbol{B}}$, such that $\tilde{\boldsymbol{B}} = \boldsymbol{C}\boldsymbol{B}$, then if $\boldsymbol{C}$ can be written as $\boldsymbol{C} = c\boldsymbol{Q}$, where c is a scalar and $\boldsymbol{Q}$ an orthogonal matrix, then these two embedding sets will give the same value for both f and g. If $\boldsymbol{C}$ is not of this form then we will still have $f(\boldsymbol{X}, \boldsymbol{A}\boldsymbol{B}) = f(\boldsymbol{X}, (\boldsymbol{A}\boldsymbol{C}^{-1})(\boldsymbol{C}\boldsymbol{B}))$, but we may not have $g(\boldsymbol{D}, \boldsymbol{B}) = g(\boldsymbol{D}, \boldsymbol{C}\boldsymbol{B})$.

The non-identifiability issue is apparent when we consider word embeddings based on SVD. Here, the objective function is

$$f = ||\boldsymbol{X} - \boldsymbol{AB}||_F^2. \tag{3.5}$$

We know that, to minimize f in Equation 3.5, the matrix product $\boldsymbol{AB}$ must satisfy

$$\boldsymbol{AB} = \boldsymbol{U}_r \boldsymbol{\Phi}_r \boldsymbol{V}_r^T,$$

where $\boldsymbol{\Phi}_r$ is a diagonal matrix with its diagonal elements equal to the r largest singular values of $\boldsymbol{X}$, and $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$ are $n \times r$ and $p \times r$ matrices with orthonormal columns equal to the corresponding singular vectors [Eckart and Young, 1936]. There are thus three matrices, which we need to decide how to divide into $\boldsymbol{A}$ and $\boldsymbol{B}$. For example, we could take

$$\boldsymbol{A} = \boldsymbol{U}_r, \qquad \boldsymbol{B} = \boldsymbol{\Phi}_r \boldsymbol{V}_r^T,$$

which is called by [Bullinaria and Levy, 2012] the "simple SVD" solution. However, we could alternatively take

$$\boldsymbol{A} = \boldsymbol{U}_r \boldsymbol{\Phi}_r, \qquad \boldsymbol{B} = \boldsymbol{V}_r^T,$$

or indeed, as is investigated in [Bullinaria and Levy, 2012] and [Turney and Littman, 2002],

$$\boldsymbol{A} = \boldsymbol{U}_r \boldsymbol{\Phi}_r^{1-\alpha}, \qquad \boldsymbol{B} = \boldsymbol{\Phi}_r^{\alpha} \boldsymbol{V}_r^T,$$

for $\alpha \in \mathbb{R}$. For each α, $\boldsymbol{\Phi}_r^{\alpha} \boldsymbol{V}_r^T$ is a different element of the solution set, with the set of such solutions forming a subset of the whole solution set (as $\{\boldsymbol{\Phi}_r^{\alpha} : \alpha \in \mathbb{R}\}$ is a subset of $\mathsf{GL}(r)$). [Bullinaria and Levy, 2012] and [Turney and Littman, 2002] investigate the performance of different values of α; although [Bullinaria and Levy, 2012] only consider a few different values, they observe that that $\alpha = \frac{1}{2}$ tends to perform better than $\alpha = 0$ or 1. Later, in Section 3.4.2, we will consider optimizing over the set of possible α's. However, we will next investigate in more depth the effect of the discrepancy between the invariances of f and those of g.

3.3.3 Incompatibility between invariances of f and g

In order to investigate the implications of non-identifiability upon the test function g, and in particular the discrepancy between the set of transformations to which g is invariant and the set of transformations to which f is invariant, it is helpful to formalize the situation using a group-theoretic framework.

We are interested in the set of transformations of the embedding set $\boldsymbol{B} \in \mathbb{R}^{r \times p}$ to which the objective function f is invariant, but not the test function g. First we characterize the sets of transformations to which each of f and g are invariant (Lemmas 3 and 4).

Lemma 3. *f is invariant to transformations of the embedding set by $\mathsf{GL}(r)$, in the sense that for all $\boldsymbol{C} \in \mathsf{GL}(r)$, $f(\boldsymbol{X}, (\boldsymbol{A}\boldsymbol{C}^{-1})(\boldsymbol{C}\boldsymbol{B})) = f(\boldsymbol{X}, \boldsymbol{A}\boldsymbol{B})$.*

Lemma 4. *g is invariant to transformations of the embedding set by elements of $c\mathsf{O}(r)$: for all $\boldsymbol{C} \in c\mathsf{O}(r)$, $g(\boldsymbol{D}, \boldsymbol{C}\boldsymbol{B}) = g(\boldsymbol{D}, \boldsymbol{B})$.*

Let $\mathcal{F}_r$ denote the set of transformations to which f is invariant but g is not:

$$\mathcal{F}_r = \{\boldsymbol{C} \in \mathbb{R}^{r \times r} : \boldsymbol{C} \in \mathsf{GL}(r), \boldsymbol{C} \notin c\mathsf{O}(r)\}.$$

The dimensions of $\mathsf{GL}(r)$ and $c\mathsf{O}(r)$ are r^2 and $\frac{1}{2}r(r-1)+1$ respectively. Hence, the dimension of $\mathcal{F}_r$ is $r^2 - \frac{1}{2}r(r-1) - 1 = \frac{1}{2}r(r+1)$.

In order to characterize this set, let $\mathsf{UT}^+(r)$ be the set of upper triangular $r \times r$ matrices with positive diagonal elements which sum to 1:

Definition 8. $\mathsf{UT}^+(r) := \{\boldsymbol{R} \in \mathsf{GL}(r) : \ r_{ij} = 0 \text{ for } j > i,\ r_{ii} > 0,\ \sum_i r_{ii} = 1\ \}$

We note that as $\boldsymbol{C} \in \mathsf{GL}(r)$ is non-singular, it can be decomposed uniquely into the form $\boldsymbol{C} = \boldsymbol{Q}\boldsymbol{R}$, where $\boldsymbol{Q} \in \mathsf{O}(r)$ and $\boldsymbol{R}$ is upper triangular [Nicholson, 2019, p. 454–455]. Since transformation of the embedding set by an orthogonal $\boldsymbol{Q}$ does not have an effect on g, we need only consider the upper triangular component $\boldsymbol{R}$. However, scale transformations also do not affect g, so we can also take out a

scale factor c and write $\boldsymbol{C} = c\boldsymbol{Q}\boldsymbol{R}$. To ensure that this decomposition is unique, we require $c > 0$ (a change of sign can be absorbed into $\boldsymbol{Q}$), and we also need a constraint on the scale of $\boldsymbol{R}$, for example requiring $\sum_{i=1}^{r} r_{ii} = 1$. Hence, each $\boldsymbol{C} \in \mathsf{GL}(r)$ can be decomposed uniquely into $\boldsymbol{C} = c\boldsymbol{Q}\boldsymbol{R}$, where $c > 0$, $\boldsymbol{Q} \in \mathsf{O}(r)$, and $\boldsymbol{R} \in \mathsf{UT}^{+}(r)$. This is useful because only the $\boldsymbol{R}$ part of the decomposition has an effect on g.

Remark. We could equivalently decompose $\boldsymbol{C}$ as $\boldsymbol{C} = c\boldsymbol{L}\boldsymbol{Q}$, where $\boldsymbol{L}$ is a lower triangular matrix with positive diagonal elements that sum to 1, and c and $\boldsymbol{Q}$ are as above.

We now use this to show that $\mathcal{F}_r$ can be identified with $\mathsf{UT}^{+}(r)$ (Proposition 1). To do this, we first need the following lemma:

Lemma 5. *$c\mathsf{O}(r)$ is a subgroup of $\mathsf{GL}(r)$.*

Proof. To show this, we need to show that each of the following hold:

1. $c\mathsf{O}(r)$ contains the identity matrix $\boldsymbol{I}_r$
2. For all $\boldsymbol{C}_1, \boldsymbol{C}_2 \in c\mathsf{O}(r)$, $\boldsymbol{C}_1\boldsymbol{C}_2 \in c\mathsf{O}(r)$
3. For all $\boldsymbol{C} \in c\mathsf{O}(r)$, $\boldsymbol{C}^{-1} \in c\mathsf{O}(r)$

Since $\boldsymbol{I}_r$ is an orthogonal matrix, it must be an element of $c\mathsf{O}(r)$, since $\mathsf{O}(r) \subset c\mathsf{O}(r)$ (with $c = 1$).

For the second condition, if $\boldsymbol{C}_1, \boldsymbol{C}_2 \in c\mathsf{O}(r)$, then there exist $c_1, c_2 > 0$ and $\boldsymbol{Q}_1, \boldsymbol{Q}_2 \in \mathsf{O}(r)$ such that $\boldsymbol{C}_1 = c_1\boldsymbol{Q}_1$ and $\boldsymbol{C}_2 = c_2\boldsymbol{Q}_2$. Then

$$\boldsymbol{C}_1\boldsymbol{C}_2 = (c_1\boldsymbol{Q}_1)(c_2\boldsymbol{Q}_2) = (c_1c_2)(\boldsymbol{Q}_1\boldsymbol{Q}_2) \in c\mathsf{O}(r),$$

since the product of two orthogonal matrices is an orthogonal matrix, and $c_1, c_2 > 0 \Rightarrow c_1c_2 > 0$.

Finally, if $\boldsymbol{C} = c\boldsymbol{Q} \in c\mathsf{O}(r)$, then $\boldsymbol{C}^{-1} = c^{-1}\boldsymbol{Q}^{-1} = c^{-1}\boldsymbol{Q}^{T} \in c\mathsf{O}(r)$. Thus, $c\mathsf{O}(r)$ is a subgroup of $\mathsf{GL}(r)$ □

Definition 9. Let $\mathsf{GL}(r)/c\mathsf{O}(r)$ denote the set of left cosets of $\mathsf{GL}(r)$ with respect to $c\mathsf{O}(r)$.

Remark. Lemma 5 is necessary to ensure that $\mathsf{GL}(r)/c\mathsf{O}(r)$ is well-defined.

Proposition 1 (Structure of $\mathcal{F}_r$)**.** *The set $\mathcal{F}_r$ can be identified with the set $\mathsf{UT}^+(r)$ of upper triangular matrices that have positive diagonal elements which sum to 1.*

Proof. First, we define an equivalence relation on $\mathsf{GL}(r)$, and show that the equivalence classes are the same as the right cosets of $c\mathsf{O}(r)$ in $\mathsf{GL}(r)$. We show that the set of equivalence classes, and hence the set $\mathsf{GL}(r)/c\mathsf{O}(r)$, is bijective with $\mathsf{UT}^+(r)$.

We showed earlier that each element $\boldsymbol{C} \in \mathsf{GL}(r)$ can be uniquely written as $c\boldsymbol{Q}\boldsymbol{R}$, where $c > 0$, $\boldsymbol{Q} \in \mathsf{O}(r)$ and $\boldsymbol{R} \in \mathsf{UT}^+(r)$. Hence, we can define a mapping $h : \mathsf{GL}(r) \to \mathsf{UT}^+(r)$ such that

$$h(\boldsymbol{C}) = \boldsymbol{R}. \tag{3.6}$$

Since $c\mathsf{O}(r)$ is a subgroup of $\mathsf{GL}(r)$, we can define an equivalence relation $\sim$ on $\mathsf{GL}(r)$ such that for two matrices $\boldsymbol{C}_1$ and $\boldsymbol{C}_2 \in \mathsf{GL}(r)$, $\boldsymbol{C}_1 \sim \boldsymbol{C}_2$ if there exists a matrix $\tilde{\boldsymbol{Q}} \in c\mathsf{O}(r)$ such that $\boldsymbol{C}_1 = \tilde{\boldsymbol{Q}}\boldsymbol{C}_2$.

We need the following proposition:

Proposition 2. *Define an equivalence relation between two matrices $\boldsymbol{C}_1$ and $\boldsymbol{C}_2 \in \mathsf{GL}(r)$ such that*

$$\boldsymbol{C}_1 \sim \boldsymbol{C}_2 \iff \text{there exist } c > 0, \boldsymbol{Q} \in \mathsf{Q}(r) \text{ such that } \boldsymbol{C}_1 = c\boldsymbol{Q}\boldsymbol{C}_2$$

Two matrices $\boldsymbol{C}_1$ and $\boldsymbol{C}_2 \in \mathsf{GL}(r)$ belong to the same equivalence class if and only if there exists $\boldsymbol{R} \in \mathsf{UT}^+(r)$ and $c_1\boldsymbol{Q}_1, c_2\boldsymbol{Q}_2 \in c\mathsf{O}(r)$ such that $\boldsymbol{C}_1 = c_1\boldsymbol{Q}_1\boldsymbol{R}$ and $\boldsymbol{C}_2 = c_2\boldsymbol{Q}_2\boldsymbol{R}$.

Proof. Suppose that $\boldsymbol{C}_1 \sim \boldsymbol{C}_2$. Then there exist $c > 0$, $\boldsymbol{Q} \in \mathsf{O}(r)$ such that $\boldsymbol{C}_1 = c\boldsymbol{Q}\boldsymbol{C}_2$. Using the QR decomposition, we can write $\boldsymbol{C}_1 = c_1\boldsymbol{Q}_1\boldsymbol{R}_1$ and $\boldsymbol{C}_2 = c_2\boldsymbol{Q}_2\boldsymbol{R}_2$. Hence, we get

$$\boldsymbol{C}_1 = c_1\boldsymbol{Q}_1\boldsymbol{R}_1 = c\boldsymbol{Q}\left(c_2\boldsymbol{Q}_2\boldsymbol{R}_2\right) = \left(cc_2\right)\left(\boldsymbol{Q}\boldsymbol{Q}_2\right)\boldsymbol{R}_2. \tag{3.7}$$

Since $cc_2 \in \mathbb{R}$ and $\boldsymbol{Q}\boldsymbol{Q}_2 \in \mathsf{O}(r)$, Equation 3.7 gives two QR decompositions for $\boldsymbol{C}_1$. By the uniqueness of the QR decomposition we must have $\boldsymbol{Q}_1 = \boldsymbol{Q}\boldsymbol{Q}_2$ and $\boldsymbol{R}_1 = \boldsymbol{R}_2$. Hence, writing $\boldsymbol{R} = \boldsymbol{R}_1 = \boldsymbol{R}_2$, we have $\boldsymbol{C}_1 = c_1\boldsymbol{Q}_1\boldsymbol{R}$ and $\boldsymbol{C}_2 = c_2\boldsymbol{Q}_2\boldsymbol{R}$.

Now, suppose that there exist $c_1, c_2 > 0$, $\boldsymbol{Q}_1, \boldsymbol{Q}_2 \in \mathsf{O}(r)$ and $\boldsymbol{R} \in \mathsf{UT}^+(r)$ such that $\boldsymbol{C}_1 = c_1\boldsymbol{Q}_1\boldsymbol{R}$ and $\boldsymbol{C}_2 = c_2\boldsymbol{Q}_2\boldsymbol{R}$. Then,

$$\boldsymbol{C}_1 = c_1\boldsymbol{Q}_1\boldsymbol{R} = c_1\boldsymbol{Q}_1\left(c_2^{-1}\boldsymbol{Q}_2\right)^T(c_2\boldsymbol{Q}_2)\,\boldsymbol{R} = c_1c_2^{-1}\boldsymbol{Q}_1\boldsymbol{Q}_2^T\boldsymbol{C}_2, \tag{3.8}$$

where $c_1c_2^{-1}\boldsymbol{Q}_1\boldsymbol{Q}_2^T \in c\mathsf{O}(r)$. Hence, $\boldsymbol{C}_1 \sim \boldsymbol{C}_2$. □

Thus, each of the equivalence classes $[\boldsymbol{C}]$ is associated with a unique element $\boldsymbol{R} \in \mathsf{UT}^+(r)$, so the set of such classes is bijective with this set. Thus, the set $\mathsf{GL}(r)/c\mathsf{O}(r)$ can be identified with the set $\mathsf{UT}^+(r)$. □

Figures 3.2 and 3.3 show how the test scores for a set of embeddings can vary when the embedding set is multiplied by a series of random elements of $\mathsf{UT}^+(r)$. In Figure 3.2, the embeddings are LSA embeddings trained on COHA, which are multiplied by a series of random diagonal matrices in $\mathsf{UT}^+(r)$; in Figure 3.3, the embeddings are SGNS embeddings trained on the Google News dataset, and are multiplied by a series of random elements of $\mathsf{UT}^+(r)$. In each case the test scores vary as the embeddings are multiplied by different non-singular matrices, with the difference between the smallest and largest scores attained being between around 0.06 and 0.15. (For context, in [Pennington et al., 2014] GloVe is reported to outperform SGNS on the WordSim-353 test set by a margin of 0.03 and SVD by 0.019, when trained on the same training data. Hence, the differences in performance in Figures 3.2 and 3.3 are of the order that would be significant for model comparison.) We find that in each case, although the test score obtained on the original embedding set falls at the upper end of the range, it is possible to obtain higher scores by transforming the embedding set by an element of $\mathsf{UT}^+(r)$. It is not clear why the original embeddings should always be at the higher end of the range: this

may be due to tuning of hyperparameters to optimize performance, or to some form of regularization being imposed during optimization.

3.4 Addressing identifiability

We now consider two methods for dealing with identifiability: imposing constraints on the solutions (Section 3.4.1) and optimizing g over $\mathsf{UT}^+(r)$ (Section 3.4.2).

3.4.1 Imposing identifiability conditions

One solution to the problem of non-identifiability is to apply constraints to the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, reducing the set of possible transformations from $\mathsf{GL}(r)$ to $c\mathsf{O}(r)$. In some cases, we may want to impose stricter constraints to reduce the set of transformations to $\{I_r\}$, so that $\boldsymbol{A}$ and $\boldsymbol{B}$ are completely uniquely identified. For example, this is necessary when we want to find the asymptotic distributions of the parameters in order to make inferences about them; the parameters must be uniquely identifiable for their asymptotic distributions to be well-defined. (This will be of interest in Chapter 5.)

Since $\boldsymbol{C}$ contains r^2 free parameters, in order to ensure complete identifiability ($\boldsymbol{C} = \boldsymbol{I}_r$) we need to impose r^2 constraints. If we only want to restrict the set of transformations to $\mathsf{GL}(r)/c\mathsf{O}(r)$, then we only need $\frac{1}{2}r(r+1) - 1$ constraints.

One way of ensuring complete identifiability is to require $\boldsymbol{A}^T\boldsymbol{A}$ to be diagonal and $\boldsymbol{B}\boldsymbol{B}^T = \boldsymbol{I}_r$ (or $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_r$ and $\boldsymbol{B}\boldsymbol{B}^T$ diagonal). This is done in SVD when we take $\boldsymbol{A} = \boldsymbol{U}_r\boldsymbol{\Phi}_r$ and $\boldsymbol{B} = \boldsymbol{V}_r^T$. (If we only want to restrict the set of possible transformations to $c\mathsf{O}(r)$, it is sufficient to require that $\boldsymbol{B}\boldsymbol{B}^T = c\boldsymbol{I}_r$ for some c, with no constraints on $\boldsymbol{A}$.) We could also require certain elements of the matrices to be 0 or 1. For example, requiring the first or last $r \times r$ block of $\boldsymbol{B}$ to be the identity matrix (as in e.g. [Amemiya and Fuller, 1987]) ensures that $\boldsymbol{A}$ and $\boldsymbol{B}$ are identifiable.

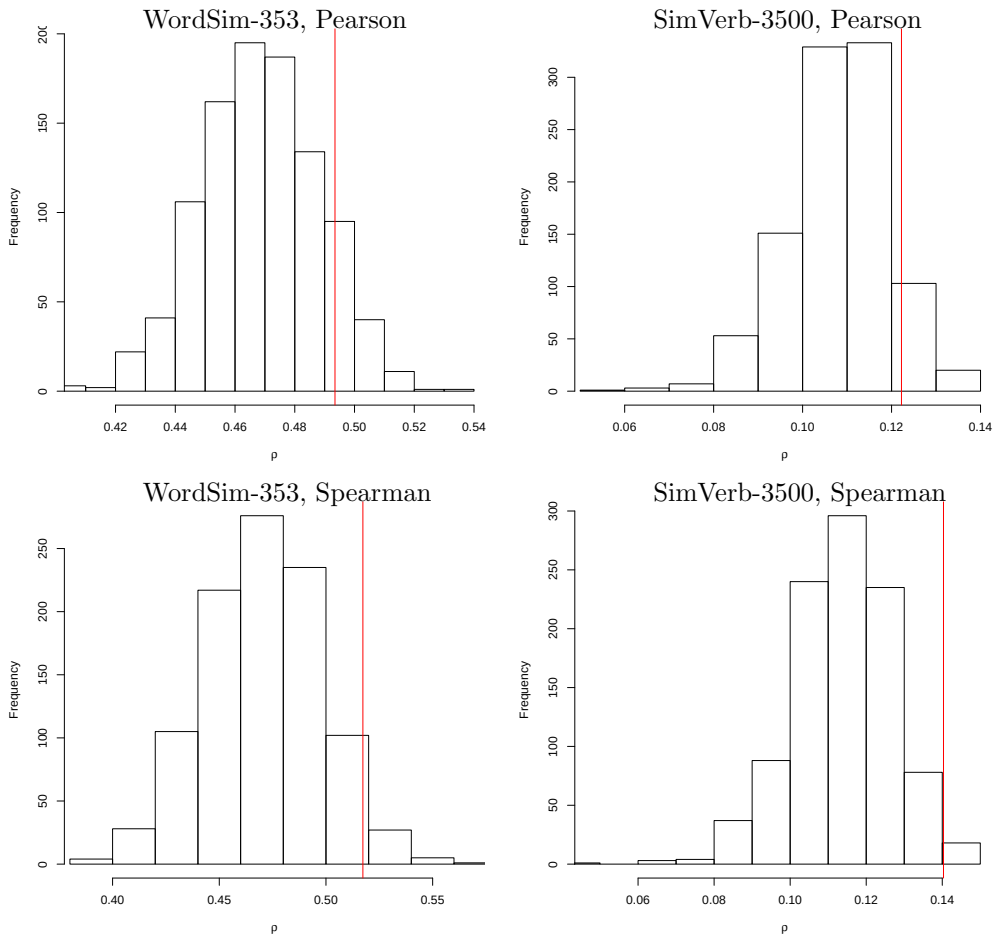

Figure 3.2: Performance of LSA embeddings trained on the COHA document-term matrix, ($r = 300$ and $\boldsymbol{B} = \boldsymbol{V}_r^T$), where the embeddings are multiplied by a series of different diagonal matrices $\boldsymbol{R} \in \mathsf{UT}^+(r)$. For each $\boldsymbol{R}$, the diagonal elements are simulated from $|N(0,1)|$, before the matrix is scaled so that $\sum_i r_{ii} = 1$. The red line on each histogram shows the performance of the original embeddings.

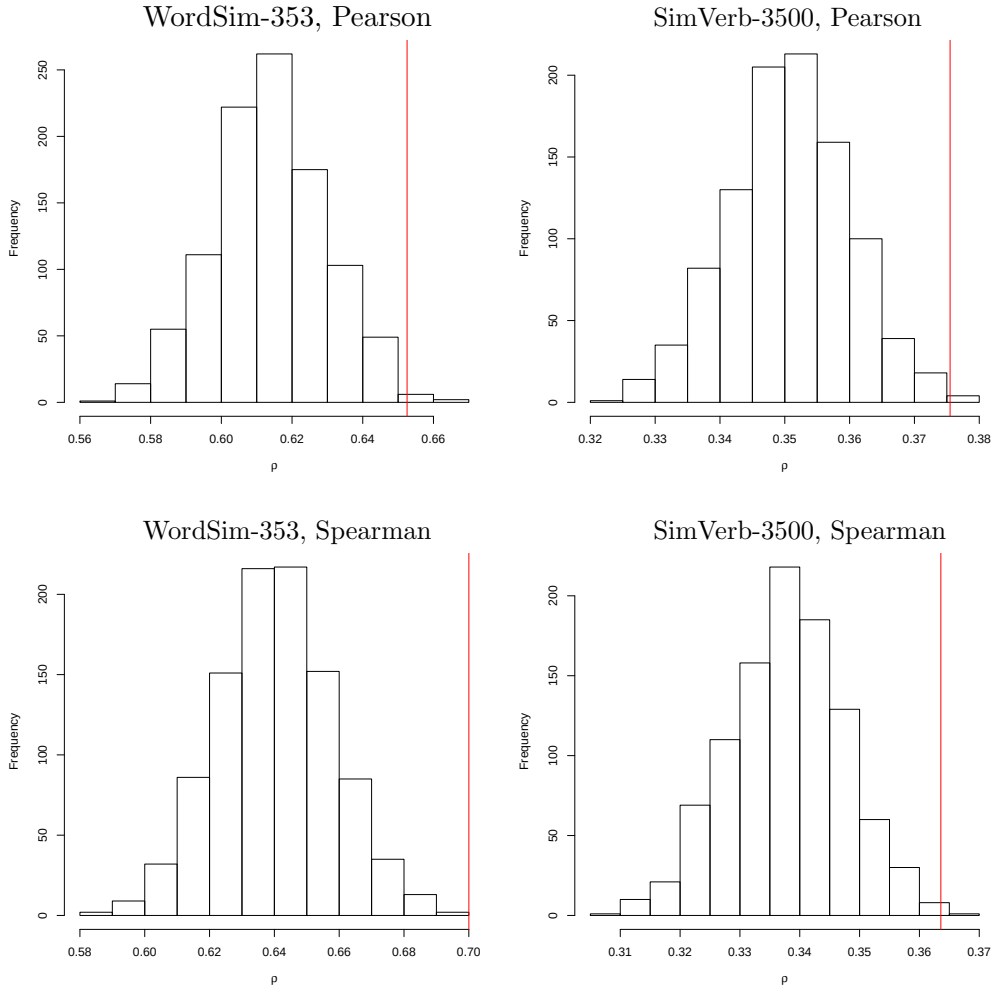

Figure 3.3: Performance of SGNS embeddings, trained on 100 billion words of the Google News dataset with $r = 300$ (downloaded from https://code.google.com/archive/p/word2vec/), where the embeddings are multiplied by a series of different and upper triangular matrices $\boldsymbol{R} \in \mathsf{UT}^{+}(r)$ (with non-zero elements simulated from $|N(0,1)|$). The red line on each histogram shows the performance of the original embeddings.

It is desirable to treat all words equally, rather than having the embeddings depend on the (perhaps arbitrary) ordering of the words, as is the case for the second option above. In SVD, both $\boldsymbol{U}$ and $\boldsymbol{V}$ have orthonormal columns; we can ensure identifiability by requiring the columns of $\boldsymbol{A}$ and rows of $\boldsymbol{B}$ to be orthogonal, and for either the columns of $\boldsymbol{A}$ or the rows of $\boldsymbol{B}$ to have norm 1 (or we can split $\boldsymbol{\Phi}$ in a different way between them). To prevent permutations of the embeddings and multiplication of columns of $\boldsymbol{A}$ and rows of $\boldsymbol{B}$ by -1, we can require the columns of $\boldsymbol{A}$ and rows of $\boldsymbol{B}$ to be ordered by singular value, and the first entry in each column of $\boldsymbol{A}$ to be positive.

Lemma 6 states that if we have two sets of word embeddings $\boldsymbol{B}$ and $\boldsymbol{B}^*$ which both minimize the objective function, then they can be related by a transformation in $c\mathsf{O}(r)$.

Lemma 6. *Let $\mathcal{B} = \{\boldsymbol{B} \in \mathbb{R}^{r\times p} : \boldsymbol{B}\boldsymbol{B}^T = c^2\boldsymbol{I}_r$ for some $c > 0\}$. Given two sets of word embeddings $\boldsymbol{B}$ and $\boldsymbol{B}^*$ that are solutions of the constrained optimization problem*

$$\arg\min_{\boldsymbol{B}\in\mathcal{B}}\{f(\boldsymbol{X}, \boldsymbol{A}\boldsymbol{B})\}, \tag{3.9}$$

we can find a matrix $c\boldsymbol{Q} \in c\mathsf{O}(r)$ such that $\boldsymbol{B}^* = c\boldsymbol{Q}\boldsymbol{B}$.

Proof. Suppose $\boldsymbol{B}$ and $\boldsymbol{B}^*$ both satisfy Equation 3.9. From earlier, we know that there exists $\boldsymbol{C} \in \mathsf{GL}(r)$ such that $\boldsymbol{B}^* = \boldsymbol{C}\boldsymbol{B}$. Since there exist $c, c^* > 0$ such that $\boldsymbol{B}\boldsymbol{B}^T = c^2\boldsymbol{I}_r$ and $\boldsymbol{B}^*\boldsymbol{B}^{*T} = c^{*2}\boldsymbol{I}_r$, we have

$$\boldsymbol{I}_r = c^{*-2}\boldsymbol{B}^*\boldsymbol{B}^{*T} = c^{*-2}\boldsymbol{C}\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{C}^T = c^2c^{*-2}\boldsymbol{C}\boldsymbol{C}^T,$$

so $\boldsymbol{C} \in c\mathsf{O}(r)$. □

Lemma 7 gives a set of constraints that will guarantee the uniqueness of $\boldsymbol{A}$ and $\boldsymbol{B}$, given some conditions on the singular values of $\boldsymbol{A}\boldsymbol{B}$:

Lemma 7. *Given an $n \times p$ matrix $\tilde{\boldsymbol{X}}_r$ of rank r, which has r distinct non-zero singular values, there is only one matrix pair $(\boldsymbol{A}, \boldsymbol{B})$, such that*

$$\tilde{\boldsymbol{X}}_r = \boldsymbol{A}\boldsymbol{B},\ \boldsymbol{A} \in \mathbb{R}^{n\times r}, \boldsymbol{B} \in \mathbb{R}^{r\times p}\ ,$$

and which satisfy the following constraints:

1. *The columns of* $\boldsymbol{A}$ *are orthogonal (*$\boldsymbol{A}^T\boldsymbol{A}$ *is diagonal)*
2. *The rows of* $\boldsymbol{B}$ *are orthonormal (*$\boldsymbol{B}\boldsymbol{B}^T = \boldsymbol{I}_r$*)*
3. *The diagonal elements of* $\boldsymbol{A}^T\boldsymbol{A}$ *are ordered in descending order*
4. *The first entry of each column of* $\boldsymbol{A}$ *is positive*

Proof. By the singular value decomposition theorem [Horn and Johnson, 1991, p. 144], we can find orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ and a diagonal matrix $\boldsymbol{\Phi}$ such that

$$\tilde{\boldsymbol{X}}_r = \boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{V}^T.$$

Since $\tilde{\boldsymbol{X}}_r$ is of rank r, this can be reduced to

$$\tilde{\boldsymbol{X}}_r = \boldsymbol{U}_r\boldsymbol{\Phi}_r\boldsymbol{V}_r^T,$$

where $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$ contain the first r columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively, and $\boldsymbol{\Phi}_r$ contains the $r \times r$ upper left corner of $\boldsymbol{\Phi}$. This decomposition is unique up to the sign of each column of $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$, as long as the r non-zero singular values of $\tilde{\boldsymbol{X}}$ are all unique [Blum et al., 2020, p. 35]. If we require the first element of each column of $\boldsymbol{U}_r\boldsymbol{\Phi}_r$ to be positive, then this decomposition is unique. Hence, $\boldsymbol{A}$ and $\boldsymbol{B}$ that satisfy the constraints can be uniquely identified as $\boldsymbol{U}_r\boldsymbol{\Phi}_r$ and $\boldsymbol{V}_r^T$ respectively. □

Remark. In factor analysis, one common way of ensuring identifiability is to have either $\boldsymbol{\Sigma}_a = \boldsymbol{I}_r$ and $\frac{1}{p}\boldsymbol{B}\boldsymbol{\Sigma}_z^{-1}\boldsymbol{B}^T$ diagonal, or $\boldsymbol{\Sigma}_a$ diagonal and $\frac{1}{p}\boldsymbol{B}\boldsymbol{\Sigma}_z^{-1}\boldsymbol{B}^T = \boldsymbol{I}$. If we require (as in the SVD case) that $\boldsymbol{\Sigma}_z$ be diagonal, then these conditions are equivalent to the SVD ones, up to a scaling factor of $\frac{1}{p}$.

Remark. If $\boldsymbol{X}$ is a symmetric matrix, as is often the case with word embedding methods based on co-occurrence counts, then the problem of generating word embeddings is symmetric in $\boldsymbol{A}$ and $\boldsymbol{B}$, and we can interpret either the rows of $\boldsymbol{A}$ or the columns of $\boldsymbol{B}$ as word embeddings. In this case it seems sensible to choose $\boldsymbol{A}$

Test set	$r = 100$		$r = 300$	
	Original score	With identifiability conditions	Original score	With identifiability conditions
Verb-143	0.439	0.501	0.460	0.518
YP-130	0.557	0.665	0.616	0.705
MTurk-287	0.695	0.693	0.701	0.644
MTurk-771	0.646	0.654	0.669	0.652
SimVerb-3500	0.269	0.329	0.311	0.385
SimLex-999	0.397	0.454	0.441	0.487
WordSim-353	0.652	0.683	0.645	0.613

Table 3.2: Test scores $g_S(\boldsymbol{D}, \boldsymbol{B})$ for seven different test sets, for SGNS embeddings trained on COHA. g_S is calculated using Pearson correlation. The results are given both before and after applying the constraint that $\boldsymbol{B}\boldsymbol{B}^T = \boldsymbol{I}_r$ (which was done in `R` using the QR decomposition). In most cases applying these identifiability conditions increases the test scores.

and $\boldsymbol{B}$ such that $\boldsymbol{A} = \boldsymbol{B}^T$. For the SVD case, this is equivalent to choosing $\alpha = \frac{1}{2}$, as for symmetric $\boldsymbol{X}$ we get $\boldsymbol{U} = \boldsymbol{V}$ in the SVD. In general, given a solution $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}})$ we can achieve symmetry by finding a $\boldsymbol{C}$ such that $\tilde{\boldsymbol{A}}\boldsymbol{C}^{-1} = \tilde{\boldsymbol{B}}^T\boldsymbol{C}^T$.

We should consider how imposing identifiability conditions affects the performance on tasks. Table 3.2 gives the performance of 100- and 300-dimensional SGNS embeddings, trained on COHA, before and after identifiability ($\boldsymbol{B}\boldsymbol{B}^T = \boldsymbol{I}_r$) is applied. In most, though not all, cases applying the identifiability conditions increases the test scores, perhaps because applying these constraints has a similar effect to using regularization during optimization. Since the performance of the embeddings after imposing identifiability conditions is generally similar to or better than the performance of the original embeddings, it seems that this is a reasonable solution to the problem of non-identifiability.

3.4.2 Optimizing over $\mathsf{UT}^+(r)$

Another way of addressing the problem of non-identifiability is to optimize the test score g over the set of possible transformations $\boldsymbol{C} \in \mathsf{GL}(r)$, for a particular set of embeddings. Since $\boldsymbol{C}$ contains r^2 total parameters, this is a very high-dimensional optimization (e.g. $r = 100$ gives 10^4 parameters to optimize over). However, as discussed previously, we can write $\boldsymbol{C} = c\boldsymbol{Q}\boldsymbol{R}$, where $c > 0$, $\boldsymbol{Q} \in \mathsf{O}(r)$ and $\boldsymbol{R} \in \mathsf{UT}^+(r)$. Since multiplication by $c\boldsymbol{Q}$ does not affect g, we need only optimize over $\boldsymbol{R} \in \mathsf{UT}^+(r)$. This reduces the number of parameters in the optimization to $\frac{1}{2}r(r+1) - 1$.

Earlier we considered the possibility of optimizing over a one-dimensional subset of $\mathsf{UT}^+(r)$, indexed by α; we now investigate this further. Figure 3.4 shows how the value of g_S changes with different values of α, where $\boldsymbol{B} = \boldsymbol{\Phi}_r^\alpha \boldsymbol{V}_r^T$, with $\boldsymbol{\Phi}_r$ and $\boldsymbol{V}_r$ coming from SVD. Table 3.3 shows the optimal values of α (found using the `optimize` function in `R`) for each of seven different test sets, using both Pearson and Spearman correlation to calculate g_S. We note that there is no overall best choice of

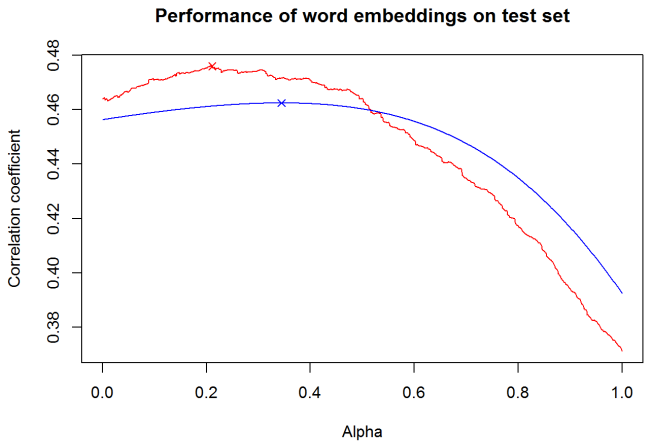

Figure 3.4: Performance of the word embeddings on the Word Similarity-353 test set, with α taking values between 0 and 1. The graph shows Spearman correlation coefficient (red) and Pearson correlation coefficient (blue) between the human-assigned similarity scores and the cosine similarities of the word pairs in the test set. The crosses mark the "best" value according to each criterion.

Test set	Pearson		Spearman	
	α	g	α	g
Verb-143	-0.675	0.398	-0.357	0.277
YP-130	2.546	0.269	3.607	0.241
MTurk-287	0.564	0.640	0.458	0.617
MTurk-771	0.492	0.503	0.287	0.546
SimVerb-3500	-0.406	0.125	-0.381	0.144
SimLex-999	0.869	0.174	-0.029	0.202
WordSim-353	0.463	0.505	0.309	0.536

Table 3.3: For LSA embeddings trained on COHA, with $\boldsymbol{B} = \boldsymbol{\Phi}^{\alpha}\boldsymbol{V}^{T}$, the optimal g with respect to α was found using the `optimize` function in `R`. The table displays the optimal values of g obtained for each test set, and the value of α at which this value was attained.

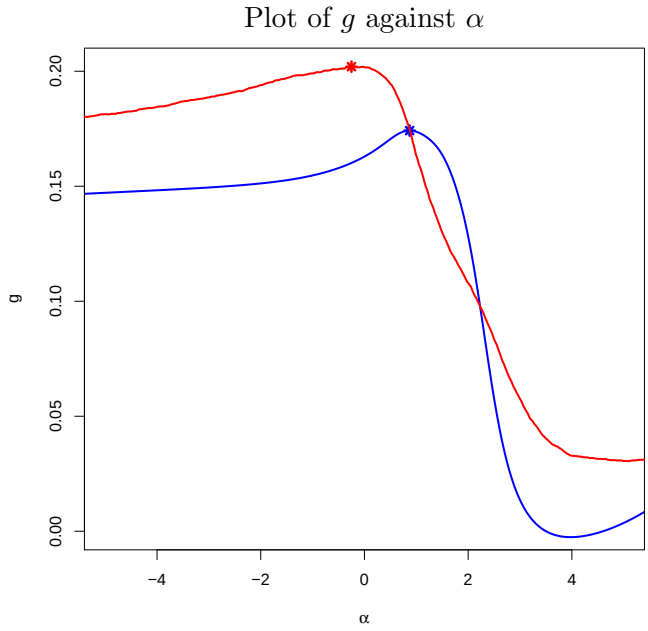

Figure 3.5: Plot of g against α for the SimLex-999 test set, using 300-dimensional LSA embeddings trained on COHA. The blue line shows g calculated using the Pearson correlation coefficient, and the red line shows g using the Spearman correlation coefficient.

α; the optimal values of α vary a lot between test sets, and for the SimLex-999 test set, the optimal α's are quite different depending on whether we use Pearson and Spearman correlation. Figure 3.5 shows how the value of g_S changes with α for the SimLex-999 test set. The blue line on the graph represents Spearman correlation and the red line is Pearson correlation. Although the optimal values of α for the different correlation coefficients are further apart for this test set than for the others, we can see that the graph has a similar shape for Pearson and Spearman correlation, with values of α that perform better on one test set also performing better on the other.

We can see that by optimizing over α, we can significantly improve the performance of the LSA embeddings (although we note that there is a danger of overfitting to a particular test set; this will be discussed more later). However, we need not restrict ourselves to this subset: we can consider optimizing over α where the em-

bedding set is

$$\boldsymbol{\Lambda}^{\alpha}\boldsymbol{B}^{*},$$

where $\boldsymbol{B}^{*}$ is an embedding set and $\boldsymbol{\Lambda}$ is any diagonal $r \times r$ matrix such that all its diagonal elements are positive and sum to 1 (as any such matrix will be an element of $\mathsf{UT}^{+}(r)$). Figure 3.6 shows the result of multiplying an embedding set by $\boldsymbol{\Lambda}^{\alpha}$ for different choices of $\boldsymbol{\Lambda}$ and α; $\boldsymbol{\Lambda} = \boldsymbol{\Phi}$ does not perform significantly better than the other choices.

We now consider optimizing g_S over $\mathsf{UT}^{+}(r)$. (We will look at analogy tasks later.) The optimization is done in `R` by running `optim` repeatedly, using the final value of one run as the starting value for the next, until the difference between the value of g_S for two successive runs falls below 10^{-5}. Since the test score must stay the same or increase for each run, this gives a better value than just running `optim` once. This process is still not guaranteed to converge to the overall maximum, but it gives a lower bound for how much test scores can be increased.

We aim to find $\boldsymbol{R}$ which solves

$$\underset{\boldsymbol{R}\in\mathsf{UT}^{+}(r)}{\arg\max}\, g_S(\boldsymbol{D}, \boldsymbol{RB}).$$

In Table 3.4, the original embeddings $\boldsymbol{B}$ are LSA embeddings trained on COHA with $r = 300$. The table shows the performance of the original embeddings $\boldsymbol{B}$ and of $\hat{\boldsymbol{R}}\boldsymbol{B}$, where $\hat{\boldsymbol{R}}$ is the optimal $\boldsymbol{R}$ found during our optimization, on each test set. Tables 3.6 and 3.7 give the same results for SGNS embeddings trained on COHA, with $r = 100$ and $r = 300$ respectively, and Table 3.8 gives results for SGNS embeddings trained on the larger Google News dataset. In each case we are able to significantly improve performance on each test set using this optimization procedure.

