MARTIJN KAGIE

Advances in Online Shopping Interfaces

Product Catalog Maps and Recommender Systems

Advances in Online Shopping Interfaces: Product Catalog Maps and Recommender Systems

Advances in Online Shopping Interfaces: Product Catalog Maps and Recommender Systems

Nieuwe methoden voor online winkelomgevingen:
Productcataloguskaarten en aanbevelingssystemen

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Erasmus Universiteit Rotterdam
op gezag van de rector magnificus

Prof.dr. H.G. Schmidt

en volgens besluit van het College voor Promoties.

De openbare verdediging zal plaatsvinden op

woensdag 19 mei 2009 om 13.30 uur

door

MARTIJN KAGIE
geboren te Voorburg.

Promotiecommissie

Promotor:	Prof.dr. P.J.F. Groenen
Overige leden:	Prof.dr.ir. R. Dekker
	Prof.dr.ir. U. Kaymak
	Prof.dr. D. van den Poel
Co-promotor:	Dr. M.C. van Wezel

Erasmus Research Institute of Management - ERIM
Rotterdam School of Management (RSM)
Erasmus School of Economics (ESE)
Erasmus University Rotterdam
Internet: http://www.erim.eur.nl

ERIM Electronic Series Portal: http://hdl.handle.net/1765/1

ERIM PhD Series Research in Management, 195
Reference number ERIM: EPS-2010-195-MKT
ISBN 978-90-5892-233-5

Design: B&T Ontwerp en advies `www.b-en-t.nl`
Print: Haveka `www.haveka.nl`

Preface

Many prefaces of doctoral dissertations start by saying or stressing how hard it is and how much effort it takes to write it. This will not be such a preface, since I really enjoyed the four years writing it. Many people are responsible for this and this is a proper place to them.

First of all, I want to thank my promotor Patrick Groenen and co-promotor Michiel van Wezel. Their enthusiasm is one of the main drivers behind the research presented in this thesis. I cannot remember any meeting which I left without a positive feeling and a handful of new ideas for further improvements. These ideas led to three chapters in this thesis, while Patrick also contributed to Chapter 6 and Michiel to Chapter 5. Furthermore, I want to thank Matthijs van der Loos and Michael Trosset for coauthoring Chapter 5 and Chapter 6 respectively. I want also thank Arjen van Zon and Jeroen Korving from Vergelijk.nl for their cooperation. The data and support from Vergelijk.nl were essential for the research presented in Chapters 2 and 4. This thesis also highly benefited from the support of ERIM. ERIM gave me the ability to use the Erasmus Behavioural Lab for the usability study presented in Chapter 3 and to present my work at international conferences in the Netherlands, Turkey, Germany, Spain, the United Kingdom, and Switzerland.

Also, I would like to thank my fellow PhD students for making the last four years such an enjoyable time. The discussions during the coffee and lunch breaks about research, politics, "verdorvenheid", or life in general were always moments to look forward to. Also, events such as the Koehandel games, the Cinerama movie day, Joost's Valentine Bash, and the ERIM survival day I will never forget. A special thanks goes to my paranymphs Nees Jan van Eck and Ludo Waltman for their friendship. I really liked the gossiping about the computer science department during the cola breaks, our victories in the Bierloop, and evenings during the SIKS courses in Vught.

Of course, I also want to thank my family for all their love and support during the last four years, but also during all those years before. Vincent, thank you for exploring half Europe and visiting obscure movies and concerts with me. Mom, thank you for always being there for me. Dad, thank you for always being proud on me. It is such a sorrow that you won't be there at my defense.

Finally, I want to thank my wife Marloes for coming in my life more than two years ago. Marloes, thank you for loving me just as I am.

Martijn Kagie
The Hague, February 2010

Contents

Preface **v**

1 Introduction **1**
1.1 Product Catalog Maps . . . 3
1.2 Recommender Systems . . . 5
1.3 Overview of this Thesis . . . 8

I Product Catalog Maps **11**

2 Map Based Visualization of Product Catalogs **13**
2.1 Introduction . . . 13
2.2 Methods for Map Based Visualization . . . 14
2.2.1 Self-Organizing Maps . . . 15
2.2.2 Treemaps . . . 16
2.2.3 Multidimensional Scaling . . . 18
2.2.4 Nonlinear Principal Components Analysis . . . 20
2.3 Product Catalog Maps . . . 20
2.3.1 Multidimensional Scaling . . . 20
2.3.2 Nonlinear Principal Components Analysis . . . 23
2.4 Determining Attribute Weights using Clickstream Analysis . . . 25
2.4.1 Poisson Regression Model . . . 25
2.4.2 Handling Missing Values . . . 26
2.4.3 Choosing Weights Using Poisson Regression . . . 26
2.4.4 Stepwise Poisson Regression Model . . . 27
2.5 E-Commerce Applications . . . 27
2.5.1 MDS Based Product Catalog Map Using Attribute Weights . . . 28
2.5.2 NL-PCA Based Product Catalog Map . . . 32
2.6 Summary, Conclusions, and Outlook . . . 34

3 A Graphical Shopping Interface Based on Product Attributes **37**
3.1 Introduction . . . 37

3.2 Related Work . . . 38
3.3 Methodology . . . 40
3.3.1 Multidimensional Scaling . . . 42
3.4 Graphical Recommender System . . . 42
3.5 Graphical Shopping Interface . . . 43
3.6 A Prototype Application to MP3 Players . . . 46
3.7 Evaluation of the Graphical Shopping Interface . . . 50
3.8 Conclusions and Discussion . . . 56

II Advances in Recommender Systems 59

4 Determination of Attribute Weights for Recommender Systems 61
4.1 Introduction . . . 61
4.2 The Use of Dissimilarities in Recommender Systems . . . 63
4.3 Determining Attribute Weights . . . 65
4.3.1 Poisson Loglikelihood and Deviance . . . 65
4.3.2 Poisson Regression . . . 66
4.3.3 Boosting Poisson Deviance . . . 67
4.4 Determining Weights for Four Real-Life Product Catalogs . . . 70
4.4.1 Data . . . 70
4.4.2 Model Performance . . . 70
4.4.3 Determined Weights . . . 72
4.5 Clickstream Evaluation . . . 72
4.5.1 Mean Average Relative Co-Occurrence . . . 73
4.5.2 Evaluation Results . . . 75
4.6 User Experiment . . . 77
4.6.1 Online Survey . . . 77
4.6.2 Mean Average Relative Relevance . . . 78
4.6.3 Evaluation Results . . . 79
4.7 Summary, Conclusions & Discussion . . . 80
4.A Adapted Gower Coefficient . . . 82
4.B Derivation of PoissonTreeBoost . . . 83

5 Including Item Characteristics in pLSA for Collaborative Filtering 85
5.1 Introduction . . . 85
5.2 Recommender Systems . . . 87
5.2.1 Explanations in Recommender Systems . . . 88
5.3 Models . . . 89
5.3.1 Probabilistic Latent Semantic Analysis for Collaborative Filtering . . . 89
5.3.2 Clusterwise Linear Regression . . . 92
5.3.3 Latent Class Regression Recommender System . . . 94
5.4 Data . . . 97

5.5 Experiments & Results 98
5.5.1 Experiment Setup 98
5.5.2 Experiment Results 100
5.5.3 Cold Start Item Problem 102
5.5.4 Model Interpretation 103
5.6 Conclusions 106

III Visualization of Large Data Sets 109

6 Multidimensional Scaling for Large Data Sets 111
6.1 Introduction 111
6.2 MDS by Iterative Majorization 113
6.3 Sparse Data 115
6.3.1 Structured Data Design 117
6.3.2 Unconnected Subsets 117
6.4 Large Scale MDS 119
6.5 Choosing Dissimilarity Measures and Weights in Large Scale MDS 121
6.6 Applications 125
6.6.1 Swiss Roll Data 125
6.6.2 Thesaurus Data Set 126
6.7 Conclusions 128

7 Discussion 133
7.1 Summary and Discussion of the Main Findings 133
7.2 Directions for Future Research 135

Nederlandse Samenvatting (Summary in Dutch) 139

Bibliography 143

Author Index 155

Curriculum Vitae 161

Chapter 1

Introduction

Over the past two decades the internet has rapidly become an important medium to retrieve information, maintain social contacts, and to do online shopping. The latter has some important advantages over traditional shopping. Products are often cheaper on the internet because companies do not need to keep a physical store and there is more competition on the internet, since companies are not bounded by their physical location. Furthermore, internet companies sell a wider collection of products, also within a product category. Amazon.com, for instance, sells 2,200 different types of MP3 players, 7,200 different types of digital cameras, and over 25 million book titles. Also, consumers can buy items whenever they like, since they are not restricted by opening hours of the shops. Finally, consumers do not need to leave their seats to buy something, since they do not need to leave their home to go to a shop.