It is not clear, however, whether optimizing over one test set necessarily leads to better quality embeddings; there is a risk of overfitting the model to one test set. Hence, for each test set, Table 3.5 gives the performance of $\hat{\boldsymbol{R}}\boldsymbol{B}$ where $\hat{\boldsymbol{R}}$ is found by optimizing over that test set, on both that set and each of the other embedding sets. The embeddings found by optimizing with respect to one test set

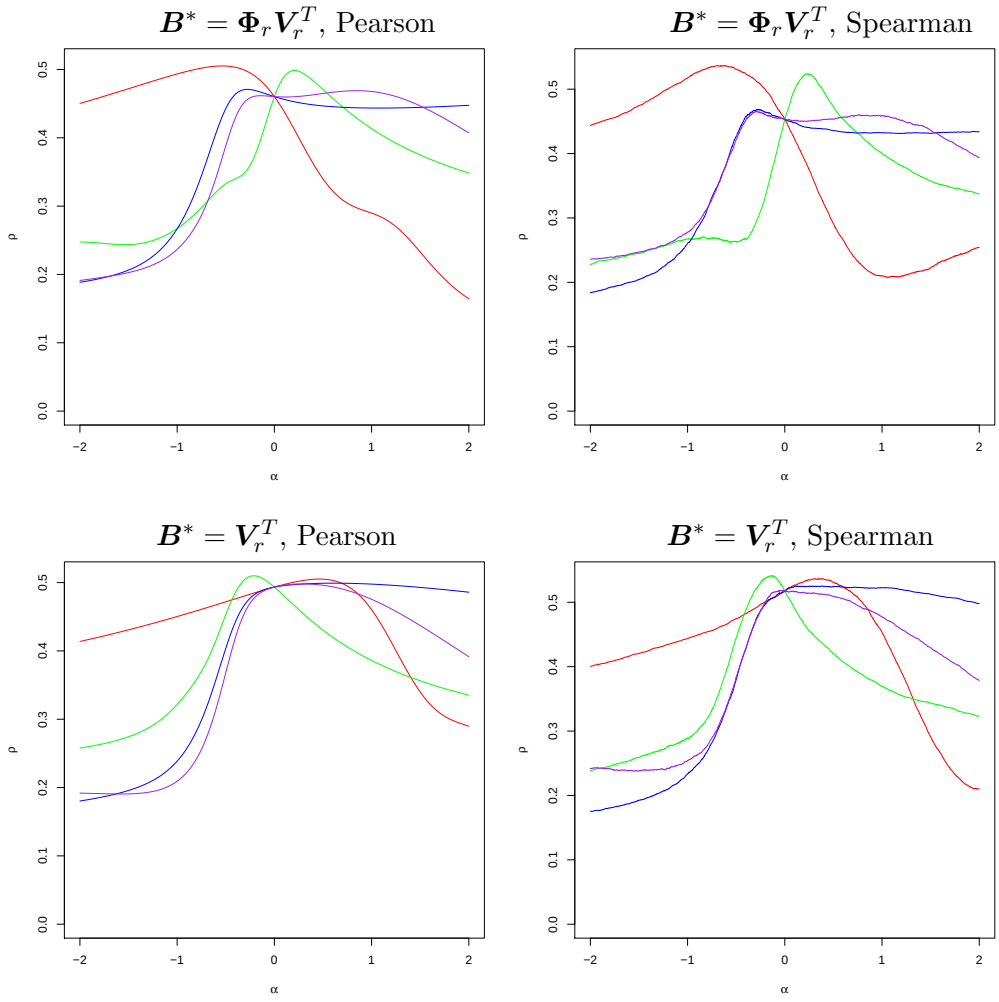

Figure 3.6: Plots showing $g_S(\boldsymbol{D}, \boldsymbol{B})$, where $\boldsymbol{D}$ is the WordSim-353 test set, when $\boldsymbol{B}$ is of the form $\boldsymbol{\Lambda}^\alpha \boldsymbol{B}^*$. To get $\boldsymbol{B}^*$, we take the rank-r SVD of the document-term matrix ($r = 300$): $\boldsymbol{X} = \boldsymbol{U}_r \boldsymbol{\Phi}_r \boldsymbol{V}_r^T$, and take $\boldsymbol{B}^*$ to be either $\boldsymbol{\Phi}_r \boldsymbol{V}_r^T$ (top) or $\boldsymbol{V}_r^T$ (bottom). Each line on the graphs corresponds to a different $\boldsymbol{\Lambda}$. In each case $\boldsymbol{\Lambda} = \text{diag}\,(\lambda_1, ..., \lambda_r)$, taken as the following: $\boldsymbol{\Lambda} = \boldsymbol{\Phi}_r$ (red lines); $\lambda_i = i$ (green); $\lambda_i \sim U(0, 1)$ (blue); and $\lambda_i \sim |N(0, 1)|$ (purple). There is not much difference in performance between the different $\boldsymbol{\Lambda}$'s. The dataset used is the Corpus of Historical American English (COHA).

Test set	Pearson		Spearman	
	Original	Best	Original	Best
Verb-143	0.380	0.882	0.276	0.480
YP-130	0.287	0.852	0.285	0.535
MTurk-287	0.627	0.811	0.599	0.847
MTurk-771	0.495	0.638	0.539	0.668
SimVerb-3500	0.122	0.278	0.140	0.298
SimLex-999	0.163	0.367	0.202	0.353
WordSim-353	0.493	0.736	0.517	0.699

Table 3.4: For each of seven different test sets, the word similarity score for a set of LSA embeddings trained on COHA, and the best score that can be achieved by multiplying the embedding matrix by a $\boldsymbol{R} \in \mathsf{UT}^+(r)$, found by using `optim` in `R`. $r = 300$.

do often perform slightly better when assessed on other test sets, but in other cases performance decreases. Hence, we should be cautious about doing this.

Analogy tasks

So far we have considered optimizing the word similarity test score g_S for a particular embedding set. In this section we consider doing this for the analogy test score g_A.

Word analogies are questions of the sort "king" is to "queen" as "man" is to "?". As discussed in Section 3.3, g_A (like g_S) is invariant to transformation by elements of $c\mathsf{O}(r)$, but not by $\mathsf{GL}(r)$.

Table 3.9 shows the performance of SGNS embeddings, multiplied by different diagonal and upper triangular matrices $\boldsymbol{C}$, on the Google Analogy test set. The test set contains 19544 different analogy tasks in 14 different categories, with the performance of the embeddings varying between categories (Figure 3.7). Unlike with the similarity tasks, none of the new sets of embeddings gives an improvement in performance over the original embeddings, and for the upper triangular matrices the accuracy is much worse. This may be because the analogy tasks are more sensitive

	Optimized on						
	Verb-143	YP-130	MT-287	MT-771	SV-3500	SL-999	WS-353
Verb-143	**0.882**	0.306	**0.567**	**0.447**	0.110	0.133	**0.461**
YP-130	**0.328**	**0.852**	**0.576**	**0.475**	0.123	0.175	**0.522**
MTurk-287	0.301	0.276	**0.811**	0.455	0.108	0.147	0.483
MTurk-771	0.278	0.237	**0.574**	**0.638**	0.104	0.188	**0.500**
SimVerb-3500	**0.211**	**0.264**	**0.585**	**0.466**	**0.278**	**0.196**	**0.500**
SimLex-999	**0.260**	**0.274**	**0.539**	**0.442**	0.144	**0.367**	**0.497**
WordSim-353	0.233	0.244	**0.573**	0.445	0.097	0.161	**0.736**

Table 3.5: Performance of LSA embeddings optimized over each test set on all test sets. All correlations in the table are Pearson correlations. The values in bold are those that exceed the test scores for the original LSA embeddings for each test set. We see that in many cases the embeddings optimized over one test set perform better on others as well. However, the embeddings optimized over SimVerb-3500 and SimLex-999 tend to perform worse on other test sets; indeed, the embeddings optimized on the MTurk-287, MTurk-771 and WordSim-353 test sets perform better on SimVerb-3500 and SimLex-999 than the embeddings optimized over these test sets. So this optimization may or may not increase performance overall.

Test set	Pearson		Spearman	
	Original	Best	Original	Best
Verb-143	0.439	0.735	0.399	0.615
YP-130	0.557	0.821	0.566	0.757
MTurk-287	0.695	0.803	0.676	0.757
MTurk-771	0.646	0.734	0.647	0.729
SimVerb-3500	0.269	0.382	0.255	0.369
SimLex-999	0.397	0.532	0.369	0.514
WordSim-353	0.652	0.777	0.663	0.768

Table 3.6: Table showing the original and optimized test scores using 100-dimensional SGNS embeddings, trained on COHA, for each of seven different test sets.

Test set	Pearson		Spearman	
	Original	Best	Original	Best
Verb-143	0.460	0.844	0.387	0.438
YP-130	0.616	0.942	0.599	0.627
MTurk-287	0.701	0.867	0.680	0.792
MTurk-771	0.669	0.780	0.676	0.761
SimVerb-3500	0.311	0.465	0.301	0.451
SimLex-999	0.441	0.595	0.423	0.548
WordSim-353	0.645	0.821	0.659	0.724

Table 3.7: Table showing the original and optimized test scores using 300-dimensional SGNS embeddings, trained on COHA, for each of seven different test sets. It should be said that this optimization process is not guaranteed to converge to the true optimum, which explains why sometimes the "optimum" score for the 300-dimensional embeddings is sometimes lower than that for the 100-dimensional embeddings.

Test set	Pearson		Spearman	
	Original	Best	Original	Best
SimLex-999	0.453	0.617	0.441	0.583
WordSim-353	0.652	0.838	0.700	0.797

Table 3.8: Table showing the original and optimized test scores using 300-dimensional SGNS embeddings, trained on the 100-billion word Google News corpus (downloaded from https://code.google.com/archive/p/word2vec), for two different test sets.

	Minimum	Mean	Maximum
Initial score		0.736	
$\boldsymbol{R}$ diagonal	0.677	0.692	0.703
$\boldsymbol{R}$ upper triangular	0.242	0.247	0.254

Table 3.9: Proportion of correctly guessed words using the Google Analogy test set, for SGNS embeddings (downloaded from `https://code.google.com/archive/p/word2vec/`) when they are multiplied by a series of random matrices $\boldsymbol{R} \in \mathsf{UT}^{+}(r)$, with non-zero elements sampled from $U(0, 1)$. For the diagonal and upper triangular cases, there were a total of 20 runs.

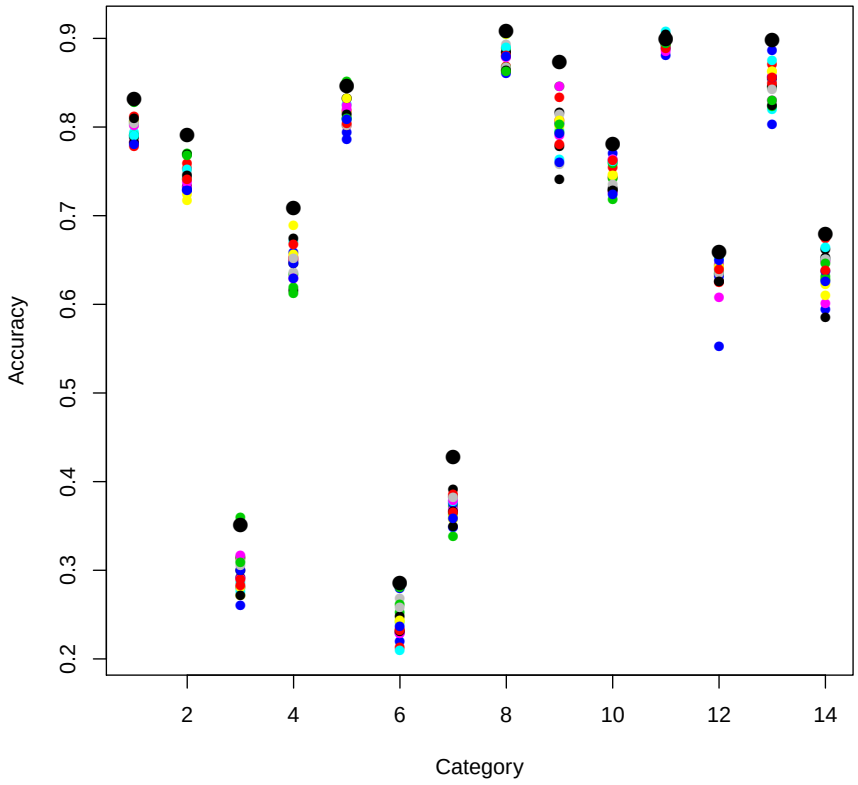

Figure 3.7: Performance of SGNS embeddings multiplied by a diagonal matrix $\boldsymbol{C}$ on the 14 different categories of analogies. The larger black points show the accuracy scores for the original embedding set. There is a lot of variation in performance across the different categories of embeddings.

to changes in the embedding set. In order to correctly solve an analogy, we must select the correct word out of tens or hundreds of thousands, so a small distortion of the embedding set may cause a significant drop in the accuracy rate.

Although the original embeddings seem to perform optimally on the analogy task set, we hypothesize that this is due to some regularization included in the optimization algorithm, which may implicitly be imposing identifiability constraints. It is also possible that hyperparameter tuning, or the use of multiple runs of the algorithm, play a role. We acknowledge that we may not always be able to improve performance of the embedding set using the methods in this chapter; however, we

	Minimum	Mean	Maximum
Initial score		0.218	
$\boldsymbol{R}$ diagonal	0.180	0.189	0.197
$\boldsymbol{R}$ upper triangular	0.067	0.073	0.071

Table 3.10: Proportion of correctly guessed words using the Google Analogy test set, using LSA embeddings trained on COHA, when they are multiplied by a series of matrices $\boldsymbol{R} \in \mathsf{UT}^+(r)$, with non-zero elements sampled from $U(0,1)$. For the diagonal and upper triangular cases, there were a total of 10 runs.

note that the objective function does not distinguish between the embeddings that give an accuracy of 73.6%, and those giving an accuracy of 24.2%.

Table 3.10 shows the same results for LSA embeddings trained on COHA. Not all of the words in the Google Analogy test set appear in COHA, so the embeddings are only evaluated on the set of analogies for which all four words appear in the dataset, which gives a total of 13,946 words.

For analogy tasks, it is too computationally expensive to optimize over the set $\mathsf{UT}^+(r)$, as we did for word similarity tasks. However, we can still investigate the effect of multiplying the embedding set by $\boldsymbol{\Lambda}^\alpha$, as we previously did for similarity tasks. In Figure 3.8, we take the rank-300 SVD of the document-term matrix of COHA, $\boldsymbol{U}_r\boldsymbol{\Phi}_r\boldsymbol{V}_r^T$, and investigate taking the embedding set to be $\boldsymbol{\Phi}_r^\alpha\boldsymbol{V}_r^T$, where α takes values between -2 and 2. We find that the embeddings perform best when $\alpha = 1.2$. Interestingly the proportion of correctly solved analogies seems to be a fairly smooth function of α; it is not clear why this is the case.

3.5 Conclusion

In this chapter we have explored the issue of non-identifiability in word embedding methods. We have shown how it arises, as multiplying the embedding set by a non-singular matrix does not change the value of the objective function. We have

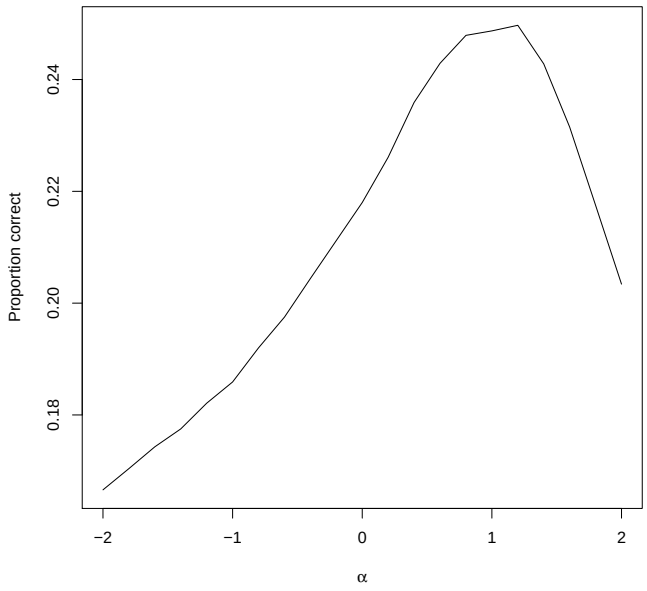

Figure 3.8: Plot showing the proportion of correctly solved analogies (using the Google Analogy test set) as a function of α, where $\boldsymbol{B} = \boldsymbol{\Phi}_r^{\alpha}\boldsymbol{V}_r^T$, where the rank-$r$ SVD of the document-term matrix is $\boldsymbol{U}_r\boldsymbol{\Phi}_r\boldsymbol{V}_r^T$. The embeddings were trained on COHA with $r = 300$. The optimal value of α seems to be around $\alpha = 1.2$.

also explored the consequences of this. In particular, we have identified that the set of transformations of the word embeddings to which the objective function f is invariant, is larger than the set of invariances to which the test function g is invariant, which means that solutions that are equally optimal with respect to the objective function may perform differently on test data.

We have proposed two solutions for dealing with this non-identifiability: imposing identifiability conditions on the embedding set in order to make it unique, and optimizing over the set of solutions with respect to a test set. We have implemented both of these methods on sets of word embeddings, where we found that embeddings with identifiability conditions imposed tend to perform fairly well on test data. We also found that, when testing the embeddings on word similarity tasks, it is possible to significantly increase the test score for a set of embeddings by multiplying the embedding set by an invertible $r \times r$ matrix $\boldsymbol{C}$. In some cases optimizing over one test set also gave superior results on others. For analogy tasks, we found that multiplying by a random diagonal or upper triangular matrix generally lead to a significant decrease in performance. However, we did find that, when using SVD to generate embeddings, we were able to slightly increase performance on analogy tasks by changing the value of α, where the embedding matrix was $\boldsymbol{\Phi}_r^\alpha \boldsymbol{V}_r^T$. We were unable to optimize g_A over $\mathsf{UT}^+(r)$ because it was too computationally expensive to do so, but this is something that could be interesting to explore.

Chapter 4

Semi-supervised word embeddings

In this chapter we explore the potential of semi-supervised word embeddings. In the previous chapter we looked at embeddings which were optimized over an objective function $f(\boldsymbol{X}, \boldsymbol{AB})$ and assessed on a test function $g(\boldsymbol{D}, \boldsymbol{B})$, where $\boldsymbol{X}$ is an unsupervised dataset, $\boldsymbol{D}$ is a supervised test dataset, $\boldsymbol{B}$ is the matrix of word embeddings, and $\boldsymbol{A}$ is an auxiliary matrix of context or document embeddings. In this chapter, we consider optimizing over an objective function in which both f and g play a role:

$$L = \lambda f(\boldsymbol{X}, \boldsymbol{B}) + (1 - \lambda) g(\boldsymbol{D}, \boldsymbol{B}).$$

The unsupervised part (f) is a function of the word embeddings $\boldsymbol{B}$ and the observed data matrix $\boldsymbol{X}$. The supervised part (g) is a function of the word embeddings and of a supervised dataset $\boldsymbol{D}$ that is human-generated; for example, a word similarity test set such as those discussed in Section 2.2.

The hope is that, by combining a large unsupervised dataset with some supervised data, we will be able to improve the performance of the word embeddings, compared to performance from training embeddings on just the unsupervised data. In order to avoid overfitting, performance must be assessed on other test sets than those on which we train the data.

We propose a semi-supervised version of multidimensional scaling, where we use

information in the data matrix $\boldsymbol{X}$ as well as information from a word embedding test set to generate a set of embeddings, which we implement on some simulated and real datasets. We will discuss briefly how identifiability issues similar to those described in the previous chapter relate to this method.

4.1 Motivation

Word embedding algorithms (such as LSA, SGNS, GloVe) are generally unsupervised. This is a necessity due to the huge amount of data involved in training the algorithms: the number of words we typically want to generate embeddings for is so large that it would be impractical to generate enough labelled data to be able to train a supervised algorithm.

However, small supervised datasets exist in the form of word similarity test sets $\boldsymbol{D}$ that are used to evaluate the performance of word embeddings (discussed in Sections 2.2 and 3.3.1). These are sets of word pairs, with human-assigned similarity scores for each pair. Word embeddings are considered to perform well if the cosine similarities between pairs of words are highly correlated with the human-assigned scores. (See Section 2.2 for more details.)

It is possible instead to formulate a word embedding model in terms of the supervised data $\boldsymbol{D}$. In this case, the aim would be to generate a set of word embeddings with the direct goal of being as consistent as possible with the human-assigned scores. However, these datasets are much too small to be able to generate useful word representations: both in terms of the number of total words included (typically a few hundred or thousand at most), and also in the number of pairwise similarity scores given for words in the test set. For example, the WordSim-353 test set [Finkelstein et al., 2001] contains 353 total word pairs, and there are many cases where, for a given word, we only have a human-assigned similarity score between that word and one other word.

In this chapter we explore how we can combine two types of data – large datasets

in the form of document-term/co-occurrence matrices, and small supervised test sets – to generate semi-supervised word embeddings. The aim is that by using an objective function that contains both an unsupervised component (similar to the objective function of an existing unsupervised word embedding algorithm) and a supervised component (based on one of these small datasets), we will be able to generate "better" word embeddings, by leveraging the information available in the supervised dataset $\boldsymbol{D}$ alongside the information in the unsupervised data matrix $\boldsymbol{X}$. This may be particularly useful in applications where $\boldsymbol{X}$ is relatively small, as we would then expect the impact from also including the supervised dataset to be greater. This may be useful in applications of word embeddings which use smaller specialised datasets. For example, [Stenetorp et al., 2012] uses the biomedical corpora containing between 90,000 and 450,000 total words, compared to 400 million words in COHA and 100 billion words in the Google News dataset.

A semi-supervised approach, combining both supervised and unsupervised data, is used in other applications where there is a large amount of unsupervised data, but supervised data is difficult or time-consuming to collect, for example speech recognition and webpage classification [Chapelle et al., 2006]. This differs from our approach, however, as the supervised and unsupervised data are usually of the same form (except that data in the supervised dataset has labels associated with each point). In our case, the training and test datasets are of different forms.

4.2 Introduction

Let the objective function be of the form

$$L(\boldsymbol{B}) = \lambda f(\boldsymbol{X}, \boldsymbol{B}) + (1 - \lambda) g(\boldsymbol{D}, \boldsymbol{B}), \tag{4.1}$$

where f is a function of the word embeddings and the data matrix $\boldsymbol{X}$, g is a function of the word embeddings and the supervised data $\boldsymbol{D}$, and λ is a hyperparameter controlling the relative importance of f and g. (Note that we will sometimes write

$f(\boldsymbol{X}, \boldsymbol{B})$ and $g(\boldsymbol{D}, \boldsymbol{B})$ as $f(\boldsymbol{B})$ and $g(\boldsymbol{B})$ for brevity.)

For example, we could take f to be the function (equivalent to the LSA embedding function when we require $\boldsymbol{B}$ to have orthogonal columns; see Chapter 3)

$$f(\boldsymbol{X}, \boldsymbol{B}) = ||\boldsymbol{X} - \boldsymbol{B}\boldsymbol{B}^T||_F, \tag{4.2}$$

and g to be the word similarity test function (Equation 2.1)

$$g(\boldsymbol{D}, \boldsymbol{B}) = \rho(\boldsymbol{z}(\boldsymbol{D}, \boldsymbol{B}), \boldsymbol{y}(\boldsymbol{D})). \tag{4.3}$$

Here, as in Section 2.2, the test data $\boldsymbol{D}$ consists of a set of triples (w_{i_1}, w_{i_2}, y_i), where w_{i_1} and w_{i_2} are words and y_i the human assigned similarity between them, and $\boldsymbol{z}$ is defined as

$$z_i = \cos\left(\langle \boldsymbol{b}_{i_1}, \boldsymbol{b}_{i_2} \rangle\right),$$

where $\boldsymbol{b}_{i_1}$ and $\boldsymbol{b}_{i_2}$ are the word embedding vectors associated with words w_{i_1} and w_{i_2} respectively.

Theorem 1 motivates our approach: it says that if we optimize L in Equation 4.1, obtaining the embedding set $\hat{\boldsymbol{B}}_L$, then $\hat{\boldsymbol{B}}_L$ will give a "better" embedding with respect to g than the embedding set optimized on f (denoted $\hat{\boldsymbol{B}}_f$), for any $\lambda < 1$.

Theorem 1. *Suppose we have functions $f(x)$ and $g(x)$. For $\lambda \in [0, 1]$, define*

$$h(x) = \lambda f(x) + (1 - \lambda) g(x),$$

and let $\hat{x}_f = \arg\min f(x)$. Then any x for which $h(x) \leq h(\hat{x}_f)$ is such that $g(x) \leq g(\hat{x}_f)$. Further, if $h(x) < h(\hat{x}_f)$, then $g(x) < g(\hat{x}_f)$.

Proof. Take any x such that $h(x) \leq h(\hat{x}_f)$. Then,

$$\lambda f(x) + (1 - \lambda) g(x) \leq \lambda f(\hat{x}_f) + (1 - \lambda) g(\hat{x}_f),$$

so

$$\lambda(f(x) - f(\hat{x}_f)) \leq (1 - \lambda)(g(\hat{x}_f) - g(x)),$$

so

$$g(\hat{x}_f) - g(x) \geq \frac{\lambda}{1-\lambda} \left(f(x) - f(\hat{x}_f) \right).$$

Since $\hat{x}_f$ minimizes f, we must have $f(\hat{x}_f) \leq f(x)$, so $f(x) - f(\hat{x}_f) \geq 0$, and hence $g(\hat{x}_f) - g(x) \geq 0$.

If $h(x) < h(\hat{x}_f)$, then most of the inequalities become strict:

$$\lambda f(x) + (1-\lambda) g(x) < \lambda f(\hat{x}_f) + (1-\lambda) g(\hat{x}_f),$$

so

$$\lambda (f(x) - f(\hat{x}_f)) < (1-\lambda)(g(x) - g(\hat{x}_f)),$$

so

$$g(x) - g(\hat{x}_f) > \frac{\lambda}{1-\lambda} \left(f(x) - f(\hat{x}_f) \right) \geq 0.$$

Hence, $g(\hat{x}_f) > g(x)$. □

Hence, if we define f and g as in Equations 4.2 and 4.3, the embeddings trained on the semi-supervised objective function (Equation 4.1) must perform at least as well with respect to g as LSA embeddings trained just on f. Hence, the embedding set trained on f gives a lower bound for performance on g with the semi-supervised embeddings. This result holds of course for the performance on the *training* data; the potential for overfitting means that it may not perform well on new test data, and this is a question we will address in this chapter.

The rest of this chapter is outlined as follows. Section 4.3 gives a brief overview of multidimensional scaling, which we use to derive the semi-supervised objective function. Section 4.4 introduces the objective function for semi-supervised word embeddings and explains two strategies we consider for optimizing it, majorization and stochastic gradient descent. Section 4.5 gives results from both small simulated datasets, and from real data. Section 4.6 summarizes the findings of this chapter and describes some extensions to this chapter that could be explored in future work.

4.3 Multidimensional scaling

Our approach to semi-supervised word embeddings is based on multidimensional scaling (MDS). MDS is a dimension reduction technique used for generating lower-dimensional representations of high-dimensional data. We use it here because it is a well-understood dimension reduction framework that is more flexible than SVD (which is a special case).

Section 2.7 gave an overview of MDS; we briefly recap this here. We have a set of objects and a matrix of pairwise dissimilarities between them, which we here denote $\boldsymbol{X}$. These could be distances between the objects in high-dimensional space, or subjective quantities such as psychological factors. We aim to generate a low-dimensional representation for each object, by minimizing the objective function

$$L_{MDS}(\boldsymbol{B}) = \sum_{i<j} w_{ij}(x_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||)^2,$$

where the x_{ij}'s are the dissimilarities, w_{ij}'s are weights (usually 0 or 1), and $\boldsymbol{B}$ is the matrix of object representations; in our case, these are word embeddings. The objective function is minimized when the Euclidean distances between the embeddings, $||\boldsymbol{b}_i - \boldsymbol{b}_j||$, are as close as possible to the dissimilarities x_{ij} (subject to the weights w_{ij}).

More generally, we can replace x_{ij} in the objective with $\delta(x_{ij}) = \delta_{ij}$, where δ is a function of $\boldsymbol{X}$:

$$L_{MDS}(\boldsymbol{B}) = \sum_{i<j} w_{ij}(\delta_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||)^2. \tag{4.4}$$

In non-metric MDS, we are only interested in the rank-order of the dissimilarities, not in their precise values. This is used when the dissimilarities are based on subjective judgments, such as the word similarity tasks in Chapter 3 (where the test function g is taken to be the correlation between the human-assigned similarity scores and the embeddings). We assume that words with higher similarity scores are more closely related than those with lower scores, but the absolute differences between the scores are difficult to interpret. In this case the function δ is not de-

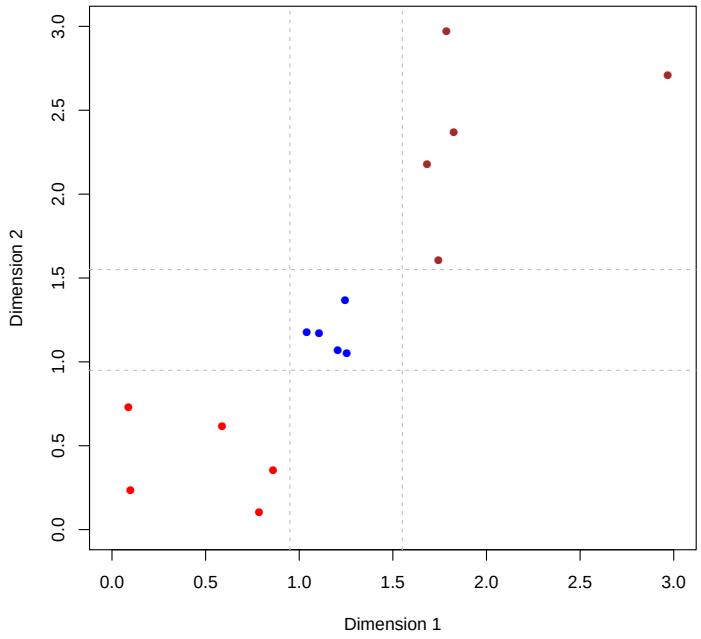

Figure 4.1: Figure illustrating the separation of points in different classes in [Witten and Tibshirani, 2011]. The points in class 2 (blue) all have greater values than points in class 1 (red) in both Dimension 1 and Dimension 2. The points in class 3 (brown) have larger values than those in both class 1 and class 2.

termined in advance, but estimated along with $\boldsymbol{B}$, subject to the constraint that $x_{ij} \leq x_{kl} \Rightarrow \delta_{ij} \leq \delta_{kl}$.

With respect to identifiability, the Euclidean distance between embeddings $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ is invariant to multiplications of the embeddings by the set of orthogonal transformations $\mathsf{O}(r)$, and hence the objective function in Equation 4.4 is invariant to orthogonal transformations of the embedding set. This is also true of the inner product, which we shall use later instead of the Euclidean distance. Later in this chapter, in Section 4.4.1, we discuss how this relates to semi-supervised word embeddings.

[Witten and Tibshirani, 2011] presents a semi-supervised variant of metric MDS. In the paper, they assume that there are class labels associated with the data points,

and the aim is that, after applying MDS, the classes will be separated in the output. This separation is achieved by requiring that $b_{ik} > b_{jk}$ when the class label of point i is greater than that of point j for all k. This means that, for example, the points in class 2 will have greater values in all embedding dimensions than the points in class 1, but smaller values than the points in class 3. (Figure 4.1 illustrates this for $r = 2$, with three classes.) Hence the objective function in the paper is (using our notation)

$$(1-\lambda)\frac{1}{2}\sum_{i,j=1}^{p}\left(x_{ij}-||\boldsymbol{b}_i-\boldsymbol{b}_j||\right)^2+\lambda\sum_{i,j:y_j>y_i}(y_j-y_i)\sum_{k=1}^{r}\left(\frac{x_{ij}}{\sqrt{r}}-(b_{jk}-b_{ik})\right)^2,$$

where the y_i's are the class labels. This is similar to what we are proposing to do with word embeddings, but as the supervised data we are using is in a different form, the supervised part of the objective function will be different.

4.3.1 MDS with inner product

The MDS objective is a function of the Euclidean distances between the embeddings. However, most word embedding algorithms (such as SGNS (Section 2.3.6), GloVe (Section 2.3.7), and LSA (Section 2.1)) are based on the inner products between the embeddings. Hence, we could define a modified objective that is a function of the inner products, rather than Euclidean distances. The metric version is

$$L^{'}_{MDS}(\boldsymbol{B})=\sum_{i<j}w_{ij}\left(x_{ij}-\boldsymbol{b}_i^T\boldsymbol{b}_j\right)^2, \tag{4.5}$$

and the non-metric version is

$$L^{'}_{MDS}(\boldsymbol{B})=\sum_{i<j}\tilde{w}_{ij}\left(\hat{\delta}_{ij}-\boldsymbol{b}_i^T\boldsymbol{b}_j\right)^2.$$

In this case, we take $\boldsymbol{X}=\boldsymbol{X'}^T\boldsymbol{X'}$, where $\boldsymbol{X'}$ is the document-term or co-occurrence matrix, so $\boldsymbol{b}_i$ approximates $\boldsymbol{x}_i^{'}$ for each i. This is thus closer to the objectives used in other word embedding algorithms. (It is the same as the objective function used in LSA, except that there we use $\boldsymbol{X'}$ instead of $\boldsymbol{X}$. However, a word embedding

Case	f	g
Case 1	$f_1 = \sum_{i<j} w_{ij} \left(x_{ij} - \|\|\boldsymbol{b}_i - \boldsymbol{b}_j\|\|\right)^2$	$g_1 = \sum_{i<j} \tilde{w}_{ij} \left(\hat{\delta}_{ij}(\boldsymbol{D}) - \|\|\boldsymbol{b}_i - \boldsymbol{b}_j\|\|\right)^2$
Case 2	$f_2 = \sum_{i<j} w_{ij} \left(x_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2$	$g_2 = \sum_{i<j} \tilde{w}_{ij} \left(\hat{\delta}_{ij}(\boldsymbol{D}) - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2$
Case 3	$f_3 = \sum_{i<j} w_{ij} \left(x_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2$	$g_3 = 1 - \rho(\boldsymbol{z}(\boldsymbol{D}, \boldsymbol{B}), \boldsymbol{y}(\boldsymbol{D}))$

Table 4.1: Summary of the three different semi-supervised objectives we are using.

matrix $\boldsymbol{B}$ which minimizes this objective will also minimize the LSA objective.) We will look at using both the standard MDS objective and this modified objective in generating semi-supervised embeddings.

4.4 Semi-supervised word embeddings with MDS

We propose the semi-supervised MDS objective function

$$L(\boldsymbol{B}) = \lambda f(\boldsymbol{X}, \boldsymbol{B}) + (1 - \lambda) g(\boldsymbol{D}, \boldsymbol{B}),$$

where $f(\boldsymbol{X}, \boldsymbol{B})$ is the unsupervised part of the objective, $g(\boldsymbol{D}, \boldsymbol{B})$ is the supervised part, and λ is a hyperparameter controlling the relative weighting of the two. We consider three cases, with different choices of f and g, which are given in Table 4.1.

Each of these three cases has different reasons for choosing it. Case 1 is directly based on ordinary MDS, with Euclidean distances; we are just combining the metric and non-metric objectives with two different datasets ($\boldsymbol{X}$ and $\boldsymbol{D}$). Case 2 is as Case 1, but uses inner products between the embeddings instead of Euclidean distances, which makes it more similar to the usual word embedding algorithms, where the objective is a function of the inner products between embeddings. Case 3 uses the word similarity test function g (Equation 2.1) as the supervised part. This is further from standard MDS, and is a more complicated function to optimize, but it is more directly related to optimizing word embedding performance on test data.

Hence, the objective functions for each of the three cases are given below:

$$L^{(1)}(\boldsymbol{B}) = \lambda \sum_{i<j} w_{ij} \left(x_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||\right)^2 + (1-\lambda) \sum_{i<j} \tilde{w}_{ij} \left(\hat{\delta}_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||\right)^2; \quad (4.6)$$

$$L^{(2)}(\boldsymbol{B}) = \lambda \sum_{i<j} w_{ij} \left(x_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 + (1-\lambda) \sum_{i<j} \tilde{w}_{ij} \left(\hat{\delta}_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2; \quad (4.7)$$

$$L^{(3)}(\boldsymbol{B}) = \lambda \sum_{i,j} w_{ij} \left(x_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 + (1-\lambda) \left(1 - \rho(\boldsymbol{z}(\boldsymbol{D}, \boldsymbol{B}), \boldsymbol{y}(\boldsymbol{D}))\right). \quad (4.8)$$

4.4.1 Identifiability

As discussed earlier (Section 4.3), the supervised part of the MDS objective function f with either Euclidean distance or inner product (Equations 4.4 and 4.5) is invariant to transformations of the embeddings by elements of $\mathsf{O}(r)$ (the set of orthogonal $r \times r$ matrices):

$$f(\boldsymbol{B}) = f(\boldsymbol{Q}\boldsymbol{B}) \text{ for } \boldsymbol{Q} \in \mathsf{O}(r) \ .$$

This is in contrast with most word embedding algorithms that involve a product of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, where the objective function is invariant to transformations of the embedding set by an arbitrary non-singular matrix (see Section 3.2). As with the test functions in the previous chapter, the supervised part of the objective function, g, is invariant to both orthogonal and scale transformations: the set $c\mathsf{O}(r) = \{c\boldsymbol{Q} : c > 0, \boldsymbol{Q} \in \mathsf{O}(r)\}$. (This holds for all three cases.)

Hence, we do not have the same problem as in Section 3.3, where there were transformations to which the training objective f was invariant but the test function g was not. In fact, in this case, if we let

$$\mathcal{F} = \{\boldsymbol{C} : f(\boldsymbol{C}\boldsymbol{B}) = f(\boldsymbol{B}) \text{ for all } \boldsymbol{B}\} = \mathsf{O}(r)$$

and

$$\mathcal{G} = \{\boldsymbol{C} : g(\boldsymbol{C}\boldsymbol{B}) = g(\boldsymbol{B}) \text{ for all } \boldsymbol{B}\} = c\mathsf{O}(r),$$

then we have $\mathcal{F} \subset \mathcal{G}$.This means that g is less sensitive than f: embedding sets that give different values for f may perform equally well according to g. However, we

could avoid this by standardizing the embedding set to have a fixed norm, thus not allowing scale transformations.

4.4.2 Optimizing the objective function

We investigate two ways of minimizing the objective function. A standard approach for minimizing the objective function in MDS is to use a majorization algorithm (see Section 2.7.1), and we derive here majorization algorithms for optimizing the semi-supervised objective in Case 1 (Equation 4.6) and Case 2 (Equation 4.7). However, majorization can be slow to converge [de Leeuw, 1988], and this approach is infeasible for large datasets due to computational issues. Hence, we also derive a stochastic gradient descent algorithm for each case. Most machine learning optimization methods are based on a form of this. This is faster, but has the disadvantage of having hyperparameters (the learning rate and batch size, which we will explain later) that must be tuned in order for the algorithm to converge well: if the learning rate is too high, the value of the objective function may increase, whereas if it is too low, convergence will be very slow. Hence, for a small problem it may be easier to use majorization. We investigate in which situations it is better to use one or the other.

4.4.3 Majorization

Majorization is a way of minimizing an objective function, where we use a secondary (majorizing) function instead of our desired objective, which is easier to minimize. (See Section 2.7.1 for more details.) In order for a function h to be a majorizing function of a function f, h must satisfy $h(x, \tilde{x}) \geq f(x)$ for all $\tilde{x}$, with equality if $x = \tilde{x}$. We generate a series of inequalities

$$f(x_{k+1}) \leq h(x_{k+1}, x_k) \leq h(x_k, x_k) = f(x_k),$$

which, as long as f is bounded below, is guaranteed to converge to a local minimum.

Case 1

The objective function is (Equation 4.6):

$$L^{(1)}(\boldsymbol{B}) = \lambda \sum_{i<j} w_{ij} \left(x_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||\right)^2 + (1-\lambda) \sum_{i<j} \tilde{w}_{ij} \left(\hat{\delta}_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||\right)^2.$$

The following theorem gives a majorizing function of $L^{(1)}$.

Theorem 2. *Let $L^{(1)}(\boldsymbol{B})$ be defined as in Equation 4.6. A majorizing function of $L^{(1)}(\boldsymbol{B})$ is*

$$m(\boldsymbol{B}, \tilde{\boldsymbol{B}}) = \tilde{\eta}_x^2 + \operatorname{tr}\left(\boldsymbol{B}^T \tilde{\boldsymbol{Y}} \boldsymbol{B}\right) - 2 \operatorname{tr}\left(\boldsymbol{B}^T \tilde{\boldsymbol{H}}(\tilde{\boldsymbol{B}}) \tilde{\boldsymbol{B}}\right),$$

where

$$\tilde{\eta}_x^2 = \sum_{i<j} w_{ij} x_{ij}^2 + \lambda \tilde{w}_{ij} \hat{\delta}_{ij}^2,$$

$$\tilde{y}_{ij} = \begin{cases} -\left(\lambda w_{ij} + (1-\lambda)\tilde{w}_{ij}\right) & \text{if } i \neq j \\ \sum_{k=1, k\neq i}^{n} \left(\lambda w_{ik} + (1-\lambda)\tilde{w}_{ij}\right) & \text{if } i = j, \end{cases}$$

$$\tilde{h}_{ij}\left(\tilde{\boldsymbol{B}}\right) = \begin{cases} -\frac{\lambda w_{ij} x_{ij} + (1-\lambda)\tilde{w}_{ij} d\hat{e}lta_{ij}}{||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j||} & \text{if } i \neq j \text{ and } ||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| > 0 \\ 0 & \text{if } i \neq j \text{ and } ||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| = 0 \\ -\sum_{k=1, k\neq i} \frac{\lambda w_{ik} x_{ik} + (1-\lambda)\tilde{w}_{ik}\hat{\delta}_{ik}}{||\boldsymbol{b}_i - \boldsymbol{b}_k||} & \text{if } i = j. \end{cases}$$

Remark. This is related to the majorizing function of metric MDS given in Equation 2.3.

Proof. To show this, we use the same method as in [Borg and Groenen, 1997] (which is outlined in Section B.1). We first divide the objective function into three parts:

$$\begin{aligned} L^{(1)}(\boldsymbol{B}) &= \sum_{i<j} \left(\lambda w_{ij} x_{ij}^2 + (1-\lambda)\tilde{w}_{ij}\hat{\delta}_{ij}^2\right) + \sum_{i<j} \left(\lambda w_{ij} + (1-\lambda)\tilde{w}_{ij}\right) ||\boldsymbol{b}_i - \boldsymbol{b}_j||^2 \\ &\quad - 2 \sum_{i<j} \left(\lambda w_{ij} x_{ij} + (1-\lambda)\tilde{w}_{ij}\hat{\delta}_{ij}\right) ||\boldsymbol{b}_i - \boldsymbol{b}_j|| \\ &= \tilde{\eta}_x^2 + \tilde{\eta}^2(\boldsymbol{B}) - 2\tilde{\rho}(\boldsymbol{B}). \end{aligned}$$

As before, $\tilde{\eta}_x^2$ does not depend on $\boldsymbol{B}$. $\tilde{\eta}^2(\boldsymbol{B})$ is the same as $\eta^2(\boldsymbol{B})$, but with w_{ij} replaced with $\lambda w_{ij} + (1-\lambda)\tilde{w}_{ij}$:

$$\begin{aligned}\tilde{\eta}^2(\boldsymbol{B}) &= \sum_{i<j} \left(\lambda w_{ij} + (1-\lambda)\tilde{w}_{ij}\right) ||\boldsymbol{b}_i - \boldsymbol{b}_j||^2 \\ &= \sum_{i<j} \operatorname{tr}\left(\boldsymbol{B}^T \left(\left(\lambda w_{ij} + (1-\lambda)\tilde{w}_{ij}\right) \boldsymbol{Y}_{ij}\right) \boldsymbol{B}\right) \\ &= \operatorname{tr}\left(\boldsymbol{B}^T \tilde{\boldsymbol{Y}} \boldsymbol{B}\right),\end{aligned}$$

where, as before, $\boldsymbol{Y}_{ij}$ is a matrix with its iith and jjth elements equal to 1, its ijth and jith elements equal to -1, and all other elements equal to 0. We now get

$$\tilde{y}_{ij} = \begin{cases} -\left(\lambda w_{ij} + (1-\lambda)\tilde{w}_{ij}\right) & \text{if } i \neq j \\ \sum_{k=1, k\neq i}^{n} \left(\lambda w_{ik} + (1-\lambda)\tilde{w}_{ik}\right) & \text{if } i = j. \end{cases}$$

For $\tilde{\rho}(\boldsymbol{B})$, as in Section B.1, we use the Cauchy-Schwarz inequality to show that

$$\begin{aligned}\sum_{k=1}^{r} \left(b_{ik} - b_{jk}\right)\left(\tilde{b}_{ik} - \tilde{b}_{jk}\right) &\leq \left(\sum_{k=1}^{r} \left(b_{ik} - b_{jk}\right)^2\right)^{1/2} \left(\sum_{k=1}^{r} \left(\tilde{b}_{ik} - \tilde{b}_{jk}\right)^2\right)^{1/2} \\ &= ||\boldsymbol{b}_i - \boldsymbol{b}_j|| \cdot ||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j||.\end{aligned}$$

Given that $||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| \neq 0$, we can rearrange this to get

$$-||\boldsymbol{b}_i - \boldsymbol{b}_j|| \leq -\frac{\sum_{k=1}^{r} \left(b_{ik} - b_{jk}\right)\left(\tilde{b}_{ik} - \tilde{b}_{jk}\right)}{||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j||} \tag{4.9}$$

When $||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| = 0$, we cannot do this, but we still have that $-||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| \leq 0$. Thus, as we still have

$$\sum_{k=1}^{r} \left(b_{ik} - b_{jk}\right)\left(\tilde{b}_{ik} - \tilde{b}_{jk}\right) = \operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{Y}_{ij} \tilde{\boldsymbol{B}}\right),$$

we get

$$\begin{aligned}-\tilde{\rho}(\boldsymbol{B}) &= -\sum_{i<j} \left(\lambda w_{ij} x_{ij} + (1-\lambda)\tilde{w}_{ij}\hat{\delta}_{ij}\right) ||\boldsymbol{b}_i - \boldsymbol{b}_j|| \\ &\leq -\sum_{i<j} \left(\frac{\lambda w_{ij} x_{ij} + (1-\lambda)\,\tilde{w}_{ij}\hat{\delta}_{ij}}{||\tilde{b}_i - \tilde{b}_j||}\right) \operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{Y}_{ij} \tilde{\boldsymbol{B}}\right) \\ &= -\sum_{i<j} \operatorname{tr}\left(\boldsymbol{B}^T \left(\frac{\lambda w_{ij} x_{ij} + (1-\lambda)\tilde{w}_{ij}\hat{\delta}_{ij})}{||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j||} \boldsymbol{Y}_{ij}\right) \tilde{\boldsymbol{B}}\right) \\ &= -\operatorname{tr}\left(\boldsymbol{B}^T \tilde{\boldsymbol{H}}(\tilde{\boldsymbol{B}}) \tilde{\boldsymbol{B}}\right),\end{aligned}$$

where $\tilde{h}_{ij}(\tilde{\boldsymbol{B}})$ is equal to

$$\tilde{h}_{ij} = \begin{cases} -\frac{\lambda w_{ij} x_{ij} + (1-\lambda)\tilde{w}_{ij}\hat{\tilde{\delta}}_{ij}}{||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j||} & \text{if } i \neq j \text{ and } ||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| > 0 \\ 0 & \text{if } i \neq j \text{ and } ||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| = 0 \\ \sum_{k=1,k\neq i} \tilde{h}_{ij}\left(\boldsymbol{B}\right) & \text{if } i = j. \end{cases}$$

Note that, if $\boldsymbol{B} = \tilde{\boldsymbol{B}}$, then there is equality in Equation 4.9, and hence we have $-\tilde{\rho}(\boldsymbol{B}) = -\operatorname{tr}\left(\tilde{\boldsymbol{B}}^T \tilde{\boldsymbol{H}}\left(\tilde{\boldsymbol{B}}\right)\tilde{\boldsymbol{B}}\right)$.

Thus,

$$L^{(1)}(\boldsymbol{B}) \leq \tilde{\eta}_x^2 + \operatorname{tr}\left(\boldsymbol{B}^T \tilde{\boldsymbol{Y}} \boldsymbol{B}\right) - 2\operatorname{tr}\left(\boldsymbol{B}^T \tilde{\boldsymbol{H}}(\tilde{\boldsymbol{B}})\tilde{\boldsymbol{B}}\right) = m\left(\boldsymbol{B}, \tilde{\boldsymbol{B}}\right),$$

with equality if $\boldsymbol{B} = \tilde{\boldsymbol{B}}$, and so $m(\boldsymbol{B}, \tilde{\boldsymbol{B}})$ majorizes $L^{(1)}(\boldsymbol{B})$. □

The majorization algorithm for $L^{(1)}$ is given in Algorithm 5. As in Section 2.7.3, $\hat{\tilde{\delta}}_{ij}$ is estimated at each stage using Kruskal's up-and-down blocks algorithm [Kruskal, 1964].

Case 2

Here, the part of the objective function in Equation 4.7 that corresponds to the unsupervised part,

$$f\left(\boldsymbol{B}\right) = \sum_{i,j} w_{ij}\left(x_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2, \tag{4.10}$$

is the same as the objective function for weighted SVD. A majorizing function for this is derived in [Groenen et al., 2003]:

$$m\left(\boldsymbol{B}, \tilde{\boldsymbol{B}}\right) = M \sum_{i,j}\left(r_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 + w_{ij}\left(1 - \frac{w_{ij}}{M}\right)\left(x_{ij} - \tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j\right)^2,$$

where $M = \max_{i,j}\{w_{ij}\}$ and $r_{ij} = \left(1 - \frac{w_{ij}}{M}\right)\tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j + \frac{w_{ij}}{M} x_{ij}$. (The proof of this is outlined in Appendix B.2; we make use of it to derive a majorizing function for Equation 4.7.)

Algorithm 5 Majorization: Case 1

Require: $\tilde{\boldsymbol{B}} = \boldsymbol{B}^{[0]}$ and $k = 0$.

Find an optimal set of values for $\hat{\delta}_{ij}$ that minimizes $\sum_{i<j} \left(\hat{\delta}_{ij} - ||\boldsymbol{b}_i^{[0]} - \boldsymbol{b}_j^{[0]}||\right)^2$, subject to the constraint in Equation 2.7.3.

Compute $L^{(1)}\left(\boldsymbol{B}^{[0]}\right)$.

while $convergence = 0$ **do**

$k = k + 1$.

$\boldsymbol{B}^{[k]} = \boldsymbol{Y}^{+}\boldsymbol{H}(\tilde{\boldsymbol{B}})\tilde{\boldsymbol{B}}$ (with x_{ij}'s replaced by $\hat{\delta}_{ij}$'s).

Find an optimal set of values for $\hat{\delta}_{ij}$ that minimizes $\sum_{i<j} \left(\hat{\delta}_{ij} - ||\boldsymbol{b}_i^{[k]} - \boldsymbol{b}_j^{[k]}||\right)^2$, subject to the constraint in Equation 2.7.3.

Compute $L^{(1)}\left(\boldsymbol{B}^{[k]}\right)$.

if $L^{(1)}\left(\boldsymbol{B}^{[k-1]}\right) - L^{(1)}\left(\boldsymbol{B}^{[k]}\right) < \epsilon$ **then**

$convergence = 1$,

else if $k = maxiter$ **then**

$convergence = 1$,

end if

$\tilde{\boldsymbol{B}} = \boldsymbol{B}^{[k]}$.

end while

The objective function we want to minimize (Equation 4.7) is:

$$\begin{aligned}
L^{(2)}(\boldsymbol{B}) &= \sum_{i,j} \lambda w_{ij} \left(x_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 + (1-\lambda)\tilde{w}_{ij} \left(\hat{\delta}_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 \\
&= \sum_{i,j} \left(\lambda w_{ij} x_{ij}^2 + (1-\lambda)\tilde{w}_{ij}\hat{\delta}_{ij}^2\right) - 2\left(\lambda w_{ij} x_{ij} + (1-\lambda)\tilde{w}_{ij}\hat{\delta}_{ij}\right) \boldsymbol{b}_i^T \boldsymbol{b}_j \\
&+ \left(\lambda w_{ij} + (1-\lambda)\tilde{w}_{ij}\right) \left(\boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2.
\end{aligned}$$

In order to derive a majorization algorithm for this objective function, we first introduce another function $h(\boldsymbol{B})$ that differs from the objective function by a constant, and is simpler to minimize than the objective function.

Let

$$h(\boldsymbol{B}) = \sum_{i,j} \gamma_{ij} \left(c_{ij} x_{ij} + (1-c_{ij})\hat{\delta}_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2,$$

where $\gamma_{ij} = \lambda w_{ij} + (1-\lambda)\,\tilde{w}_{ij}$, and, if $w_{ij} = \tilde{w}_{ij} = 0$, $c_{ij} = \tilde{c}_{ij} = 0$; otherwise

$$c_{ij} = \frac{\lambda w_{ij}}{\lambda w_{ij} + (1-\lambda)\tilde{w}_{ij}} = \frac{\lambda w_{ij}}{\gamma_{ij}}.$$

Let

$$\sigma_{ij}(\boldsymbol{B}) = \lambda w_{ij} \left(x_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 + (1-\lambda)\tilde{w}_{ij} \left(\hat{\delta}_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2$$

and

$$h_{ij}(\boldsymbol{B}) = \gamma_{ij} \left(c_{ij} x_{ij} + (1-c_{ij})\hat{\delta}_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2.$$

If $w_{ij} = \tilde{w}_{ij} = 0$, then $\sigma_{ij}(\boldsymbol{B}) = h_{ij}(\boldsymbol{B}) = 0$. Otherwise,

$$\begin{aligned}
h_{ij}(\boldsymbol{B}) &= \gamma_{ij} \left(c_{ij} x_{ij} + (1-c_{ij})\hat{\delta}_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 \\
&= \gamma_{ij} \left(\left(c_{ij} x_{ij} + (1-c_{ij})\hat{\delta}_{ij}\right)^2 - 2\left(c_{ij} x_{ij} + (1-c_{ij})\hat{\delta}_{ij}\right) \boldsymbol{b}_i^T \boldsymbol{b}_j + \left(\boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2\right) \\
&= f_{ij}^1 \left(x_{ij}, \hat{\delta}_{ij}, w_{ij}, \tilde{w}_{ij}, \lambda\right) - 2\gamma_{ij} \left(c_{ij} x_{ij} + (1-c_{ij})\hat{\delta}_{ij}\right) \boldsymbol{b}_i^T \boldsymbol{b}_j + \gamma_{ij} \left(\boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 \\
&= f_{ij}^1 \left(x_{ij}, \hat{\delta}_{ij}, w_{ij}, \tilde{w}_{ij}, \lambda\right) - 2\left(\lambda w_{ij} x_{ij} + (1-\lambda)\tilde{w}_{ij}\hat{\delta}_{ij}\right) \boldsymbol{b}_i^T \boldsymbol{b}_j + \gamma_{ij} \left(\boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2,
\end{aligned}$$

and

$$\begin{aligned}
\sigma_{ij}(\boldsymbol{B}) &= \lambda w_{ij} \left(x_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 + (1-\lambda)\tilde{w}_{ij} \left(\hat{\delta}_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 \\
&= \left(\lambda w_{ij} x_{ij}^2 + (1-\lambda)\tilde{w}_{ij}\hat{\delta}_{ij}^2\right) - 2\left(\lambda w_{ij} x_{ij} + (1-\lambda)\tilde{w}_{ij}\hat{\delta}_{ij}\right) \boldsymbol{b}_i^T \boldsymbol{b}_j + \gamma_{ij} \left(\boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 \\
&= f_{ij}^2 \left(x_{ij}, \hat{\delta}_{ij}, w_{ij}, \tilde{w}_{ij}, \lambda\right) - 2\left(\lambda w_{ij} x_{ij} + (1-\lambda)\tilde{w}_{ij}\hat{\delta}_{ij}\right) \boldsymbol{b}_i^T \boldsymbol{b}_j + \gamma_{ij} \left(\boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2.
\end{aligned}$$

Hence,

$$h(\boldsymbol{B}) - L^{(2)}(\boldsymbol{B}) = \sum_{i,j} f_{ij}^1\left(x_{ij}, \hat{\delta}_{ij}, w_{ij}, \tilde{w}_{ij}, \lambda\right) - f_{ij}^2\left(x_{ij}, \hat{\delta}_{ij}, w_{ij}, \tilde{w}_{ij}, \lambda\right)$$
$$= f\left(x_{ij}, \hat{\delta}_{ij}, w_{ij}, \tilde{w}_{ij}, \lambda\right),$$

which does not depend on $\boldsymbol{B}$. Thus, the difference between $h(\boldsymbol{B})$ and $L^{(2)}(\boldsymbol{B})$ is constant in $\boldsymbol{B}$, so the two functions are minimized by the same value of $\boldsymbol{B}$. Therefore, we can minimize $h(\boldsymbol{B})$ instead of $L^{(2)}(\boldsymbol{B})$, which is a weighted least-squares problem in $\boldsymbol{B}$, as is the case for Equation 4.10.

Minimizing $h(\boldsymbol{B})$ is equivalent to a weighted SVD problem. To solve weighted SVD we minimize the objective function

$$W = \sum_{i,j} \omega_{ij}\left(\xi_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2,$$

which is equivalent to $h(\boldsymbol{B})$, where

$$\omega_{ij} = \gamma_{ij} = \lambda w_{ij} + (1-\lambda)\tilde{w}_{ij},$$

and

$$\xi_{ij} = c_{ij}\delta_{ij} + (1 - c_{ij})\hat{\delta}_{ij}.$$

Thus, $h(\boldsymbol{B})$ can be majorized by the function

$$m(\boldsymbol{B}, \tilde{\boldsymbol{B}}) = M \sum_{i,j} \left(r_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2$$
$$+ \gamma_{ij}\left(1 - \frac{\lambda w_{ij} + (1-\lambda)\tilde{w}_{ij}}{M}\right)\left(c_{ij}x_{ij} + (1-c_{ij})\hat{\delta}_{ij} - \tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j\right)^2,$$

where $M = \max_{i,j}\{\gamma_{ij}\}$ and

$$r_{ij} = \left(1 - \frac{\lambda w_{ij} + (1-\lambda)\tilde{w}_{ij}}{M}\right)\tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j + \frac{\lambda w_{ij} + (1-\lambda)\tilde{w}_{ij}}{M}\left(c_{ij}x_{ij} + (1-c_{ij})\hat{\delta}_{ij}\right).$$

4.4.4 Stochastic Gradient Descent

Stochastic gradient descent is an optimization algorithm commonly used in machine learning: an overview is given in Section 2.7.2.

In order to apply stochastic gradient descent to our semi-supervised objective functions, we must first calculate $\frac{\partial L}{\partial \boldsymbol{B}}$ for each case.

Case 1

The objective function is

$$L^{(1)}(\boldsymbol{B}) = \lambda \sum_{i=1}^{p}\sum_{j=1}^{p} w_{ij}\left(x_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||\right)^2 + (1-\lambda)\sum_{i=1}^{p}\sum_{j=1}^{p}\tilde{w}_{ij}\left(\hat{\delta}_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||\right)^2$$
$$= 2\lambda \sum_{i=1}^{p-1}\sum_{j=i+1}^{p} w_{ij}\left(x_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||\right)^2 + 2(1-\lambda)\sum_{i=1}^{p-1}\sum_{j=i+1}^{p}\tilde{w}_{ij}\left(\hat{\delta}_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||\right)^2,$$

since $x_{ii} = ||\boldsymbol{b}_i - \boldsymbol{b}_i|| = 0$, and $x_{ji} = x_{ij}$ and $||\boldsymbol{b}_i - \boldsymbol{b}_j|| = ||\boldsymbol{b}_j - \boldsymbol{b}_i||$.

We can rewrite this as

$$L^{(1)}(\boldsymbol{B}) = 2L_1 - 4L_2 + 2L_3,$$

where

$$L_1 = \sum_{i=1}^{p-1}\sum_{j=i+1}^{p}\left(\lambda w_{ij}x_{ij}^2 + (1-\lambda)\,\tilde{w}_{ij}\hat{\delta}_{ij}^2\right),$$

$$L_2 = \sum_{i=1}^{p-1}\sum_{j=i+1}^{p}\left(\lambda w_{ij}x_{ij} + (1-\lambda)\,\tilde{w}_{ij}\hat{\delta}_{ij}\right)||\boldsymbol{b}_i - \boldsymbol{b}_j||,$$

and

$$L_3 = \sum_{i=1}^{p-1}\sum_{j=i+1}^{p}\left(\lambda w_{ij} + (1-\lambda)\,\tilde{w}_{ij}\right)||\boldsymbol{b}_i - \boldsymbol{b}_j||^2,$$

L_1 is not a function of $\boldsymbol{B}$, so $\frac{\partial L_1}{\partial \boldsymbol{B}} = \boldsymbol{0}$.

We can rewrite L_2 as

$$L_2 = \sum_{i=1}^{p-1}\sum_{j=i+1}^{p}\left(\lambda w_{ij}x_{ij} + (1-\lambda)\,\tilde{w}_{ij}\hat{\delta}_{ij}\right)\left(\sum_{k=1}^{r}(b_{ik} - b_{jk})^2\right)^{1/2}.$$

Then

$$\frac{\partial L_2}{\partial b_{lm}} = \sum_{j=1,j\neq l}^{p}\left(\lambda w_{lj}x_{lj} + (1-\lambda)\,\tilde{w}_{lj}\hat{\delta}_{lj}\right)\cdot\frac{1}{2}\cdot 2\,(b_{lm} - b_{jm})\left(\sum_{k=1}^{r}(b_{lk} - b_{jk})^2\right)^{-1/2}$$
$$= \sum_{j=1,j\neq l}^{p}\left(\lambda w_{lj}x_{lj} + (1-\lambda)\,\tilde{w}_{lj}\hat{\delta}_{lj}\right)(b_{lm} - b_{jm})\,||\boldsymbol{b}_l - \boldsymbol{b}_j||^{-1}.$$

We can rewrite L_3 as

$$L_3 = \sum_{i=1}^{p-1}\sum_{j=i+1}^{p}\left(\lambda w_{ij} + (1-\lambda)\,\tilde{w}_{ij}\right)\sum_{k=1}^{r}(b_{ik} - b_{jk})^2.$$

Then

$$\frac{\partial L_3}{\partial b_{lm}} = \sum_{j=1, j\neq 1}^{p} \left(\lambda w_{ij} + (1-\lambda)\,\tilde{w}_{ij}\right) \cdot 2\left(b_{lm} - b_{jm}\right)$$
$$= 2\sum_{j=1, j\neq 1}^{p} \left(\lambda w_{ij} + (1-\lambda)\,\tilde{w}_{ij}\right)\left(b_{lm} - b_{jm}\right).$$

Hence,

$$\frac{\partial L^{(1)}}{\partial b_{lm}} = 4\sum_{j=1, j\neq 1}^{p} \left(\gamma_{lj}^{(1)} - \gamma_{lj}^{(2)} ||\boldsymbol{b}_l - \boldsymbol{b}_j||^{-1}\right)\left(b_{lm} - b_{jm}\right),$$

where

$$\gamma_{lj}^{(1)} = \lambda w_{ij} + (1-\lambda)\,\tilde{w}_{ij}$$

and

$$\gamma_{lj}^{(2)} = \lambda w_{lj} x_{lj} + (1-\lambda)\,\tilde{w}_{lj}\hat{\delta}_{lj}.$$

Hence,

$$\frac{\partial L^{(1)}}{\partial \boldsymbol{b}_l} = 4\sum_{j=1, j\neq l}^{p} \left(\gamma_{lj}^{(1)} - \gamma_{lj}^{(2)} ||\boldsymbol{b}_l - \boldsymbol{b}_j||^{-1}\right)\left(\boldsymbol{b}_l - \boldsymbol{b}_j\right).$$

Now, let $\boldsymbol{\Xi}$ be a matrix such that

$$\xi_{ij} = \begin{cases} \gamma_{ij}^{(1)} - \gamma_{ij}^{(2)} ||\boldsymbol{b}_i - \boldsymbol{b}_j||^{-1} & \text{if } i \neq j \\ 0 & \text{if } i = j. \end{cases}$$

Then,

$$\frac{\partial L^{(1)}}{\partial \boldsymbol{b}_i} = 4\sum_{j=1}^{p} \xi_{ij}\left(\boldsymbol{b}_i - \boldsymbol{b}_j\right) = 4\left(\left(\boldsymbol{\xi}_i^T \mathbf{1}_p\right)\boldsymbol{b}_i + \boldsymbol{\xi}_i^T \boldsymbol{B}\right).$$

Hence,

$$\frac{\partial L^{(1)}}{\partial \boldsymbol{B}} = 4\left(\sum_{j=1}^{p} \operatorname{diag}\left(\boldsymbol{\xi}_j\right)\boldsymbol{B} - \sum_{j=1}^{p} \boldsymbol{\xi}_j \boldsymbol{b}_j^T\right).$$

Hence, to obtain an estimate of the gradient with respect to a batch $\mathcal{S} = \{j_1, ..., j_q\} \subset \{1, ..., p\}$, we calculate

$$\frac{\partial L^{(1)}}{\partial \boldsymbol{B}} \approx 4\left(\sum_{j\in\mathcal{S}} \operatorname{diag}\left(\boldsymbol{\xi}_j\right)\boldsymbol{B} - \sum_{j\in\mathcal{S}} \boldsymbol{\xi}_j \boldsymbol{b}_{j_k}^T\right).$$

Case 2

Using the inner product formulation, the objective function (Equation 4.7) is

$$L^{(2)}(\boldsymbol{B}) = \lambda \sum_{i,j} w_{ij} \left(x_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 + (1-\lambda) \sum_{i,j} \tilde{w}_{ij} \left(\hat{\delta}_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2.$$

In order to calculate the gradient for each b_{kj}, rewrite $L^{(2)}$ as

$$\begin{aligned} L^{(2)}(\boldsymbol{B}) = &\sum_{i=1}^{p} \left(\lambda w_{ii} \left(x_{ii} - \sum_{k=1}^{r} b_{ik}^2 \right)^2 + (1-\lambda) \tilde{w}_{ii} \left(\hat{\delta}_{ii} - \sum_{k=1}^{r} b_{ik}^2 \right)^2 \right) \\ &+ \sum_{i=1}^{p} \sum_{j=1, j\neq i}^{p} \left(\lambda w_{ij} \left(x_{ij} - \sum_{k=1}^{r} b_{ik} b_{jk} \right)^2 + (1-\lambda) \sum_{i,j} \tilde{w}_{ij} \left(\hat{\delta}_{ij} - \sum_{k=1}^{r} b_{ik} b_{jk} \right)^2 \right). \end{aligned}$$

Then,

$$\begin{aligned} \frac{\partial L}{\partial b_{il}} &= (-4b_{il}) \left(\lambda w_{ii} \left(x_{ii} - \sum_{k=1}^{r} b_{ik}^2 \right) + (1-\lambda)\, \tilde{w}_{ii} \left(\hat{\delta}_{ii} - \sum_{k=1}^{r} b_{ik}^2 \right) \right) \\ &\quad + 2 \sum_{j=1, j\neq i}^{p} (-2b_{jl}) \left(\lambda w_{ij} \left(x_{ij} - \sum_{k=1}^{r} b_{ik} b_{jk} \right) + (1-\lambda) \tilde{w}_{ij} \left(\hat{\delta}_{ij} - \sum_{k=1}^{r} b_{ik} b_{jk} \right) \right) \\ &= -4\lambda \left(b_{il} w_{ii} \left(x_{ii} - \sum_{k=1}^{r} b_{ik}^2 \right) + \sum_{j=1, j\neq i}^{p} b_{jl} w_{ij} \left(x_{ij} - \sum_{k=1}^{r} b_{ik} b_{jk} \right) \right) \\ &\quad - 4\,(1-\lambda) \left(b_{il} \tilde{w}_{ii} \left(\hat{\delta}_{ii} - \sum_{k=1}^{r} b_{ik}^2 \right) + \sum_{j=1, j\neq i}^{p} b_{jl}\,(1-w_{ij}) \left(\hat{\delta}_{ij} - \sum_{k=1}^{r} b_{ik} b_{jk} \right) \right) \\ &= -4 \sum_{j=1}^{p} b_{jl} \left(\lambda w_{ij} \left(x_{ij} - \sum_{k=1}^{r} b_{ik} b_{jk} \right) + (1-\lambda)\, \tilde{w}_{ij} \left(\hat{\delta}_{ij} - \sum_{k=1}^{r} b_{ik} b_{jk} \right) \right). \end{aligned}$$

Hence, the gradient with respect to $\boldsymbol{B}$ is

$$\frac{\partial L^{(2)}}{\partial \boldsymbol{B}} = -4 \left(\lambda \boldsymbol{W} \otimes \left(\boldsymbol{X} - \boldsymbol{B}\boldsymbol{B}^T \right) + (1-\lambda)\, \tilde{\boldsymbol{W}} \otimes \left(\hat{\boldsymbol{\Delta}} - \boldsymbol{B}\boldsymbol{B}^T \right) \right) \boldsymbol{B},$$

where $\hat{\boldsymbol{\Delta}}$ is a matrix whose ijth element is $\hat{\delta}_{ij}$.

We can write this as a sum

$$\frac{\partial L}{\partial \boldsymbol{B}} = -4 \sum_{j=1}^{p} \left(\lambda \boldsymbol{W} \otimes \left(\boldsymbol{X} - \boldsymbol{B}\boldsymbol{B}^T \right) + (1-\lambda)\, \tilde{\boldsymbol{W}} \otimes \left(\hat{\boldsymbol{\Delta}} - \boldsymbol{B}\boldsymbol{B}^T \right) \right)_{,j} \boldsymbol{b}_j^T,$$

where $\boldsymbol{C}_{,j}$ denotes the jth column of a matrix $\boldsymbol{C}$, and $\boldsymbol{b}_j$ is the jth row of $\boldsymbol{B}$.

For stochastic gradient descent, we use only a subset of the j's, $\mathcal{S} \subset \{1, ..., j\}$. To calculate the approximate gradient with respect to a batch $\mathcal{S}$, let $\tilde{\boldsymbol{B}}$ be a matrix containing the rows of $\boldsymbol{B}$ that correspond to the indices in $\mathcal{S}$. Then the approximation of $\frac{\partial L}{\partial \boldsymbol{B}}$ we use is

$$\frac{\partial L}{\partial \boldsymbol{B}} \approx -4\left(\lambda \boldsymbol{W} \otimes \left(\boldsymbol{X} - \boldsymbol{B}\tilde{\boldsymbol{B}}^T\right) + (1-\lambda)\,\tilde{\boldsymbol{W}} \otimes \left(\hat{\boldsymbol{\Delta}} - \boldsymbol{B}\tilde{\boldsymbol{B}}^T\right)\right)\tilde{\boldsymbol{B}}.$$

Case 3

In order to calculate the gradient, we split the objective function into its supervised and unsupervised components. We denote the unsupervised part by $f(\boldsymbol{B})$ and the supervised part by $g(\boldsymbol{B})$: so

$$f(\boldsymbol{B}) = \lambda \sum_{i,j} w_{ij}\left(x_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2$$

and

$$g(\boldsymbol{B}) = (1-\lambda)\left(1 - \rho(\boldsymbol{z}(\boldsymbol{D}, \boldsymbol{B}), \boldsymbol{y}(\boldsymbol{D}))\right),$$

where ρ is the Pearson correlation coefficient between $\boldsymbol{z}$ and $\boldsymbol{y}$.

The derivative of f we have already calculated, as it is the same as the first part of the objective function in Case 2. To calculate the derivative of g with respect to $\boldsymbol{B}$, we use the chain rule.

$$\frac{\partial g}{\partial \boldsymbol{B}} = \frac{\partial g}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{B}}.$$

To calculate $\frac{\partial g}{\partial \boldsymbol{z}}$, we use the derivative of the Pearson correlation function [Strickert et al., 2008]

$$\frac{\partial \rho\left(\boldsymbol{z}, \boldsymbol{y}\right)}{\partial z_i} = \frac{(y_i - \bar{y}) - \frac{S_{yz}}{S_z}(z_i - \bar{z})}{\left(S_y S_z\right)^{1/2}}, \tag{4.11}$$

where

$$S_{yz} = \sum_{k=1}^{m} (z_k - \bar{z})(y_k - \bar{y})$$

$$S_z = \sum_{k=1}^{m} (z_k - \bar{z})^2$$

$$S_y = \sum_{k=1}^{m} (y_k - \bar{y})^2.$$

Hence,

$$\frac{\partial g}{\partial z_i} = -(1-\lambda)\left(\frac{(y_i-\bar{y}) - \frac{S_{yz}}{S_z}(z_i-\bar{z})}{(S_y S_z)^{1/2}}\right). \tag{4.12}$$

Each z_i is a function of two word embeddings, which we denote $\boldsymbol{b}_{i_1}$ and $\boldsymbol{b}_{i_2}$. For each z_i, we have

$$z_i = \cos\left(\langle \boldsymbol{b}_{i_1}, \boldsymbol{b}_{i_2}\rangle\right) = \frac{\sum_{k=1}^r b_{i_1,k} b_{i_2,k}}{\left(\sum_{k=1}^r b_{i_1,k}^2\right)^{1/2}\left(\sum_{k=1}^r b_{i_2,k}^2\right)^{1/2}} = \frac{\tilde{S}_{12}}{\tilde{S}_1^{1/2}\tilde{S}_2^{1/2}},$$

where

$$\begin{aligned}\tilde{S}_{12} &= \sum_{k=1}^r b_{i_1,k} b_{i_2,k}\\ \tilde{S}_1 &= \sum_{k=1}^r b_{i_1,k}^2\\ \tilde{S}_2 &= \sum_{k=1}^r b_{i_2,k}^2.\end{aligned}$$

So, we get

$$\frac{\partial z_i}{\partial b_{i_1,l}} = \frac{\tilde{S}_1^{1/2}\tilde{S}_2^{1/2}\cdot\frac{\partial}{\partial b_{i_1,l}}\left(\tilde{S}_{12}\right) - \tilde{S}_{12}\cdot\frac{\partial}{\partial b_{i_1,k}}\left(\tilde{S}_1^{1/2}\tilde{S}_2^{1/2}\right)}{\left(\tilde{S}_1^{1/2}\tilde{S}_2^{1/2}\right)^2}.$$

Now,

$$\begin{aligned}\frac{\partial \tilde{S}_{12}}{\partial b_{i_1,l}} &= \frac{\partial}{\partial b_{i_1,l}}\left(\sum_{k=1}^r b_{i_1,k} b_{i_2,k}\right) = b_{i_2,l},\\ \frac{\partial \tilde{S}_1}{\partial b_{i_1,l}} &= \frac{\partial}{\partial b_{i_1,l}}\left(\sum_{k=1}^r b_{i_1,k}^2\right) = 2b_{i_1,l},\end{aligned}$$

and

$$\frac{\partial}{\partial b_{i_1,l}}\left(\tilde{S}_1^{1/2}\tilde{S}_2^{1/2}\right) = \frac{1}{2}\tilde{S}_1^{-1/2}\cdot 2b_{i_1,l}\cdot\tilde{S}_2^{1/2} = \left(\frac{\tilde{S}_2}{\tilde{S}_1}\right)^{1/2} b_{i_1,l}.$$

So,

$$\frac{\partial z_i}{\partial b_{i_1,l}} = \frac{\left(\tilde{S}_1\tilde{S}_2\right)^{1/2} b_{i_2,l} - \tilde{S}_{12}\left(\frac{\tilde{S}_2}{\tilde{S}_1}\right)^{1/2} b_{i_1,l}}{\tilde{S}_1\tilde{S}_2} = \frac{b_{i_2,l} - \tilde{S}_{12}\tilde{S}_1^{-1} b_{i_1,l}}{\left(\tilde{S}_1\tilde{S}_2\right)^{1/2}} = \frac{b_{i_2,l}}{\left(\tilde{S}_1\tilde{S}_2\right)^{1/2}} - \frac{b_{i_1,l} z_i}{\tilde{S}_1},$$

and, by symmetry,

$$\frac{\partial z_i}{\partial b_{i_2,l}} = \frac{b_{i_1,l}}{\left(\tilde{S}_1\tilde{S}_2\right)^{1/2}} - \frac{b_{i_2,l} z_i}{\tilde{S}_2}.$$

Hence,

$$\frac{\partial z_i}{\partial \boldsymbol{b}_{i_1}} = \left(\tilde{S}_1\tilde{S}_2\right)^{-1/2}\boldsymbol{b}_{i_2} - z_i\tilde{S}_1^{-1}\boldsymbol{b}_{i_1},$$

and

$$\frac{\partial z_i}{\partial \boldsymbol{b}_{i_1}} = \left(\tilde{S}_1\tilde{S}_2\right)^{-1/2}\boldsymbol{b}_{i_1} - z_i\tilde{S}_2^{-1}\boldsymbol{b}_{i_2}.$$

Since a particular embedded word $\boldsymbol{b}_j$ can appear in more than one row of the test set — in other words, there may be multiple i_q's $(q = 1, 2)$ with $j = i_q$ — we now need to take account of this. We can use the indicator function, so that we can write

$$\frac{\partial z_i}{\partial \boldsymbol{b}_j} = \mathbb{I}_{j=i_1}\left(\tilde{S}_1\tilde{S}_2\right)^{-1/2}\boldsymbol{b}_{i_2} - z_i\tilde{S}_1^{-1}\boldsymbol{b}_{i_1} + \mathbb{I}_{j=i_2}\left(\tilde{S}_1\tilde{S}_2\right)^{-1/2}\boldsymbol{b}_{i_1} - z_i\tilde{S}_2^{-1}\boldsymbol{b}_{i_2}.$$

Putting this together with Equation 4.12 gives

$$\frac{\partial g}{\partial \boldsymbol{b}_j} = \sum_{i=1}^{m}\frac{\partial g}{\partial z_i}\frac{\partial z_i}{\partial \boldsymbol{b}_j} = -(1-\lambda)\sum_{i=1}^{m} h_1^{(i)}(\boldsymbol{y}, \boldsymbol{z})\boldsymbol{h}_{\boldsymbol{2}}^{(i)}(\boldsymbol{z}, \boldsymbol{b}_j),$$

where

$$h_1^{(i)}(\boldsymbol{y}, \boldsymbol{z}) = \frac{(y_i - \bar{y}) - \frac{S_{yz}}{S_z}(z_i - \bar{z})}{(S_y S_z)^{1/2}}$$

and

$$\boldsymbol{h}_{\boldsymbol{2}}^{(i)}(\boldsymbol{z}, \boldsymbol{b}_j) = \mathbb{I}_{j=i_1}\left(\tilde{S}_1\tilde{S}_2\right)^{-1/2}\boldsymbol{b}_{i_2} - z_i\tilde{S}_1^{-1}\boldsymbol{b}_{i_1} + \mathbb{I}_{j=i_2}\left(\tilde{S}_1\tilde{S}_2\right)^{-1/2}\boldsymbol{b}_{i_1} - z_i\tilde{S}_2^{-1}\boldsymbol{b}_{i_2}.$$

Hence,

$$\frac{\partial L^{(3)}(\boldsymbol{B})}{\partial \boldsymbol{b}_j} = -4\lambda\sum_{i=1}^{p} w_{ij}\left(x_{ij} - \boldsymbol{b}_i^T\boldsymbol{b}_j\right)\boldsymbol{b}_i - (1-\lambda)\sum_{k=1}^{m} h_1^{(k)}(\boldsymbol{y}, \boldsymbol{z})\boldsymbol{h}_{\boldsymbol{2}}^{(k)}(\boldsymbol{z}, \boldsymbol{b}_j).$$

Hence, the batch gradient is

$$\frac{\partial L^{(3)}(\boldsymbol{B})}{\partial \boldsymbol{b}_j} \approx -4\lambda\sum_{i\in\mathcal{S}_1} w_{ij}\left(x_{ij} - \boldsymbol{b}_i^T\boldsymbol{b}_k\right)\boldsymbol{b}_i - (1-\lambda)\sum_{k\in\mathcal{S}_2} h_1^{(k)}(\boldsymbol{y}, \boldsymbol{z})\boldsymbol{h}_{\boldsymbol{2}}^{(k)}(\boldsymbol{z}, \boldsymbol{b}_j), \quad (4.13)$$

where we take $\mathcal{S}_1$ to be a sample from $\{1, ..., p\}$ and $\mathcal{S}_2$ to be a sample from $\{1, ..., m\}$.

4.5 Results

We now investigate generating semi-supervised embeddings using simulated and real datasets.

4.5.1 Optimization via Majorization and Stochastic Gradient Descent

In this section we apply the majorization and stochastic gradient descent algorithms to two simulated datasets. In the first example, the data is simulated with $p = 100$, and we find that majorization is both faster than gradient descent and achieves a lower value for the objective function. In the second example, with $p = 10000$, we show that the stochastic gradient descent algorithm converges much faster than majorization, but majorization achieves a lower value of the objective function.

Small example

To compare using majorization and stochastic gradient descent to minimize the objective, we use a small simulated dataset, with $p = 100$ and $r = 2$. $\boldsymbol{X}$ is simulated as $\boldsymbol{D}\boldsymbol{D}^T$, where elements of $\boldsymbol{D}$ are simulated independently from $U(0, 1)$. $\boldsymbol{V}$ is generated by selecting 50 random pairs of indices (i, j), where we choose i uniformly from $\{1, ..., p - 1\}$, and j uniformly from $\{i + 1, ..., p\}$. Each (i, j) pair is assigned a value in $U(0, 1)$. (All other entries of $\boldsymbol{V}$ are assumed to be unobserved.) We use the objective for Case 2 (Equation 4.7). For stochastic gradient descent, we use a batch size of 50.

First, we investigate using stochastic gradient descent with different learning rates. Table 4.2 and Figure 4.2 give the times and objectives after 1000 iterations, for different learning rates. For each learning rate, the algorithm was run 10 times, and the minimum and mean values are given. This is to give an idea of what might be a good learning rate, without running the algorithm all the way to convergence,

Learning rate	Min time (secs)	Mean time (secs)	Min objective	Mean objective
0.001	4.29	5.06	6.56	17.21
0.002	4.74	5.47	3.05	4.51
0.005	4.29	5.20	3.05	3.06
0.010	4.25	4.52	3.03	3.08
0.015	4.53	4.95	32.93	122.97
0.020	4.28	4.53	14.47	922.07

Table 4.2: Time taken and objective reached after the first 1000 iterations, for different learning rates

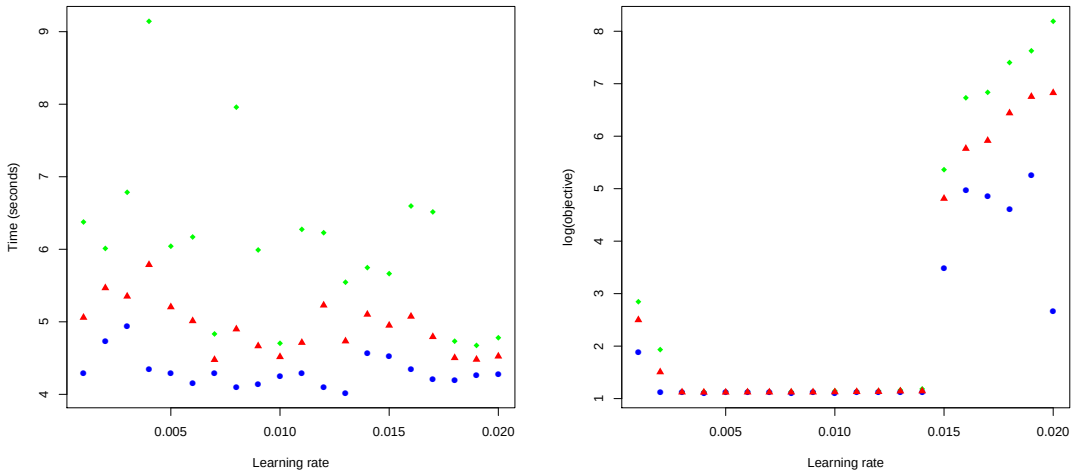

Figure 4.2: Plots showing the times and objectives for 1000 iterations, for different learning rates. Each plot shows the minimum (blue circle), mean (red triangle), and maximum (green diamond) values for learning rates between 0.001 and 0.02, across 10 runs of the algorithm. The value of the objective is plotted on a log scale for better visualization.