On the other hand, the current state of online shops still has two major disadvantages over 'real' shops. The first disadvantage is that products are often much harder to find than in traditional shops. Generally, internet stores provide two ways to search for a product. The first way is via a search option in which a search string provided by the user is (partly) matched to the name or description of some products. The second way is by selecting a specific category of products; the categories are often based on characteristics of the products. Both methods often provide a very long list of products that require much scrolling by the consumer to locate the product best suiting her needs.

In Figure 1.1, we show a screenshot of one of the largest internet stores, Amazon.com[1] to illustrate this. The top of this screenshot shows that we arrived at this page by searching for an 'iPod' using the search bar. At the left of the page there are several options to restrict the list of results further by, for example, selecting a certain type of product or brand. The main part of the webpage is the list of products. Only six products fit in a single screen, so the consumer needs to do much scrolling to navigate through this webpage.

Another disadvantage of online over traditional shops, is that there is no salesman to help you. Although websites nowadays present products with ample information, such as product characteristics, images, and user reviews, it is up to consumer to process this information. To assist in such situations, the research field of recommender systems has emerged. Recommender

[1] `http://www.amazon.com`

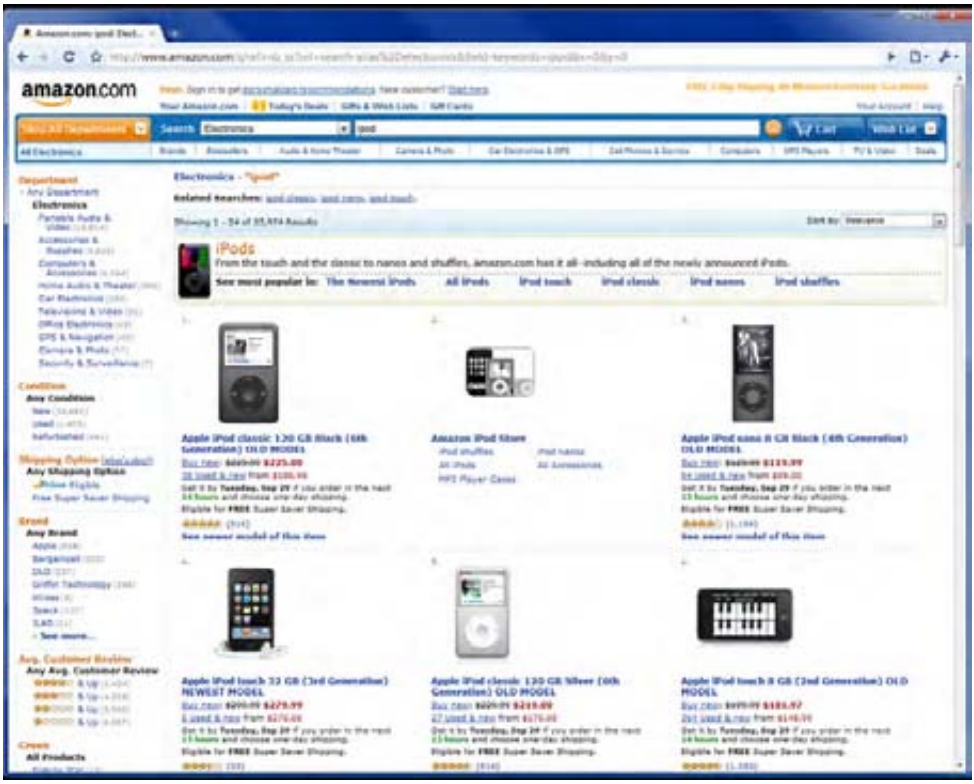


Figure 1.1: Screenshot of the Amazon.com website.

systems (Resnick & Varian, 1997) are systems that suggest products you might like and, hence, act as a substitute for a salesman in some way.

In this thesis, we will address both these disadvantages. In the first part, we will focus on the development of user interfaces that make it easier to users to find products. These interfaces are built on the idea to represent products in a two dimensional map instead of a list. These maps are constructed in such a way that similar products are located close to each other in the map. We also combine these map based representations with recommendation techniques. We provide working prototypes of these interfaces on `http://www.browsingmap.com`. In the second part of the thesis, we focus on recommender systems. First, we introduce a way to determine weights for case based recommendation based on product popularity. Furthermore, a new hybrid recommender system is proposed that is able to explain why a certain recommendation is given to a user. Finally, in the last part, we introduce an algorithm that can be used to create maps, such as the maps of products, on very large data sets.

In the remainder of this introduction, we will introduce the two key subjects of this thesis,

product catalog maps and recommender systems. Finally, we give an outline of the individual chapters.

1.1 Product Catalog Maps

There is a wide variety of methods that are able to create maps, that is, two dimensional representations, out of data. Since these methods essentially reduce the dimensionality of the data, these methods are generally known as dimension reduction techniques. The most well known dimension reduction techniques are principal components analysis (Jolliffe, 2002) and multidimensional scaling (MDS, see Borg & Groenen, 2005), although other methods exist of which some are discussed in Chapter 2.

MDS, the method heavily used in this thesis, differs from most of the other methods that it does not have a data matrix as input in which items are described by a number of attributes, but a square dissimilarity matrix. In this dissimilarity matrix elements represent the dissimilarities between items. It is possible to create this dissimilarity matrix from some data matrix, but this is not necessary as we show in an example later. The goal of MDS is to represent this dissimilarity matrix in some low dimensional Euclidean space as good as possible. In this thesis we will only consider the case that we want a two dimensional solution, that is, a map.

An example of a dissimilarity matrix is shown in Table 1.1. In this table, travel times (in minutes) between the largest ten cities in the European Union are shown[2]. We performed MDS with the dissimilarity matrix of Table 1.1 as input to make a map based on these travel times by train. This map is shown in Figure 1.2. As can been seen in the plot cities are not on their geographical location. The reason for this is that travel times do not map one-to-one to geometric distances, e.g., since some trajectories consist of high speed train connections and tracks do not always follow a straight line. The main conclusion that can be derived from the plot is that travel times west of Berlin are relatively much shorter than to the east. The fast ICE and TGV trains are important contributors to explain this effect.

This example illustrates there is much freedom in choosing the dissimilarity measure. In this thesis, we will focus on recommender systems that use MDS to visualize their recommendations. In such a case we need to have a dissimilarity measure that measures dissimilarity between pairs of products. When developing a dissimilarity measure between products, the dissimilarity measure should have some specific properties. These properties are not the same for the whole range of products, but are the same for the group of products, consumer electronics, considered in this thesis. This kind of products are generally described by a relatively large number of numerical and categorical attributes (say between 20 and 50). Also, the data is often incomplete. That is, there are quite a lot missing values, partially due to the fact that different manufacturers provide a different set of attributes. Finally, users do not consider all attributes to be equally important and, therefore, attributes should be weighted differently.

In Chapters 2, 3, and 4, we use a dissimilarity measure that has all these specific properties

[2]According to `http://en.wikipedia.org/wiki/Largest_cities_of_the_European_Union_by_population_within_city_limits`, accessed at October 1, 2009.

Table 1.1: Travel times by train between the 10 largest cities in the European Union

	Berl.	Madr.	Rome	Paris	Buch.	Hamb.	Buda.	Wars.	Vien.	Barc.
Berlin	0									
Madrid	1507	0								
Rome	1051	1349	0							
Paris	501	840	714	0						
Bucharest	1584	2699	1956	1881	0					
Hamburg	96	1399	979	490	1578	0				
Budapest	717	1819	964	953	782	924	0			
Warsaw	351	1820	1255	1028	1500	494	667	0		
Vienna	567	1574	765	705	983	561	172	457	0	
Barcelona	1318	177	1386	550	2443	1282	1640	1741	1401	0

Figure 1.2: MDS solutions representing the ten largest cities in the European union based on travel times by train.

Table 1.2: Small product catalog of eight MP3 players

Name	Brand	Price	HDD	Memory Size	Weight
iPod Touch	Apple	391.27	no	16	120.0
Zen	Creative	199.50	no	8	43.5
SA6024	Philips	118.69	no	2	75.0
X20	iRiver	240.50	yes	8	71.0
iPod Classic	Apple	231.68	yes	80	158.0
iPod Shuffle	Apple	76.00	no	1	15.5
Zen Vision W	Creative	342.50	yes	30	276.0
Audiodiva Premium	Packard Bell	64.94	no	2	33.0

for both the creation of product catalog maps and in a recommendation setting. This dissimilarity measure is an adaptation of the General Coefficient of Similarity proposed by Gower (1971). We refer to these chapters for more details.

We illustrate the concept of product catalog maps using a small product catalog as example. Table 1.2 shows a small product catalog consisting of 8 MP3-players. In this example, we only use the five product attributes shown in this table to create the map. Two of these attributes are categorical (Brand and HDD) and the other three are numerical. Applying the dissimilarity measure discussed above to this product catalog results in the dissimilarity matrix shown in Table 1.3. When using this dissimilarity matrix as input for the MDS algorithm, we get the product catalog map shown in Figure 1.3.

In this map, similar products are located close to each other. The iPod Shuffle and Audiodiva Premium, for instance, are very close in this map. When we have a look at Table 1.3, we see that these MP3 players have the smallest dissimilarity, that is, a dissimilarity of 0.541. In Table 1.2 we can see that these two MP3-players are the cheapest and lightest in the product catalog. Also, they have no hard disk drive and only limited memory size.

In general, the MP3 players with more memory available are located at the top of the map. The MP3 players with a hard disk drive are positioned in the top right. Also, there is an ordering on brand visible in the map: the three Apple MP3 players are located on the left of the map and also the two Creative MP3 players are positioned at about the same horizontal position. In general, price and weight also increase like the memory size in the vertical direction of the map, although there is some error in the map on these attributes.

1.2 Recommender Systems

The most important feature of a recommender system is that it recommends items to a user. There are various types of recommender systems that vary on a number of dimensions. J. B. Schafer, Konstan, and Riedl (2001) distinguish six dimensions in their taxonomy for recommender systems.

Table 1.3: Dissimilarity matrix of the eight products of Table 1.2

	Touch	Zen	SA6024	X20	Classic	Shuffle	Vision	Audiodiva
iPod Touch	0.000							
Zen	0.849	0.000						
SA6024	0.902	0.676	0.000					
X20	0.990	0.849	0.911	0.000				
iPod Classic	1.078	1.200	1.238	0.988	0.000			
iPod Shuffle	0.879	0.719	0.652	1.003	1.221	0.000		
Zen Vision W	1.052	1.107	1.246	0.986	1.008	1.320	0.000	
Audiodiva Premium	0.988	0.699	0.629	0.988	1.302	0.541	1.309	0.000

1. **Targeted customer inputs**: Input from the current user. The input can be explicit such as a search query or specifying ratings for items or implicit like the user's viewing behavior. It is also possible that no input is required by the user, but then the recommender system is not able to provide personalized recommendations.

2. **Community input**: The data the system has collected of all users. This input can exist of global measures of product popularity, but also of user ratings, or text comments. Some systems do not use any community inputs, but use other kind of data sources.

3. **Output**: There exist three popular ways in which recommender systems provide their results. Many systems just give the user a single suggestion of which the system think that he will like most. Other systems provide a set of suggestions that may be ranked in some Top-n. Finally, some systems provide predictions of the rating the user will give to some item.

4. **Recommendation method**: The recommendation algorithm used to compute which items should be recommended. We will discuss this dimension in more detail later in this section.

5. **Degree of personalization**: J. B. Schafer et al. (2001) distinguish three degrees of personalization for recommender systems. The first class of recommender systems are non-personalized and provide the same recommendations to all users. The second class of systems only uses the current user input. Finally, the fully personalized recommender systems use user profiles to provide individual recommendations for users.

6. **Delivery**: This is the way how the system communicates its recommendation to users. This can be using a push strategy by providing the user unasked with recommendations by, for example, email, or a pull strategy in which the user has to go to some specific website to ask for a recommendation. Alternatively, the system can be fully integrated in the website of the company.

Note that choices in one dimension can restrict the options of another dimension. For example, when no community inputs are used, it is impossible to use some kinds of algorithms, such as

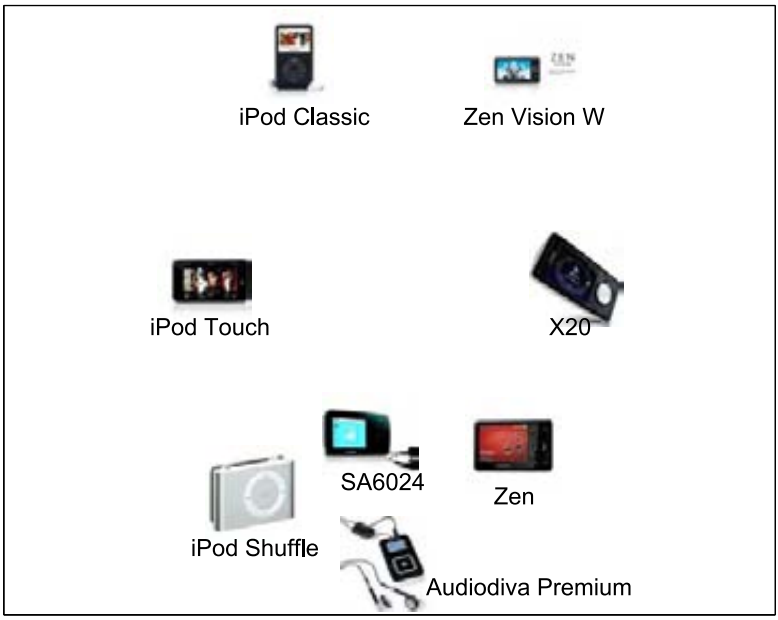


Figure 1.3: Product catalog map made using MDS based on the dissimilarity matrix in Table 1.3.

collaborative filtering algorithms which are discussed later, and restricting the customer inputs can affect the degree of personalization.

We now return to various types of recommendation methods (Dimension 4) that can be distinguished. According to Adomavicius and Tuzhilin (2005), there are three groups of algorithms that mainly differ in which kind of data they use to provide recommendations. Collaborative filtering (CF) algorithms recommend items that similar users liked in the past. A popular CF method is using Pearson correlation to determine which users have a similar rating pattern as the current user. The more similar a user is to the current user the more the rating of this user for the item to be rated is taken into account. The main advantage of CF based algorithms is that no information is needed about the product domain, such as specific characteristics of the items. Content-based recommender systems, on the other hand, do use these item characteristics to provide recommendations. In fact, they recommend items that are similar to items the user liked in the past. In some cases, the recommendation is only based on the item on the currently viewed or some search query. Then, some (dis)similarity measure is needed to compute which item is most similar to this current item. In the next section, we discuss this in somewhat more detail. The content-based approach has as an advantage over CF methods that it is able to recommend items that are new in the system and of which no rating data exists. The third group of recommender systems, the hybrid recommender systems integrate CF and content-based methods to

benefit from the advantages of both systems. In Chapter 5, we introduce a new type of hybrid recommender system.