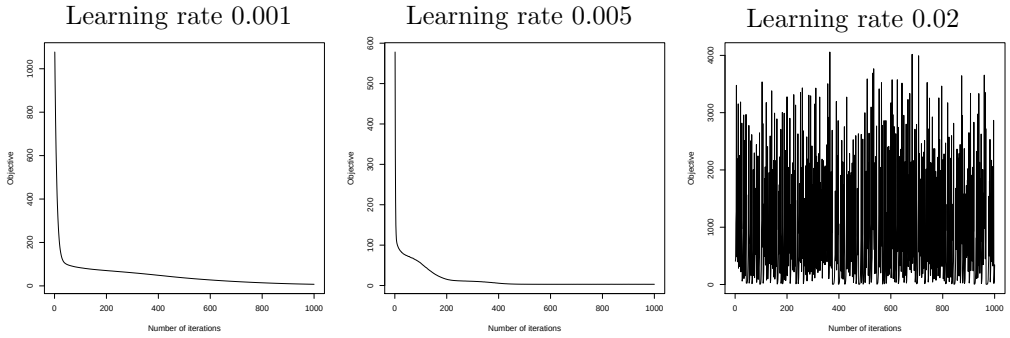

Figure 4.3: Plots showing how the objective function changes over the first 1000 iterations, for different values of the learning rate.

which could take a very long time if the learning rate is much too small. There does not seem to be a pattern in the times, with differences in run times probably being due to random variation. For the objective, we see that increasing the learning rate at first allows the objective to decrease more quickly, but if the learning rate is too high the objective will increase, and there is much more variation in its values (see Figure 4.3). The algorithm seems to do best when the learning rate is between about 0.003 and 0.014.

Figure 4.4 shows that increasing the rate of decay results in a higher value of the final objective, especially when the initial learning rate is higher. This is because decaying the learning rate too fast means the algorithm is more likely to get stuck in a local minimum. The right-hand plot shows that the time to convergence decreases as the rate of decay increases, but the initial learning rate does not seem to have much influence on the time to convergence. Thus, a higher rate of decay means that the algorithm converges to a limit faster, but it converges to a higher value of the objective function. There is therefore a trade-off to be made between having faster convergence versus converging to a better value of the objective.

Using majorization, the algorithm converges to an objective value of 2.998 in 0.407 seconds. This is a better result than stochastic gradient descent in terms of both objective value and time to converge. Hence, for a small dataset it makes sense to use majorization.

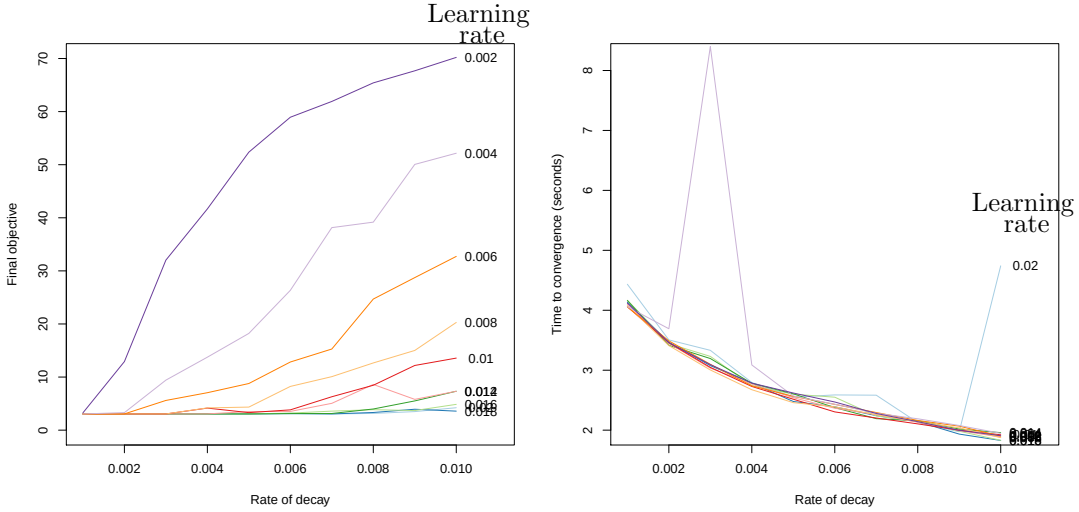

Figure 4.4: Plots showing the value of the objective function (left) and time to convergence (right) for different values of the initial learning rate and rate of decay. Each line corresponds to a different initial learning rate. The values given are the average values over 10 runs of the algorithm.

Larger example

We repeated this analysis with a larger example, where $p = 10000$ and $r = 100$. The data is simulated in the same way as previously (with 500 entries in $\boldsymbol{V}$). As before, we investigated using stochastic gradient descent with different initial learning rates and rates of decay, with a batch size of 50. We found similar patterns as we did previously: increasing the rate of decay leads to faster convergence, but usually a higher value of the final objective (Figures 4.5, 4.6, 4.7).

With stochastic gradient descent, the lowest value of the final objective we achieved was 17,552,469; this took 2414 seconds, or about 40 minutes, to converge. It is possible that a better result would have been possible, using a different combination of initial learning rate, rate of decay, and batch size, since it is of course not possible to try all possible combinations. With majorization, after 100 iterations we reached a lower objective value of 16,375,798, but this took 3.6 days to achieve. It is likely that a faster implementation of the algorithm could reduce this time, but

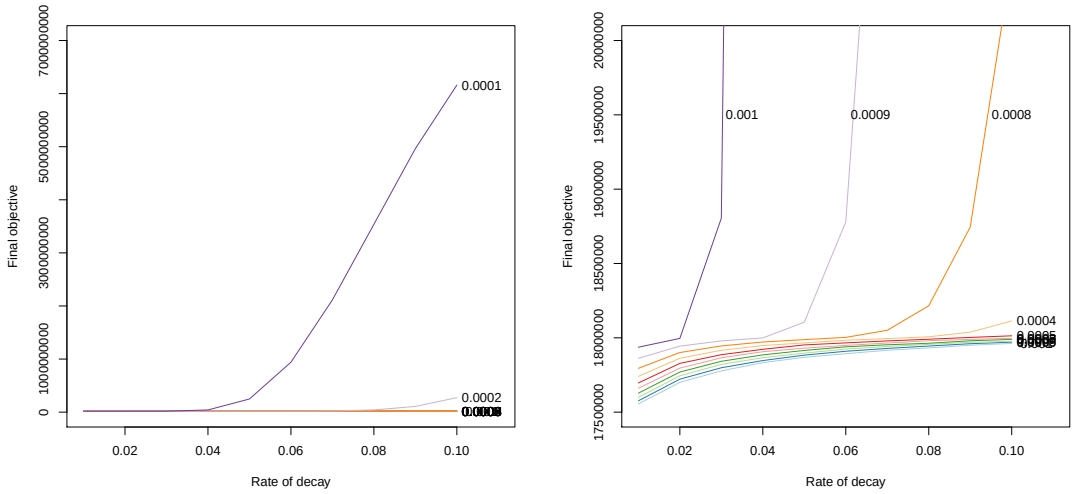

Figure 4.5: Plots showing how the value of the objective function changes for different values of the learning rate and rate of decay. The right-hand graph shows the same as the left-hand one, but with a different scale for the y-axis so it is easier to see. Each line corresponds to a different initial learning rate between 0.0001 and 0.001. The values given are the averages over 10 runs of the algorithm.

this is also true for stochastic gradient descent. So it seems certain that stochastic gradient descent is much faster. However, majorization seems to give a better result, and has the advantage of not having to find good values for the learning rate and rate of decay, by trial and error. Hence, which method we choose to use for the optimization will depend upon the time and resources available.

4.5.2 Implementation on subset of COHA

Table 4.3 shows the result of applying Case 3 (Equation 4.8) to some real data. We take the unsupervised dataset $\boldsymbol{X}$ to be the portion of the document-term matrix corresponding to the 10,000 most frequent words in COHA, and the supervised dataset $\boldsymbol{D}$ to be the WordSim-353 test set (restricted to the word pairs where both words are in $\boldsymbol{X}$). Table 4.3 gives the performance of the semi-supervised embeddings on the WordSim-353 and three other test sets, for different values of λ. We see that

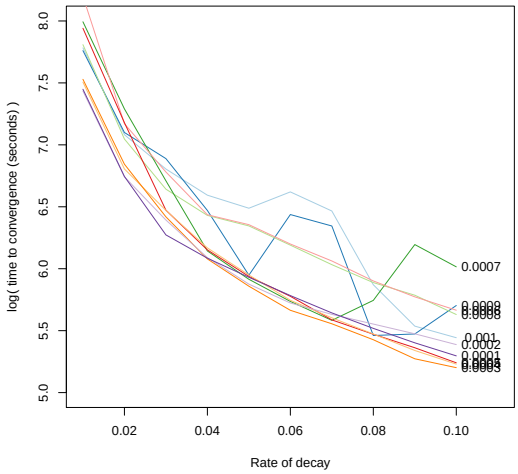

Figure 4.6: Plot showing how the time to the convergence (shown on the log scale) changes with different learning rates and rates of decay. There is some variation in the results, but generally using a higher learning rate results in faster convergence.

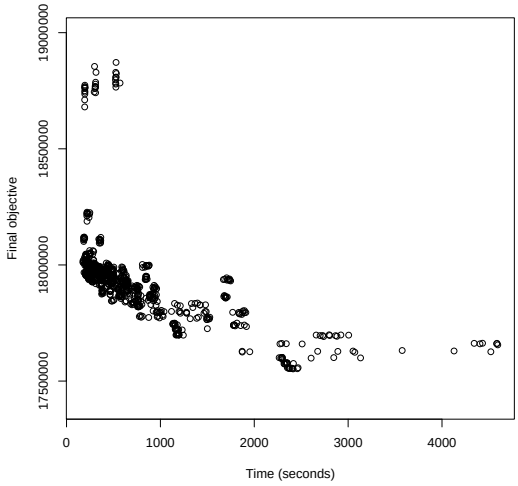

Figure 4.7: Plot of the objective function against the time taken to run the algorithm, for different runs (with different learning rates and rates of decay). (Some points with a very high final objective value are omitted, so that the graph is easier to see.)

performance on the WordSim-353 test set increases as λ decreases, which we would expect, as a lower value of λ means that the supervised part of the algorithm has a greater weight in the algorithm. However, the performance of the embeddings does not change much on the other test sets. We hypothesize that this is because including the supervised part of the algorithm only has a significant effect on the small subset of words that occur in the WordSim-353 test set, and the embeddings corresponding to words not in the test set are not much affected. (There is some overlap between test sets in terms of individual words — for example, the SimLex-999 test set contains 1028 unique words, 124 of which also occur in WordSim-353 — but there are no word pairs in the WordSim-353 test set that occur in any of the other test sets.) If the supervised dataset were larger, relative to the unsupervised dataset, we would expect to see more of an impact, and the value of λ having a larger effect on the test scores for the different test sets. Hence, the choice of λ would have a more significant effect on the results.

4.6 Conclusion

In this chapter we have explored the possibility of generating semi-supervised word embeddings, which combine a large unsupervised dataset with a small supervised dataset. The aim was to generate higher-quality embeddings compared to using unsupervised data alone.

We introduced three objective functions for generating semi-supervised embeddings. The first two of these were based on the MDS objective function, whilst the third was closer to a combination of standard word embedding training and test objectives. We use two different methods for optimizing the objectives, majorization and stochastic gradient descent. The choice of which optimization algorithm we should use depends on the situation; majorization is more accurate and does not require tuning of hyperparameters, whilst stochastic gradient descent converges faster, especially for larger datasets. We confirmed this through simulated examples.

	Test set			
λ	WordSim-353	SimLex-999	SimVerb-3500	MTurk-771
1	0.494	0.179	0.082	0.504
0.2	0.801	0.181	0.015	0.463
0.01	0.912	0.177	0.016	0.460
0.005	0.928	0.174	0.018	0.457
0.001	0.937	0.172	0.020	0.458

Table 4.3: Results of implementing Case 3 (Equation 4.8) with stochastic gradient descent (Algorithm 3, using the gradient in Equation 4.13), for some different values of λ. (The values of f and g are of a similar order for $\lambda = 0.01$. For larger λ, f dominates, and for smaller λ, g dominates.) The dataset the embeddings are trained on is the COHA document-term matrix, restricted to the 10,000 most common words in COHA. The WordSim-353 test set is used in the objective function and we can see that performance on this test set increases, and more so as λ decreases (as the importance of the supervised part of the objective function increases). However, performance on other test sets does not change much.

We created semi-supervised embedding sets using a subset of COHA, for different values of the hyperparameter λ. This showed that although (as we would expect) optimizing the semi-supervised objective leads to embeddings that perform better on the supervised test set $\boldsymbol{D}$ used in the training, the performance on other test sets does not significantly improve. We hypothesize that this is because $\boldsymbol{D}$ is very small compared to the unsupervised dataset $\boldsymbol{X}$, and including $\boldsymbol{D}$ in the semi-supervised embedding algorithm only seems to affect the words which occur in $\boldsymbol{D}$. However, it is possible that we would gain more from using semi-supervised embeddings, compared to unsupervised embeddings, when $\boldsymbol{X}$ is small.

In future, it would be interesting to investigate the potential of semi-supervised word embeddings on smaller datasets. In addition, in each of the three objective functions we have used for generating semi-supervised embeddings, the supervised part of the objective is related to the LSA objective, which we have chosen because it is a simple model. However, it would be interesting to explore the possibilities for generating semi-supervised word embeddings used objectives based on other models, such as GloVe and SGNS.

Chapter 5

Dynamic word embeddings: Testing for time dependence

In this chapter we investigate testing whether words change over time using word embeddings, including how this relates to the identifiability issues discussed in the previous chapter. We discuss carrying out statistical tests for whether a set of words has changed across time using a time-dependent extension of LSA. We make use of a factor analysis test and show, using simulations, that it has power against the presence of time dependence in our model. However, we show that although such a test is invariant to the choice of identifiability criteria (see Chapter 3), it is unable to distinguish between the presence of time dependence and a misspecified embedding dimension.

5.1 Motivation

A particular area of interest in recent research is the development of dynamic (time-dependent) word embeddings that aim to represent changing relationships between words over time. In Section 2.5 we discussed some of the recent work in this area and outlined a number of challenges that need to be solved. We discuss here issues with identifiability for time-dependent embeddings (the static case was covered in

Chapter 3), which we use to motivate the material introduced in this chapter.

Usually, time-dependent word embeddings are generated by dividing the time period up into smaller periods, such as years or decades, and using a static word embedding method to generate a set of embeddings for each period. The set of embeddings for each time period is non-identifiable: multiplying the embedding set by a matrix $\boldsymbol{C} \in \mathsf{GL}(r)$ (where $\mathsf{GL}(r)$ is the set of non-singular $r \times r$ matrices) does not change the value of the objective function. However, there are additional issues when comparing words across time periods. The amount of change a word has undergone between times t_{i_1} and t_{i_2} is measured as the cosine similarity between the embeddings for that word at times t_{i_1} and t_{i_2}:

$$change = \cos\left(\langle \boldsymbol{b}_j(t_{i_1}), \boldsymbol{b}_j(t_{i_2})\rangle\right). \tag{5.1}$$

Multiplying an embedding set at one time period, say times t_{i_1}, by a matrix $\boldsymbol{C} \in \mathsf{GL}(r)$ will change the value of this:

$$\cos\left(\langle \boldsymbol{b}_j(t_{i_1}), \boldsymbol{b}_j(t_{i_2})\rangle\right) \neq \cos\left(\langle \boldsymbol{C}\boldsymbol{b}_j(t_{i_1}), \boldsymbol{b}_j(t_{i_2})\rangle\right).$$

In the static case, cosine similarities between embeddings remained the same if the embedding set was multiplied by a matrix of the form $c\boldsymbol{Q}$, where $c > 0$ and $\boldsymbol{Q} \in \mathsf{O}(r)$. This is because, in the static case, all the embeddings were multiplied by the same transformation, and the cosine similarity between two vectors multiplied by the same transformation matrix is not affected by these transformations. However, in the time-dependent case, embeddings at different time periods can be multiplied by different transformation matrices, so unlike in the static case orthogonal and scale transformations are an issue here.

This issue was partially addressed by [Hamilton et al., 2016]: for each time period, they find the orthogonal transformation of the embedding set for that time period that is as close as possible to the embedding set for the previous time period. However, they do not appear to consider any transformations outside of the set $\mathsf{O}(r)$, although any element of $\mathsf{GL}(r)$ will lead to the same problem.

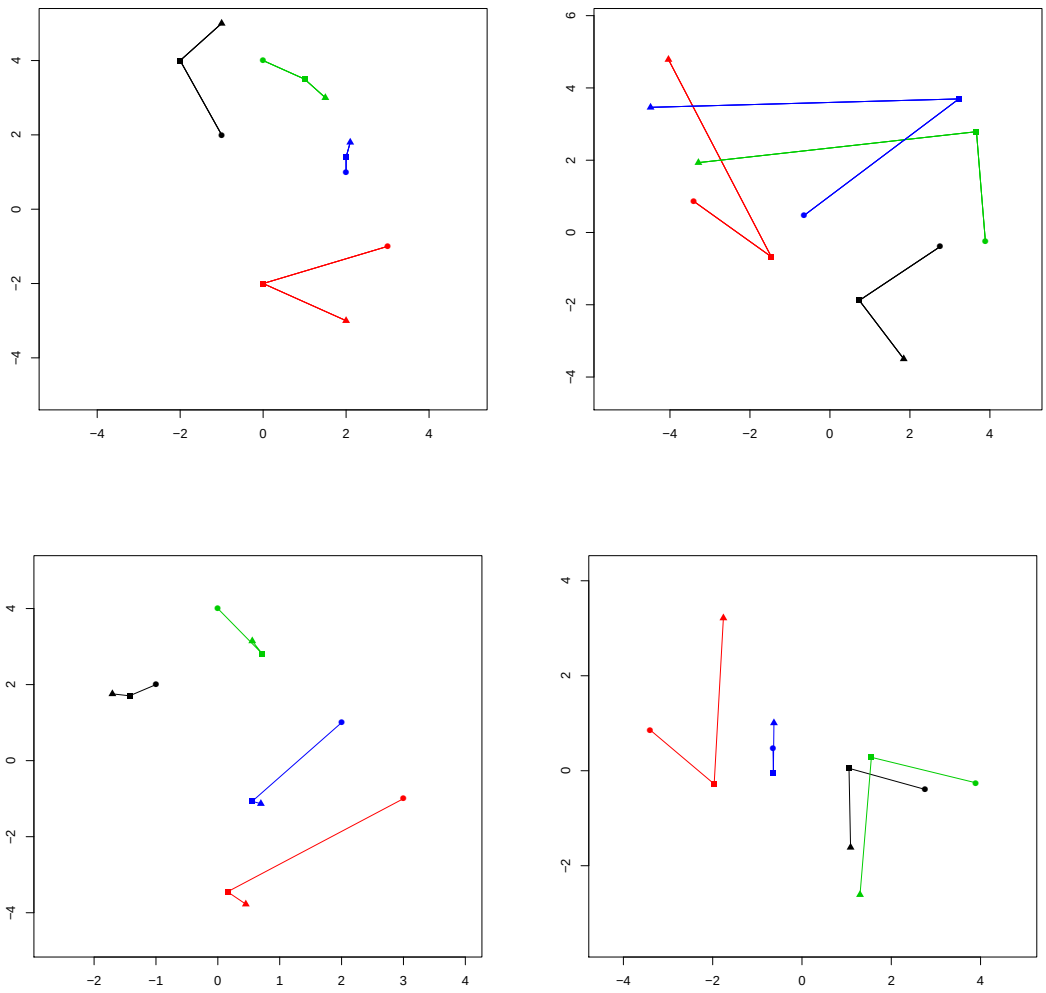

Figure 5.1: Figure showing how considering only orthogonal embeddings can be an issue, with four words and three time points. The top two graphs represent the same set of embeddings, but for each time point the embeddings used in the right-hand graph have been multiplied by a different matrix $\boldsymbol{C} \in \mathsf{GL}(r)$. Below, the two graphs have been aligned across time using a Procrustes algorithm. For each graph, the four words are represented by four different colours, and the circle, square and triangle correspond to times 0, 1 and 2 respectively.

Word	$\cos(\langle b_i^0, b_i^1 \rangle)$	$\cos(\langle b_i^0, b_i^2 \rangle)$	$\cos(\langle b_i^1, b_i^2 \rangle)$
1	0.492	0.587	0.994
2	0.779	0.813	0.268
3	0.755	-0.893	-0.379
4	-0.091	0.999	-0.060
1	0.982	0.671	0.519
2	0.925	0.681	0.352
3	0.970	0.503	0.279
4	0.758	0.926	0.456

Table 5.1: Cosine similarities between the positions of each word at different time periods, using embeddings corresponding to graphs (c) and (d) in Figure 5.1. The top half of the table is for Figure 5.1 (c), and the bottom half is for Figure 5.1 (d).

Figure 5.1 shows how this can be an issue. The top two graphs show the same set of embeddings across three time periods, where for one the embeddings at each time points have been transformed by an element $\boldsymbol{C} \in \mathsf{GL}(r)$ (by a different $\boldsymbol{C}$ for each time points). The bottom two graphs show the embeddings after they have been aligned by finding the closest orthogonal transformation of the time-1 embeddings to time-0 embeddings, and then the closest orthogonal transformation of time-2 embeddings to the time-1 embeddings. We would want these graphs to look the same, so that the embeddings are not affected by multiplication by elements of $\mathsf{GL}(r)$, but this is clearly not the case.

Since the amount a word is changed is determined by measuring the cosine similarity between the positions of that word in successive time periods (Equation 5.1), the fact that (as we have shown) non-singular transformations of the word embeddings at each time period can lead to these cosine similarities being quite different is problematic.

Hence, in this chapter, we attempt to develop a test for whether words have changed over time, which is not affected by multiplication of the embeddings by a

non-singular matrix, or by the choice of identifiability conditions imposed.

5.2 Introduction

This chapter mainly focuses on developing a more rigorous statistical framework to determine whether the embeddings associated with a given set of words have changed significantly over time. To do this, we first introduce a time-dependent word embedding model that we will make use of. We then consider the properties we would like such a test to have; in particular, we would like it to be invariant to the choice of identifiability conditions in the model. In Chapter 3 (Section 3.2), we showed that a word embedding matrix can be multiplied by a non-singular matrix without changing the value of the objective function, so to get a unique solution it is necessary to impose constraints on the embeddings. However, we would not like the test to be affected by the choice of constraints, as the choice is somewhat arbitrary.

We employ a factor analysis test for adequacy of the static model. We show that it has power to reject the null hypothesis when there is time dependence in the data, and show that the test statistic is impervious to transformations of the embedding set by a non-singular matrix, and hence to the choice of identifiability conditions. However, we also show that the test is unable to distinguish between the presence of time dependence and a misspecified embedding dimension.

In the first part of this chapter (Sections 5.2 to 5.5), we introduce a time-dependent extension of LSA, and discuss issues of identifiability and some properties of this model. Then, in Sections 5.6 to 5.8, we develop a statistical framework for testing whether a word or set of words significantly changes over time within a particular dataset.

5.3 Dynamic LSA model

We introduce here a dynamic (time-dependent) version of Latent Semantic Analysis (LSA) that will be used in this chapter.

Recall from Section 2.3.1 that LSA takes the $n \times p$ document-term matrix $\boldsymbol{X}$, and approximates it by

$$\boldsymbol{X} \approx \boldsymbol{AB},$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are $n \times r$ and $r \times p$ matrices of document and word vectors respectively, with $r << \min\{n, p\}$.

For the dynamic case, we make use of the fact that we can associate a time with each row of the document-term matrix. To incorporate these times into the model, we allow the word embeddings $\boldsymbol{b}_j$ to be functions of time: $\boldsymbol{b}_j = \boldsymbol{b}_j(t)$. The simplest case incorporating time dependence is when the functions $\boldsymbol{b}_j(t)$ are linear:

$$\boldsymbol{b}_j(t) = \boldsymbol{b}_j^0 + \boldsymbol{b}_j^1 t,$$

where both $\boldsymbol{b}_j^0$ and $\boldsymbol{b}_j^1$ are r-dimensional vectors. Hence, the objective function is

$$J(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1 | \boldsymbol{X}) = ||\boldsymbol{X} - \boldsymbol{AB}_0 + \boldsymbol{TAB}_1||_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{p} \left(x_{ij} - \boldsymbol{a}_i^T \boldsymbol{b}_j^0 - t_i \boldsymbol{a}_i^T \boldsymbol{b}_j^1\right)^2. \quad (5.2)$$

We saw in Section 2.3.1 that minimization of the objective function for LSA with respect to $\boldsymbol{A}$ and $\boldsymbol{B}$ was equivalent to maximizing the likelihood of the model

$$x_{ij} \overset{ind}{\sim} N\left(\boldsymbol{a}_i^T \boldsymbol{b}_j, \sigma^2\right) \quad i = 1, ..., n;, \; j = 1, ..., p,$$

or, equivalently,

$$\boldsymbol{X} = \boldsymbol{AB} + \boldsymbol{Z},$$

where $z_{ij} \overset{iid}{\sim} N(0, \sigma^2)$. In the same way, for the dynamic case we have the model

$$\boldsymbol{X} = \boldsymbol{AB}_0 + \boldsymbol{TAB}_1 + \boldsymbol{Z}, \quad (5.3)$$

where

$$\boldsymbol{A} = \left(\boldsymbol{a}_1^T, ..., \boldsymbol{a}_n^T\right)^T,$$

$$\boldsymbol{B}_0 = \left(\boldsymbol{b}_1^0, ..., \boldsymbol{b}_p^0\right),$$

$$\boldsymbol{B}_1 = \left(\boldsymbol{b}_1^1, ..., \boldsymbol{b}_p^1\right),$$

$$\boldsymbol{T} = \text{diag}\left(t_1, ..., t_n\right).$$

$\boldsymbol{A}$ is $n \times r$, $\boldsymbol{B}_0$ and $\boldsymbol{B}_1$ are $r \times p$, $\boldsymbol{Z}$ is $n \times p$ $\boldsymbol{T}$ is $n \times n$.

Remark. Observe that if $\boldsymbol{T} = \boldsymbol{0}$, then we recover the static LSA model.

This model has the advantage that it avoids dividing the data up into broad time periods (such as decades or years). In addition, using the document-term matrix makes it easier to incorporate time dependence than when we use the co-occurrence matrix, because each document (and thus each row of the document-term matrix) has a time associated with it. Since the co-occurrence matrix aggregates data across the corpus, to retain temporal information we would need to produce a co-occurrence matrix for each time point, which would be much more computationally challenging.

It is necessary to consider whether this is a suitable model for the data. The Normal distribution assumes data is continuous and can take positive or negative values, whereas the data that we have is discrete and non-negative. Probabilistic Latent Semantic Indexing (PLSI) uses a Multinomial distribution for the document-term matrix (see Section 2.3.2), which seems more appropriate. However, this can be approximated by a Normal distribution as long as the length of the documents is sufficiently large; so a Normal distribution seems reasonable as long as the counts are away from zero. Having zero (or very small) counts in the document-term matrix may still be an issue, as this is on the boundary of the Multinomial parameter space and so the Normal approximation will not work well. Despite this issue, we proceed with this model as it is the simplest means of implementing a time-dependent model. It is hoped that the results could later be generalized to other, more realistic, models.

We also note that while a dynamic version of the factor model exists (see e.g. [Bai, 2003]), where $\boldsymbol{a}_i$ is modelled as a multivariate time series, this does not seem

an appropriate model here, as it involves a time-dependent model for $\boldsymbol{A}$, but not $\boldsymbol{B}$, which is what we are interested in. (The time series model also seems inappropriate for $\boldsymbol{a}_i$, as it would necessitate having only one document per time point, which is not the case for our data.)

5.3.1 Identifiability

Similarly to the static methods we looked at in Chapter 3, the solution to the minimization problem

$$\arg\min_{\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1} J(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1)$$

is not unique. In particular, for any $(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1)$, and any $\boldsymbol{C} \in \mathsf{GL}(r)$ (where $\mathsf{GL}(r)$ is the set of invertible $r \times r$ matrices),

$$J\left(\boldsymbol{AC},\boldsymbol{C}^{-1}\boldsymbol{B}_0,\boldsymbol{C}^{-1}\boldsymbol{B}_1\right) = J(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1).$$

Identifiability is an issue for developing statistical tests (see Section 5.5) where we have to estimate the true values of parameters; they must be identifiable in order for their true values to be well-defined. In addition, since there are multiple solutions which optimize the objective function, we do not want the conclusion drawn from a test to be dependent upon which solution we choose. Therefore, we want the test statistic to be invariant to the transformation $(\boldsymbol{A},\boldsymbol{B}_0,\boldsymbol{B}_1) \rightarrow (\boldsymbol{AC},\boldsymbol{C}^{-1}\boldsymbol{B}_0,\boldsymbol{C}^{-1}\boldsymbol{B}_1)$.

As in Section 3.4.1, the solution can be made unique by adding constraints to the matrices $\boldsymbol{A}$, $\boldsymbol{B}_0$ and $\boldsymbol{B}_1$. There are a number of ways this can be done. We present two ways below in Theorems 3 and 4, and then discuss why we might prefer one of these over the other.

Theorem 3 (Identifiability of dynamic model: 1)**.** *Assume that:*

i. $\boldsymbol{A}$ *has full column rank;*

ii. For all $i = 1, ..., n$, $t_i \neq 0$*;*

iii. $J(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1) = J(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1) \Rightarrow \boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1 = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}_0 + \boldsymbol{T}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}_1$

Then the following constraints guarantee uniqueness of the solution set $(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)$*:*

1. $\boldsymbol{A}$ *has orthonormal columns.*

2. *The* i*th column of* $\boldsymbol{A}$ *has its first* $r - i$ *entries equal to* 0*.*

3. *The first non-zero entry in each column of* $\boldsymbol{A}$ *is positive.*

That is, if there are two solutions $(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)$ and $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1)$ that both satisfy these constraints, such that $J(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1) = J(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1)$, then $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1) = (\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)$. In addition, for any given solution $(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)$ that does not satisfy the constraints, it is possible to find a solution that has the same value of the objective function and that does satisfy the constraints. The proof of this theorem will be given below.

Alternatively, we can use similar constraints to SVD:

Theorem 4 (Identifiability of dynamic model: 2)**.** *Under the same assumptions as Theorem 3, with the additional assumption that the first* r *singular values of* $\boldsymbol{A}\boldsymbol{B}_0$ *are all distinct, the following constraints guarantee uniqueness of the solution set up to multiplication by a permutation matrix:*

1. $\boldsymbol{A}$ *has orthogonal columns.*

2. $\boldsymbol{B}_0$ *has orthonormal rows.*

3. *The first non-zero entry of each column of* $\boldsymbol{A}$ *is positive*

The proof of Theorem 4 is given below.

Remark. We can make $\boldsymbol{A}$, $\boldsymbol{B}_0$ and $\boldsymbol{B}_1$ uniquely identifiable by fixing the order of the columns of $\boldsymbol{A}$ and rows of $\boldsymbol{B}_0$ and $\boldsymbol{B}_1$, for example by ordering them by the magnitude of the corresponding singular values of $\boldsymbol{A}\boldsymbol{B}_0$.

Remark. It is not necessary to impose constraints on $\boldsymbol{B}_1$: if $\boldsymbol{A}$ and $\boldsymbol{B}_0$ are fixed, then there is a unique $\boldsymbol{B}_1$ that minimizes the objective function. Thus, if we can uniquely identify $\boldsymbol{A}$ and $\boldsymbol{B}_0$, then $\boldsymbol{B}_1$ will also be identifiable.

The constraints imposed may affect how we interpret the embeddings we obtain, so we should consider this when choosing which constraints to use. The first set of constraints treat the first few rows of $\boldsymbol{A}$ (and hence documents in $\boldsymbol{X}$) differently to the others, by imposing zeros in their corresponding document vectors, which is undesirable unless we have a particular justification for doing this. For the second case, all rows of $\boldsymbol{A}$ are placed under the same constraints, and the same is true for the columns of $\boldsymbol{B}_0$. Therefore, it seems that the second set of constraints make more sense from the point of view of interpretation.

The model in Equation 5.3 contains $nr + 2pr$ parameters. Both sets of identifiability conditions reduce the number of degrees of freedom by r^2. For the first set of conditions, there are $\frac{1}{2}(r-1)r$ parameters fixed at 0; the requirement that the columns of $\boldsymbol{A}$ are orthogonal adds another $\frac{1}{2}(r-1)r$ constraints, and another r constraints are added by the requirement that the columns have norm 1. For the second set of constraints, we still have that the columns of $\boldsymbol{A}$ are orthogonal and have norm 1, giving a total of $\frac{1}{2}r(r+1)$ constraints, and the requirement that the rows of $\boldsymbol{B}_0$ are orthogonal adds another $\frac{1}{2}(r-1)r$, bringing the total to r^2 again.

Proof of Theorem 3

Proof. Assume that $\boldsymbol{A}$ has full column rank, and that $t_i \neq 0$ for all $i = 1, ..., n$. The latter can be assumed without loss of generality: if there is an i for which $t_i = 0$, this can be fixed by adding a constant each t_i. (It will be shown in Section 5.3.2 that doing this does not change the value of the objective function (Equation 5.2).) Assume also that if $J(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1) = J(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1)$, then $\boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1 = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}_0 + \boldsymbol{T}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}_1$. This implies that if, given that $\boldsymbol{X}$ is fixed, we have $J(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1) = J(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1)$, then there exists $\boldsymbol{C} \in \mathsf{GL}(r)$ such that $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1) = (\boldsymbol{A}\boldsymbol{C}, \boldsymbol{C}^{-1}\boldsymbol{B}_0, \boldsymbol{C}^{-1}\boldsymbol{B}_1)$.

First, we show that if two solution sets that satisfy the constraints in Theorem 3 give the same value for the objective function, then they must be identical.

Suppose that both $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}} = \boldsymbol{AC}$ satisfy the given constraints:

1. $\boldsymbol{A}$ has orthonormal columns.

2. The ith column of $\boldsymbol{A}$ has its first $r - i$ entries equal to 0.

3. The first non-zero entry in each column of $\boldsymbol{A}$ is positive.

Constraint 1 implies that

$$\boldsymbol{I}_r = (\boldsymbol{AC})^T \boldsymbol{AC} = \boldsymbol{C}^T \boldsymbol{A}^T \boldsymbol{AC} = \boldsymbol{C}^T \boldsymbol{I}_r \boldsymbol{C} = \boldsymbol{C}^T \boldsymbol{C},$$

so $\boldsymbol{C}$ is an orthogonal matrix.

Constraint 2 implies that, for $i \in \{1, ..., n\}, k \in \{1, ..., r\}$, if $i \leq r - k$ (or equivalently if $k \leq r - i$), then $a_{ik} = (\boldsymbol{AC})_{ik} = 0$. Hence for all i, k

$$(\boldsymbol{AC})_{ik} = \sum_{l=1}^{r} a_{il} c_{lk} = \sum_{l=r-i+1}^{r} a_{il} c_{lk},$$

which equals 0 if $i \leq r - k$.

Taking $k = 1$, and letting $i = 1, ..., r - k$, we get the $r - 1$ equations:

$$\begin{aligned} a_{1r}\, c_{r1} &= 0; \\ a_{2,r-1}\, c_{r-1,1} + a_{2r}\, c_{r1} &= 0; \\ &\vdots \\ a_{r-1,2}\, c_{21} + a_{r-1,3}\, c_{31} + ... + a_{r-1,r}\, c_{r1} &= 0. \end{aligned}$$

Assuming $a_{1r} \neq 0$, the first equation gives $c_{r1} = 0$. Substituting this into the second equation gives $c_{r-1,1} = 0$, and so on, until we get $c_{21} = 0$ from the $(r - 1)$-th equation. This leaves only c_{11} from the first column of $\boldsymbol{b}$. Since we have shown that the columns of $\boldsymbol{b}$ have norm 1, this must be equal to ± 1.

For $k = 2$, $i = 1, ..., r-2$, we get:

$$\begin{aligned} a_{1r}\, c_{r2} &= 0; \\ a_{2,r-1}\, c_{r-1,2} + a_{2r}\, c_{r2} &= 0; \\ &\vdots \\ a_{r-2,3}\, c_{3,2} + ... + a_{r-2,r}\, c_{r2} &= 0. \end{aligned}$$

Following the same process as before, we get $c_{i2} = 0$ for $i = 3, ..., r$. Also, since the second column of $\boldsymbol{C}$ must be orthogonal to the first we get

$$\sum_{l=1}^{r} c_{l1} c_{l2} = c_{11} c_{12} = 0,$$

and since $c_{11} \neq 0$, this gives that $c_{12} = 0$. This leaves only c_{22} from the second column of $\boldsymbol{b}$ so, by the same argument as before, this must be equal to ± 1.

By a similar process, we get that for $k = 3, ..., r-1$, the equations given by $\sum_{l=r-i+1}^{r} a_{il} c_{lk}$, $i = 1, ..., r-k$ give that $c_{jk} = 0$ for $j = k+1, ..., r$, and orthogonality with previous columns gives $c_{jk} = 0$ for $j = 1, ..., k-1$. Hence, the only non-zero element of the kth column of $\boldsymbol{C}$ is c_{kk}, which must be equal to ± 1.

Therefore, in order to satisfy the first two constraints $\boldsymbol{C}$ must be a diagonal matrix with all its entries equal to ± 1. This means that the entries of $\boldsymbol{AC}$ would be the same as those of $\boldsymbol{A}$, except that each column may have a sign change. However, Constraint 3 implies that the first non-zero element of each column of both $\boldsymbol{A}$ and $\boldsymbol{AC}$ must be positive, so the columns of $\boldsymbol{A}$ and $\boldsymbol{AC}$ must have the same signs. Therefore, we must have $\boldsymbol{C} = \boldsymbol{I}$.

It remains to be shown that, for any $(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)$, it is possible to find $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1)$ that satisfy the constraints such that $J\left(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1\right) = J(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)$. This can be done by finding a $\boldsymbol{C} \in \mathsf{GL}(r)$ such that $\boldsymbol{AC}$ satisfies the constraints. Then $(\boldsymbol{AC}, \boldsymbol{C}^{-1}\boldsymbol{B}_0, \boldsymbol{C}^{-1}\boldsymbol{B}_1)$ will be a valid solution set with the same objective function as $(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)$.

First, to make the columns of $\boldsymbol{A}$ orthonormal, we use the QR decomposition: we can find $\boldsymbol{Q} \in \mathbb{R}^{n \times r}$ with orthonormal columns and $\boldsymbol{R} \in \mathsf{UT}(r)$ such that $\boldsymbol{A} = \boldsymbol{QR}$.

Let $\boldsymbol{A}^* = \boldsymbol{Q}$, $\tilde{\boldsymbol{B}}_0 = \boldsymbol{R}\boldsymbol{B}_0$ and $\tilde{\boldsymbol{B}}_1 = \boldsymbol{R}\boldsymbol{B}_1$. Then $\boldsymbol{A}^*$ has orthonormal columns. We note that this will also be true for $\boldsymbol{A}^*\boldsymbol{C}$ where $\boldsymbol{C} \in \mathsf{O}(r)$.

To satisfy Constraint 2 we need

$$(\boldsymbol{A}^*\boldsymbol{C})_{ij} = \sum_{k=1}^{r} a_{ik}c_{kj} = 0 \text{ for } 2 \leq i \leq r, 1 \leq j \leq i-1.$$

Since we also require $\boldsymbol{C} \in \mathsf{O}(r)$, we get the following set of equations (of which there are r^2 in total):

$$\begin{aligned} \sum_{k=1}^{r} c_{ki}\, c_{kj} = 0 \text{ for } 2 \leq i \leq r, 1 \leq j \leq i-1; \\ \sum_{k=1}^{r} c_{ki}^2 = 1 \text{ for } 1 \leq i \leq r; \\ \sum_{k=1}^{r} a_{ik}\, c_{kj} = 0 \text{ for } 2 \leq i \leq r, 1 \leq j \leq i-1. \end{aligned} \tag{5.4}$$

where the first two sets of equations come from the columns of $\boldsymbol{C}$ being orthogonal, and each having norm 1.

For $j = 1$, we must solve

$$\begin{aligned} \sum_{k=1}^{r} a_{ik}\, c_{k1} = 0 \text{ for } 1 \leq j \leq r-1; \\ \sum_{k=1}^{r} c_{k1}^2 = 1. \end{aligned}$$

The first set of equations can be written as

$$\begin{pmatrix} a_{11} & \ldots & a_{1r} \\ \ldots & \ldots & \ldots \\ a_{r-1,1} & \ldots & a_{r-1,r} \end{pmatrix} \begin{pmatrix} c_{11} \\ \ldots \\ c_{r1} \end{pmatrix} = \begin{pmatrix} 0 \\ \ldots \\ 0 \end{pmatrix}.$$

This is a system of linear equations, but there is one fewer equation than there are coefficients. It can be rewritten as

$$\begin{pmatrix} a_{11} & \ldots & a_{1,r-1} \\ \vdots & \ldots & \vdots \\ a_{r-1,1} & \ldots & a_{r-1,r-1} \end{pmatrix} \begin{pmatrix} c_{11} \\ \vdots \\ c_{r-1,1} \end{pmatrix} = - \begin{pmatrix} a_{1r} \\ \vdots \\ a_{r-1,r} \end{pmatrix} c_{r1}.$$

If we set $c_{r1} = 1$ initially, then we can solve for the other c_{i1}'s in terms of c_{r1}. Then dividing through by $\sum_{k=1}^{r} c_{k1}^2$ will give the solution for $\boldsymbol{C}$ that satisfies Equation 5.4.