1.3 Overview of this Thesis

This thesis consists of three parts. In Part I, we focus on the use of product catalog maps to improve online shopping interfaces. Part I comprises two chapters. Part II focuses on recommender systems and also comprise two chapters. Finally, Part III contains a single chapter concerning an algorithm for large scale visualization. Each chapter can be read independently of the other chapters. Three chapters are published or accepted for publication in the literature. The other two chapters are ready to be submitted. Below, we give a brief overview of each chapter.

In Chapter 2, we develop two shopping interfaces based on a map based representation of a product catalog. In these maps, products that are similar in terms of their characteristics should, in general, be located close to each other in the map. We use two different visualization techniques to create these interfaces. One is based on multidimensional scaling (MDS, see Borg & Groenen, 2005), while the second is based on nonlinear principal components analysis (NLPCA, see, e.g., Linting, Meulman, Groenen, & Van der Kooij, 2007). Furthermore, we introduce a method to estimate which characteristics of a product are considered important by users based on some measure of product popularity. This method can be used to set weights in the interfaces. We show applications of both interfaces on a real life product catalog of MP3 players. Chapter 2 is based on Kagie, Van Wezel, and Groenen (2010) and is accepted for publication in the *Recommender Systems Handbook*.

In Chapter 3, we develop a shopping interface that combines the idea of a map based representation of a product catalog as introduced in Chapter 2 with product recommendation. First we introduce a flexible dissimilarity measure that can be used as input for multidimensional scaling and for recommendation. Then, we introduce two map based recommender systems. The graphical recommender system (GRS) visualizes an ideal product specification together with some recommendations similar to this product specification in a map. The graphical shopping interface (GSI) combines the map based representations with the recommendation by proposing (Shimazu, 2002) principle, in which the user can navigate to a product of her liking in a limited number of steps. We show prototypes of these systems on a product catalog of MP3 players. We test both the map quality and the recommending by proposing algorithm in a simulation study. Also, we test the overall quality of the GSI in a usability study. Chapter 3 is based on Kagie, Van Wezel, and Groenen (2008b) and has been published in *Decision Support Systems*.

In Chapter 4, we go deeper into the issue of determining important product characteristics that has also been considered in Chapter 2. Two models are discussed to determine attribute importance based on some measure of product popularity, such as sales, based on the assumption that attributes that are able to distinguish popular products from non popular products are attributes that are considered to be important by users. The first approach is based on a Poisson regression model and the second approach is based on a novel boosting algorithm that is able to handle a Poisson deviance loss function. We use both models to determine weights in a dissimilarity measure and use this measure to recommend products to users. Both approaches are tested

using two novel evaluation approaches, in which the weighting approaches are compared to an equal weighted dissimilarity measure. The first approach is based on a clickstream analysis. We use four real life product catalogs (MP3 players, Digital Cameras, Notebooks, and Microwave Ovens) in this evaluation. The second approach is based on a user experiment for which we use the Microwave Oven product catalog. Chapter 4 is partly based on Kagie, Van Wezel, and Groenen (2008a) which has been published in the *Proceedings of the 2nd ACM Conference on Recommender Systems*.

In Chapter 5, we introduce a novel hybrid recommendation algorithm combining collaborative filtering and content-based recommendation. Basis of this system is probabilistic latent semantic analysis for collaborative filtering (pLSA-CF) developed by Hofmann (2004). Disadvantage of this approach is that item characteristics are not used by the model. We combine pLSA-CF with ideas from clusterwise linear regression (DeSarbo & Cron, 1988), such that we are able to integrate item characteristics in the model. This has two major advantages: first, we can use the regression coefficients estimated for these item characteristics to explain to the user why a certain recommendation is given; second, using this approach we are able recommend items that are (relatively) new in the system. Although the performance of our method is slightly worse on a movie data set than of the original pLSA-CF method, our method performs much better on relatively new items. Also, we show how our model can be used to explain recommendations on this data set. Chapter 5 is based on Kagie, Van der Loos, and Van Wezel (2009) and has been published in *AI Communications*.

In Chapter 6, we develop an algorithm to perform multidimensional scaling (MDS) on large sparse dissimilarity matrices. Ordinary MDS algorithms have a complexity that is quadratic in the number of items, while the algorithm we propose is linear in the number of nonmissing dissimilarities, which is much faster when the dissimilarity matrix is highly sparse. When the dissimilarity matrix is not sparse, we show how such a matrix can be made sparse using a specific design. Furthermore, we address two other issues that are important when handling large sparse dissimilarity matrices. The first issue is that sparse data may consist of a number of unconnected subsets, such that an MDS solution should be obtained for each subset. We show a simple algorithm to find these unconnected subsets. The second issue is that large scale MDS solutions tend to be dominated by the most frequent dissimilarity value. Therefore, it is essential to use a proper dissimilarity measure and weighting scheme and we show how this can be done. We use our large scale MDS algorithm in two applications. First, we use it to unroll a Swiss roll only using the smallest dissimilarities. The algorithm leads to a high quality solution, only in a fraction of the time original MDS needs. Second, we apply large scale MDS to a Thesaurus data set showing the similarity between 23,000 words. Chapter 6 is joint work with Patrick J.F. Groenen and Michael W. Trosset.

Part I

Product Catalog Maps

Chapter 2

Map Based Visualization of Product Catalogs*

2.1 Introduction

Traditionally, recommender systems present recommendations in ranked lists to the user. In content- and knowledge-based recommender systems, these lists are often sorted on some notion of similarity with a query, ideal product specification, or sample product. However, a lot of information is lost in this way, since two products with the same similarity to a query can differ from this query on a completely different set of product characteristics. When using a two dimensional map based visualization of the recommendations, it is possible to retain part of this information. In the map, we can then position recommendations that are similar to each other in the same area of the map.

A domain in which this map based approach can be very useful is electronic commerce, since electronic commerce stores sell a large number of products, often described by a large number of characteristics, but that are, in certain product categories, easily recognizable by an image of the product at hand. E-commerce domains for which this holds are, for example, consumer electronics and real estate. In the industry, one tries nowadays to improve usability of this kind of websites, using a trend called visual shopping. An example is CrispyShop.com[1], which compares products on one characteristic and price using a chart. Another approach taken in visual shopping is selecting items based on visual similarity (like color and shape), such as is done by Like.com[2] and Modista[3] which is especially useful in fashion related fields. An approach more closely related to the map based approach, is taken at BrowseGoods.com[4], where the products are shown

*This chapter is based on Kagie, Van Wezel, and Groenen (2010)

[1] http://www.crispyshop.com

[2] http://www.like.com

[3] http://www.modista.com

[4] http://www.browsegoods.com

in a map ordered as a department store. Both Musicovery[5] and LivePlasma[6] show a map of songs, where the latter also created a similar map for movies. Finally, YouTube[7] has an option called Warp!, which recommends movies similar to a movie you watched and shows them in a map.

Despite this wide list of commercial map based interfaces, these interfaces lack a, publicly available, scientific foundation. In this chapter, we discuss several issues in product catalog map interfaces which we have published earlier in Kagie, Van Wezel, and Groenen (2007, 2008a, 2008c). This chapter combines these issues and shows applications on a real e-commerce product catalog.

In the next section, we first review some scientific methods that can be used to create such maps that are used in the e-commerce domain or in related fields. Two of these methods, namely multidimensional scaling (MDS, see Borg & Groenen, 2005) and nonlinear principal components analysis (NL-PCA, see Gifi, 1990; Linting, Meulman, Groenen, & Van der Kooij, 2007; Michailidis & De Leeuw, 1998), are discussed in more detail in Section 2.3, where they are used in two different approaches to create a product catalog map interface. The first which is based on MDS uses a flexible dissimilarity measure between products to create the map. This approach has been published earlier in Kagie et al. (2007). The second approach based on NL-PCA and published earlier in Kagie et al. (2008c) has the advantage that it also shows points representing category values of attributes that can be used for navigation over the map.

One problem in both map based visualization and content-based recommendation is that different characteristics of products are not considered to be equally important by users. In Section 2.4, we introduce a way to determine attribute weights using clickstream data. We counted how often products were sold. Based on the assumption that attributes that have a high influence on sales of products are attributes that are considered to be important by users, we estimate a Poisson regression model (McCullagh & Nelder, 1989; Nelder & Wedderburn, 1972) and derive weights from that. This approach has earlier been described in Kagie et al. (2008a).

All these methods are applied to a new real commercial product catalog of MP3 players in Section 2.5. Prototypes of these methods are available at `http://www.browsingmap.com/mapvis.html`. This product catalog was provided by Compare Group, owner of the Dutch price comparison site `http://www.vergelijk.nl`. Finally, we will draw conclusions and give directions for future research in Section 2.6.

2.2 Methods for Map Based Visualization

The use of maps displaying areas with similar items has become increasingly popular in fields where users need to browse through relatively large collections of data, such as, web searching, image browsing, and managing music playlists. Also, map based visualization fits in the trend of visual shopping interfaces introduced in industry. In this section, we discuss four scientific methods for map based visualizations in the fields mentioned above, where we focus on the

[5] `http://www.musicovery.com`
[6] `http://www.liveplasma.com`
[7] `http://www.youtube.com`

Table 2.1: Example data set

	Name	Price	HDD	Memory Size (GB)	Weight
1.	Apple iPod Nano	180.55	0	8	40.0
2.	Apple iPod Video (30GB)	248.41	1	30	130.0
3.	Apple iPod Video (60GB)	73.50	1	60	156.0
4.	Apple iPod Video (80GB)	324.00	1	80	156.0
5.	Apple iPod Shuffle	76.00	0	1	15.5
6.	Creative Zen Nano Plus	76.00	0	1	22.0
7.	Creative Zen Vision M (30GB)	199.50	1	30	163.0
8.	Creative Zen Vision M (60GB)	250.50	1	60	178.0
9.	Sandisk Sansa e280	129.99	0	8	75.2
10.	Sony NW-A1200	143.90	1	8	109.0

visualization methods used and their advantages and disadvantages when used to browse through product catalogs. In this discussion, we specifically pay attention to the type of data normally contained in (commercial) product catalogs. These catalogs often contain numerical, categorical, and multi-valued categorical attributes. Also, there can be a lot of missing values, due to, for instance, different product specifications by different manufacturers. Note that, some of the visualization methods could also be used for visualizations in three dimensions. However, we think these 3D visualizations will be too complex for users and, therefore, we only discuss 2D visualizations.

To give somewhat more insight in the differences between the four visualization methods we discuss next, we apply them to a small product catalog. This product catalog, shown in Table 2.1, consists of ten popular MP3 players derived from the larger MP3 player product catalog we use later to show our applications on. Furthermore, we limited the number of attributes to four, which where all chosen to be numerical or binary, such that all four methods could handle them easily. Also, we selected products that did not have any missing values on these four attributes.

2.2.1 Self-Organizing Maps

In the field of web (search) visualization, Kohonen's Self-Organizing Map (SOM, see Kohonen, 1990, 1998, 2001) is among the most popular methods to create maps. Following the early work of H. Chen, Houston, Sewell, and Schatz (1998), SOMs have, for example, been used in applications by Chung (2008), Chung, Bonillas, Lain, Xi, and Chen (2006) and C. C. Yang, Chen, and Hong (2003). Ong, Chen, Sung, and Zhu (2005) use a SOM to create a map for online news and Van Gulik, Vignoli, and Van der Wetering (2004) for a music collection on an MP3 player.

Self-organizing maps use an unsupervised neural network to cluster and reduce dimensionality at the same time. First, a grid needs to be chosen that represents the structure of the map. Often, this is a rectangular or a hexagonal grid. Informally, the SOM training process works in the following way. All grid points (often called models, weights, or prototype vectors in SOM

literature) are initialized with their location on the grid and a vector in the original attribute space. Then, following an incremental-learning approach items are randomly shown to the SOM. For an item, we first compute which model represents the item best. This model is called the winner or best matching unit (BMU) and is often determined using Euclidean distance. Then, the model and neighbors of the model, to a lesser extent, are adapted to better fit the item at hand. This is done for all items and after that the next iteration is started. Each iteration, a seen item gets less influence on both the model (grid point) and its neighbors, such that the SOM converges to a solution. There also exists a batch-learning algorithm (Kohonen, 2001) for SOM which is faster.

The main advantage of SOM is that it generally provides nice clustered maps, in which clusters are formed by a set of neighboring grid points to which items are connected and relative many empty grid points. Also, neighboring models are similar to each other providing a similarity interpretation of the map. However, this similarity interpretation is not always valid for the complete map. Although a neighboring model is generally similar to another model, it can be that there exists a model even more similar in another part of the map.

Another disadvantage is that items have the same position in the map, when they are assigned to the same grid point (model), although this can be overcome by specifying a larger grid or making a hierarchical SOM, in which a SOM is created for each model in the original SOM. Also, the original SOM is not able to handle missing values or (multi-valued) categorical attributes, but with an adaptation of the comparison metric used to determine the BMU this is possible (Kohonen, 1998).

A SOM based on the example product catalog of Table 2.1 and shown in Figure 2.1 was created using the SOM Toolbox for Matlab (Alhoniemi, Himberg, Parviainen, & Vesanto, 1999) with a 10×10 rectangular grid. This grid is also shown in the map. As can be seen, each of the products is assigned to one of the squares (grid points). The complete map is used, that is, all corners have a product assigned to them, but there seem not to be real clusters of products. Looking at the ordering of the products on the map, the vertical dimension clearly corresponds to memory size and whether the MP3 player has a hard disk drive (HDD). The horizontal dimension captures the other two attributes, but these effects are not consistent over the map, due to the nonlinear character of SOMs.

2.2.2 Treemaps

Another popular visualization method which was also used to visualize web search results (Turetken & Sharda, 2004), but also to visualize the news in the commercial application NewsMap[8], is the treemap algorithm (Schneiderman, 1992).

A treemap uses a tree, for example, resulting from a hierarchical clustering as its input. Each leaf node represents a single item. All nodes in the tree are labeled by a weight value. For non-leaf nodes, this weight value is the sum of weights of its children. When all leaf nodes are given a weight of 1, the weight value of all nodes represents the cluster size. Alternatively, items can be labeled by some measure of popularity to make these items more important in the visualization.

[8] `http://marumushi.com/apps/newsmap/index.cfm`.

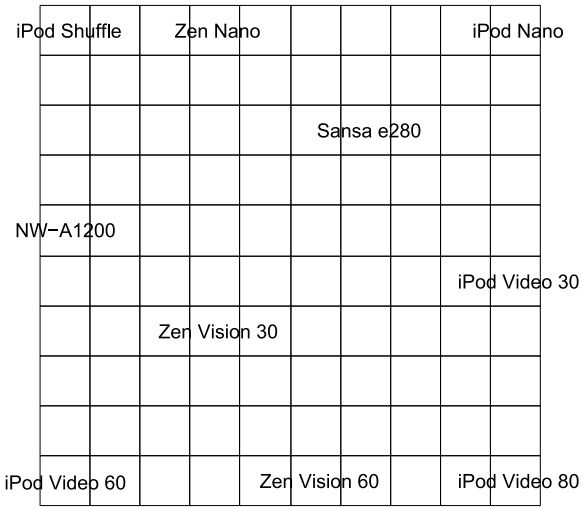


Figure 2.1: Example of a self-organizing map on the product catalog shown in Table 2.1.