For $i = 2$ the last row of the equation matrix is replaced by $c_{11}, ..., c_{1,r-1}$ and the last element of the vector by c_{1r}, as there is one fewer of the first type of equations, but we now need to ensure that $\boldsymbol{c}_2$ is orthogonal to $\boldsymbol{c}_1$. Then for $i = 3$ the second-to-last row of the matrix is replaced with $\boldsymbol{c}_2$ (minus the last entry), and so on, until for $i = r$ the matrix is just the first $(r-1)$ rows of $\boldsymbol{C}$ minus the rth column, and the vector is equal to minus this column. Thus we can determine the elements of $\boldsymbol{C}$ using this method, such that $\boldsymbol{AC}$ satisfies the zero constraint.

To satisfy the constraint that the first non-zero element of each column of $\boldsymbol{A}$ should be positive, it may be necessary to multiply some columns of $\boldsymbol{A}^*\boldsymbol{C}$ by -1. Let $\boldsymbol{F}$ be a diagonal entries with

$$f_{ii} = \begin{cases} 1 & \text{if } a_{1i} > 0; \\ -1 & \text{otherwise.} \end{cases}$$

Let $\boldsymbol{C}^* = \boldsymbol{CF}$. Then, we get the solution $\tilde{\boldsymbol{A}} = \boldsymbol{A}^*\boldsymbol{C}^*$, with $\tilde{\boldsymbol{B}}_0 = (\boldsymbol{C}^*)^{-1}\boldsymbol{RB}_0$ and $\tilde{\boldsymbol{B}}_1 = (\boldsymbol{C}^*)^{-1}\boldsymbol{RB}_1$. Provided that $\boldsymbol{C}^*$ is invertible this will always provide a valid solution that satisfies $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}_0 + \boldsymbol{T}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}_1 = \boldsymbol{AB}_0 + \boldsymbol{TAB}_1$. □

Proof of Theorem 4

Proof. Since the matrix product $\boldsymbol{AB}_0$ is of rank r, it is equal to its rank-r SVD:

$$\boldsymbol{AB}_0 = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^T.$$

Let

$$\boldsymbol{A} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r, \qquad \boldsymbol{B}_0 = \boldsymbol{V}^T.$$

Then $\boldsymbol{A}$ and $\boldsymbol{B}_0$ satisfy Constraints 1 and 2. If the first r singular values of $\boldsymbol{AB}_0$ are all distinct, then the corresponding normalized left and right singular vectors

of $\boldsymbol{AB}_0$ will be unique, up to a change of sign [Blum et al., 2020, p. 35]. For the rank-r SVD, we only use the first r singular vectors, so we only require the r largest singular values to be distinct. Hence, $\boldsymbol{U}_r$, $\boldsymbol{\Sigma}_r$ and $\boldsymbol{V}_r$ (and thus $\boldsymbol{A}$ and $\boldsymbol{B}$) will be unique up to multiplication by a permutation matrix ($\tilde{\boldsymbol{U}}_r = \boldsymbol{U}_r\boldsymbol{P}$, $\tilde{\boldsymbol{\Sigma}}_r = \boldsymbol{P}\boldsymbol{\Sigma}_r\boldsymbol{P}$, $\tilde{\boldsymbol{V}}_r^T = \boldsymbol{P}\boldsymbol{V}_r^T$). The sign change is taken care of by Constraint 3, the requirement that the first non-zero entry of each column of $\boldsymbol{A}$ be positive.

Hence, given $\left(\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}_0, \hat{\boldsymbol{B}}_1\right)$, to find a solution that satisfies these identifiability criteria we do the following:

1. Find the SVD of $\hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_0$: $\hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_0 = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$.

2. Set $\tilde{\boldsymbol{A}} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r$ and $\tilde{\boldsymbol{B}}_0 = \boldsymbol{V}_r^T$, where $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$ contain the first r columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively, $\boldsymbol{\Sigma}_r$ is the top left $r \times r$ block of $\boldsymbol{\Sigma}$. Since $\hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_0$ has rank r, we will get $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}_0 = \hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_0$.

3. For any columns of $\tilde{\boldsymbol{A}}$ for which the first element is negative, multiply that column and the corresponding row of $\tilde{\boldsymbol{B}}_0$ by -1, to get $\boldsymbol{A}$ and $\boldsymbol{B}_0$.

4. Now, we want to find $\boldsymbol{B}_1$ such that $\boldsymbol{AB}_0 + \boldsymbol{TAB}_1 = \hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_0 + \boldsymbol{T}\hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_1$. Since we have $\boldsymbol{AB}_0 = \hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_0$, we require $\boldsymbol{TAB}_1 = \boldsymbol{T}\hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_1$, and so $\boldsymbol{AB}_1 = \hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_1$, given that $\boldsymbol{T}$ is invertible. This is solved by $\boldsymbol{B}_1 = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_1$.

□

Remark. Although in the constraints we specify that $\boldsymbol{B}_0$ has orthonormal rows, we could equally set $\tilde{\boldsymbol{A}} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r^{1/2}$ and $\tilde{\boldsymbol{B}}_0 = \boldsymbol{\Sigma}_r^{1/2}\boldsymbol{V}^T$, which would make the constraints symmetrical in $\boldsymbol{A}$ and $\boldsymbol{B}_0$, as discussed in Chapter 3.

5.3.2 Properties of the dynamic model

We consider in this section how robust the method is to certain arbitrary choices: (i) permutation of the columns of $\boldsymbol{X}$ (re-ordering the words), (ii) scaling of $\boldsymbol{T}$ (changing

the scale used to measure time), and (iii) translation of $\boldsymbol{T}$ (changing the starting point from which time is measured).

Proposition 3 (Properties of the dynamic model)**.** *Let* $(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)$ *be a set of matrices that satisfy the identifiability conditions given in Theorem 3. Then, under the following three transformations of the data:*

1. *Permutation of the columns of* $\boldsymbol{X}$*;*
2. *Scaling of* $\boldsymbol{T}$ *by a constant;*
3. *Translation of* $\boldsymbol{T}$ *by a constant,*

we can find $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1)$ *that also satisfy the constraints, such that*

$$J(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}_0, \tilde{\boldsymbol{B}}_1 | \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{T}}) = J(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1 | \boldsymbol{X}, \boldsymbol{T}),$$

where $\tilde{\boldsymbol{X}}$ *and* $\tilde{\boldsymbol{T}}$ *are the transformed* $\boldsymbol{X}$ *and* $\boldsymbol{T}$*. Thus, the minimum value of the objective function remains unchanged.*

Proof. (1) As $\boldsymbol{X}$ is $n \times p$, permuting the columns of $\boldsymbol{X}$ is equivalent to right-multiplying $\boldsymbol{X}$ by a $p \times p$ permutation matrix $\boldsymbol{P}$. Since multiplying by a permutation matrix does not change the value of the Frobenius norm $||\cdot||_F$ [Li and Mehta, 1995], we get

$$\begin{aligned} ||\boldsymbol{X} - \boldsymbol{A}\boldsymbol{B}_0 - \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1|| &= ||\left(\boldsymbol{X} - \boldsymbol{A}\boldsymbol{B}_0 - \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1\right)\boldsymbol{P}|| \\ &= ||\boldsymbol{X}\boldsymbol{P} - \boldsymbol{A}\boldsymbol{B}_0\boldsymbol{P} - \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1\boldsymbol{P}||, \end{aligned}$$

so

$$J(\boldsymbol{A}, \boldsymbol{B}_0\boldsymbol{P}, \boldsymbol{B}_1\boldsymbol{P} | \boldsymbol{X}\boldsymbol{P}, \boldsymbol{T}) = J(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1 | \boldsymbol{X}, \boldsymbol{T}).$$

(2) If $\tilde{\boldsymbol{T}} = \gamma\boldsymbol{T}$ for some constant γ, then

$$\boldsymbol{A}\boldsymbol{B}_0 + \tilde{\boldsymbol{T}}\boldsymbol{A}\boldsymbol{B}_1 = \boldsymbol{A}\boldsymbol{B}_0 + (\gamma\boldsymbol{T})\,\boldsymbol{A}\boldsymbol{B}_1 = \boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{T}\boldsymbol{A}\,(\gamma\boldsymbol{B}_1)\,,$$

so by replacing $\boldsymbol{B}_1$ with $\tilde{\boldsymbol{B}}_1 = \frac{1}{\lambda}\boldsymbol{B}_1$, we can obtain the same value for the objective function.

(3) If $\tilde{\boldsymbol{T}} = \boldsymbol{T} + \lambda \boldsymbol{I}$, then

$$\begin{aligned} \boldsymbol{A}\boldsymbol{B}_0 + \tilde{\boldsymbol{T}}\boldsymbol{A}\boldsymbol{B}_1 &= \boldsymbol{A}\boldsymbol{B}_0 + (\boldsymbol{T} + \lambda \boldsymbol{I})\, \boldsymbol{A}\boldsymbol{B}_1 \\ &= \boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1 + \lambda \boldsymbol{A}\boldsymbol{B}_1 \\ &= \boldsymbol{A}\left(\boldsymbol{B}_0 + \lambda \boldsymbol{B}_1\right) + \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1, \end{aligned}$$

so we can replace $\boldsymbol{B}_0$ with $\boldsymbol{B}_0 + \lambda \boldsymbol{B}_1$ and retain the same value for the objective function.

In each case we can find new values of $\boldsymbol{A}$, $\boldsymbol{B}_0$ and $\boldsymbol{B}_1$ that give the same value of the objective function as before. In addition, using the first set of identifiability conditions, $\boldsymbol{A}$ is unchanged in each of these cases. Using the second set of constraints, where there are conditions on both $\boldsymbol{A}$ and $\boldsymbol{B}_0$, applying the constraints in the translation case will change the values in $\boldsymbol{A}$. □

5.4 Extensions

The linear model used so far allows general trends across time to be captured, but cannot pick up more complex trends. For example, if there were a pair of words that become increasingly related in the first half of the time period, but move further apart in the second half, it is quite likely that the linear model would not pick this up. Adding more terms to the model would allow for the observation of more complicated relationships:

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1 + ... + \boldsymbol{T}^q\boldsymbol{A}\boldsymbol{B}_q + \boldsymbol{Z},$$

or, more generally,

$$\boldsymbol{X} = \sum_{k=0}^{q} \boldsymbol{T}_k \boldsymbol{A}\boldsymbol{B}_q + \boldsymbol{Z}, \tag{5.5}$$

where each $\boldsymbol{T}_k$ is a diagonal matrix with $(\boldsymbol{T}_k)_{ii} = f_k(t_i)$, where the f_k's are a set of basis functions.

We can use the same identifiability conditions as in either Theorem 3 or Theorem 4: they apply only to $\boldsymbol{A}$ and $\boldsymbol{B}_0$ but are sufficient to fix identifiability for the

extended model. This is because we can rewrite Equation 5.5 as

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{T}_0\boldsymbol{A} & \boldsymbol{T}_1\boldsymbol{A} & ... & \boldsymbol{T}_q\boldsymbol{A} \end{pmatrix} \begin{pmatrix} \boldsymbol{B}_0 \\ \boldsymbol{B}_1 \\ \vdots \\ \boldsymbol{B}_q \end{pmatrix} + \boldsymbol{Z},$$

so, if $\boldsymbol{A}$ and the $\boldsymbol{T}_k$'s are fixed, there is a unique solution (which can be found using least squares) for $\tilde{\boldsymbol{B}} = \left(\boldsymbol{B}_0^T, \boldsymbol{B}_1^T, ..., \boldsymbol{B}_q^T\right)^T$, and hence for each $\boldsymbol{B}_k$.

We must be careful, however, not to over-parameterize the model. The number of parameters is $nr+(q+1)pr$, where $q+1$ is the number of basis functions (minus r^2 parameters for identifiability). This must be lower than the number of data points, np, for the model to be well-defined, and in order for the model to be useful we would want $nr + (q+1)pr << np$. So the number of basis functions we can reasonably use depends on r: the higher r is, the simpler the functions need to be. On the other hand, using a smaller embedding dimension would allow us to use a greater number of basis functions.

One option is to consider fixing most of the words, and allowing a small subset of words of interest to change across time. This requires making the assumption that the vast majority of the words will not change over the time period. We acknowledge that this is a fairly strong assumption, which may or may not be justified depending on our knowledge of the dataset, and the length of time it covers: it would seem reasonable to assume that most words will not change in the short-term at least. Introducing this assumption reduces the number of parameters in the model, as most of the entries of $\boldsymbol{B}_k$ will be fixed at zero for $k \geq 1$. If we denote the set of words which are allowed to move by S, then this model can be written as

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}_0 + \sum_{k=1}^{q} \boldsymbol{T}^k \boldsymbol{A}\boldsymbol{B}_k^S + \boldsymbol{Z},$$

where $\boldsymbol{B}_k^S$ has $\boldsymbol{b}_j^k = \boldsymbol{0}$ for $j \notin S$, where $\boldsymbol{b}_j^k$ is the jth column of $\boldsymbol{B}_k$. This has the advantage of allowing more complicated functions to be used, as the number of extra

parameters required for each basis function we add is proportional to the size of the set of words allowed to change over time, so if this is small we will be able to use a greater number of basis functions without increasing the number of parameters too much. However, there is still a risk of overfitting with respect to the words in S, and we also need to consider whether the assumption about most words being fixed is valid for the dataset we are using.

5.4.1 Factor model

We also consider a variant of the dynamic LSA model where instead of having $z_{ij} \overset{iid}{\sim} N(0, \sigma^2)$, we have $z_{ij} \overset{ind}{\sim} N(0, \sigma_j^2)$. As is usual in factor analysis, we use $\boldsymbol{\mu}$ to represent the mean of the $\boldsymbol{x}_i$'s, so that $E(\boldsymbol{a}_i) = \mathbf{0}$. Hence, the model is

$$\boldsymbol{x}_i = \boldsymbol{\mu} + (\boldsymbol{B}_0 + t_i \boldsymbol{B}_1)^T \boldsymbol{a}_i + \boldsymbol{z}_i, \tag{5.6}$$

where

$$\boldsymbol{z}_i \overset{iid}{\sim} N(\mathbf{0}, \boldsymbol{\Sigma}_z),$$

with $\boldsymbol{\Sigma}_z$ diagonal. Note that we can rewrite Equation 5.6 as

$$\boldsymbol{x}_i - \boldsymbol{\mu} = (\boldsymbol{B}_0 + t_i \boldsymbol{B}_1)^T \boldsymbol{a}_i + \boldsymbol{z}_i. \tag{5.7}$$

Hence, we shall generally assume that $\boldsymbol{X}$ has been centred so that $\frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i = 0$, which is equivalent to estimating $\boldsymbol{\mu}$ (by $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i$) and subtracting it from $\boldsymbol{x}_i$.

When $\boldsymbol{T} = \mathbf{0}$, the static model we regain is the factor analysis model from Section 2.6. Using this model will allow us to use some existing asymptotic results from factor analysis literature in the testing section. It is also a more realistic assumption, as the variances for more frequent words will likely be greater than those for less frequent words. The main disadvantage is that is is harder to optimize the likelihood.

As stated in Section 2.6, the $\boldsymbol{a}_i$'s can be regarded as either fixed or Normally distributed, with these cases being asymptotically equivalent. Hence, unless otherwise stated, we shall assume that $\boldsymbol{a}_i \overset{iid}{\sim} N(\mathbf{0}, \boldsymbol{\Sigma}_a)$.

5.5 Developing testing framework

In this section we discuss testing for whether there is significant change across time in a dataset. This is something that has not been done rigorously (see Section 2.5.2), but is necessary to determine whether any findings are statistically significant. In particular, we need to control for false positive results, which is not done in, for example, [Hamilton et al., 2016].

We first define formally the test we would like to carry out, and list some desirable properties that we would like the test to have. (In particular, we want it to be invariant to the choice of identifiability conditions.) We then explain why we cannot use a standard likelihood ratio test. Using the factor analysis variant of the dynamic model allows us to use a test for adequacy of the null model. Using a simulation study, we investigate how sensitive the test is to deviations from the null hypothesis, and discuss the difficulty of distinguishing misspecified r from the presence of time dependence.

We then show theoretically that the test is unable to distinguish between data that comes from the dynamic model with embedding dimension r, and data from the static model with embedding dimension $2r$. We show that the test statistic is impervious to transformations of the embedding set by a non-singular matrix, and hence that any test statistic that is invariant to this transformation will have the same issue.

5.5.1 Set-up

Suppose we want to test whether there is change over time for words in a text corpus. First, we need to define mathematically what we mean by this. Firstly, we need a model which we can use to develop a test. We want this to be a simple model so that the test is not too complicated to develop. It also must be a parametric model.

Hence, we use the dynamic LSA model defined earlier in this chapter:

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1 + \boldsymbol{Z}. \tag{5.8}$$

A word changes over time if the column of $\boldsymbol{B}_1$ associated with that word is non-zero. If there is no change in the full set of words, then $\boldsymbol{B}_1 = \boldsymbol{0}$. So the null hypothesis (that there is no change) is

$$H_0 : \boldsymbol{B}_1 = \boldsymbol{0},$$

whilst our alternative hypothesis is

$$H_1 : \boldsymbol{B}_1 \neq \boldsymbol{0}.$$

We may want to investigate only whether words in a particular subset of interest have changed across time. In this case, we are only interested in whether the columns of $\boldsymbol{B}_1$ that correspond to words in this subset are equal to 0. Let $\mathcal{S} = \{s_1, ..., s_q\}$ denote the set of indices of the words of interest (where $q < p$). The columns of $\boldsymbol{B}_1$ corresponding to words not in the subset $\mathcal{S}$ can either all be assumed to be 0 (Equation 5.9) – in which case we assume that these words are all stationary – or can be left unspecified (Equation 5.10; we allow them to be moving over time, but are not interested in whether they are moving).

If we denote the jth column of $\boldsymbol{B}_1$ by $\boldsymbol{B}_{1j}$, then in the first case the two hypotheses are

$$\begin{aligned} &H_0 : \boldsymbol{B}_1 = \boldsymbol{0}, \\ &H_1 : \boldsymbol{B}_{1j} \neq \boldsymbol{0} \text{ for all } j \in \mathcal{S}; \boldsymbol{B}_{1j} = \boldsymbol{0} \text{ for all } j \notin \mathcal{S}. \end{aligned} \tag{5.9}$$

In the second case, we have

$$\begin{aligned} H_0 :&\boldsymbol{B}_{1j} = \boldsymbol{0} \text{ for all } j \in \mathcal{S}, \\ H_1 :&\boldsymbol{B}_{1j} \neq \boldsymbol{0} \text{ for all } j \in \mathcal{S}. \end{aligned} \tag{5.10}$$

We would like such a test to have the following properties:

- Invariance to choice of identifiability conditions: the choice of identifiability constraints used to ensure uniqueness of $\boldsymbol{B}_1$ should not affect the result of the test. (Naturally, any test statistic will be (implicitly or explicitly) dependent on having a consistent estimate of $\boldsymbol{B}_1$.)

- Power against H_1: the test should reject H_0 when the data are from the dynamic model in Equation 5.8, with $\boldsymbol{B}_1 \neq \boldsymbol{0}$.

- Ability to distinguish between the presence of time dependence in the dataset and a misspecified static model under H_0.

We proceed looking at the case where only a subset of the words are of interest, with hypotheses defined in Equation 5.10. (The case where all words are moving is a special case of this, where $\mathcal{S} = \{1, ...p\}$.)

The likelihood ratio statistic is

$$-2\log\Lambda = -2\log\left(\frac{\max L\left(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1^{0^{\mathcal{S}}}\right)}{\max L(\boldsymbol{A}, \boldsymbol{B}_0, \boldsymbol{B}_1)}\right),$$

where $\boldsymbol{B}_1^{0^{\mathcal{S}}}$ has the restriction that the columns corresponding to the elements in $\mathcal{S}$ are equal to $\boldsymbol{0}$.

By Wilks' theorem [Wilks, 1938], under certain conditions this quantity asymptotically follows a χ^2_{qr} distribution, where $q = |\mathcal{S}|$. However, one of the necessary conditions for this theorem to apply is that the number of parameters in the model be fixed as the number of data points, n, increases. In our model, the number of parameters is $nr + pr + qr$ (or $nr + 2pr$ if we allow all words to move), which depends on n, so this condition is not satisfied. In addition, the maximum likelihood estimates of the parameters of interest $\boldsymbol{B}_0$ and $\boldsymbol{B}_1$ may not be consistent (converge to their true values) as $n \to \infty$ [Neyman and Scott, 1948].

If we knew $\boldsymbol{A}$, then the distribution of the likelihood ratio statistic would be exactly χ^2_{qr} (Theorem 5). But in practice, since we do not know $\boldsymbol{A}$, we cannot guarantee that $-2\log\Lambda$ converges to the correct distribution. Figure 5.2 shows that

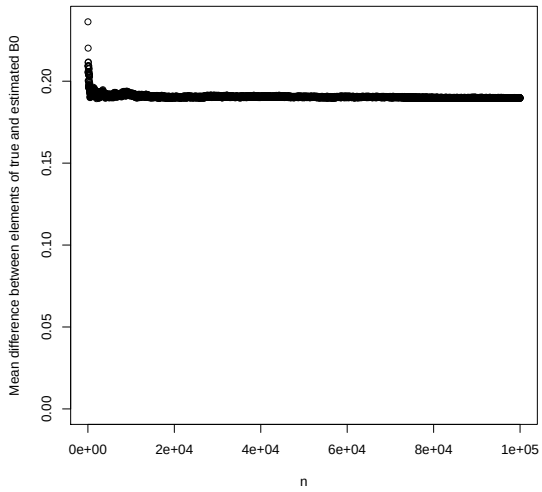

Figure 5.2: Plot of $\frac{1}{n}\sum_{i,j}|\boldsymbol{b}_{ij}^0 - \hat{\boldsymbol{b}}_{ij}^0|$ against n under H_0 ($p = 10, r = 2$). Identifiability conditions are applied so that there is a unique "true" $\boldsymbol{B}_0$. It appears that $\hat{\boldsymbol{B}}_0$ is not converging to the true value of $\boldsymbol{B}_0$.

under H_0, the maximum likelihood estimate of $\boldsymbol{B}_0$ does not converge to the true value of $\boldsymbol{B}_0$ as $n \to \infty$.

Theorem 5 (Distribution of likelihood ratio statistic when $\boldsymbol{A}$ is known)**.** *Assuming $\boldsymbol{A}$ is known, the distribution of the likelihood ratio statistic is exactly χ^2_{qr} distribution if σ^2 is known, or $F_{qr,nr-qr}$ if σ^2 is unknown.*

The proof of this theorem is given in the Appendix.

To summarise: we would like to test for time dependence in the corpus, but we cannot use Wilks' theorem because of the increasing number of parameters as n increases. The scenario we want to investigate is

$$H_0 : \boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{Z} \text{ vs. } H_1 : \boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1 + \boldsymbol{Z}. \tag{5.11}$$

Using the factor analysis version of the dynamic model, with $\boldsymbol{\Sigma}_z$ diagonal, we can use a factor analysis test for the adequacy of the model under H_0:

$$H_0 : \boldsymbol{\Sigma}_x = \boldsymbol{B}_0^T\boldsymbol{\Sigma}_a\boldsymbol{B}_0 + \boldsymbol{\Sigma}_z \text{ vs. } H_1 : \boldsymbol{\Sigma}_x \text{ positive definite}, \tag{5.12}$$

where

$$\boldsymbol{\Sigma}_x = E\left(\frac{1}{n-1}\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right)\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right)^T\right),$$

with $\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i$. The null hypotheses of the two tests in Equations 5.11 and 5.12 are the same, but the alternative hypothesis in the test in Equation 5.12 is more general than the alternative hypothesis in the first test. Hence, we might expect that if H_0 is rejected in the first test it will also be rejected in the second, but that this might not be the case the other way around.

The likelihood of the model is

$$L\left(\boldsymbol{\Sigma}_x, \boldsymbol{\mu}\right) = (2\pi)^{-np/2}\left|\boldsymbol{\Sigma}_x\right|^{-n/2}\exp\left(\frac{1}{2}\left(\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \boldsymbol{\mu}\right)\boldsymbol{\Sigma}_x^{-1}\left(\boldsymbol{x}_i - \boldsymbol{\mu}\right)\right)\right).$$

The maximum likelihood estimate of $\boldsymbol{\mu}$ (under H_0 or H_1) is simply $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i$. The maximum likelihood estimate of $\boldsymbol{\Sigma}_x$ under H_0 is $\hat{\boldsymbol{\Sigma}}_x^0 = \hat{\boldsymbol{B}}_0^T\hat{\boldsymbol{\Sigma}}_a\hat{\boldsymbol{B}}_0 + \hat{\boldsymbol{\Sigma}}_z$. The maximum likelihood estimate under H_1 for $\boldsymbol{\Sigma}_x$ is $\hat{\boldsymbol{\Sigma}}_x^1 = \boldsymbol{M}_x = \frac{1}{n-1}\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right)\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right)^T$.

[Anderson and Rubin, 1956] states that, under the assumption that the $\boldsymbol{a}_i \sim N(\boldsymbol{0}, \boldsymbol{\Sigma}_a)$, the likelihood ratio statistic $-2\log\Lambda$ converges asymptotically to the χ^2_ν distribution, where ν is equal to the difference in the number of parameters under H_0 and under H_1:

$$\nu = \frac{1}{2}p\left(p+1\right) - \left(pr + p + \frac{1}{2}r\left(r+1\right) - r^2\right) = \frac{1}{2}p\left(p+1\right) + \frac{1}{2}r\left(r-1\right) - p\left(r+1\right).$$

The likelihood ratio statistic [Anderson and Rubin, 1956] is

$$-2\log\Lambda = -2\log\left(\frac{|\boldsymbol{M}_x|^{n/2}e^{np/2}}{|\hat{\boldsymbol{\Sigma}}_x|^{n/2}e^{n/2\,\mathrm{tr}\left(\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}\right)}}\right). \tag{5.13}$$

[Amemiya and Anderson, 1990] gives an alternative formulation:

$$\mathcal{T} = n\,\mathrm{tr}\,\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1} - n\log|\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}| - np. \tag{5.14}$$

These are equivalent:

$$\begin{aligned}
-2\log\Lambda &= -2\left(\frac{n}{2}\log|\boldsymbol{M}_x| + \frac{np}{2} - \frac{n}{2}\log|\hat{\boldsymbol{\Sigma}}_x| - \frac{n}{2}\operatorname{tr}\left(\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}\right)\right) \\
&= n\left(\log|\hat{\boldsymbol{\Sigma}}_x| + \operatorname{tr}\left(\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}\right) - \log|\boldsymbol{M}_x| - p\right) \\
&= n\left(\operatorname{tr}\left(\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}\right) - \log\left(\frac{|\boldsymbol{M}_x|}{\hat{\boldsymbol{\Sigma}}_x}\right) - p\right) \\
&= n\left(\operatorname{tr}\left(\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}\right) - \log|\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}| - p\right) = \mathcal{T}.
\end{aligned}$$

Hence, we can use either formulation.

Fixed $\boldsymbol{a}_i$

Suppose now that instead of following a Normal distribution, the $\boldsymbol{a}_i$'s are regarded as fixed parameters. We use $\boldsymbol{M}_a = \frac{1}{n-1}\left(\boldsymbol{a}_i - \bar{\boldsymbol{a}}\right)\left(\boldsymbol{a}_i - \bar{\boldsymbol{a}}\right)^T$, rather than $\boldsymbol{\Sigma}_a$, as the parameter of interest. This is assumed to tend to some limit as $n \to \infty$.

The asymptotic distribution of $\mathcal{T}$ given in the previous section is derived under the assumption that $\boldsymbol{a}_i \overset{iid}{\sim} N(\boldsymbol{0}, \boldsymbol{\Sigma}_a)$, but [Anderson and Rubin, 1956] claims that it also holds (with $\boldsymbol{\Sigma}_a$ replaced with $\boldsymbol{M}_a$) under the assumption that the $\boldsymbol{a}_i$'s are fixed.

5.5.2 The test statistic under the dynamic model

In this section we ascertain, via simulations, how large n needs to be relative to p for the distribution of $\mathcal{T}$ to be approximately the same as its asymptotic distribution.

All simulations were carried out in `R`. The elements of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}_0$ were simulated independently from $N(0,1)$ unless otherwise stated. Details of the simulation of $\boldsymbol{B}_1$ are given in the relevant sections below. The test was implemented using `R`'s `factanal` function, which estimates the model parameters and calculates an estimate of $\mathcal{T}$. The number of runs for each simulation was 1000. When time dependence is included, the n times were equally spaced between -1 and 1.

We simulated under H_0:

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{Z}, \tag{5.15}$$

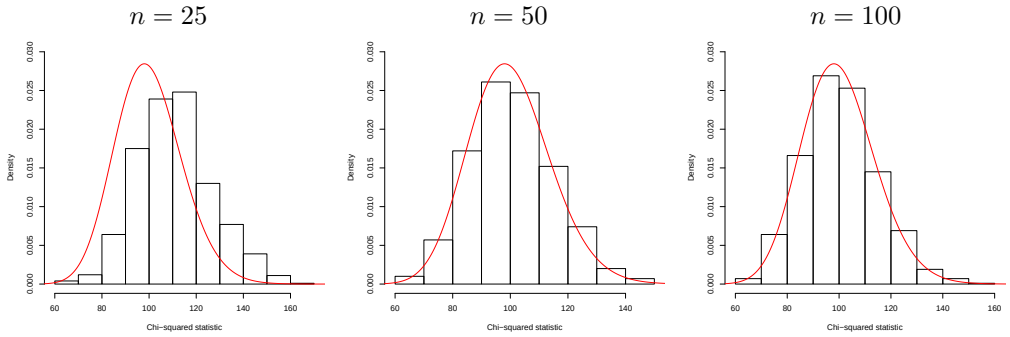

Figure 5.3: Histograms of $\mathcal{T}$ (Equation 5.14) under H_0, with $r = 5$ and $p = 20$, and n taking values 25, 50, 100.

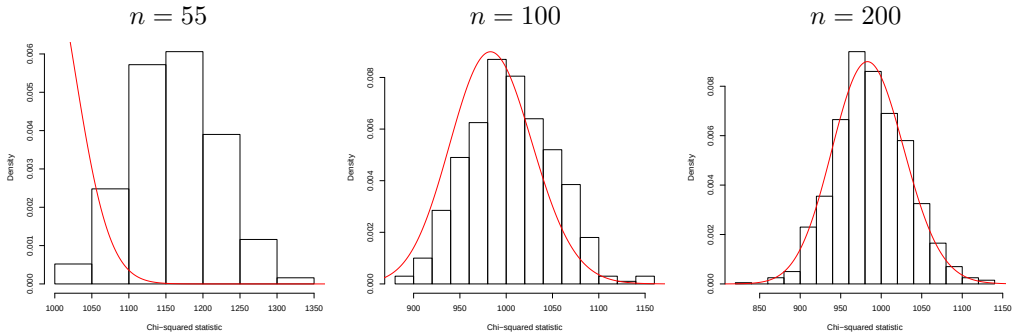

Figure 5.4: Histograms of $\mathcal{T}$ (Equation 5.14) under H_0, with $r = 5$ and $p = 50$, and n taking values 55, 100, 200.

where $\boldsymbol{z}_i \sim N(\mathbf{0}, \boldsymbol{\Sigma}_z)$ ($\boldsymbol{\mu}$ is taken to be 0), and plotted histograms of $\mathcal{T}$ (Equation 5.14) to ascertain that it follows the expected distribution. In particular, since we only know the asymptotic distribution of $\mathcal{T}$, it is necessary to find out how large n has to be, relative to p and r, for the distribution of $\mathcal{T}$ to be close enough to its asymptotic distribution for the test to be useful.

Hence, Figures 5.3 and 5.4 show data simulated under H_0 (Equation 5.15) for different values of n and p. The red line on each graph shows the theoretical distribution which $\mathcal{T}$ should follow under H_0. In both cases we can see that the distribution of $\mathcal{T}$ does not match the theoretical distribution when n is close to p, but the distributions match when n is larger (at least several times p).

We now investigate the distribution of $\mathcal{T}$ when we simulate under the dynamic

model

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1. \tag{5.16}$$

We investigate changing the magnitude of the elements of $\boldsymbol{B}_1$, simulating $\boldsymbol{B}_1$ from different distributions, and conducting simulations where only some of the columns of $\boldsymbol{B}_1$ are non-zero (to represent the situation where a subset of the words are moving across time).

Figures 5.5 and 5.6 show histograms of $\mathcal{T}$ in Equation 5.14 under the dynamic model (Equation 5.16). These two sets of histograms are simulated under the same scenario, both with $n = 1000$, but with different values of p and r; in Figure 5.5, $p = 20$ and $r = 4$, and in Figure 5.6, $p = 50$ and $r = 5$. In both cases, the elements of $\boldsymbol{B}_1$ are simulated independently from $N(0, \phi^2)$, where ϕ^2 takes different values between 0.0001 and 1. This is in order to investigate the effect of the magnitude of $\boldsymbol{B}_1$ on $\mathcal{T}$; when the variances of the entries of $\boldsymbol{B}_1$ are larger, their magnitude will in general also be larger (since we are simulating from a distribution which has mean 0), and so we would expect $\boldsymbol{B}_1$ to have a greater effect on $\mathcal{T}$. The vector of times is taken to be a set of equally spaced values between -1 and 1.

This is shown to be the case in Figures 5.5 and 5.6; $\mathcal{T}$ takes larger values when ϕ^2 is larger, but even for $\phi^2 = 0.01$, where the entries of $\boldsymbol{B}_1$ will be very small compared to those in $\boldsymbol{A}$ and $\boldsymbol{B}_0$, we can see that the distribution of $\mathcal{T}$ is different from its distribution under H_0, and that H_0 will usually be rejected. Hence, the test is sensitive to the presence of time dependence in the simulated data.

In Figure 5.7, we simulate the elements of $\boldsymbol{B}_1$ from three different distributions: $N(0, 0.01)$, $U(0, 0.1)$, and $Exp(10)$. In each case the estimated value of $\mathcal{T}$ takes much larger values than those it would take under H_0.

In Figure 5.8 we move only a subset of the words (so most of the columns of $\boldsymbol{B}_1$ equal to $\mathbf{0}$). Hence,

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{T}\boldsymbol{A}\tilde{\boldsymbol{B}}_1,$$

where $\tilde{\boldsymbol{B}}_1$ has all of its columns equal to 0 except for those corresponding to a small

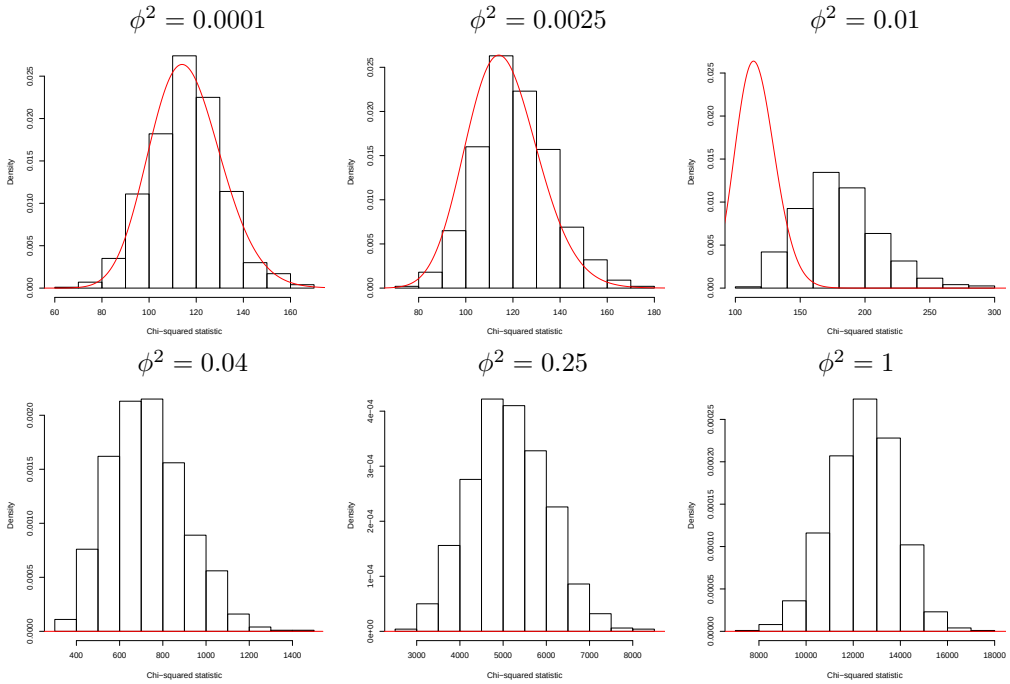

Figure 5.5: Histograms of $\mathcal{T}$ (Equation 5.14) under H_1, with $n = 1000$, $p = 20$, and $r = 4$. In each case the elements of $\boldsymbol{B}_1$ simulated as i.i.d. random variables with distribution $N(0, \phi^2)$, where $\phi^2 = \{0.0001, 0.0025, 0.01, 0.04, 0.25, 1\}$. We see that even small values of $\boldsymbol{B}_1$ can have a significant effect on the distribution and rejection rate.

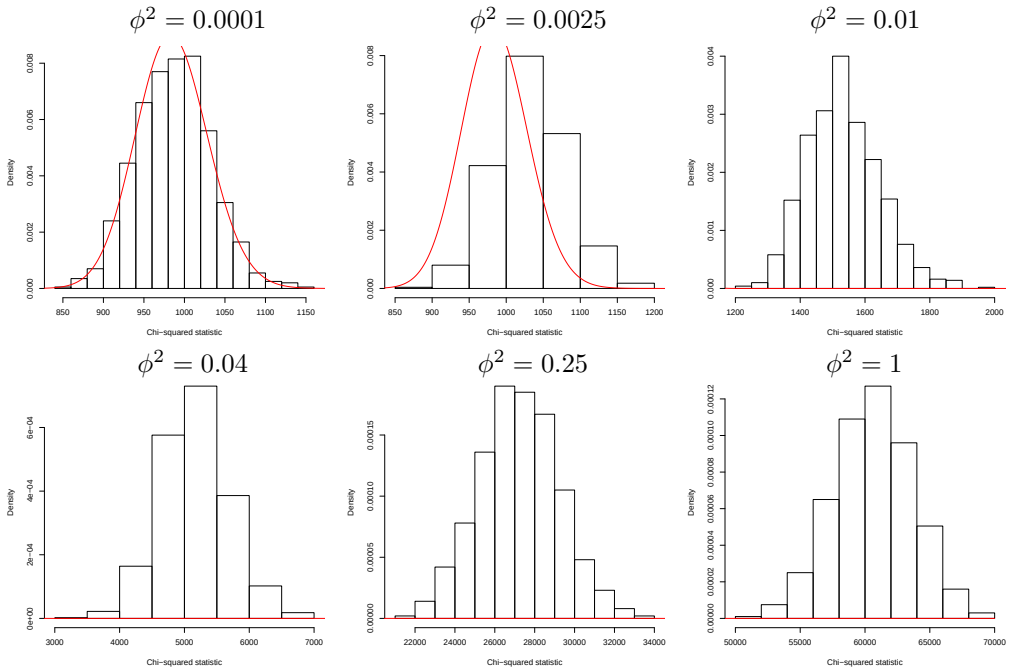

Figure 5.6: Histograms of $\mathcal{T}$ (Equation 5.14) under H_1, with $n = 1000$, $p = 50$, and $r = 5$. In each case the elements of $\boldsymbol{B}_1$ are i.i.d. random variables with distribution $N(0, \phi^2)$, where $\phi^2 = \{0.0001, 0.0025, 0.01, 0.04, 0.25, 1\}$. $\boldsymbol{B}_0$ is simulated so that its entries have distribution $N(0, 1)$. We see that even small values of $\boldsymbol{B}_1$ can have a significant effect on the distribution and rejection rate.

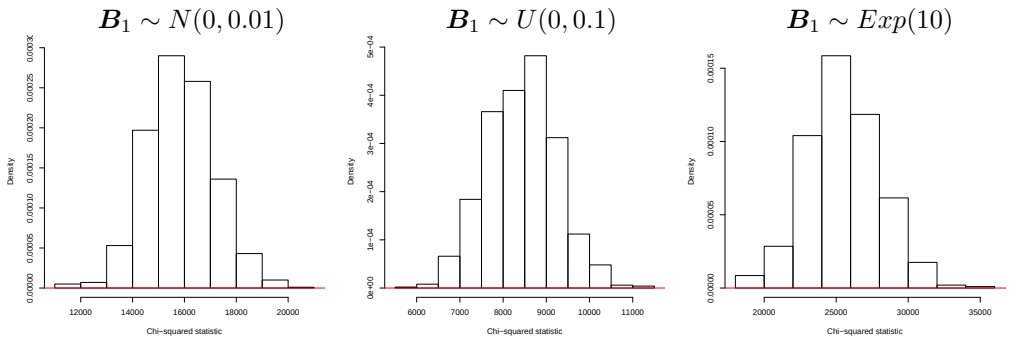

Figure 5.7: Histograms of $\mathcal{T}$ (Equation 5.14) under $H_1 : \boldsymbol{X} = \boldsymbol{AB}_0 + \boldsymbol{TAB}_1$, where $\boldsymbol{B}_1$ is simulated from (i) $N(0, 0.01)$; (ii) $U(0, 0.1)$; (iii) Exp(10).

number of words, q. The non-zero elements of $\boldsymbol{B}_1$ are simulated from $N(0, \phi^2)$ where ϕ^2 takes values 0.0001, 0.01, and 0.25, so we can also investigate the effect that the size of the non-zero elements of $\boldsymbol{B}_1$ has on $\mathcal{T}$. We use different values of p, r, and q for the two sets of histograms (see figures and captions).