Starting at the root node, we divide the map vertically into as many parts as the root has children. The size of the resulting rectangles is relative to the weight values of the nodes. Then, we divide each of the rectangles horizontally using the children of the corresponding node. Alternating vertical and horizontal division, we can continue in this way until the bottom of the tree is reached and each item has a rectangle with an area relative to its weight on the map.

A main advantage of treemaps is that the complete map is filled, there are no spaces without any items and there is no overlap between items in the map. Also, its hierarchical structure is very useful for implementing zooming functionality. Since treemaps can be used in combination with hierarchical clustering algorithms, which are based on a similarity measure, they are also very flexible. A similarity measure can be chosen that is very suitable to the data at hand. For instance, when visualizing a product catalog, one could use the dissimilarity measure we introduce in the next section, that is able to handle missing values and mixed attribute types.

The treemap approach has two drawbacks. First of all, the original treemap algorithm produces often tall rectangles as results for small clusters or single items, which makes visualization of single products quite hard. This problem can be partly overcome by using squarified treemaps (Bruls, Huizing, & Van Wijk, 2000) or, even better, quantum treemaps (Bederson, Schneiderman, & Wattenberg, 2002), which guarantee a certain aspect ratio for the rectangles representing the items. The second drawback is that, although similar products are clustered together, there is no clear distance interpretation and ordering between clusters might be lost, that is, two quite similar products that are assigned to two different clusters might not be close to each other in the map.

A treemap based on the example product catalog of Table 2.1 is depicted in Figure 2.2. To create this map we first used Matlab's average linkage hierarchical clustering algorithm based on

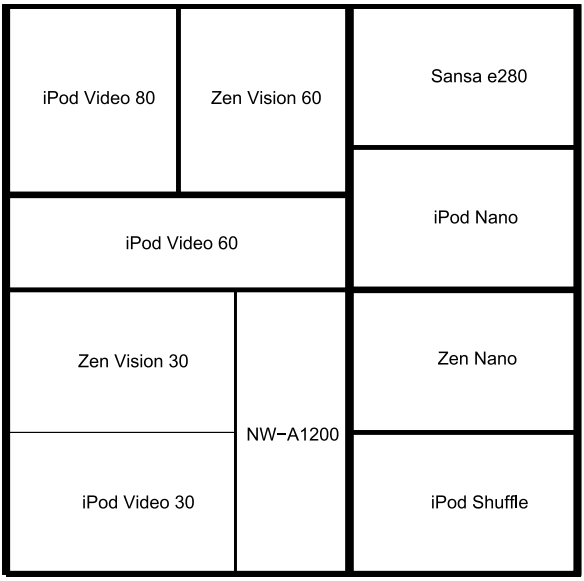


Figure 2.2: Example of a treemap on the product catalog shown in Table 2.1.

normalized Euclidean distances and visualized the resulting tree using a treemap algorithm for Matlab written by Hicklin (2007). In the map, wider lines correspond to higher level divisions of the data. Each rectangle, which all have the same area, represents a product. Compared to the SOM, the products that were at the top are on the right in the treemap. These products are the MP3 players without HDD. On the left side, we see now that the Apple iPod Video 60GB is closer to the Apple iPod Video 30GB, the Creative Zen Vision M 30GB, and the Sony NW-A1200 (since the line dividing the Apple iPod Video 80GB and the Creative Zen Vision M 60GB from the Apple iPod Video 60GB is wider than the line under it) than to the Apple iPod Video 80GB and the Creative Zen Vision M 60GB.

2.2.3 Multidimensional Scaling

A third class of applications uses multidimensional scaling (MDS, see Borg & Groenen, 2005) algorithms to visualize relative large data collections. There exists a wide variety of MDS algorithms, such as classical scaling (Torgerson, 1952; e.g. used to visualize music playlists by Donaldson, 2007) , nonmetric MDS (Kruskal, 1964), Sammon Mapping (Sammon, 1969; e.g. used to visualize a image collection by Pečenović, Do, Vetterli, & Pu, 2000), and SMACOF (De Leeuw, 1988; e.g. used to visualize a roller skates catalog by Stappers, Pasman, & Groenen, 2000). However, they are all based on mapping a (symmetric) dissimilarity matrix into low dimensional Euclidean space, such that distances between pairs of points represent the dissimi-

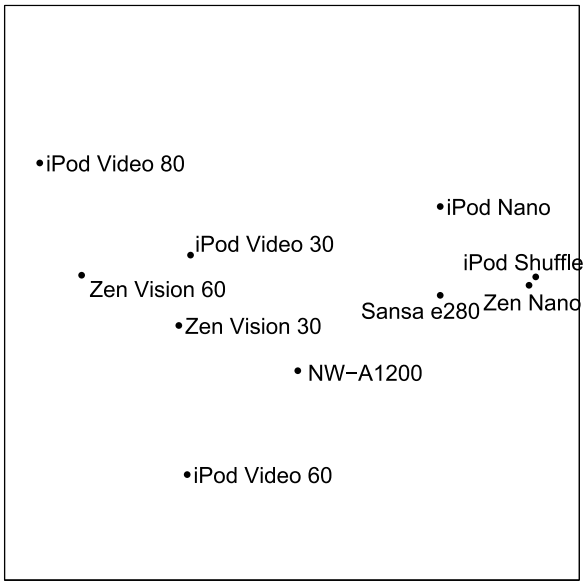


Figure 2.3: Example of a map made with multidimensional scaling on the product catalog shown in Table 2.1.

larities as closely as possible. MDS is one of the methods we use to create product catalog maps and is, therefore, in more detail described in the next section.

The main advantage of MDS is that the distances in the map really correspond to the similarity between items. Secondly, when using a flexible dissimilarity measure MDS can handle missing values and mixed attribute types. Disadvantages of MDS compared to SOMs and TreeMap are that there may be a lot of empty space in and/or around the map and very similar items may (partially) overlap each other in the map. Empty spaces are a disadvantage, since more zooming is necessary on specific parts of the map to be able to use the map in a satisfying way.

In Figure 2.3, a map based on the example product catalog made using MDS is shown. We created this map using PROXSCAL, available under SPSS Categories (Meulman & Heiser, 2007), which uses the SMACOF algorithm. The dissimilarity matrix used was computed based on normalized Euclidean distances. As was expected, the corners of the map are more empty compared to the SOM. However, the positions of the products on the map (after rotation) do not differ much. Nevertheless, the MDS configuration maps the dissimilarities better, but on the other hand this makes the map less organized. Note that it is, in most situations, impossible to map all dissimilarities perfectly in two dimensions. Therefore, the solution is a compromise in which some dissimilarities are mapped better than others.

2.2.4 Nonlinear Principal Components Analysis

Probably the most well-known method to perform dimension reduction and, thus, for creating maps is principal components analysis (PCA). PCA has a numerical data matrix as input and linearly transforms the data, such that the first dimension, normally plotted in horizontal direction, explains the variance in the data as much as possible. The next dimensions (in case of a map only the second) try to do the same for the remaining variance. Also, all dimensions are uncorrelated with each other. Nonlinear principal components analysis (NL-PCA, see Gifi, 1990; Linting et al., 2007; Michailidis & De Leeuw, 1998) is a method that does the same as PCA, but is also able to handle categorical attributes and missing values. Also, using ordinal transformation, numerical attributes can be transformed nonlinearly. Additionally, the categories of the attributes also have a location in the map, that is in the center of the items belonging to that category. These category points can be used to give an interpretation to the map, but also as a navigation tool in the map based interface as is done in our product catalog map using NL-PCA, which is introduced in the next section.