Figure 5.8 shows histograms of $\mathcal{T}$ where a subset q of the words are moving, for different values of q. For $q = 2$ and $q = 5$, the distribution of $\mathcal{T}$ does not deviate much from its distribution under H_0. For $q = 7$, this is still true when $\phi^2 = 0.0001$ or $\phi^2 = 0.01$, but when $\phi^2 = 0.25$ the distributions are different, although the curve and the histogram overlap.

Thus, we have found that the impact of adding $\boldsymbol{B}_1$ to the model depends upon the size of the entries of $\boldsymbol{B}_1$ and the number of words which have non-zero dynamic components in $\boldsymbol{B}_1$. When the entries of $\boldsymbol{B}_1$ are very small compared to those of $\boldsymbol{A}$ and $\boldsymbol{B}_0$, or only a small number of words are moving, then the distribution of $\mathcal{T}$ does not deviate much from its distribution under H_0 and hence the test may not detect the presence of time dependence. However, as long as there are enough words moving, or the entries of $\boldsymbol{B}_1$ are sufficiently large, the test is quite sensitive to the presence of time dependence.

5.5.3 Testing on COHA

We implement the test on subsets of the Corpus of Historical American English (COHA). The whole corpus contains 116597 documents and 49564 words (after removing infrequent words), which means that the document-term matrix is too large for implementing the test on all of the data to be feasible. Instead, we take random subsets of $p = 100$ and $p = 500$ words; all documents are included, as the size of n is not an issue computationally. The document-term matrix has been normalized so that all its rows sum to 1, to remove the effect of document length.

In this case, we do not know the "true" value of r. Hence, we carry out the test for several different values of r. The rejection rate given is the proportion of

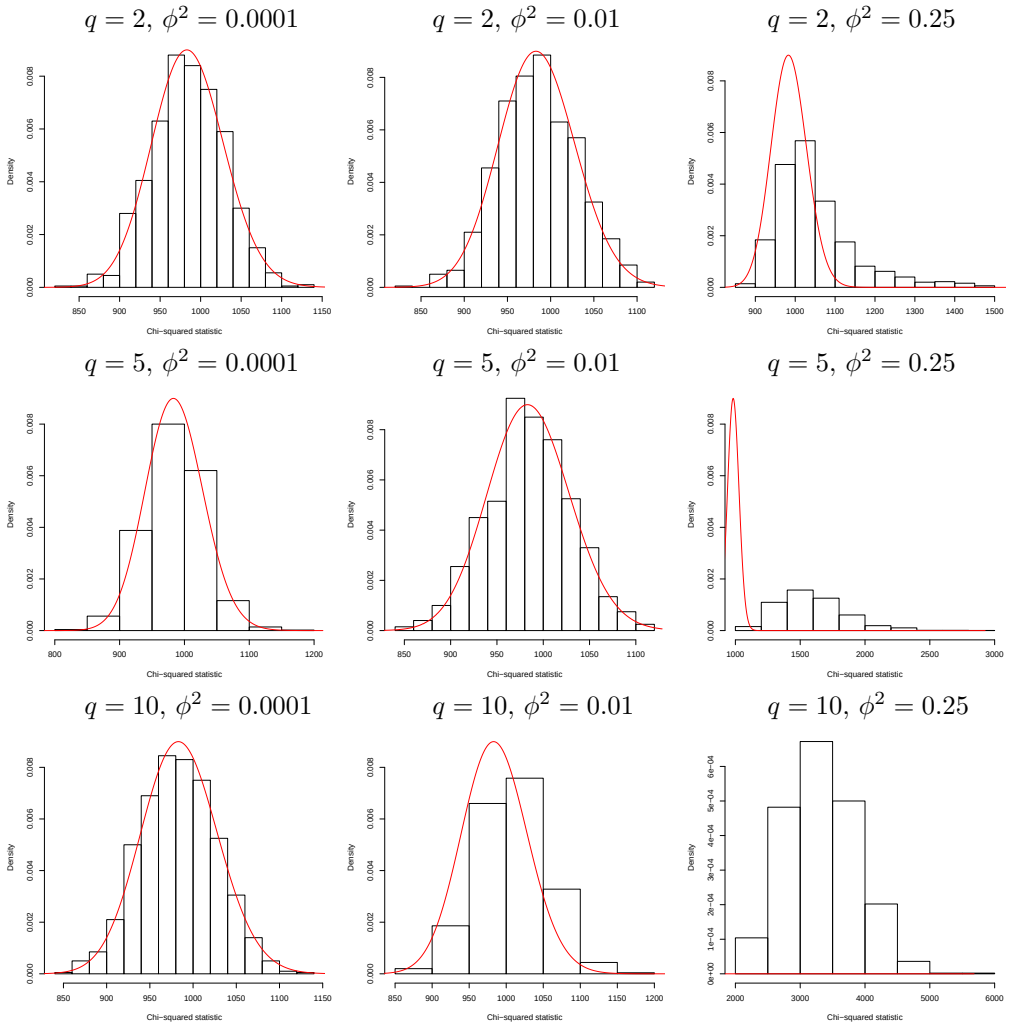

Figure 5.8: Histograms of $\mathcal{T}$ (Equation 5.14) under H_1 where only a subset of the words are moving ($n = 1000$, $r = 5$). q is the number of words moving, out of $p = 50$ total words. For words that are moving, $b_{ij}^1 \overset{iid}{\sim} N(0, \phi^2)$, where $\phi^2 = 0.0001, 0.01, 0.25$.

p	r	Rejection rate
100	5	0.112
100	10	0.001
100	25	0.002
500	5	1
500	10	1

times H_0 is rejected out of 1000 implementations. For $p = 100$, the rejection rate decreases as we increase r. For $p = 500$, H_0 is always rejected, which suggests either the presence of time dependence, or that the embedding dimension is too small. However, since we do not know the "correct" value of r, we cannot tell which of these reasons is the true one.

5.5.4 Eigenvalues

We now seek to understand more about $\mathcal{T}$ by understanding how it changes under deviations from H_0. We first make explicit the link between $\mathcal{T}$ and the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$.

Let $\lambda_1, ..., \lambda_p$ denote the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ (such that $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p$). We can rewrite Equation 5.14 as:

$$\begin{aligned}\mathcal{T} &= n \operatorname{tr}\left(\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}\right) - n\log|\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}| - np \\ &= n\sum_{i=1}^{p}\lambda_i - n\log\left(\prod_{i=1}^{p}\lambda_i\right) - np \\ &= n\sum_{i=1}^{p}\left(\lambda_i - \log\lambda_i - 1\right).\end{aligned} \tag{5.17}$$

If the model fits perfectly ($\boldsymbol{M}_x = \hat{\boldsymbol{\Sigma}}_x$) then $\mathcal{T} = 0$ ($\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1} = \boldsymbol{I}_p$, so $\lambda_i = 1$ for $i = 1, ..., p$). Let

$$f(\lambda) = \lambda - \log\lambda - 1. \tag{5.18}$$

Figure 5.9 shows a graph of $f(\lambda)$ for λ between 0 and 10. f takes its minimum at $\lambda = 1$, with $f(1) = 0$, and increases as λ moves away from 1. When λ is close to 0,

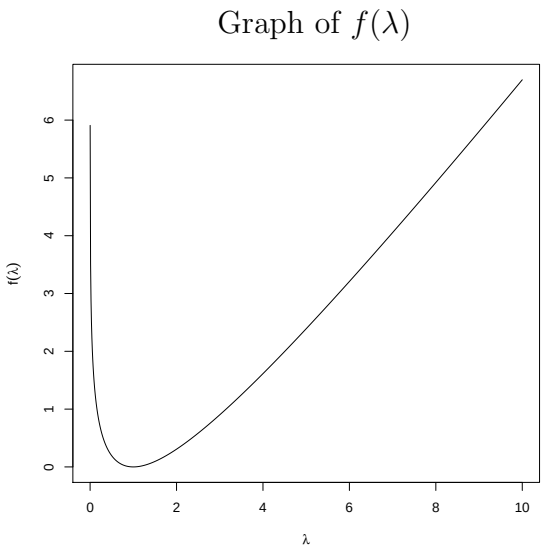

Figure 5.9: Graph of $f(\lambda)$ (see Equation 5.18).

$f(\lambda) \approx -\log\lambda$, and when $\lambda > 1$, $f(\lambda) \approx \lambda - 1$, so f is approximately linear in λ. Hence, $\mathcal{T}$ is larger when the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ are further away from 1.

Figures 5.10 and 5.11 show plots of the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ for 1000 simulations from H_0 and H_1. The elements of $\boldsymbol{B}_1$ are simulated from $N(0, \phi^2)$, where ϕ^2 takes values $\{0.0001, 0.0025, 0.01, 0.04, 0.25\}$. In Figure 5.10 $n = 1000$, $p = 20$ and $r = 4$; in Figure 5.11 $n = 1000$, $p = 50$ and $r = 5$. In both cases, the smallest and largest eigenvalues deviate most from 1 when the elements of $\boldsymbol{B}_1$ are larger (that is, when ϕ^2 is larger). Hence, we would expect $\mathcal{T}$ to be larger for these $\boldsymbol{B}_1$'s, as was indeed shown to be the case by earlier simulations (Figures 5.5 and 5.6).

Table 5.2 shows some results for simulations under the dynamic model ($\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}_0 + \boldsymbol{T}\boldsymbol{A}\boldsymbol{B}_1 + \boldsymbol{Z}$), where $\boldsymbol{B}_1$ is simulated from different distributions. The values of $\mathcal{T}$ are stated, along with the largest and smallest eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$, in each case. We can see that the cases for which $\mathcal{T}$ is higher are those for which the eigenvalues deviate farthest from 1 – the largest eigenvalues are larger and the smaller ones are smaller. When $\mathcal{T}$ is smaller, the eigenvalues are all close to 1.

Figure 5.12 shows plots of the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ where the data is simulated from the dynamic model with only a subset of words moving. As the number of

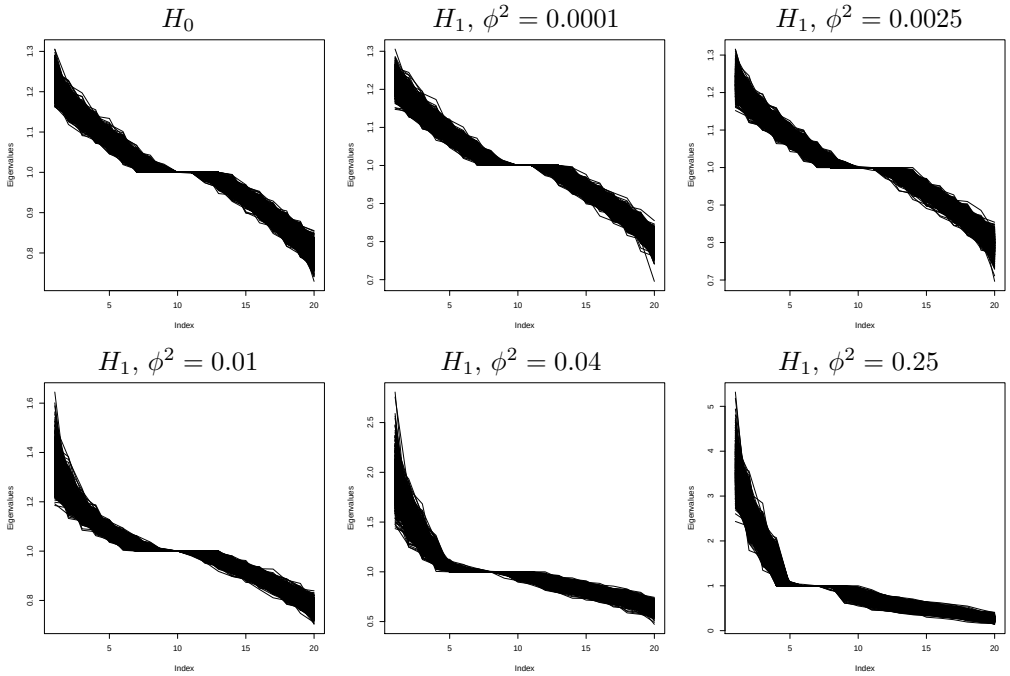

Figure 5.10: Eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ for 1000 simulations, under H_0 and under H_1 where $b_{ij}^1 \overset{iid}{\sim} N(0, \phi^2)$. $(n = 1000, p = 20, r = 4.)$

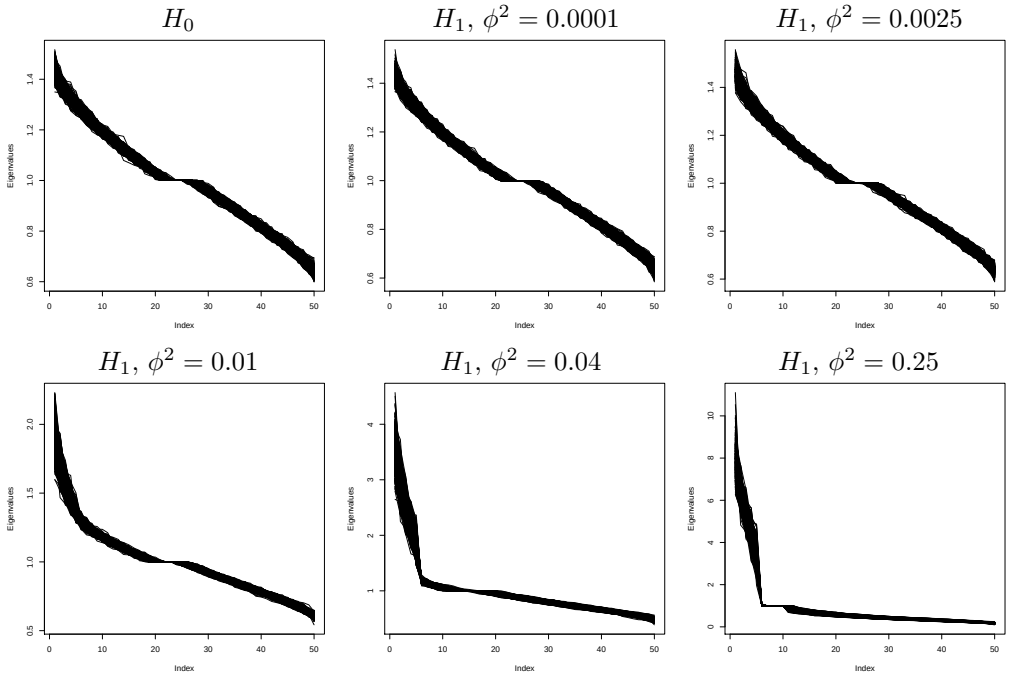

Figure 5.11: Eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ for 1000 simulations, under H_0 and under H_1, where $b_{ij}^1 \overset{iid}{\sim} N(0, \phi^2)$. $(n = 1000, p = 50, r = 5.)$

Run	$\mathcal{T}$	Largest eigenvalue	Smallest eigenvalue
1	138 000	3.279	0.007
2	133 000	3.540	0.017
3	85 000	4.989	0.055
4	71 000	4.395	0.074
5	69 000	4.806	0.090
6	50 000	4.363	0.125
7	42 000	3.367	0.127
8	12 000	1.811	0.160
9	600	1.319	0.873
10	500	1.250	0.854

Table 5.2: Results of simulating from the dynamic model. We simulated $\boldsymbol{B}_1$ from ten different distributions. For each simulation, the table gives the value of $\mathcal{T}$ and the largest and smallest eigenvalues of $\boldsymbol{M}\hat{\boldsymbol{\Sigma}}_x^{-1}$. We observe that $\mathcal{T}$ is larger when the largest and smallest eigenvalues are further from 1.

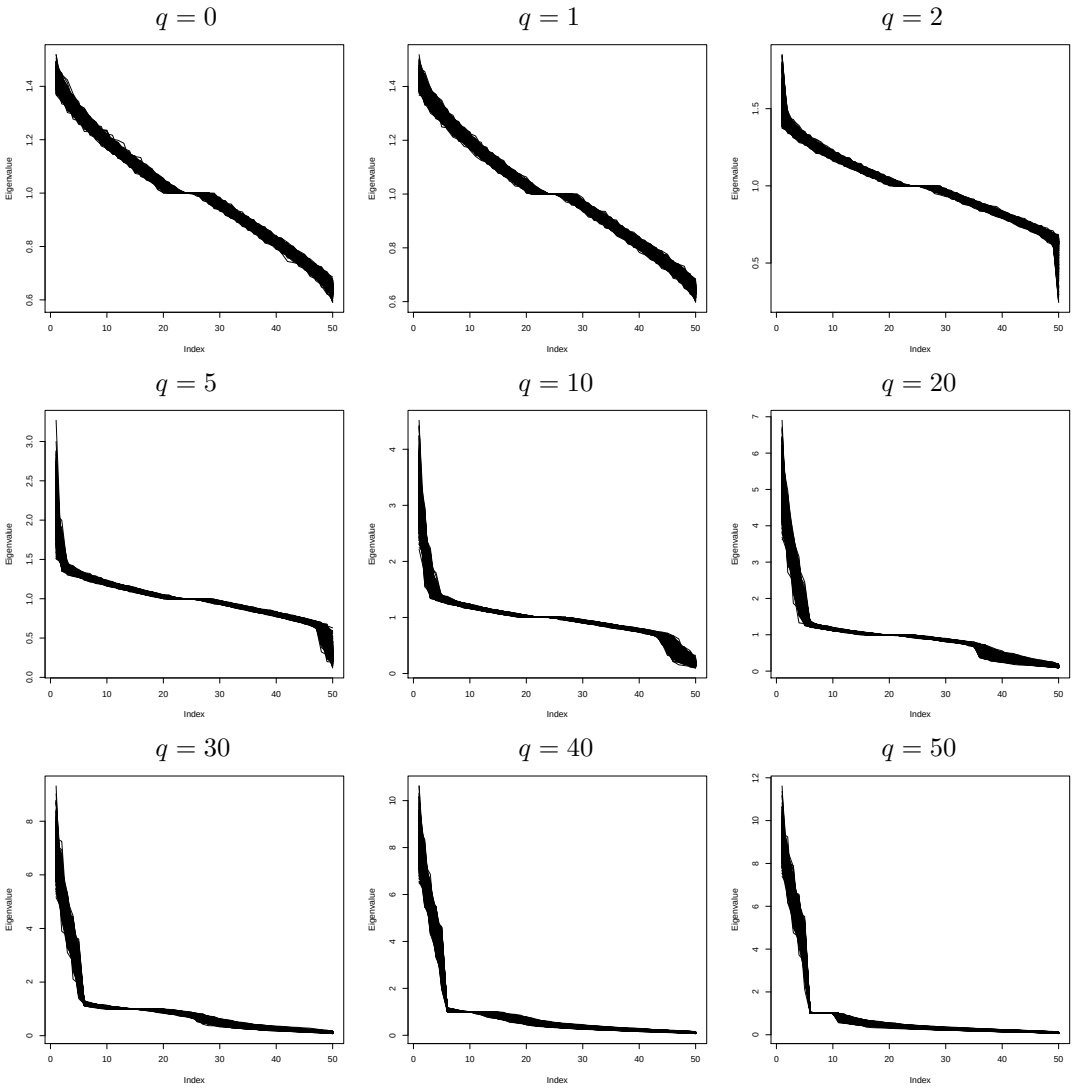

Figure 5.12: Eigenvalues of $\boldsymbol{M}_x \hat{\boldsymbol{\Sigma}}_x^{-1}$ for 1000 simulations, simulating under the dynamic model with q words moving (where q is between 0 and p). ($p = 50, r = 5, n = 1000$.)

words moving increases, the eigenvalues become more spread out.

We now investigate further what happens to $\boldsymbol{M}_x$ and $\boldsymbol{\Sigma}_x$ under H_1. Recall the dynamic version of the factor model (Section 5.4.1),

$$\boldsymbol{x}_i = \left(\boldsymbol{B}_0 + t_i \boldsymbol{B}_1\right)^T \boldsymbol{a}_i + \boldsymbol{z}_i,$$

where $\boldsymbol{z}_i \overset{iid}{\sim} N(\boldsymbol{0}, \boldsymbol{\Sigma}_z)$, with $\boldsymbol{\Sigma}_z$ diagonal.

As in Section 2.6, let $\boldsymbol{M}_x$ be defined as

$$\boldsymbol{M}_x = \frac{1}{n-1} \sum_{i=1}^{n} \left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right) \left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right)^T.$$

In the static model, where $\boldsymbol{x}_i = \boldsymbol{B}_0^T \boldsymbol{a}_i + \boldsymbol{z}_i$, this is equal to

$$\boldsymbol{M}_x^0 = \boldsymbol{B}_0^T \boldsymbol{M}_a \boldsymbol{B}_0 + \boldsymbol{B}_0^T \boldsymbol{M}_{az} + \boldsymbol{M}_{az}^T \boldsymbol{B}_0 + \boldsymbol{M}_z,$$

where

$$\boldsymbol{M}_a = \frac{1}{n-1} \sum_{i=1}^{n} \left(\boldsymbol{a}_i - \bar{\boldsymbol{a}}\right) \left(\boldsymbol{a}_i - \bar{\boldsymbol{a}}\right)^T,$$

$$\boldsymbol{M}_{az} = \frac{1}{n-1} \sum_{i=1}^{n} \left(\boldsymbol{a}_i - \bar{\boldsymbol{a}}\right) \left(\boldsymbol{z}_i - \bar{\boldsymbol{z}}\right)^T,$$

and

$$\boldsymbol{M}_z = \frac{1}{n-1} \sum_{i=1}^{n} \left(\boldsymbol{z}_i - \bar{\boldsymbol{z}}\right) \left(\boldsymbol{z}_i - \bar{\boldsymbol{z}}\right)^T.$$

We also get

$$\boldsymbol{\Sigma}_x^0 = E\left(\boldsymbol{M}_x^0\right) = \boldsymbol{B}_0^T \boldsymbol{\Sigma}_a \boldsymbol{B}_0 + \boldsymbol{\Sigma}_z, \tag{5.19}$$

as $E\left(\boldsymbol{M}_{az}\right) = \boldsymbol{0}$.

In the dynamic model, we get

$$\begin{aligned} \boldsymbol{M}_x^1 = {} & \boldsymbol{B}_0^T \boldsymbol{M}_a \boldsymbol{B}_0 + \boldsymbol{B}_0^T \boldsymbol{M}_a^t \boldsymbol{B}_1 + \boldsymbol{B}_1^T \boldsymbol{M}_a^{tT} \boldsymbol{B}_0 + \boldsymbol{B}_1^T \boldsymbol{M}_a^{tt} \boldsymbol{B}_1 + \boldsymbol{B}_0^T \boldsymbol{M}_{az} \\ & + \boldsymbol{M}_{az}^T \boldsymbol{B}_0 + \boldsymbol{B}_1^T \boldsymbol{M}_{az}^t + \boldsymbol{M}_{az}^{t\ T} \boldsymbol{B}_1 + \boldsymbol{M}_z, \end{aligned}$$

where

$$\boldsymbol{M}_a^t = \frac{1}{n-1} \sum_{i=1}^{n} \left(\boldsymbol{a}_i - \bar{\boldsymbol{a}}\right) \left(t_i \boldsymbol{a}_i - \frac{1}{n} \sum_{j=1}^{n} \left(t_j \boldsymbol{a}_i\right)\right)^T,$$

$$\boldsymbol{M}_a^{tt} = \frac{1}{n-1} \sum_{i=1}^{n} \left(t_i \boldsymbol{a}_i - \frac{1}{n} \sum_{j=1}^{n} \left(t_j \boldsymbol{a}_j\right)\right) \left(t_i \boldsymbol{a}_i - \frac{1}{n} \sum_{j=1}^{n} \left(t_j \boldsymbol{a}_i\right)\right)^T,$$

and

$$\boldsymbol{M}_{az}^{t} = \frac{1}{n-1}\sum_{i=1}^{n}\left(t_i\boldsymbol{a}_i - \frac{1}{n}\sum_{j=1}^{n}\left(t_j\boldsymbol{a}_j\right)\right)\left(\boldsymbol{z}_i - \bar{\boldsymbol{z}}\right)^T.$$

Proposition 4. *If* $\boldsymbol{a}_i \overset{iid}{\sim} N(\boldsymbol{0}, \boldsymbol{\Sigma}_a)$ $(i = 1, ..., n)$*, then*

$$\boldsymbol{\Sigma}_x^1 = \boldsymbol{B}_0^T\boldsymbol{\Sigma}_a\boldsymbol{B}_0 + \left(\frac{1}{n}\sum_{i=1}^{n}t_i\right)\left(\boldsymbol{B}_0^T\boldsymbol{\Sigma}_a\boldsymbol{B}_1 + \boldsymbol{B}_1^T\boldsymbol{\Sigma}_a\boldsymbol{B}_0\right) + \left(\frac{1}{n}\sum_{i=1}^{n}t_i^2\right)\boldsymbol{B}_1^T\boldsymbol{\Sigma}_a\boldsymbol{B}_1 + \boldsymbol{\Sigma}_z. \tag{5.20}$$

Proof. We get

$$\begin{aligned}\boldsymbol{\Sigma}_x^1 = E\left(\boldsymbol{M}_x^1\right) = {} & \boldsymbol{B}_0^T E\left(\boldsymbol{M}_a\right)\boldsymbol{B}_0 + \boldsymbol{B}_0^T E\left(\boldsymbol{M}_a^t\right)\boldsymbol{B}_1 + \boldsymbol{B}_1^T E\left(\boldsymbol{M}_a^t\right)^T\boldsymbol{B}_0 \\ & + \boldsymbol{B}_1^T E\left(\boldsymbol{M}_a^{tt}\right)\boldsymbol{B}_1 + \boldsymbol{B}_0^T E\left(\boldsymbol{M}_{az}\right) + E\left(\boldsymbol{M}_{az}\right)^T\boldsymbol{B}_0 + \boldsymbol{B}_1^T E\left(\boldsymbol{M}_{az}^t\right) \\ & + E\left(\boldsymbol{M}_{az}^t\right)^T\boldsymbol{B}_1 + E\left(\boldsymbol{M}_z\right).\end{aligned}$$

Now,

$$\begin{aligned}E\left(\boldsymbol{M}_a\right) &= \boldsymbol{\Sigma}_a, \\ E\left(\boldsymbol{M}_z\right) &= \boldsymbol{\Sigma}_z, \\ E\left(\boldsymbol{M}_{az}\right) &= \frac{1}{n-1}\sum_{i=1}^{n}E\left(\left(\boldsymbol{a}_i - \bar{\boldsymbol{a}}\right)\left(\boldsymbol{z}_i - \bar{\boldsymbol{z}}\right)^T\right) = 0,\end{aligned}$$

and

$$E\left(\boldsymbol{M}_{az}^t\right) = \frac{1}{n-1}\sum_{i=1}^{n}E\left(\left(t_i\boldsymbol{a}_i - \frac{1}{n}\sum_{j=1}^{n}t_j\boldsymbol{a}_j\right)\left(\boldsymbol{z}_i - \bar{\boldsymbol{z}}\right)^T\right) = 0.$$

For $E\left(\boldsymbol{M}_a^t\right)$, we get

$$\begin{aligned}E\left(\boldsymbol{M}_a^t\right) &= \frac{1}{n-1}\sum_{i=1}^{n}E\left(\left(\boldsymbol{a}_i - \bar{\boldsymbol{a}}\right)\left(t_i\boldsymbol{a}_i - \frac{1}{n}\sum_{j=1}^{n}t_j\boldsymbol{a}_j\right)^T\right) \\ &= \frac{1}{n-1}\left(\sum_{i=1}^{n}t_iE\left(\boldsymbol{a}_i\boldsymbol{a}_i^T\right) - \frac{2}{n}\sum_{i,j=1}^{n}t_iE\left(\boldsymbol{a}_j\boldsymbol{a}_i^T\right) + \frac{1}{n^2}\sum_{i,j,k=1}^{n}t_jE\left(\boldsymbol{a}_k\boldsymbol{a}_j^T\right)\right) \\ &= \frac{1}{n-1}\left(\left(\sum_{i=1}^{n}t_i\right)\boldsymbol{\Sigma}_a - \frac{2}{n}\left(\sum_{i=1}^{n}t_i\right)\boldsymbol{\Sigma}_a + \frac{1}{n}\left(\sum_{i=1}^{n}t_i\right)\boldsymbol{\Sigma}_a\right) \\ &= \frac{1}{n-1}\left(1 - \frac{1}{n}\right)\left(\sum_{i=1}^{n}t_i\right)\boldsymbol{\Sigma}_a \\ &= \frac{1}{n}\left(\sum_{i=1}^{n}t_i\right)\boldsymbol{\Sigma}_a.\end{aligned}$$

and for $E\left(\boldsymbol{M}_a^{tt}\right)$, we get

$$\begin{aligned}
E\left(\boldsymbol{M}_a^{tt}\right) &= \frac{1}{n-1}\sum_{i=1}^{n} E\left(\left(t_i\boldsymbol{a}_i - \frac{1}{n}\sum_{j=1}^{n} t_j\boldsymbol{a}_j\right)\left(t_i\boldsymbol{a}_i - \frac{1}{n}\sum_{k=1}^{n} t_k\boldsymbol{a}_k\right)^T\right) \\
&= \frac{1}{n-1}\left(\sum_{i=1}^{n} t_i^2 E\left(\boldsymbol{a}_i\boldsymbol{a}_i^T\right) - \frac{2}{n}\sum_{i,j=1}^{n} t_i t_j E\left(\boldsymbol{a}_j\boldsymbol{a}_i^T\right) + \frac{1}{n^2}\sum_{i,j,k=1}^{n} t_j t_k E\left(\boldsymbol{a}_j\boldsymbol{a}_k^T\right)\right) \\
&= \frac{1}{n-1}\left(\left(\sum_{i=1}^{n} t_i^2\right)\boldsymbol{\Sigma}_a - \frac{2}{n}\left(\sum_{i=1}^{n} t_i^2\right)\boldsymbol{\Sigma}_a + \frac{1}{n}\left(\sum_{i=1}^{n} t_i^2\right)\boldsymbol{\Sigma}_a\right) \\
&= \frac{1}{n-1}\left(1-\frac{1}{n}\right)\left(\sum_{i=1}^{n} t_i^2\right)\boldsymbol{\Sigma}_a \\
&= \frac{1}{n}\left(\sum_{i=1}^{n} t_i^2\right)\boldsymbol{\Sigma}_a.
\end{aligned}$$

Thus,

$$\begin{aligned}
\boldsymbol{\Sigma}_x^1 &= \boldsymbol{B}_0^T\boldsymbol{\Sigma}_a\boldsymbol{B}_0 + \boldsymbol{B}_0^T\left(\frac{1}{n}\sum_{i=1}^{n} t_i\boldsymbol{\Sigma}_a\right)\boldsymbol{B}_1 + \boldsymbol{B}_1^T\left(\frac{1}{n}\sum_{i=1}^{n} t_i\boldsymbol{\Sigma}_a\right)\boldsymbol{B}_0 \\
&\quad + \boldsymbol{B}_1^T\left(\frac{1}{n}\sum_{i=1}^{n} t_i^2\boldsymbol{\Sigma}_a\right)\boldsymbol{B}_1 + \boldsymbol{B}_0^T\boldsymbol{0} + \boldsymbol{0}^T\boldsymbol{B}_0 + \boldsymbol{B}_1^T\boldsymbol{0} + \boldsymbol{0}^T\boldsymbol{B}_1 + \boldsymbol{\Sigma}_z \\
&= \boldsymbol{B}_0^T\boldsymbol{\Sigma}_a\boldsymbol{B}_0 + \frac{1}{n}\sum_{i=1}^{n} t_i\left(\boldsymbol{B}_0^T\boldsymbol{\Sigma}_a\boldsymbol{B}_1 + \boldsymbol{B}_1^T\boldsymbol{\Sigma}_a\boldsymbol{B}_0\right) + \frac{1}{n}\sum_{i=1}^{n} t_i^2\boldsymbol{B}_1^T\boldsymbol{\Sigma}_a\boldsymbol{B}_1 + \boldsymbol{\Sigma}_z.
\end{aligned}$$

□

Corollary 1. *If* $\frac{1}{n}\sum_{i=1}^{n} t_i = 0$ *and* $\frac{1}{n}\sum_{i=1}^{n} t_i^2 = 1$, *then*

$$\boldsymbol{\Sigma}_x^1 = \boldsymbol{B}_0^T\boldsymbol{\Sigma}_a\boldsymbol{B}_0 + \boldsymbol{B}_1^T\boldsymbol{\Sigma}_a\boldsymbol{B}_1 + \boldsymbol{\Sigma}_z. \tag{5.21}$$

Proof. This can be clearly seen by substituting $\frac{1}{n}\sum_{i=1}^{n} t_i = 0$ and $\frac{1}{n}\sum_{i=1}^{n} t_i^2 = 1$ into Equation 5.20. □

This is useful because we can scale and translate the time vector without affecting the model (Proposition 3); hence, we can replace the times t_i with

$$\tilde{t}_i = \frac{t_i - \frac{1}{n}\sum_{i=1}^{n} t_i}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(t_i - \frac{1}{n}t_j^2\right)^2}},$$

and so make use of Equation 5.21. If we take as an identifiability condition that $\boldsymbol{\Sigma}_a = \boldsymbol{I}$, Equation 5.21 reduces to

$$\boldsymbol{\Sigma}_x = \boldsymbol{B}_0^T\boldsymbol{B}_0 + \boldsymbol{B}_1^T\boldsymbol{B}_1 + \boldsymbol{\Sigma}_z. \tag{5.22}$$

Remark. The times t_i $(i = 1, ..., n)$ do not appear in this equation.

Proposition 5. *If* $\boldsymbol{X}$ *is simulated under the static model with embedding dimension* $2r$*: that is,*

$$\boldsymbol{x}_i = \boldsymbol{B}^T \boldsymbol{a}_i + \boldsymbol{z}_i, \tag{5.23}$$

where $\boldsymbol{B}$ *is a* $2r \times p$ *matrix, and* $\boldsymbol{a}_i$ *a* $2r \times 1$ *vector; and we write*

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{B}_0 \\ \boldsymbol{B}_1 \end{pmatrix}, \tag{5.24}$$

then, under the application of suitable identifiability conditions, we can write

$$\boldsymbol{\Sigma}_x = \boldsymbol{B}_0^T \boldsymbol{B}_0 + \boldsymbol{B}_1^T \boldsymbol{B}_1 + \boldsymbol{\Sigma}_z.$$

Proof. Combining Equations 5.23 and 5.24 gives

$$\boldsymbol{x}_i = \begin{pmatrix} \boldsymbol{B}_0^T & \boldsymbol{B}_1^T \end{pmatrix} \boldsymbol{a}_i + \boldsymbol{z}_i = \boldsymbol{B}_0^T \boldsymbol{a}_i^0 + \boldsymbol{B}_1^T \boldsymbol{a}_i^1 + \boldsymbol{z}_i.$$

If we take $\boldsymbol{\Sigma}_a = \boldsymbol{I}_{2r}$, then $\boldsymbol{a}_i^0, \boldsymbol{a}_i^1 \overset{iid}{\sim} N(\boldsymbol{0}, \boldsymbol{I}_r)$. Hence, $\boldsymbol{x}_i \overset{iid}{\sim} N(\boldsymbol{0}, \boldsymbol{\Sigma}_x)$, where

$$\boldsymbol{\Sigma}_x = \boldsymbol{B}_0^T \boldsymbol{I}_r \boldsymbol{B}_0 + \boldsymbol{B}_0^T \boldsymbol{0}_r \boldsymbol{B}_1 + \boldsymbol{B}_1^T \boldsymbol{0}_r \boldsymbol{B}_0 + \boldsymbol{B}_1^T \boldsymbol{I}_r \boldsymbol{B}_1 + \boldsymbol{\Sigma}_z = \boldsymbol{B}_0^T \boldsymbol{B}_0 + \boldsymbol{B}_1^T \boldsymbol{B}_1 + \boldsymbol{\Sigma}_z.$$

□

Hence, a test based on $\boldsymbol{M}_x$ and $\boldsymbol{\Sigma}_x$ will not be able to distinguish between the dynamic model with embedding dimension r and the static model with embedding dimension $2r$. This is an issue when, in practice, we do not know the correct embedding dimension.

5.5.5 Choice of embedding dimension

We investigate further the problem of misspecifying the embedding dimension, via simulations.

Figure 5.13 shows simulations of $\mathcal{T}$ (Equation 5.14), where the data is simulated under H_0 (Equation 5.15) with $r = 4$, and the test is implemented with four different

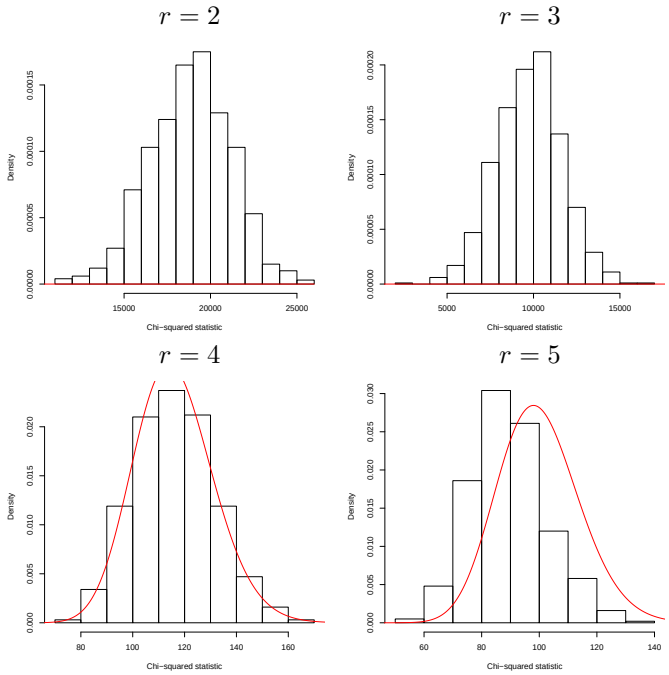

Figure 5.13: Histograms of $\mathcal{T}$ when different values of r are used than the true value. The data is simulated under H_0 with $n = 1000$, $p = 20$ and $r = 4$, but the model is fitted using four different values of r $(2, 3, 4, 5)$. In the first two cases $\mathcal{T}$ is much larger than it should be, so that H_0 is rejected in every case. (The 5 % critical values of the χ^2 distribution for these cases are 180.7 and 160.9 respectively, which are clearly much smaller than the values in the histogram.) In the third case, with $r = 4$, the χ^2 curve matches the shape of the histogram, as we would expect. In the fourth case, where the fitted r is too large, $\mathcal{T}$ is smaller than it should be, and H_0 is rejected in only 6 out of 1000 cases.

values of r. When the test is carried out with $r < 4$, $\mathcal{T}$ takes values much larger than would be expected under H_0 with the correct embedding dimension. When $r = 4$, as we would expect, $\mathcal{T}$ follows the expected theoretical distribution. When $r = 5$, the values of $\mathcal{T}$ are smaller than would usually be expected, so the histogram sits a little to the left of the curve representing the theoretical distribution. This would result in H_0 being rejected less often than the size of the test.

Figure 5.14 shows histograms of $\mathcal{T}$ when data is simulated under both H_0 (Equation 5.15) and H_1 (Equation 5.16), and the parameters of the model are estimated using several different values of r. The first three histograms are simulated under

H_0 with $r = 2$, and the model is using three different values of r. As before, the histogram of $\mathcal{T}$ sits to the left of the curve showing the theoretical distribution when the value of r used to fit the model is larger than the value under which the data is simulated.

The bottom row of histograms are simulated under H_1, again with $r = 2$, and in each case the parameters are estimated under the null model with three different values of r. As expected, when $r = 2$, $\mathcal{T}$ is larger than it should be according to the theoretical distribution. However, when $r = 5$ and $r = 8$, despite the data having been simulated under H_1, $\mathcal{T}$ is smaller than it should be according to the theoretical distribution and hence H_0 will rarely be rejected in these cases. Thus, the test seems unable to detect the presence of time dependence in the data if the embedding dimension used to fit the model is too large.

Figure 5.15 shows the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ for three different situations: in the top row the data is simulated and the model is fitted under H_0 (the static case); in the middle row the data is simulated from the static model with embedding dimension $2r$, but the model is fitted with embedding dimension r; and in the third row the data is simulated from the dynamic model, with embedding dimension r, and the model is fitted under the static model, with the same embedding dimension. We see that for the last two cases, the eigenvalue plots look the same, so a test statistic based on the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ will be unable to distinguish between these two scenarios.

5.5.6 Invariance of the test

In order to implement the test, we need to be able to estimate $\hat{\boldsymbol{\Sigma}}_x$ consistently, which requires consistent estimation of $\hat{\boldsymbol{\Sigma}}_a$, $\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{\Sigma}}_z$. For this to be possible, we must apply identifiability conditions to $\boldsymbol{\Sigma}_a$ and $\boldsymbol{B}$ to ensure that they can be uniquely identified, so that their estimates will converge to a unique limit. As previously discussed (Section 5.2), this choice is fairly arbitrary, so we want the test

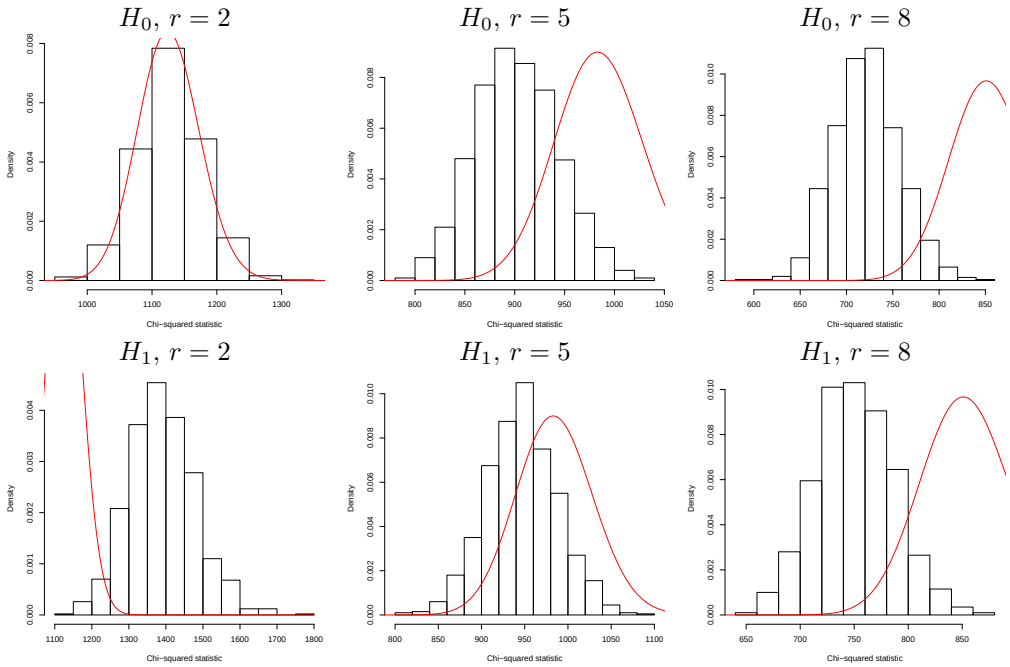

Figure 5.14: Histograms of $\mathcal{T}$ where the value of r used is too large. In the first row, the data is simulated under H_0 with $r = 2$ ($n = 1000$, and $p = 50$). The model is fitted using $r = 2, 5, 8$, and the test implemented. When the value of r used is too large, the histogram sits to the left of where it should be, so H_0 is rejected less often than it should be. In the second row, the data is simulated under H_1, where $\boldsymbol{B}_1 \sim N(0, 0.01)$, again with $r = 2$. As expected, the curve sits to the left of the histogram when the model is fitted with $r = 2$, meaning that H_0 is rejected more often than it should be under H_0. However, when the model is fitted with higher values of r, the curve is to the right of the histogram, so H_0 is rarely rejected, meaning that the presence of time dependence in the data is not detected.

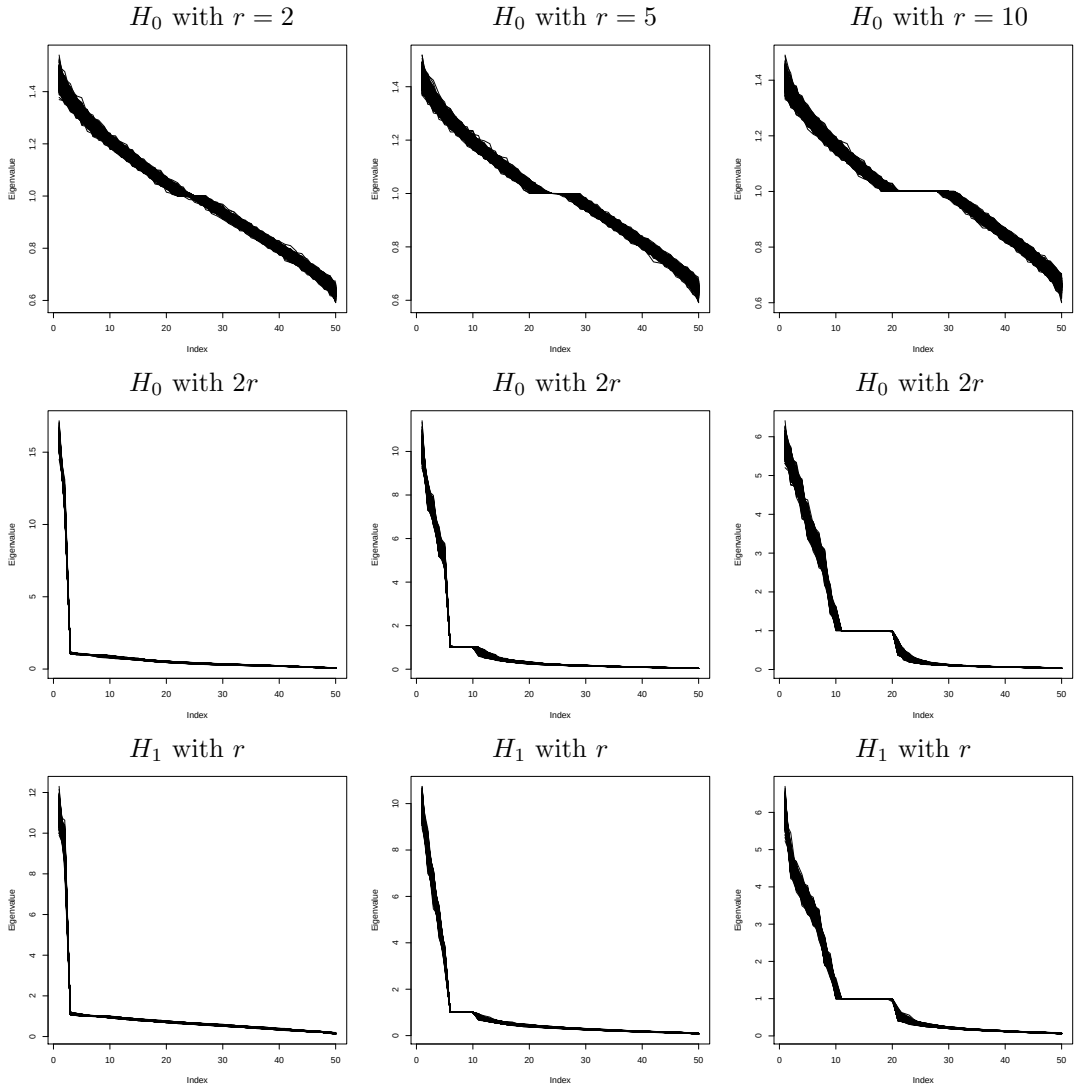

Figure 5.15: The top row of plots are of the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$, where the data is simulated under the null model, and the model fitted under the same model, for three different values of r ($r = 2, 5, 10$). The middle row of plots show the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ when the data is simulated under the null model with embedding dimension $2r$, but the model is fitted with embedding dimension r. In the bottom row the data is simulated under the dynamic model with embedding dimension r, but the model is fitted under the null model (also with embedding dimension r). There does not appear to be a significant difference between the last two cases, so the test is unable to distinguish between the presence of time dependence in the dataset and a misspecified choice of r.

statistic $\mathcal{T}$ to be invariant to the choice of identifiability conditions we impose. The test statistic we have been using so far is a function of the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ (Equation 5.17), so for $\mathcal{T}$ to be invariant to the choice of identifiability conditions, it is sufficient to show that the set of eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ is invariant to the identifiability conditions.

Theorem 6 states that the set of eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ (where $\hat{\boldsymbol{\Sigma}}_x$ is estimated under H_0) is impervious to transformations of the form $(\boldsymbol{A}, \boldsymbol{B}_0) \to (\boldsymbol{AC}, \boldsymbol{C}^{-1}\boldsymbol{B}_0)$. Applying such a transformation to $\boldsymbol{A}$ and $\boldsymbol{B}_0$ is equivalent to applying a different set of identifiability conditions.

Theorem 6. *Given that either:*
(i) $pr - \frac{1}{2}r(r-1) \leq \frac{1}{2}p(p-1)$*, or*
(ii) $\boldsymbol{\Sigma}_z$ *is assumed to be of the form* $\boldsymbol{\Sigma}_z = \sigma^2\boldsymbol{I}$*,*
the set of eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ *is impervious to the choice of identifiability conditions (or, equivalently, to transformations of the form* $(\boldsymbol{\Sigma}_a, \boldsymbol{B}_0) \to (\boldsymbol{C}^{-T}\boldsymbol{\Sigma}_a\boldsymbol{C}^{-1}, \boldsymbol{CB}_0)$*). That is, (a) applying such a transformation does not change the values of the eigenvalues of* $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$*; (b) if we have two estimates* $\hat{\boldsymbol{\Sigma}}_x^{-1}$ *and* $\tilde{\boldsymbol{\Sigma}}_x^{-1}$*, such that the eigenvalues of* $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ *and* $\boldsymbol{M}_x\tilde{\boldsymbol{\Sigma}}_x^{-1}$ *are the same, then there exists a* $\boldsymbol{C} \in \mathsf{GL}(r)$ *such that the estimates of* $(\hat{\boldsymbol{\Sigma}}_a, \hat{\boldsymbol{B}})$ *and* $(\tilde{\boldsymbol{\Sigma}}_a, \tilde{\boldsymbol{B}})$ *can be related by a transformation:* $\hat{\boldsymbol{B}} = \boldsymbol{C}\tilde{\boldsymbol{B}}$ *and* $\hat{\boldsymbol{\Sigma}}_a = \boldsymbol{C}^{-T}\tilde{\boldsymbol{\Sigma}}_a\boldsymbol{C}^{-1}$*.*

Proof. First, we show that the eigenvalues are invariant to such a transformation.

Case 1: $\boldsymbol{a}_i \overset{iid}{\sim} N(\boldsymbol{0}, \boldsymbol{\Sigma}_a)$. Let $\tilde{\boldsymbol{\Sigma}}_a = \boldsymbol{C}^{-T}\hat{\boldsymbol{\Sigma}}_a\boldsymbol{C}^{-1}$ and $\tilde{\boldsymbol{B}}_0 = \boldsymbol{C}\hat{\boldsymbol{B}}_0$. We have

$$\begin{aligned}
\tilde{\boldsymbol{B}}_0^T\tilde{\boldsymbol{\Sigma}}_a\tilde{\boldsymbol{B}}_0 &= \left(\boldsymbol{C}^{-1}\hat{\boldsymbol{B}}_0\right)^T\left(\boldsymbol{C}^T\hat{\boldsymbol{\Sigma}}_a\boldsymbol{C}\right)\left(\boldsymbol{C}^{-1}\hat{\boldsymbol{B}}_0\right) \\
&= \hat{\boldsymbol{B}}_0\boldsymbol{C}^{-T}\boldsymbol{C}^T\hat{\boldsymbol{\Sigma}}_a\boldsymbol{C}\boldsymbol{C}^{-1}\hat{\boldsymbol{B}}_0 \\
&= \hat{\boldsymbol{B}}_0^T\hat{\boldsymbol{\Sigma}}_a\hat{\boldsymbol{B}}_0.
\end{aligned}$$

Hence,

$$\tilde{\boldsymbol{\Sigma}}_x = \tilde{\boldsymbol{B}}_0^T\tilde{\boldsymbol{\Sigma}}_a\tilde{\boldsymbol{B}}_0 + \hat{\boldsymbol{\Sigma}}_z = \hat{\boldsymbol{B}}_0^T\hat{\boldsymbol{\Sigma}}_a\hat{\boldsymbol{B}}_0 + \hat{\boldsymbol{\Sigma}}_z = \hat{\boldsymbol{\Sigma}}_x.$$

Case 2: The $\boldsymbol{a}_i$'s are fixed. Let $\tilde{\boldsymbol{M}}_a = \boldsymbol{C}^{-T}\hat{\boldsymbol{M}}_a\boldsymbol{C}^{-1}$ (which is equivalent to $\tilde{\boldsymbol{A}} = \hat{\boldsymbol{A}}\boldsymbol{C}^{-1}$) and $\tilde{\boldsymbol{B}}_0 = \boldsymbol{C}\hat{\boldsymbol{B}}_0$. Then $\hat{\boldsymbol{\Sigma}}_x = \hat{\boldsymbol{B}}^T\hat{\boldsymbol{M}}_a\hat{\boldsymbol{B}} + \hat{\boldsymbol{\Sigma}}_z$. We get

$$\begin{aligned}\tilde{\boldsymbol{B}}_0^T\tilde{\boldsymbol{M}}_a\tilde{\boldsymbol{B}}_0 &= \left(\boldsymbol{C}^{-1}\hat{\boldsymbol{B}}_0\right)^T\left(\boldsymbol{C}^T\hat{\boldsymbol{M}}_a\boldsymbol{C}\right)\left(\boldsymbol{C}^{-1}\hat{\boldsymbol{B}}_0\right)\\ &= \hat{\boldsymbol{B}}_0\boldsymbol{C}^{-T}\boldsymbol{C}^T\hat{\boldsymbol{M}}_a\boldsymbol{C}\boldsymbol{C}^{-1}\hat{\boldsymbol{B}}_0\\ &= \hat{\boldsymbol{B}}_0^T\hat{\boldsymbol{M}}_a\hat{\boldsymbol{B}}_0.\end{aligned}$$

Hence,

$$\tilde{\boldsymbol{\Sigma}}_x = \tilde{\boldsymbol{B}}_0^T\tilde{\boldsymbol{M}}_a\tilde{\boldsymbol{B}}_0 + \boldsymbol{\Sigma}_z = \hat{\boldsymbol{B}}_0^T\hat{\boldsymbol{M}}_a\hat{\boldsymbol{B}}_0 + \hat{\boldsymbol{\Sigma}}_z = \hat{\boldsymbol{\Sigma}}_x.$$

In either case, $\hat{\boldsymbol{\Sigma}}_x$ is unchanged and as $\boldsymbol{M}_x$ is free from $\boldsymbol{A}$ and $\boldsymbol{B}_0$, $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ will be the same, and hence its eigenvalues will also remain unchanged.

To show (b), we first show that, if $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ and $\boldsymbol{M}_x\tilde{\boldsymbol{\Sigma}}_x^{-1}$ have the same eigenvalues, then $\tilde{\boldsymbol{\Sigma}}_x = \hat{\boldsymbol{\Sigma}}_x$. Then, we show that $\tilde{\boldsymbol{\Sigma}}_x = \hat{\boldsymbol{\Sigma}}_x$ implies that both $\tilde{\boldsymbol{B}}_0^T\tilde{\boldsymbol{\Sigma}}_a\tilde{\boldsymbol{B}}_0 = \hat{\boldsymbol{B}}_0^T\hat{\boldsymbol{\Sigma}}_a\hat{\boldsymbol{B}}_0$ and $\tilde{\boldsymbol{\Sigma}}_z = \hat{\boldsymbol{\Sigma}}_z$. The result follows.

Step 1. Suppose $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ and $\boldsymbol{M}_x\tilde{\boldsymbol{\Sigma}}_x^{-1}$ have the same eigenvalues. Using the Singular Value Decomposition, we can write

$$\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1} = \boldsymbol{Y}\boldsymbol{\Lambda}\boldsymbol{Y}^T$$

and

$$\boldsymbol{M}_x\tilde{\boldsymbol{\Sigma}}_x^{-1} = \tilde{\boldsymbol{Y}}\boldsymbol{\Lambda}\tilde{\boldsymbol{Y}}^T,$$

where $\boldsymbol{Y}$ and $\tilde{\boldsymbol{Y}}$ are orthogonal matrices, and $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues. Hence,

$$\boldsymbol{\Lambda} = \boldsymbol{Y}^T\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}\boldsymbol{Y} = \tilde{\boldsymbol{Y}}^T\boldsymbol{M}_x\tilde{\boldsymbol{\Sigma}}_x^{-1}\tilde{\boldsymbol{Y}},$$

and so

$$\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1} = \boldsymbol{Y}\tilde{\boldsymbol{Y}}^T\boldsymbol{M}_x\tilde{\boldsymbol{\Sigma}}_x^{-1}\tilde{\boldsymbol{Y}}\boldsymbol{Y}^T.$$

However, since $\boldsymbol{M}_x = \boldsymbol{Y}\tilde{\boldsymbol{Y}}^T\boldsymbol{M}_x$, we must have $\boldsymbol{Y}\tilde{\boldsymbol{Y}}^T = \boldsymbol{I}$, and so $\boldsymbol{Y} = \tilde{\boldsymbol{Y}}$. Hence, $\boldsymbol{M}_x\tilde{\boldsymbol{\Sigma}}_x^{-1} = \boldsymbol{Y}\boldsymbol{\Lambda}\boldsymbol{Y}^T = \boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$. Since $\boldsymbol{M}_x$ is invertible (provided $n > p$), $\tilde{\boldsymbol{\Sigma}}_x^{-1} = \hat{\boldsymbol{\Sigma}}_x^{-1}$, and so $\tilde{\boldsymbol{\Sigma}}_x = \hat{\boldsymbol{\Sigma}}_x$.

Step 2. We show that, given that an exact decomposition of $\hat{\boldsymbol{\Sigma}}_x$ into the sum of a rank-r matrix and a diagonal matrix exists, the decomposition is unique.

First, we consider the simpler case where $\boldsymbol{\Sigma}_z = \phi^2 \boldsymbol{I}$. Let $\hat{\boldsymbol{\Xi}} = \hat{\boldsymbol{B}}_0^T \hat{\boldsymbol{\Sigma}}_a \hat{\boldsymbol{B}}_0$. In this case, we have

$$\hat{\boldsymbol{\Sigma}}_x = \hat{\boldsymbol{\Xi}} + \hat{\sigma}^2 \boldsymbol{I}, \tag{5.25}$$

where $\hat{\boldsymbol{\Xi}}$ is of rank r. Suppose there is also a matrix $\tilde{\boldsymbol{\Xi}}$ of rank r, and a positive number $\tilde{\sigma}^2$, such that

$$\hat{\boldsymbol{\Sigma}}_x = \tilde{\boldsymbol{\Xi}} + \tilde{\sigma}^2 \boldsymbol{I}. \tag{5.26}$$

Let the eigenvalues of $\hat{\boldsymbol{\Xi}}$, $\tilde{\boldsymbol{\Xi}}$ and $\hat{\boldsymbol{\Sigma}}_x$ be denoted respectively by $\{\lambda_1^{\hat{\xi}}, ..., \lambda_p^{\hat{\xi}}\}$,$\{\lambda_1^{\tilde{\xi}}, ..., \lambda_p^{\tilde{\xi}}\}$, and $\{\lambda_1^{\Sigma}, ..., \lambda_p^{\Sigma}\}$, where, in each case, the λ_i's are in descending order, so $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p$. By Equation 5.25, the eigenvalues of $\hat{\boldsymbol{\Sigma}}_x$ solve

$$|\left(\hat{\boldsymbol{\Xi}} + \hat{\sigma}^2 \boldsymbol{I}\right) - \lambda^{\Sigma} \boldsymbol{I}| = 0,$$

or, equivalently,

$$|\hat{\boldsymbol{\Xi}} - \left(\lambda^{\Sigma} - \hat{\sigma}^2\right) \boldsymbol{I}| = 0,$$

and, by Equation 5.26, they also solve

$$|\tilde{\boldsymbol{\Xi}} - \left(\lambda^{\Sigma} - \tilde{\sigma}^2\right) \boldsymbol{I}| = 0.$$

Hence, we get

$$\lambda_i^{\Sigma} = \lambda_i^{\xi} + \hat{\sigma}^2 = \lambda_i^{\tilde{\xi}} + \tilde{\sigma}^2.$$

Since both $\hat{\boldsymbol{\Xi}}$ and $\tilde{\boldsymbol{\Xi}}$ are of rank r, their last $p - r$ eigenvalues will all be 0, and so for $j > r$, we get

$$\lambda_j^{\Sigma} = \hat{\sigma}^2 = \tilde{\sigma}^2.$$

Hence, $\tilde{\sigma}^2 = \hat{\sigma}^2$, and $\tilde{\boldsymbol{\Xi}} = \hat{\boldsymbol{\Sigma}}_x - \hat{\sigma}^2 \boldsymbol{I} = \hat{\boldsymbol{\Xi}}$.

Now let $\boldsymbol{\Sigma}_z$ be diagonal. We want to show that if $\hat{\boldsymbol{\Sigma}}_x$ can be expressed as

$$\hat{\boldsymbol{\Sigma}}_x = \hat{\boldsymbol{\Xi}} + \hat{\boldsymbol{\Sigma}}_z, \tag{5.27}$$

where $\hat{\boldsymbol{\Xi}}$ is a symmetric positive definite matrix of rank r and $\hat{\boldsymbol{\Sigma}}_z$ is diagonal, then this representation is unique. Suppose that Equation 5.27 holds, with the conditions on $\hat{\boldsymbol{\Xi}}$ and $\hat{\boldsymbol{\Sigma}}_z$. Since $\boldsymbol{\Sigma}_z$ is diagonal, we know that the off-diagonal elements of $\hat{\boldsymbol{\Xi}}$ must be equal to those of $\hat{\boldsymbol{\Sigma}}_x$. Since $\hat{\boldsymbol{\Xi}}$ is symmetric and positive definite, there exists a $p \times r$ matrix $\boldsymbol{Y}$ such that

$$\hat{\boldsymbol{\Xi}} = \boldsymbol{Y}\boldsymbol{Y}^T.$$

Hence, the matrix $\boldsymbol{Y}$ must solve

$$\sum_{k=1}^{r} y_{ik}y_{jk} = \left(\hat{\boldsymbol{\Sigma}}_x\right)_{ij}$$

for $i < j$. This gives a total of $\frac{1}{2}p(p-1)$ equations. If r is too large relative to p then there will be more parameters in $\boldsymbol{Y}$ than equations and $\hat{\boldsymbol{\Xi}}$ will not be unique. Since $\boldsymbol{Y}\boldsymbol{Y}^T$ is invariant to orthogonal transformations of $\boldsymbol{Y}$, the effective number of free parameters is $pr - \frac{1}{2}r(r-1)$, so we need r to be small enough that $pr - \frac{1}{2}r(r-1) \leq \frac{1}{2}p(p-1)$, in order to have at least as many equations as parameters. Since we know that a solution exists, we do not need to worry about having more equations than parameters; in this case some equations will be redundant. A unique solution will exist as long as we have $pr - \frac{1}{2}r(r-1)$ independent equations.

If the $\boldsymbol{a}_i$'s are fixed instead of random, we can replace $\boldsymbol{\Sigma}_a$ with $\boldsymbol{M}_a$, and Step 2 of the proof proceeds in the same way.

Step 3. We have shown that, if $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$ and $\boldsymbol{M}_x\tilde{\boldsymbol{\Sigma}}_x^{-1}$ have the same eigenvalues (where $\hat{\boldsymbol{\Sigma}}_x = \hat{\boldsymbol{B}}_0^T\hat{\boldsymbol{\Sigma}}_a\hat{\boldsymbol{B}}_0 + \hat{\boldsymbol{\Sigma}}_z$ and $\tilde{\boldsymbol{\Sigma}}_x = \tilde{\boldsymbol{B}}_0^T\tilde{\boldsymbol{\Sigma}}_a\tilde{\boldsymbol{B}}_0 + \tilde{\boldsymbol{\Sigma}}_z$), then $\hat{\boldsymbol{\Sigma}}_z = \tilde{\boldsymbol{\Sigma}}_z$ and $\hat{\boldsymbol{B}}_0^T\hat{\boldsymbol{\Sigma}}_a\hat{\boldsymbol{B}}_0 = \tilde{\boldsymbol{B}}_0^T\tilde{\boldsymbol{\Sigma}}_a\tilde{\boldsymbol{B}}_0$. It then follows that $\hat{\boldsymbol{B}}_0$ and $\tilde{\boldsymbol{B}}_0$ must be related through a transformation of the form $\tilde{\boldsymbol{B}}_0 = \boldsymbol{C}\boldsymbol{B}_0$, where $\boldsymbol{C} \in \mathsf{GL}(r)$. □

5.6 Conclusion

In this chapter we have focused on methods for generating dynamic word embeddings, and testing for whether words have changed across time. We have introduced

a new model for dynamic word embeddings based on LSA, and have demonstrated how we can estimate the parameters and make the model identifiable. We have also shown that the model is not affected by the scale we use to measure time, or by the order in which the documents are placed.

We investigated testing procedures for determining whether there is significant change across time within a dataset using a modification of our previously introduced model, where we allowed the variances for each word to be different, rather than requiring them all to be the same. In the static case this is equivalent to the factor model. This allowed us to make use of asymptotic theory from factor analysis, which we used to develop a test for adequacy of the null (static) model.

We used simulations to verify that, under H_0, $\mathcal{T}$ followed the expected distribution, as long as the number of data points was sufficiently large (n approximately equal to three or four times p appeared to be sufficient). We also showed that, when simulating from the dynamic model, H_0 was rejected, unless the values in $\boldsymbol{B}_1$ were very small compared to those in $\boldsymbol{A}$ and $\boldsymbol{B}_0$. We showed that it is possible for H_0 to be rejected when only a few words are moving, depending on the size of the entries in $\boldsymbol{B}_1$ corresponding to those words. However, we also found that H_0 is rejected when the value of r used to fit the model is smaller than the true value.

We showed that if we re-scaled the times to have mean 0 and variance 1, then it was possible to equate $\boldsymbol{\Sigma}_x$ under the dynamic model with embedding dimension r with $\boldsymbol{\Sigma}_x$ under the static model with embedding dimension $2r$. Further, we showed in Section 5.5.6 that the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$, of which $\mathcal{T}$ is a function, are impervious to the transformation of $\boldsymbol{\Sigma}_a$ and $\boldsymbol{B}$ by a non-singular matrix. This means that the test is invariant to the choice of identifiability conditions (as changing the identifiability conditions is equivalent to multiplying the embeddings by a non-singular matrix), and this will apply to any test statistic which is a function of the eigenvalues of $\boldsymbol{M}_x\hat{\boldsymbol{\Sigma}}_x^{-1}$. However, such a test will not be able to distinguish between the dynamic model with embedding dimension r and the static model with embedding

dimension $2r$ (Proposition 5).

It seems likely that similar problems will arise with any other method for generating time-dependent embeddings, unless we are able to determine the "correct" dimension of the embeddings. If adding a time-dependent component to the model effectively increases the dimensionality of the embeddings (as it does here), then increasing the dimension of the static component of the embeddings may hide any time dependence in the data, as the time-dependent component will be absorbed into the extra dimensions of the static component. On the other hand, if the embedding dimension is too small the reverse may happen, with a time component appearing to be present in the data, when it is not. Thus, any method for testing for time dependence based on word embeddings must be either independent of the embedding dimension, or dependent upon having some way of determining the correct embedding dimension for the data.

Chapter 6

Conclusion

In this thesis we have looked at several aspects of word embeddings: identifiability, semi-supervised embeddings, and testing for time dependence within a dataset.

In Chapter 3, we looked at the issue of non-identifiability in word embedding methods. We explored some of the consequences of this, in particular the implications of the different sets of invariances for the objective function f, which the embeddings are optimized with respect to, and the test function g, on which the word embeddings are evaluated. The function f is invariant to non-singular transformations of the embedding set, whereas g is invariant only to orthogonal and scale transformations. As a result, different embedding sets which perform equally well with respect to the objective function may perform differently on test data, which we showed to be the case.

We explored two possible ways of resolving this issue. The first was to impose identifiability conditions on the embeddings, ensuring that the set of embeddings that optimized the objective function could be uniquely determined (up to possible orthogonal and scale transformations of the embeddings, which do not affect g). We found that embeddings with such conditions imposed performed well on test data.

The second solution we explored was to optimize over the set of non-singular transformations of the embeddings, with respect to g. In order to prevent overfitting, we evaluated the performance of the embeddings generated in this way on

other test sets, as well as the test set used for optimization. For word similarity tasks, we found that it was possible to significantly increase performance on a test set for similarity tasks, and that optimizing over one test set usually gave improved results for other test sets, although this was not always the case.

The results for analogy tasks were less promising: transforming the embeddings by random diagonal or upper triangular matrices led to a significant decrease in performance. However, we did explore, for SVD, multiplying the embedding set by $\mathbf{\Lambda}^{\alpha}$, and found that it is worth exploring values of α (at least slightly) outside the range $(0, 1)$. Interestingly, the accuracy as a function of α appeared to be a fairly smooth curve.

We were not able to optimize over g for word analogy tasks as we did for word similarity tasks, because the analogy test function takes a much longer time to evaluate, which rendered doing this computationally infeasible. However, this would be something worth investigating in the future: it would be interesting to see whether it is possible to increase performance on analogy tasks by optimizing over the set of non-singular transformations of the embeddings. If this were possible, it would be necessary to find some way of determining whether the embeddings generated using this method were indeed "better" than the original embeddings, or just better at solving these particular kinds of analogies, since the Google Analogy test set contains only a small number of kinds of analogies.

In Chapter 4, we explored the possibility of generating semi-supervised word embeddings using methods based on multidimensional scaling, where the objective function combines an unsupervised dataset $\boldsymbol{X}$ with a supervised dataset $\boldsymbol{D}$. We implemented our algorithm on simulated and real data, showing that we could improve results on test data by using the semi-supervised algorithm, compared to an unsupervised algorithm. We also compared the benefits of using two different algorithms to optimize the objective function, majorization and stochastic gradient descent. We found that majorization gives better convergence, but is much slower when the

number of words is large. Hence, which algorithm we choose to use would depend upon the size of the dataset and the amount of computing resources available.

Although we have demonstrated the potential of semi-supervised word embeddings based on MDS, there is more work to be done in terms of determining what value they may have in practice. In particular, it would be interesting to explore the potential of semi-supervised embeddings in applications where we have smaller datasets available; as unsupervised learning requires large amounts of data, we would expect the benefits of using some supervised data to be larger in these cases.

In Chapter 5, we looked at time-dependent word embeddings, and in particular the problem of testing for time dependence. We introduced a linear model for time-dependent word embeddings, which is an extension of LSA in the static case. We explored some of the properties of this model and tested it on simulated data. We then looked at testing for time dependence, using a modified version of the LSA model so that we could use results from factor analysis. We used a test for adequacy of the null model, and found, through simulations, that the test statistic is rejected when simulating under the time-dependent model, even if only a few words are moving (unless $\boldsymbol{B}_1$ is very small). However, we found that it was not possible, using this model and testing framework, to distinguish between the presence of time dependence in the data and a misspecified embedding dimension. Since the choice of embedding dimension is somewhat arbitrary, this means that such a test may not be useful in practice. Developing a test which could distinguish between these two situations, perhaps based on a different time-dependent embedding model, would be a subject for further research.

Bibliography

Karim M. Abadir and Jan R. Magnus. *Matrix Algebra*. Cambridge: Cambridge University Press, 2005.

Joshua Acosta, Norissa Lamaute, Mingxiao Luo, Ezra Finkelstein, and Andreea Cotoranu. Sentiment analysis of Twitter messages using word2vec. *Proceedings of Student-Faculty Research Day, CSIS, Pace University*, 2017.

Yasuo Amemiya and T. W. Anderson. Asymptotic chi-square tests for a large class of factor analysis models. *The Annals of Statistics*, 18(3):1453–1463, 1990.

Yasuo Amemiya and Wayne A. Fuller. The asymptotic distribution of some estimators of a factor model. *Journal of Multivariate Analysis*, 22:51–64, 1987.

T. W. Anderson and Yasuo Amemiya. The asymptotic normal distribution of estimators in factor analysis under general conditions. *The Annals of Statistics*, 16 (2):759–771, 1988.

T. W. Anderson and Herman Rubin. Statistical inference in factor analysis. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*, volume 5, pages 111–150. University of California Press, 1956.

Anita Bai, Swati Hira, and P. S. Deshpande. An application of factor analysis in the evaluation of country economic rank. *Procedia Computer Science*, 54:311–317, 2015.

Jushan Bai. Inferential theory for factor models of large dimensions. *Econometrica*, 71(1):135–171, 2003.

Simon Baker, Roi Reichart, and Anna Korhonen. An unsupervised model for instance level subcategorization acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 278–289. Association for Computational Linguistics, 2014.

Robert Bamler and Stephan Mandt. Dynamic word embeddings. *arXiv preprint arXiv 1702.08359v2*, 2017.

Oren Barkan. Bayesian neural word embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 3135–3143. Association for the Advancement of Artificial Intelligence, 2017.

Dennis S Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas with Application to Linear Systems Theory*. Princeton, N.J.; Oxford: Princeton University Press, 2005.

David M. Blei and John D. Lafferty. Correlated topic models. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, pages 147–154. MIT Press, 2005.

David M. Blei and John D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 113–120. ACM, 2006.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of Data Science*. Cambridge University Press, 2020.

Ingwer Borg and Patrick Groenen. *Modern Multidimensional Scaling: Theory and Applications*. New York: Springer, 1997.

John A. Bullinaria and Joseph P. Levy. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD. *Behavior Research Methods*, 44(3):890–907, 2012.

Olivier Chapelle, Bernhard Sch olkopf, and Alexander Zien. *Introduction to Semi-Supervised Learning*, pages 1–12. MITP, 2006.

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

Mark Davies. The Corpus of Contemporary American English as the first reliable monitor corpus of English. *Literary and Linguistic Computing*, 25(4):447–464, 2010.

Mark Davies. Expanding horizons in historical linguistics with the 400-million word Corpus of Historical American English. *Corpora*, 7(2):121–157, 2012.

Jan de Leeuw. Convergence of the majorization method for multidimensional scaling. *Journal of Classification*, 5(2):163–180, 1988.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

Morris H DeGroot and Mark J Schervish. *Probability and Statistics*. Pearson Education, 2012.

Haim Dubossarsky, Yulia Tsvetkov, Chris Dyer, and Eitan Grossman. A bottom up approach to category mapping and meaning change. In *Word Structure and Word Usage*, pages 66–70, 2015.

Haim Dubossarsky, Daphna Weinshall, and Eitan Grossman. Verbs change more than nouns: a bottom-up computational approach to semantic change. *Lingue e linguaggio*, 15(1):5–25, 2016.

Haim Dubossarsky, Eitan Grossman, and Daphna Weinshall. Outta control: Laws of semantic change and inherent biases in word representation models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1136–1145. Association for Computational Linguistics, 2017.

George H. Dunteman. *Basic Concepts of Principal Components Analysis*. SAGE Publications Ltd: London, 1989.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3), 1936.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, pages 406–414, 2001.

J. Kevin Ford, Robert C. MacCallum, and Marianne Tait. The application of exploratory factor analysis in applied psychology: A critical review and analysis. *Personnel Psychology*, 39:291–314, 1986.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. SimVerb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869*, 2016.

Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. *arXiv preprint arXiv:1606.07154v1*, 2016.

P. J. F. Groenen, P. Giaquinto, and H. A. L. Kiers. Weighted majorization algorithms for weighted least squares decomposition models. Technical report, Econometric Institute Report EI 2003-09, 2003.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM*

SIGKDD international conference on Knowledge discovery and data mining, pages 1406–1414. ACM, 2012.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv preprint arXiv:1605.09096*, 2016.

Johannes Hellrich and Udo Hahn. Bad company–neighborhoods in neural embedding spaces considered harmful. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2785–2796, 2016.

Felix Hill, Roi Reichart, and Anna Korhonen. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4): 665–695, 2015.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

Carina Jacobi, Wouter van Atteveldt, and Kasper Welbers. Quantitative analysis of large amounts of journalistic texts using topic modelling. *Digital Journalism*, 4(1):89–106, 2016.

Stefan Jansen. Word and phrase translation with word2vec. *arXiv preprint arXiv:1705.03127*, 2017.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882v2*, 2014.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. Temporal analysis of language through neural language models. *arXiv preprint arXiv:1405.3515*, 2014.

J. B. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, 1964.

Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. Diachronic word embeddings and semantic shifts: a survey. *arXiv preprint arXiv:1806.03537*, 2018.

Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting for the Association for Computational Linguistics (Short Papers)*, pages 302–308, 2014a.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 2177–2185, 2014b.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

Chi-Kwong Li and Paras P. Mehta. Permutation invariant norms. *Linear Algebra and its Applications*, 219:93–110, 1995.

Haixia Liu. Sentiment analysis of citations using word2vec. *arXiv preprint arXiv:1704.00177*, 2017.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*, 2014.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013b.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119, 2013c.

Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 2265–2273, 2013.

J. Neyman and Elizabeth L. Scott. Consistent estimates based on partially consistent observations. *Econometrica*, 16(1):1–32, 1948.

Jun-Ping Ng and Viktoria Abrecht. Better summarization evaluation with word embeddings for ROUGE. *arXiv preprint arXiv:1508.06034*, 2015.

W. Keith Nicholson. *Linear Algebra With Applications*. Lyryx, 2019.

V. Paul Pauca, Farial Shahnaz, Michael W. Berry, and Robert J. Plemmons. Text mining using non-negative matrix factorizations. In *Proceedings of the 2004 SIAM Conference on Data Mining*, pages 452–456. SIAM, 2004.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Assoication for Computational Linguistics, 2014.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online learning of social representations. *arXiv preprint arXiv:1403.6652v2*, 2014.

Christopher Purdy, Xinyu Wang, Larry He, and Mark Riedl. Predicting generated story quality with quantitative measures. In *Proceedings of the Fourteenth Artificial Inteligence and Interactive Digital Entertainment Conference (AIIDE 2018)*, 2018.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM, 2011.

Juan Ramos. Using TF-IDF to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, pages 133–142. Piscataway, NJ, 2003.

Alex Rosenfeld and Katrin Erk. Deep neural models of semantic shift. In *Proceedings of NAACL-HLT 2018*, pages 474–484. Association for Computational Linguistics, 2018.

Erhard Schmidt. Zur Theorie der linearen und nichtlinearen Integralgleichungen. I. Teil: Entwicklung willk urlicher Funktionen nach Systemen vorgeschriebener. *Mathematische Annalen*, 63:433–476, 1907.

Pontus Stenetorp, Hubert Soyer, Sampo Pyysalo, Sophia Ananiadou, and Takashi Chikayama. Size (and domain) matters: Evaluating semantic word space representations for biomedical text. *Proceedings of SMBM'12*, 2012.

Marc Strickert, Frank-Michael Schleif, and Udo Seiffert. Gradients of Pearson correlation for the analysis of biomedical data. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 12(37):37–44, 2008.

Jun Suzuki and Masaaki Nagata. A unified learning framework of skip-grams and global vectors. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural*

Language Processing (Short Papers), pages 186–191. Association for Computational Linguistics, 2015.

Jun Suzuki and Masaaki Nagata. Learning compact neural word embeddings by parameter space sharing. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 2046–2052, 2016.

Peter D. Turney and Michael L. Littman. Unsupervised learning of semantic orientation from a hundred-billion-word corpus. *arXiv preprint arXiv:cs/0212012*, 2002.

Xuerui Wang and Andrew McCallum. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM, 2006.

Derry Tanti Wijaya and Reyyan Yeniterzi. Understanding semantic change of words over centuries. In *Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web*, pages 35–40. ACM, 2011.

S. S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9(1):60–62, 1938.

Daniela M. Witten and Robert Tibshirani. Supervised multidimensional scaling for visualization, classification, and bipartite ranking. *Computational Statistics & Data Analysis*, 55(1):789–801, 2011.

Wei Xu, Xin Liu, and Yihong Gong. Document clustering based oon non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 267–273, 2003.

Dongqiang Yang and David M. W. Powers. Measuring semantic similarity in the taxonomy of WordNet. In *Proceedings of the Twenty-eighth Australasian confer-*

ence on Computer Science, pages 315–322. Australian Computer Society, Inc., 2005.

Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 673–681, 2018.

Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xuejie Zhang. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 534–539, 2017.

Wen Zhang, Taketoshi Yoshida, and Xijin Tang. A comparative study of TF*IDF, LSI and multi-words for text classification. *Expert Systems with Applications*, 38 (3):2758–2765, 2011.

Appendix A

Proof of the distribution of A under the assumption that B_0 is fixed

A.1 Notation

In this section the following notation is used:

- $\boldsymbol{x} = \left(\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, ..., \boldsymbol{x}_p^T\right)^T$: vectorized $\boldsymbol{X}$,
- $\boldsymbol{b}_0 = \left(\left(\boldsymbol{b}_1^0\right)^T, ..., \left(\boldsymbol{b}_p^0\right)^T\right)^T$: vectorized $\boldsymbol{B}_0$,
- $\boldsymbol{b}_1 = \left(\left(\boldsymbol{b}_1^1\right)^T, ..., \left(\boldsymbol{b}_p^1\right)^T\right)^T$: vectorized $\boldsymbol{B}_1$,
- $\boldsymbol{b} = \left(\boldsymbol{b}_0^T, \boldsymbol{b}_1^T\right)^T$,
- $\boldsymbol{A}_*$: block diagonal matrix containing $\boldsymbol{A}$ repeated p times along the diagonal,
- $\boldsymbol{T}_*$: diagonal matrix containing the vectors of times $\boldsymbol{t}$ repeated p times along the diagonal,
- $\boldsymbol{A}_{**}$: block diagonal matrix containing $(\boldsymbol{A}, \boldsymbol{TA})$ repeated p times along the diagonal,

where $\boldsymbol{x}_j$ denotes here the jth column of $\boldsymbol{X}$ and $\boldsymbol{b}_j^q$ denotes the jth column of $\boldsymbol{B}_q$.

Using this notation the model under H_0 is

$$H_0 : \boldsymbol{x} \sim N\left(\boldsymbol{A}_* \boldsymbol{b}_0, \sigma^2 \boldsymbol{I}\right),$$

and under H_1,

$$H_1 : \boldsymbol{x} \sim N\left(\boldsymbol{A}_* \boldsymbol{b}_0 + \boldsymbol{T}_* \boldsymbol{A}_* \boldsymbol{b}_1, \sigma^2 \boldsymbol{I}\right),$$

or alternatively,

$$H_1 : \boldsymbol{x} \sim N\left(\boldsymbol{A}_{**} \boldsymbol{b}, \sigma^2 \boldsymbol{I}\right).$$

Let the estimated values of the parameters under H_0 be denoted $\hat{\boldsymbol{A}}_*$ and $\hat{\boldsymbol{b}}_0$, and the estimated values under H_1 be $\tilde{\boldsymbol{A}}^*$, $\tilde{\boldsymbol{b}}_0$, and $\tilde{\boldsymbol{b}}_1$. Then the likelihood ratio statistic can be written as

$$\begin{aligned} -2\log\Lambda &= \frac{1}{\sigma^2}\sum_{j=1}^{np}\left(\left(x_j - \left(\hat{\boldsymbol{A}}_*\hat{\boldsymbol{b}}_0\right)_j\right)^2 - \left(x_j - \left(\tilde{\boldsymbol{A}}_*\tilde{\boldsymbol{b}}_0 - \boldsymbol{T}_*\tilde{\boldsymbol{A}}_*\tilde{\boldsymbol{b}}_1\right)_j\right)^2\right) \\ &= \frac{1}{\sigma^2}\left(\left(\boldsymbol{x} - \hat{\boldsymbol{A}}_*\hat{\boldsymbol{b}}_0\right)^T\left(\boldsymbol{x} - \hat{\boldsymbol{A}}_*\hat{\boldsymbol{b}}_0\right) - \left(\boldsymbol{x} - \tilde{\boldsymbol{A}}_*\tilde{\boldsymbol{b}}_0 - \boldsymbol{T}_*\tilde{\boldsymbol{A}}_*\tilde{\boldsymbol{b}}_1\right)^T\left(\boldsymbol{x} - \tilde{\boldsymbol{A}}_*\tilde{\boldsymbol{b}}_0 - \boldsymbol{T}_*\tilde{\boldsymbol{A}}_*\tilde{\boldsymbol{b}}_1\right)\right). \end{aligned}$$

The proof is divided into four sections: one when $\boldsymbol{A}_*$, $\boldsymbol{b}_0$ and σ^2 are all known; one when just $\boldsymbol{A}_*$ and $\boldsymbol{b}_0$ are known; one when $\boldsymbol{A}_*$ and σ^2 are known, and one when just $\boldsymbol{A}_*$ is known.