Figure 2.4 shows the NL-PCA configuration of the example product catalog which was made using CATPCA also available under SPSS Categories (Meulman & Heiser, 2007). All numerical attributes where treated as ordinal variables, such that nonlinear transformation was possible. However, due to the large flexibility of NL-PCA and the small data set, the resulting map has a lot of overlapping products. In fact, NL-PCA created clearly three clusters. Although the products in the clusters were also close together in the maps of the other three methods, they did not really form clusters. This example shows, in extreme, the ability of NL-PCA to cluster products having the same category value together.

2.3 Product Catalog Maps

In this section, we introduce two ways to create a product catalog map. We start by describing a product catalog map based on multidimensional scaling (MDS), which is combined with a k-means clustering algorithm to highlight some prototypical products. Thereafter, a product catalog map based on nonlinear principal components analysis (NL-PCA) is introduced which uses the category points to provide a navigation mechanism. These two methodologies were published earlier in Kagie et al. (2007) and Kagie et al. (2008c).

In the description of these methods (and in the remainder of this chapter), we use the following mathematical notation. A product catalog D contains I products $\mathbf{x}_i$ having K attributes $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$. The final two-dimensional map can be described by an $I \times 2$ matrix $\mathbf{Z}$ containing the coordinates of the products in the map.

2.3.1 Multidimensional Scaling

The first approach to create a product catalog map that we describe is based on MDS. As was mentioned in Section 2.2.3, the basis of an MDS map is a dissimilarity matrix. To compute a dissimilarity matrix $\mathbf{\Delta}$ from the product catalog, we need a dissimilarity measure. This measure

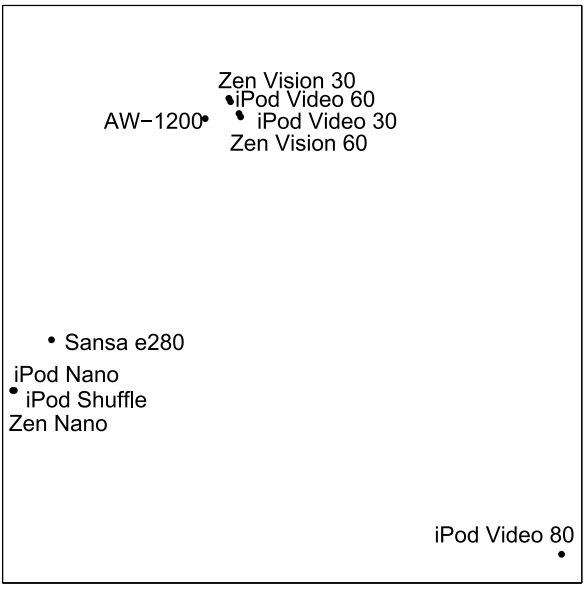


Figure 2.4: Example of a map made with nonlinear principal components analysis on the product catalog shown in Table 2.1.

should be able to cope with the specific data contained in a product catalog, that is, it should be able to handle missing values and numerical, categorical, and multi-valued categorical attributes.

Many popular (dis)similarity measures such as the Euclidean distance, Pearson's correlation coefficient, and the Jaccard similarity measure are not able to handle all of these attribute types. They can also not handle missing values naturally. Therefore, we use a dissimilarity measure which is an adaptation of the general coefficient of similarity proposed by Gower (1971) and was introduced in Kagie, Van Wezel, and Groenen (2008b). Note that the MDS based product catalog map approach can also be used with other dissimilarity measures, such as co-purchases or item-item correlations.

The dissimilarity δ_{ij} between products i and j is defined as the square root of the average of nonmissing dissimilarity scores δ_{ijk} on the K attributes

$$\delta_{ij} = \sqrt{\frac{\sum_{k=1}^{K} w_k m_{ik} m_{jk} \delta_{ijk}}{\sum_{k=1}^{K} w_k m_{ik} m_{jk}}} \quad , \tag{2.1}$$

where w_k is the weight of attribute k and m_{ik} and m_{jk} are binary indicators having a value of 0 when attribute k is missing for product i or j respectively and 1 otherwise. The weights w_k can be used to make some attributes more important than others. In Section 2.4, we show how these weights could be assigned automatically to match users' general preferences. The definition of the dissimilarity score δ_{ijk} depends on the type of attribute k. For all attribute types we use the

same kind of normalization that ensures that the average nonmissing dissimilarity score for each attribute is equal to 1 in the product catalog. This normalization is used, to make the dissimilarity scores equally important and independent of the number of missing values.

The numerical dissimilarity score is based on the absolute distance

$$\delta_{ijk}^{N} = \frac{|x_{ik} - x_{jk}|}{\left(\sum_{i<j} m_{ik}m_{jk}\right)^{-1} \sum_{i<j} m_{ik}m_{jk}|x_{ik} - x_{jk}|} \quad . \tag{2.2}$$

The dissimilarity score for categorical attributes is computed as

$$\delta_{ijk}^{C} = \frac{1(x_{ik} \neq x_{jk})}{\left(\sum_{i<j} m_{ik}m_{jk}\right)^{-1} \sum_{i<j} m_{ik}m_{jk}1(x_{ik} \neq x_{jk})} \quad , \tag{2.3}$$

where $1()$ is the indicator function returning a value of 1 when the condition is true and 0 otherwise.

To handle also multi-valued categorical attributes, which are categorical attributes for which a product can belong to multiple categories, this dissimilarity framework was extended in Kagie et al. (2008a). There, the dissimilarity score for multi-valued categorical attributes was defined as

$$\delta_{ijk}^{M} = \frac{|x_{ik} \Delta x_{jk}|}{\left(\sum_{i<j} m_{ik}m_{jk}\right)^{-1} \sum_{i<j} m_{ik}m_{jk}(|x_{ik} \Delta x_{jk}|)} \quad , \tag{2.4}$$

where both x_{ik} and x_{jk} are sets of values and Δ is the symmetric difference set operator. This measure counts how many categories there are to which one of the products at hand belongs and the other not.

As mentioned in Section 2.2.3, the aim of MDS is to map a dissimilarity matrix $\mathbf{\Delta}$ (having elements δ_{ij} and in our approach computed using (2.1)) as good as possible to distances between points in a low dimensional Euclidean space. This objective can be formalized by minimizing the raw Stress function (Kruskal, 1964)

$$\sigma_r(\mathbf{Z}) = \sum_{i<j} w_{ij}(\delta_{ij} - d_{ij}(\mathbf{Z}))^2 \quad . \tag{2.5}$$

In this equation, $\mathbf{Z}$ is an $I \times 2$ coordinate matrix which forms the basis for the map, δ_{ij} is the dissimilarity between items i and j and $d_{ij}(\mathbf{Z})$ is the Euclidean distance between the coordinates of i and j, that is

$$d_{ij}(\mathbf{Z}) = \sqrt{\sum_{s=1}^{2}(z_{is} - z_{js})^2} \quad . \tag{2.6}$$

Also, weights w_{ij} can be specified to force some dissimilarities to be fit better than others.

The dissimilarity measure we use is able to handle missing values. However, dissimilarities based on only a couple of nonmissing (and maybe even unimportant) attributes are more unreliable than dissimilarities for which no dissimilarity scores were missing. Therefore, the latter

should receive higher weights. This can be done by defining the weights to be used in (2.5) as the weighted proportion of nonmissing attributes used for pair ij

$$w_{ij} = \frac{\sum_{k=1}^{K} w_k m_{ik} m_{jk}}{\sum_{k=1}^{K} w_k} \quad . \tag{2.7}$$

We minimize $\sigma_r(\mathbf{Z})$ using the SMACOF algorithm (De Leeuw, 1988) which is based on majorization. One of the advantages of this method is that it is reasonable fast and that the iterations yield monotonically improved Stress values and the difference between subsequent coordinate matrices $\mathbf{Z}$ converges to zero (De Leeuw, 1988). This property has an important and vital consequence for dynamic visualizations: The algorithm produces smooth changes to the points in the display leading to a (local) minimum solution of (2.5). In effect, the objects follow a smooth trajectory on the screen.