A.2 Case 1: $\boldsymbol{A}$, $\boldsymbol{B}_0$, and σ^2 fixed

If $\boldsymbol{A}_*$, $\boldsymbol{b}_0$ and σ^2 are fixed, then under H_0 there is nothing to estimate, and under H_1 we only have to estimate $\boldsymbol{b}_1$. This is equivalent to minimizing

$$||\left(\boldsymbol{x} - \boldsymbol{A}_*\boldsymbol{b}_0\right) - \left(\boldsymbol{T}_*\boldsymbol{A}_*\right)\boldsymbol{b}_1||$$

by least squares, and so we get

$$\tilde{\boldsymbol{b}}_1 = \left(\left(\boldsymbol{A}_*\boldsymbol{T}_*\right)^T\boldsymbol{A}_*\boldsymbol{T}_*\right)^{-1}\left(\boldsymbol{A}_*\boldsymbol{T}_*\right)^T\left(\boldsymbol{x} - \boldsymbol{A}_*\boldsymbol{b}_0\right).$$

Let $\boldsymbol{c} = \boldsymbol{x} - \boldsymbol{A}_*\boldsymbol{b}_0$ and $\boldsymbol{G} = \boldsymbol{T}_*\boldsymbol{A}_*\left(\left(\boldsymbol{T}_*\boldsymbol{A}_*\right)^T\boldsymbol{T}_*\boldsymbol{A}_*\right)^{-1}\left(\boldsymbol{T}_*\boldsymbol{A}_*\right)^T$. Then

$$\boldsymbol{T}_*\boldsymbol{A}_*\tilde{\boldsymbol{b}}_1 = \boldsymbol{G}\boldsymbol{c},$$

and so the likelihood ratio statistic is

$$\frac{1}{\sigma^2}\left(\boldsymbol{c}^T\boldsymbol{c} - (\boldsymbol{c}-\boldsymbol{G}\boldsymbol{c})^T(\boldsymbol{c}-\boldsymbol{G}\boldsymbol{c})\right).$$

Multiplying out gives

$$-2\log\Lambda = \frac{1}{\sigma^2}\left(\boldsymbol{c}^T\boldsymbol{c} - \boldsymbol{c}^T\boldsymbol{c} + \boldsymbol{c}^T\boldsymbol{G}\boldsymbol{c} + \boldsymbol{c}^T\boldsymbol{G}^T\boldsymbol{c} - \boldsymbol{c}^T\boldsymbol{G}^T\boldsymbol{G}\boldsymbol{c}\right).$$

Now $\boldsymbol{G}$ is symmetric and idempotent, so $\boldsymbol{G} = \boldsymbol{G}^T = \boldsymbol{G}^T\boldsymbol{G}$. Thus most of the terms cancel and we are left with

$$-2\log\Lambda = \frac{1}{\sigma^2}\boldsymbol{c}^T\boldsymbol{G}\boldsymbol{c}.$$

$\boldsymbol{G}$ is $np \times np$, but is of rank rp, so we can compute the rank-rp SVD of $\boldsymbol{G}$ and get $\boldsymbol{G} = \boldsymbol{U}_{rp}\boldsymbol{\Sigma}_{rp}\boldsymbol{V}_{rp}^T$, where $\boldsymbol{U}_{rp}$ and $\boldsymbol{V}_{rp}$ are $np \times rp$ matrices with orthonormal columns, and $\boldsymbol{\Sigma}_{rp}$ is an $rp \times rp$ diagonal matrix. Using $\boldsymbol{G} = \boldsymbol{G}^T\boldsymbol{G}$, we get

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T.$$

Right-multiplying by $\boldsymbol{V}\boldsymbol{\Sigma}^{-1}$ gives

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}^{-1} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}^{-1},$$

which simplifies to

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{\Sigma}^{-1},$$

and then to

$$\boldsymbol{U} = \boldsymbol{V}\boldsymbol{\Sigma},$$

so $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{U}\boldsymbol{U}^T$. Thus,

$$-2\log\Lambda = \frac{1}{\sigma^2}\boldsymbol{c}^T\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{c} = \left(\frac{1}{\sigma}\boldsymbol{U}^T\boldsymbol{c}\right)^T\left(\frac{1}{\sigma}\boldsymbol{U}^T\boldsymbol{c}\right).$$

Under H_0,

$$\boldsymbol{x} \sim N\left(\boldsymbol{A}_*\boldsymbol{b}_0, \sigma^2\boldsymbol{I}\right),$$

so

$$\boldsymbol{c} \sim N\left(\boldsymbol{0}, \sigma^2\boldsymbol{I}\right),$$

so

$$\frac{1}{\sigma}\boldsymbol{U}^T\boldsymbol{c} \sim N(\boldsymbol{0}, \boldsymbol{U}^T\boldsymbol{U}).$$

But $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$, as $\boldsymbol{U}$ has orthonormal columns. So

$$\frac{1}{\sigma}\boldsymbol{U}^T\boldsymbol{c} \sim N(\boldsymbol{0}, \boldsymbol{I}),$$

so, letting $\boldsymbol{\nu} = \frac{1}{\sigma^2}\boldsymbol{U}^T\boldsymbol{c}$, we get

$$-2\log\Lambda = \boldsymbol{\nu}^T\boldsymbol{\nu} = \sum_{j=1}^{rp}\nu_i^2,$$

and so under H_0 the likelihood ratio statistic is the sum of the squares of rp independent Normal random variables with mean 0 and variance 1, and therefore follows a χ^2_{rp} distribution.

A.3 Case 2: $\boldsymbol{A}$ and $\boldsymbol{B}_0$ fixed, σ^2 unknown

The F-statistic is

$$F = \frac{\left(\sum_{j=1}^{np}\left(x_j - (\boldsymbol{A}_*\boldsymbol{b}_0)_j\right)^2 - \sum_{j=1}^{np}\left(x_j - (\boldsymbol{A}_*\boldsymbol{b}_0)_j - (\boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{b}_1)_k\right)^2\right)/rp}{\left(\sum_{j=1}^{np}\left(x_j - (\boldsymbol{A}_*\boldsymbol{b}_0)_j - (\boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{b}_1)_k\right)^2\right)/(np - rp)}.$$

By definition, if $Y_1 \sim \chi^2_{\nu_1}$ and $Y_2 \sim \chi^2_{\nu_2}$, and Y_1 and Y_2 are independent, then $\frac{Y_1/\nu_1}{Y_2/\nu_2} \sim F_{\nu_1,\nu_2}$ [DeGroot and Schervish, 2012].

Defining $\boldsymbol{c}$ and $\boldsymbol{G}$ as before, we get

$$F = \frac{\boldsymbol{c}^T\boldsymbol{G}\boldsymbol{c}/rp}{\boldsymbol{c}^T\left(\boldsymbol{I} - \boldsymbol{G}\right)\boldsymbol{c}/(np - rp)}.$$

As shown previously, $\boldsymbol{c}^T\boldsymbol{G}\boldsymbol{c} = \left(\boldsymbol{U}^T\boldsymbol{c}\right)^T\boldsymbol{U}^T\boldsymbol{c}$, where $\frac{1}{\sigma}\boldsymbol{U}^T\boldsymbol{c} \sim N_{rp}\left(\boldsymbol{0}, \boldsymbol{I}\right)$, so $\frac{1}{\sigma^2}\boldsymbol{c}^T\boldsymbol{G}\boldsymbol{c} \sim \chi^2_{rp}$.

As $\boldsymbol{G}$ is idempotent, so is $(\boldsymbol{I} - \boldsymbol{G})$, as $(\boldsymbol{I} - \boldsymbol{G})(\boldsymbol{I} - \boldsymbol{G}) = \boldsymbol{I} - \boldsymbol{G} - \boldsymbol{G} + \boldsymbol{G}^2 = \boldsymbol{I} - \boldsymbol{G}$. The rank of an idempotent matrix is equal to its trace Abadir and Magnus [2005], so we get

$$rank\left(\boldsymbol{I} - \boldsymbol{G}\right) = trace\left(\boldsymbol{I} - \boldsymbol{G}\right) = trace(\boldsymbol{I}) - trace(\boldsymbol{G}) = np - rank(\boldsymbol{G}) = np - rp.$$

Therefore, using the same process as for Case 1m we can write $\boldsymbol{I} - \boldsymbol{G} = \boldsymbol{V}\boldsymbol{V}^T$, where $\boldsymbol{V}$ is an $np \times (np - rp)$ matrix with orthonormal columns. So

$$\boldsymbol{c}^T (\boldsymbol{I} - \boldsymbol{G}) \boldsymbol{c} = \boldsymbol{c}^T \boldsymbol{V}\boldsymbol{V}^T \boldsymbol{c} = \left(\boldsymbol{V}^T \boldsymbol{c}\right)^T \boldsymbol{V}^T \boldsymbol{c},$$

where $\boldsymbol{V}^T \boldsymbol{c} \sim N_{np-rp}\left(\boldsymbol{0}, \boldsymbol{V}^T (\sigma^2 \boldsymbol{I}) \boldsymbol{V}\right)$, i.e.

$$\boldsymbol{V}^T \boldsymbol{c} \sim N_{np-rp}\left(\boldsymbol{0}, \sigma^2 \boldsymbol{I}\right),$$

so

$$\frac{1}{\sigma^2} \boldsymbol{c}^T (\boldsymbol{I} - \boldsymbol{G}) \boldsymbol{c} \sim \chi^2_{np-rp}.$$

Also,

$$\text{Cov}(\boldsymbol{U}^T \boldsymbol{c}, \boldsymbol{V}^T \boldsymbol{c}) = \boldsymbol{U}^T \cdot var\,(\boldsymbol{c}) \cdot \boldsymbol{V} = \sigma^2 \boldsymbol{U}^T \boldsymbol{V}.$$

We know that

$$\boldsymbol{U}\boldsymbol{U}^T + \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{G} + \boldsymbol{I} - \boldsymbol{G} = \boldsymbol{I}.$$

Multiplying all terms by $\boldsymbol{U}^T$ on the left and $\boldsymbol{V}$ on the right gives

$$\boldsymbol{U}^T \boldsymbol{U}\boldsymbol{U}^T \boldsymbol{V} + \boldsymbol{U}^T \boldsymbol{V}\boldsymbol{V}^T \boldsymbol{V} = \boldsymbol{U}^T \boldsymbol{V},$$

which simplifies to

$$\boldsymbol{U}^T \boldsymbol{V} + \boldsymbol{U}^T \boldsymbol{V} = \boldsymbol{U}^T \boldsymbol{V},$$

i.e.

$$\boldsymbol{U}^T \boldsymbol{V} = \boldsymbol{0},$$

so

$$\text{Cov}\left(\boldsymbol{U}^T \boldsymbol{c}, \boldsymbol{V}^T \boldsymbol{c}\right) = \boldsymbol{0}.$$

So, we have that $\boldsymbol{c}^T \boldsymbol{G} \boldsymbol{c}$ and $\boldsymbol{c}^T (\boldsymbol{I} - \boldsymbol{G}) \boldsymbol{c}$ are independent χ^2 random variables with degrees of freedom rp and $np - rp$ respectively, and so we get

$$\frac{(\sigma^2)^{-1} \boldsymbol{c}^T \boldsymbol{G} \boldsymbol{c}/rp}{(\sigma^2)^{-1} \boldsymbol{c}^T (\boldsymbol{I} - \boldsymbol{G}) \boldsymbol{c}/(np - rp)} \sim F_{rp,np-rp}$$

exactly. The σ^2 terms cancel to give the required F statistic.

A.4 Case 3: $\boldsymbol{A}$ and σ^2 fixed, $\boldsymbol{B}_0$ unknown

If $\boldsymbol{A}_*$ is fixed, but not $\boldsymbol{b}_0$, then under H_0 we get

$$\hat{\boldsymbol{b}}_0 = \arg\min_{\boldsymbol{b}_0} ||\boldsymbol{x} - \boldsymbol{A}_* \boldsymbol{b}_0||^2 = \left((\boldsymbol{A}_*)^T \boldsymbol{A}_* \right)^{-1} (\boldsymbol{A}_*)^T \boldsymbol{x},$$

and under H_1, we get

$$\tilde{\boldsymbol{b}} = \arg\min_{\boldsymbol{b}} ||\boldsymbol{x} - \boldsymbol{A}_{**} \boldsymbol{b}||^2 = \left((\boldsymbol{A}_{**})^T \boldsymbol{A}_{**} \right)^{-1} (\boldsymbol{A}_{**})^T \boldsymbol{x}.$$

The likelihood ratio statistic can then be written as

$$-2 \log \Lambda = \frac{1}{\sigma^2} \left(\boldsymbol{x} - \boldsymbol{A}_* \hat{\boldsymbol{b}}_0 \right)^T \left(\boldsymbol{x} - \boldsymbol{A}_* \hat{\boldsymbol{b}}_0 \right) - \left(\boldsymbol{x} - \boldsymbol{A}_{**} \tilde{\boldsymbol{b}} \right)^T \left(\boldsymbol{x} - \boldsymbol{A}_{**} \tilde{\boldsymbol{b}} \right),$$

which is equal to

$$\frac{1}{\sigma^2} (\boldsymbol{x} - \boldsymbol{G}\boldsymbol{x})^T (\boldsymbol{x} - \boldsymbol{G}\boldsymbol{x}) - (\boldsymbol{x} - \boldsymbol{H}\boldsymbol{x})^T (\boldsymbol{x} - \boldsymbol{H}\boldsymbol{x}),$$

where $\boldsymbol{G} = \boldsymbol{A}_* \left(\boldsymbol{A}_*^T \boldsymbol{A}_* \right)^{-1} \boldsymbol{A}_*^T$ and $\boldsymbol{H} = \boldsymbol{A}_{**} \left((\boldsymbol{A}_{**})^T \boldsymbol{A}_{**} \right)^{-1} (\boldsymbol{A}_{**})^T$. Hence

$$-2 \log \Lambda = \frac{1}{\sigma^2} \left(\boldsymbol{x}^T (\boldsymbol{I} - \boldsymbol{G}) \boldsymbol{x} - \boldsymbol{x}^T (\boldsymbol{I} - \boldsymbol{H}) \boldsymbol{x} \right) = \frac{1}{\sigma^2} \boldsymbol{x}^T (\boldsymbol{H} - \boldsymbol{G}) \boldsymbol{x},$$

using that $(\boldsymbol{I} - \boldsymbol{G})$ and $(\boldsymbol{I} - \boldsymbol{H})$ are both symmetric and idempotent.

Under H_0, where $\boldsymbol{x} = \boldsymbol{A}_x \boldsymbol{b}_0 + \boldsymbol{z}$, this is equal to

$$-2 \log \Lambda = \frac{1}{\sigma^2} (\boldsymbol{A}_* \boldsymbol{b}_0 + \boldsymbol{z})^T (\boldsymbol{H} - \boldsymbol{G}) (\boldsymbol{A}_* \boldsymbol{b}_0 + \boldsymbol{z}),$$

where $z \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$.

In order to show that this follows a χ^2_{pr} distribution, we need to show that $(\boldsymbol{H} - \boldsymbol{G})$ is an idempotent matrix of rank pr.

We have that

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{A}_* & \boldsymbol{T}_* \end{pmatrix} \begin{pmatrix} \boldsymbol{A}_*^T \boldsymbol{A}_* & \boldsymbol{A}_*^T \boldsymbol{T}_* \boldsymbol{A}_* \\ \boldsymbol{A}_*^T \boldsymbol{T}_* \boldsymbol{A}_* & \boldsymbol{A}_*^T \boldsymbol{T}_*^2 \boldsymbol{A}_* \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{A}_*^T \\ \boldsymbol{A}_*^T \boldsymbol{T}_* \end{pmatrix}.$$

Using the formula [Bernstein, 2005]

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{X} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1} \boldsymbol{b} (\boldsymbol{X} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \boldsymbol{C}\boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B} (\boldsymbol{X} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \\ -(\boldsymbol{X} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \boldsymbol{C}\boldsymbol{A}^{-1} & (\boldsymbol{X} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \end{pmatrix},$$

the required inverse is equal to

$$\begin{pmatrix} \boldsymbol{Y}_{11} & \boldsymbol{Y}_{12} \\ \boldsymbol{Y}_{21} & \boldsymbol{Y}_{22} \end{pmatrix},$$

where

$$\boldsymbol{Y}_{11} = \left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1} + \left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{T}_*^2\boldsymbol{A}_* - \boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1},$$

$$\boldsymbol{Y}_{12} = -\left(\boldsymbol{A}_*^T\boldsymbol{T}_*^2\boldsymbol{A}_* - \boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1},$$

$$\boldsymbol{Y}_{21} = -\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{T}_*^2\boldsymbol{A}_* - \boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\right)^{-1},$$

and

$$\boldsymbol{Y}_{22} = \left(\boldsymbol{A}_*^T\boldsymbol{T}_*^2\boldsymbol{A}_* - \boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\right)^{-1}.$$

Letting

$$\boldsymbol{F} = \left(\boldsymbol{A}_*^T\boldsymbol{T}_*^2\boldsymbol{A}_* - \boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\right)^{-1} = \left(\boldsymbol{A}_*^T\boldsymbol{T}_*\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{T}_*\boldsymbol{A}\right)^{-1},$$

this becomes

$$\begin{pmatrix} \left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1} + \left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{F}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1} & -\boldsymbol{F}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1} \\ -\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{F} & \boldsymbol{F} \end{pmatrix},$$

and hence $\boldsymbol{H}$ is equal to

$$\begin{aligned}\boldsymbol{H} =& \boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T + \boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{F}\boldsymbol{A}*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T \\ & - \boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{F}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T - \boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{F}\boldsymbol{A}_*^T\boldsymbol{T}_* + \boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{F}\boldsymbol{A}_*^T\boldsymbol{T}_* \\ & = \boldsymbol{G} + \boldsymbol{G}\boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{F}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{G} - \boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{F}\boldsymbol{A}_*^T\boldsymbol{T}_*\boldsymbol{G} - \boldsymbol{G}\boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{F}\boldsymbol{A}_*^T\boldsymbol{T}_* + \boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{F}\boldsymbol{A}_*^T\boldsymbol{T}.\end{aligned}$$

Letting $\boldsymbol{K} = \boldsymbol{T}_*\boldsymbol{A}_*\boldsymbol{F}\boldsymbol{A}_*^T\boldsymbol{T}_*$, we get

$$\boldsymbol{H} = \boldsymbol{G} + \boldsymbol{G}\boldsymbol{K}\boldsymbol{G} - \boldsymbol{K}\boldsymbol{G} - \boldsymbol{G}\boldsymbol{K} + \boldsymbol{K}, \tag{A.1}$$

so

$$\begin{aligned}\boldsymbol{H}-\boldsymbol{G} &= \boldsymbol{GKG}-\boldsymbol{KG}-\boldsymbol{GK}+\boldsymbol{K}\\ &= (\boldsymbol{G}-\boldsymbol{I})\,\boldsymbol{KG}-(\boldsymbol{G}-\boldsymbol{I})\,\boldsymbol{K}\\ &= (\boldsymbol{G}-\boldsymbol{I})\,(\boldsymbol{KG}-\boldsymbol{K})\\ &= (\boldsymbol{G}-\boldsymbol{I})\,\boldsymbol{K}\,(\boldsymbol{G}-\boldsymbol{I})\\ &= (\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{K}\,(\boldsymbol{I}-\boldsymbol{G})\,.\end{aligned}$$

As $\boldsymbol{I}-\boldsymbol{G}$ is idempotent, we get,

$$((\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{K}\,(\boldsymbol{I}-\boldsymbol{G}))\,((\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{K}\,(\boldsymbol{I}-\boldsymbol{G})) = (\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{K}\,(\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{K}\,(\boldsymbol{I}-\boldsymbol{G})\,.$$

But

$$\begin{aligned}\boldsymbol{K}\,(\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{K} &= \left(\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{T}_*\,(\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{T}_*\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\right)\\ &\quad\times(\boldsymbol{I}-\boldsymbol{G})\left(\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{T}_*\,(\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{T}_*\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\right)\\ &= \left(\boldsymbol{T}_*\boldsymbol{A}_*\left(\boldsymbol{A}_*^T\boldsymbol{T}_*\,(\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{T}_*\boldsymbol{A}_*\right)^{-1}\boldsymbol{A}_*^T\boldsymbol{T}_*\right)\\ &= \boldsymbol{K},\end{aligned}$$

so we get

$$((\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{K}\,(\boldsymbol{I}-\boldsymbol{G}))^2 = (\boldsymbol{I}-\boldsymbol{G})\,\boldsymbol{K}\,(\boldsymbol{I}-\boldsymbol{G})\,,$$

and so $\boldsymbol{H}-\boldsymbol{G}$ is idempotent.

Alternatively, we can see this using

$$(\boldsymbol{H}-\boldsymbol{G})\,(\boldsymbol{H}-\boldsymbol{G}) = \boldsymbol{H}-\boldsymbol{HG}-\boldsymbol{GH}+\boldsymbol{G}.$$

Using Equation A.1, we get that

$$\boldsymbol{HG} = (\boldsymbol{G}+\boldsymbol{GKG}-\boldsymbol{KG}-\boldsymbol{GK}+\boldsymbol{K})\,\boldsymbol{G} = \boldsymbol{G}+\boldsymbol{GKG}-\boldsymbol{KG}-\boldsymbol{GKG}+\boldsymbol{KG} = \boldsymbol{G},$$

and

$$\boldsymbol{GH} = \boldsymbol{G}\,(\boldsymbol{G}+\boldsymbol{GKG}-\boldsymbol{KG}-\boldsymbol{GK}+\boldsymbol{K}) = \boldsymbol{G}+\boldsymbol{GKG}-\boldsymbol{GKG}-\boldsymbol{GK}+\boldsymbol{GK} = \boldsymbol{G}.$$

So

$$(\boldsymbol{H}-\boldsymbol{G})\,(\boldsymbol{H}-\boldsymbol{G}) = \boldsymbol{H}-\boldsymbol{G}-\boldsymbol{G}+\boldsymbol{G} = \boldsymbol{H}-\boldsymbol{G},$$

so $\boldsymbol{H}-\boldsymbol{G}$ is idempotent.

As $\boldsymbol{H}$ is of rank $2pr$ and $\boldsymbol{G}$ is of rank pr, $\boldsymbol{H}-\boldsymbol{G}$ must have rank pr, as

$$rank\left(\boldsymbol{H}-\boldsymbol{G}\right)=trace\left(\boldsymbol{H}-\boldsymbol{G}\right)=trace\left(\boldsymbol{H}\right)-trace\left(\boldsymbol{G}\right)=rank\left(\boldsymbol{H}\right)-rank\left(\boldsymbol{G}\right).$$

Also, using $\boldsymbol{x}=\boldsymbol{A}_{*}\boldsymbol{b}_{0}+\boldsymbol{z}$, where $z\sim N\left(\boldsymbol{0},\sigma^{2}\boldsymbol{I}\right)$ under H_0, we get

$$\begin{aligned}\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{x} &= \left(\boldsymbol{I}-\boldsymbol{A}_{*}\left(\boldsymbol{A}_{*}^{T}\boldsymbol{A}_{*}\right)^{-1}\boldsymbol{A}_{*}^{T}\right)\left(\boldsymbol{A}_{*}\boldsymbol{b}_{0}+\boldsymbol{z}\right)\\ &= \left(\boldsymbol{A}_{*}\boldsymbol{b}_{0}+\boldsymbol{z}-\boldsymbol{A}_{*}\boldsymbol{b}_{0}-\boldsymbol{A}_{*}\left(\boldsymbol{A}_{*}^{T}\boldsymbol{A}_{*}\right)^{-1}\boldsymbol{A}_{*}^{T}\boldsymbol{z}\right)\\ &= \left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{z},\end{aligned}$$

so $\left(\boldsymbol{H}-\boldsymbol{G}\right)\boldsymbol{x}=\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{K}\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{x}=\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{K}\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{z}=\left(\boldsymbol{H}-\boldsymbol{G}\right)\boldsymbol{z}$. Therefore,

$$\boldsymbol{x}^{T}\left(\boldsymbol{H}-\boldsymbol{G}\right)\boldsymbol{x}=\boldsymbol{z}^{T}\left(\boldsymbol{H}-\boldsymbol{G}\right)\boldsymbol{z}=\boldsymbol{z}^{T}\boldsymbol{W}\boldsymbol{W}^{T}\boldsymbol{z},$$

where $\boldsymbol{W}$ is an $np\times rp$ matrix with orthonormal columns. So, under H_0,

$$-2\log\Lambda=\frac{1}{\sigma^{2}}\boldsymbol{x}^{T}\left(\boldsymbol{H}-\boldsymbol{G}\right)\boldsymbol{x}\sim\chi_{rp}^{2}.$$

A.5 Case 4: $\boldsymbol{A}$ fixed; $\boldsymbol{B}_0$ and σ^2 unknown

From the previous section, we know that

$$\frac{1}{\sigma^{2}}\left(\sum_{j=1}^{np}\left(x_{j}-\left(\boldsymbol{A}_{*}\boldsymbol{b}\right)_{j}\right)^{2}-\sum_{j}\left(x_{j=1}^{np}-\left(\left(\boldsymbol{A}_{*}\ \boldsymbol{T}_{*}\boldsymbol{A}_{*}\right)\boldsymbol{b}\right)_{j}\right)^{2}\right)=\frac{1}{\sigma^{2}}\boldsymbol{x}^{T}\left(\boldsymbol{H}-\boldsymbol{G}\right)\boldsymbol{x}\sim\chi_{rp}^{2},$$

and that

$$\frac{1}{\sigma^{2}}\sum_{j=q}^{np}\left(x_{j}-\left(\left(\boldsymbol{A}_{*}\ \boldsymbol{T}_{*}\boldsymbol{A}_{*}\right)\boldsymbol{b}\right)_{j}\right)^{2}=\frac{1}{\sigma^{2}}\boldsymbol{x}^{T}\left(\boldsymbol{I}-\boldsymbol{H}\right)\boldsymbol{x},$$

where $\boldsymbol{H}$ is an idempotent matrix of rank $2rp$. Since $\boldsymbol{H}$ is idempotent, $\boldsymbol{I}-\boldsymbol{H}$ is an idempotent matrix of rank $rank(I)-rank(H)=np-2rp$. Substituting $\boldsymbol{H}=\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{K}\left(\boldsymbol{I}-\boldsymbol{G}\right)+\boldsymbol{G}$ and using $\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{x}=\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{z}$ gives:

$$\begin{aligned}\left(\boldsymbol{I}-\boldsymbol{H}\right)\boldsymbol{x} &= \left(\boldsymbol{I}-\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{K}\left(\boldsymbol{I}-\boldsymbol{G}\right)-\boldsymbol{G}\right)\boldsymbol{x}\\ &= \left(\boldsymbol{I}-\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{K}\right)\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{x}\\ &= \left(\boldsymbol{I}-\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{K}\right)\left(\boldsymbol{I}-\boldsymbol{G}\right)\boldsymbol{z}\\ &= \left(\boldsymbol{I}-\boldsymbol{H}\right)\boldsymbol{z},\end{aligned}$$

so

$$(\boldsymbol{I}-\boldsymbol{H})\,\boldsymbol{x} \sim N_{np-2rp}\left(\boldsymbol{0}, \sigma^2\boldsymbol{I}\right),$$

and so

$$\frac{1}{\sigma^2}\boldsymbol{x}^T\,(\boldsymbol{I}-\boldsymbol{H})\,\boldsymbol{x} \sim \chi^2_{np-2rp}.$$

Also

$$\begin{aligned}
\operatorname{Cov}\left((\boldsymbol{H}-\boldsymbol{G})\,\boldsymbol{x}, (\boldsymbol{I}-\boldsymbol{H})\,\boldsymbol{x}\right) &= (\boldsymbol{H}-\boldsymbol{G})\,\sigma^2\boldsymbol{I}\,(\boldsymbol{I}-\boldsymbol{H}) \\
&= \sigma^2\,(\boldsymbol{H}-\boldsymbol{G})\,(\boldsymbol{I}-\boldsymbol{H}) \\
&= \sigma^2\left(\boldsymbol{H}-\boldsymbol{H}^2-\boldsymbol{G}+\boldsymbol{G}\boldsymbol{H}\right) \\
&= \boldsymbol{0},
\end{aligned}$$

as $\boldsymbol{H}^2 = \boldsymbol{H}$ and $\boldsymbol{G}\boldsymbol{H} = \boldsymbol{G}$, as shown previously. So the two distributions are independent, and thus,

$$F = \frac{\boldsymbol{x}^T\,(\boldsymbol{H}-\boldsymbol{G})\,\boldsymbol{x}/rp}{\boldsymbol{x}^T\,(\boldsymbol{I}-\boldsymbol{H})\,\boldsymbol{x}/(np-2rp)} \sim F_{rp,np-2rp}.$$

Appendix B

Derivation of majorizing functions for MDS

B.1 Euclidean distance

In this section we show that

$$g(\boldsymbol{B}, \tilde{\boldsymbol{B}}) = \eta_\delta^2 + \operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{Y} \boldsymbol{B}\right) - 2\operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{H}(\tilde{\boldsymbol{B}})\tilde{\boldsymbol{B}}\right)$$

is a majorizing function of the objective

$$L_{MDS}(\boldsymbol{B}) = \sum_{i<j} w_{ij}(\delta_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||)^2.$$

Write

$$\begin{aligned} L_{MDS}(\boldsymbol{B}) &= \sum_{i<j} w_{ij} \left(\delta_{ij} - ||\boldsymbol{b}_i - \boldsymbol{b}_j||\right)^2 \\ &= \sum_{i<j} w_{ij}\delta_{ij}^2 + \sum_{i<j} w_{ij}||\boldsymbol{b}_i - \boldsymbol{b}_j||^2 - 2\sum_{i<j} w_{ij}\delta_{ij}||\boldsymbol{b}_i - \boldsymbol{b}_j|| \\ &= \eta_\delta^2 + \eta^2(\boldsymbol{B}) - 2\rho(\boldsymbol{B}), \end{aligned}$$

where $\eta_\delta^2 = \sum_{i<j} w_{ij}\delta_{ij}^2$, $\eta^2(\boldsymbol{B}) = \sum_{i<j} w_{ij}||\boldsymbol{b}_i - \boldsymbol{b}_j||^2$, and $\rho(\boldsymbol{B}) = \sum_{i<j} w_{ij}\delta_{ij}||\boldsymbol{b}_i - \boldsymbol{b}_j||$.

To show that g is a majorizing function of L_{MDS}, we need to show that (i) $L_{MDS}(\boldsymbol{B}) \leq g(\boldsymbol{B}, \tilde{\boldsymbol{B}})$ for all $\boldsymbol{B}$ and $\tilde{\boldsymbol{B}}$; and (ii) $L_{MDS}(\tilde{\boldsymbol{B}}) = g(\tilde{\boldsymbol{B}}, \tilde{\boldsymbol{B}})$.

First, we rewrite the expressions $\eta^2(\boldsymbol{B})$ and $\rho(\boldsymbol{B})$ to get them in a more convenient form. Since η_δ^2 is constant in $\boldsymbol{B}$, we can ignore it for now.

For $\eta^2(\boldsymbol{B})$, we note that for each (i, j) pair, we can rewrite $||\boldsymbol{b}_i - \boldsymbol{b}_j||^2$ as

$$||\boldsymbol{b}_i - \boldsymbol{b}_j||^2 = \operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{Y}_{ij} \boldsymbol{B}\right),$$

where $\boldsymbol{Y}_{ij}$ is an $n \times n$ matrix with its iith and jjth elements equal to 1, its ijth and jith elements equal to -1, and all other entries equal to 0. Thus we get

$$\begin{aligned}
\eta^2(\boldsymbol{B}) = \sum_{i<j} w_{ij} ||\boldsymbol{b}_i - \boldsymbol{b}_j||^2 &= \sum_{i<j} w_{ij} \operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{Y}_{ij} \boldsymbol{B}\right) \\
&= \sum_{i<j} \operatorname{tr}\left(\boldsymbol{B}^T \left(w_{ij} \boldsymbol{Y}_{ij}\right) \boldsymbol{B}\right) \\
&= \operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{Y} \boldsymbol{B}\right),
\end{aligned}$$

where $\boldsymbol{Y} = \sum_{i<j} w_{ij} \boldsymbol{Y}_{ij}$, and hence has elements

$$y_{ij} = \begin{cases} \sum_{k=1, k\neq i}^{n} w_{ik} & \text{if } i = j, \\ -w_{ij} & \text{otherwise.} \end{cases}$$

Hence $\eta^2(\boldsymbol{B})$ is a quadratic function of $\boldsymbol{B}$.

For $\rho(\boldsymbol{B})$, we use the Cauchy-Schwarz inequality

$$\sum_{k=1}^{r} p_k q_k \leq \left(\sum_{k=1}^{r} p_k^2\right)^{1/2} \left(\sum_{k=1}^{r} q_k^2\right)^{1/2}.$$

Substituting p_k and q_k with $(b_{ik} - b_{jk})$ and $(\tilde{b}_{ik} - \tilde{b}_{jk})$ gives

$$\begin{aligned}
\sum_{k=1}^{r} (b_{ik} - b_{jk})(\tilde{b}_{ik} - \tilde{b}_{jk}) &\leq \left(\sum_{k=1}^{r} (b_{ik} - b_{jk})^2\right)^{1/2} \left(\sum_{k=1}^{r} \left(\tilde{b}_{ik} - \tilde{b}_{jk}\right)^2\right)^{1/2} \\
&= ||\boldsymbol{b}_i - \boldsymbol{b}_j|| \cdot ||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j||.
\end{aligned}$$

If $||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| \neq 0$, we can rearrange this to get

$$-||\boldsymbol{b}_i - \boldsymbol{b}_j|| \leq -\frac{\sum_{k=1}^{r} (b_{ik} - b_{jk}) \left(\tilde{b}_{ik} - \tilde{b}_{jk}\right)}{||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j||}.$$

If $||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| = 0$, we cannot do this since the right-hand side of the inequality will be undefined. However, since $||\boldsymbol{b}_i - \boldsymbol{b}_j||$ must be non-negative, we know that $-||\boldsymbol{b}_i - \boldsymbol{b}_j|| \leq 0$. Defining $\boldsymbol{Y}_{ij}$ as before, we can write

$$\sum_{k=1}^{r} (b_{ik} - b_{jk}) \left(\tilde{b}_{ik} - \tilde{b}_{jk}\right) = \operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{Y}_{ij} \tilde{\boldsymbol{B}}\right),$$

and combining the previous two equations gives

$$\begin{aligned} -\rho(\boldsymbol{B}) = -\sum_{i<j} w_{ij}\delta_{ij}||\boldsymbol{b}_i - \boldsymbol{b}_j|| &\leq -\sum_{i<j} \operatorname{tr}\left(\boldsymbol{B}^T \left(\frac{w_{ij}\delta_{ij}}{||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j||}\boldsymbol{Y}_{ij}\right)\tilde{\boldsymbol{B}}\right) \\ &= -\operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{H}(\tilde{\boldsymbol{B}})\tilde{\boldsymbol{B}}\right), \end{aligned}$$

where the ijth element of $\boldsymbol{H}(\tilde{\boldsymbol{B}})$ is

$$h_{ij} = \begin{cases} -\frac{w_{ij}\delta_{ij}}{||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j||} & \text{if } i \neq j \text{ and } ||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| \neq 0, \\ 0 & \text{if } i \neq j \text{ and } ||\tilde{\boldsymbol{b}}_i - \tilde{\boldsymbol{b}}_j|| = 0, \\ -\sum_{k=1, k\neq i}^{n} b_{ik} & \text{if } i = j. \end{cases}$$

Thus, we get

$$-\rho(\boldsymbol{B}) = -\operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{H}(\boldsymbol{B})\boldsymbol{B}\right) \leq -\operatorname{tr}\left(\boldsymbol{B}^T \boldsymbol{H}(\tilde{\boldsymbol{B}})\tilde{\boldsymbol{B}}\right),$$

with equality if $\boldsymbol{B} = \tilde{\boldsymbol{B}}$; and so

$$L_{MDS}(\boldsymbol{B}) = \eta_\delta^2 + \eta^2(\boldsymbol{B}) - 2\rho(\boldsymbol{B}) \leq g(\boldsymbol{B}, \tilde{\boldsymbol{B}}),$$

with equality if $\boldsymbol{B} = \tilde{\boldsymbol{B}}$.

B.2 Inner product

In this section we outline a proof (from [Groenen et al., 2003]) that

$$g\left(\boldsymbol{B}, \tilde{\boldsymbol{B}}\right) = m \sum_{i,j} \left(r_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 + w_{ij}\left(1 - \frac{w_{ij}}{m}\right)\left(d_{ij} - \tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j\right)^2$$

is a majorizing function of

$$L_{MDS}\left(\boldsymbol{B}\right) = \sum_{i,j} w_{ij}\left(\delta_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2. \tag{B.1}$$

Let $e_{ij} = \delta_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j$. Let $\tilde{e}_{ij} = \delta_{ij} - \tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j$ for some $\tilde{\boldsymbol{B}}$. Let $m = \max_{i,j}\{w_{ij}\}$. Then,

$$w_{ij} \left(e_{ij} - \tilde{e}_{ij}\right)^2 \leq m \left(e_{ij} - \tilde{e}_{ij}\right)^2,$$

and so

$$w_{ij} e_{ij}^2 - 2 w_{ij} e_{ij} \tilde{e}_{ij} + w_{ij} \tilde{e}_{ij}^2 \leq m e_{ij}^2 - 2 m e_{ij} \tilde{e}_{ij} + m \tilde{e}_{ij}^2.$$

Thus,

$$\begin{aligned}
w_{ij} e_{ij}^2 &\leq m e_{ij}^2 - 2 m e_{ij} \tilde{e}_{ij} + 2 w_{ij} e_{ij} \tilde{e}_{ij} - w_{ij} \tilde{e}_{ij}^2 + m \tilde{e}_{ij}^2 \\
&= m e_{ij}^2 - 2 m e_{ij} \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij} + 2m \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}^2 \\
&= m \left(e_{ij}^2 - 2 e_{ij} \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}\right) + m \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}^2 \\
&= m \left(e_{ij} - \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}\right)^2 - m \left(1 - \frac{w_{ij}}{m}\right)^2 \tilde{e}_{ij}^2 + m \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}^2 \\
&= m \left(e_{ij} - \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}\right)^2 + w_{ij} \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}^2.
\end{aligned}$$

Then,

$$\begin{aligned}
\left(e_{ij} - \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}\right)^2 &= \left(\left(\delta_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right) - \left(1 - \frac{w_{ij}}{m}\right) \left(\delta_{ij} - \tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j\right)\right)^2 \\
&= \left(\delta_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j - \delta_{ij} + \tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j + \frac{w_{ij}}{m} \delta_{ij} - \frac{w_{ij}}{m} \tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j\right)^2 \\
&= \left(\left(1 - \frac{w_{ij}}{m}\right) \tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j + \frac{w_{ij}}{m} \delta_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 \\
&= \left(r_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2,
\end{aligned}$$

where $r_{ij} = \left(1 - \frac{w_{ij}}{m}\right) \tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j + \frac{w_{ij}}{m} \delta_{ij}$.

Thus,

$$\begin{aligned}
\sum_{i,j} w_{ij} \left(\delta_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 &= \sum_{i,j} w_{ij} e_{ij}^2 \\
&\leq m \sum_{i,j} \left(r_{ij} - \boldsymbol{b}_i^T \boldsymbol{b}_j\right)^2 + w_{ij} \left(1 - \frac{w_{ij}}{m}\right) \left(d_{ij} - \tilde{\boldsymbol{b}}_i^T \tilde{\boldsymbol{b}}_j\right)^2 \\
&= g\left(\boldsymbol{B}, \tilde{\boldsymbol{B}}\right).
\end{aligned}$$

The second part of $g\left(\boldsymbol{B}, \tilde{\boldsymbol{B}}\right)$ is constant in $\boldsymbol{B}$, depending only on $\tilde{\boldsymbol{B}}$. The first part is an unweighted least-squares problem, solvable via SVD. Also, if $\boldsymbol{B} = \tilde{\boldsymbol{B}}$, and

hence $e_{ij} = \tilde{e}_{ij}$ for all i, j,

$$\begin{aligned}
g\left(\tilde{\boldsymbol{B}}, \tilde{\boldsymbol{B}}\right) &= m \sum_{i,j} \left(r_{ij} - \boldsymbol{z}_i^T \boldsymbol{z}_j\right)^2 + w_{ij} \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}^2 \\
&= \sum_{i,j} m \left(\tilde{e}_{ij} - \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}\right)^2 + w_{ij} \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}^2 \\
&= \sum_{i,j} m \left(\frac{w_{ij}}{m}\right)^2 \tilde{e}_{ij}^2 + w_{ij} \left(1 - \frac{w_{ij}}{m}\right) \tilde{e}_{ij}^2 \\
&= \sum_{i,j} \left(\frac{w_{ij}^2}{m} + w_{ij} - \frac{w_{ij}^2}{m}\right) \tilde{e}_{ij}^2 \\
&= \sum_{i,j} w_{ij} \tilde{e}_{ij}^2 \\
&= L_{MDS}\left(\tilde{\boldsymbol{B}}\right),
\end{aligned}$$

so $g\left(\boldsymbol{B}, \tilde{\boldsymbol{B}}\right)$ is a majorizing function of $L_{MDS}\left(\boldsymbol{B}\right)$.