The resulting maps may look overwhelming to the user when the number of products is large. To make the map more appealing to the user, a small number of products will be highlighted by showing larger sized and full color images, while other products are represented by a smaller monochrome image. These highlighted products will be helpful to the user to get a quick overview of the map. Therefore, it would be nice, when these products represent different groups of products in this map. This was done, by first clustering the products in the map using k-means clustering (Hartigan & Wong, 1979). We decided to perform a k-means clustering on the map $\mathbf{Z}$ instead of a hierarchical clustering procedure on the original dissimilarities for two reasons. First, this procedure is faster and, second, it is consistent with the visualization, that is, there is no overlap between the clusters in the map. In each cluster, one product is chosen to be highlighted, that is, the product closest to the cluster center based on Euclidean distance. In Section 2.5, we show a prototype of this approach using a product catalog of MP3-players.

2.3.2 Nonlinear Principal Components Analysis

The second approach for creating a product catalog map discussed here is based on NL-PCA. In this approach, a map is created in which not only the products are plotted, but also the category values of the attributes. These can then be used for navigation and selection. NL-PCA is a generalization of ordinary principal components analysis to ordinal (nonlinearly ordered) and categorical attributes. When only having numerical attributes, NL-PCA simplifies to ordinary PCA and when all attributes are categorical and a so-called multiple nominal transformation is chosen, then NL-PCA is identical to homogeneity or multiple correspondence analysis.

In homogeneity analysis, the $I \times K$ data matrix $\mathbf{X}$ is modeled by an indicator matrix $\mathbf{G}_k$ for every attribute. Let L_k denote the number of categories of attribute k. Every category $\ell, \ell = 1, \ldots, L_k$ has its own column in the $I \times L_k$ matrix $\mathbf{G}_k$, in which a 1 denotes that the object belongs to this category and a 0 that it does not. Multi-valued categorical attributes are modeled using an $I \times 2$ indicator matrix for every category. Missing values are incorporated using an $I \times I$ binary diagonal matrix $\mathbf{M}_k$ for all attributes having a value of 1 on the diagonal for nonmissing values for attribute k and 0 otherwise. Using this data representation, we can define the following

loss function for homogeneity analysis (Gifi, 1990; Michailidis & De Leeuw, 1998) by

$$\sigma(\mathbf{Z}; \mathbf{Y}_1, \ldots, \mathbf{Y}_K) = K^{-1} \sum_{k=1}^{K} \text{tr}(\mathbf{Z} - \mathbf{G}_k\mathbf{Y}_k)'\mathbf{M}_k(\mathbf{Z} - \mathbf{G}_k\mathbf{Y}_k) \ , \tag{2.8}$$

where $\mathbf{Z}$ is a $I \times 2$ matrix representing the objects in 2D Euclidean space and the matrix $\mathbf{Y}_k$ is the 2 dimensional representations of the category values of the attribute k. Both $\mathbf{Z}$ and all $\mathbf{Y}_k$'s can be plotted in a joint space which is called a biplot (Gower & Hand, 1996). Essentially, $\mathbf{Z} - \mathbf{G}_k\mathbf{Y}_k$ gives the differences (or error) between the position of the individual products and the positions of the category centroids they belong to for variable k. Ideally, no error would exist and all products in the same category would be fully homogeneous and coincide with the position of their category. As there are more attributes and the products fill in different categories on the different attributes, (2.8) simply measures the squared distances of the products relative to their category centroids, hence how homogeneous the products are. The matrix $\mathbf{M}_k$ removes the contribution to the error for any product having a missing value for attribute k. Equation (2.8) can be minimized using an alternating least squares (ALS) procedure called Homals (Gifi, 1990; Michailidis & De Leeuw, 1998).

From Homals, it is an easy change to NL-PCA by imposing extra restrictions on the category points. Numerical and ordinal attributes can be incorporated in the Homals framework when we impose a rank-1 restriction of the form

$$\mathbf{Y}_k = \mathbf{q}_k\mathbf{a}_k' \tag{2.9}$$

for the numerical and ordinal attributes, where $\mathbf{q}_k$ is a L_k dimensional column vector of category quantifications and $\mathbf{a}_k$ is a 2 dimensional column vector of weights. Using this restriction the category points are forced to be on a line. However, this restriction is not sufficient to preserve the order for ordinal attributes or even the relative distance for numerical attributes. Therefore, $\mathbf{q}_k$ is constrained every ALS iteration to satisfy such restrictions. In the case of ordinal attributes this transformation is done by a weighted monotone regression (De Leeuw, 1977b) and in the case of numerical attributes this results in replacing $\mathbf{q}_k$ by the attribute's original values $\mathbf{x}_k$. A detailed description of the ALS algorithm for NL-PCA can be found in Gifi (1990); Linting et al. (2007); Michailidis and De Leeuw (1998).

NL-PCA solutions have a number of advantageous properties that make it very suitable to create maps that contain both products and attribute category values.

- NL-PCA provides a joint space of object and attribute category points, called a biplot (Gower & Hand, 1996), while other visualization methods only provide the object points. The category points can be used to provide an easier interpretation of the map and to navigate through the map by selecting subsets of products.

- For categorical attributes, the category points are located in the centroids of the object points belonging to that category. This implies that when a certain attribute is well represented in the map, the object points are clustered around their corresponding category value of that attribute and a selection of all points belonging to this category will lead to

a selection of a subspace of the map. For ordinal attributes, the category point will be the point on the line closest to the centroid of the object points.

- The third advantage is shared by NL-PCA and most other visualization methods. That is, the distance between object points in the map is, in general, related to dissimilarity between objects.

In Section 2.5, we also show an application of this approach to the MP3 player data set.

2.4 Determining Attribute Weights using Clickstream Analysis

Both product catalog map approaches introduced in the previous section consider all attributes of a product as equally important to the user. However, we showed that attribute weights can be incorporated in the dissimilarity measure used for MDS and also NL-PCA can be adapted to incorporate different weights for attributes. This holds for most visualization methods, including self-organizing maps and treemaps which were described in Section 2.2.

In this section, we will introduce an approach to determine these weights automatically using clickstream data. For every product, we count how often it was sold during some period. In our application, shown in Section 2.5, we actually counted outclicks, which are clicks out of the site (we used data of a price comparison site) to a shop where the product can be bought. For ease of readability, we use the term sales instead of outclicks during the remainder of this chapter. Using these counts and the product attributes, we estimate a Poisson regression model, which is a model belonging to the class of generalized linear models. Using the coefficients of this model and their corresponding standard errors, we compute t-values which form the basis of our attribute weights. This approach was described earlier in (Kagie et al., 2008a).

2.4.1 Poisson Regression Model

A collection of models frequently used in the field of statistics are the generalized linear models (GLM, see McCullagh & Nelder, 1989; Nelder & Wedderburn, 1972). To model a dependent count variable being discrete and nonnegative, such as sales in our domain, we use an appropriate member of the GLM family, that is, the Poisson regression model. In Poisson regression, we cannot use (multi-valued) categorical attributes directly, so we have to create dummy attributes instead. Therefore, every categorical attribute is represented by $L_k - 1$ dummies $x_{ik\ell}$, which are 1 for the category where the item belongs to and 0 for all other attributes, where L_k is the number of different categories for attribute k. When an item belongs to the last category L_k all dummies for this attribute will be 0. This representation is chosen to avoid multicollinearity. For multi-valued categorical attributes the same approach is used, only now all categories are represented by, in total, L_k dummies. For numerical attributes, we can just use the attribute itself. Hence, $x_{ik} = x_{ik1}$ and $L_k = 1$. We collect all $x_{ik\ell}$ for item i in vector $\mathbf{x}_i$. Also, an intercept term x_{i0}

is incorporated in this vector, which equals 1 for all items. Furthermore, we have the dependent count variable value y_i for all I items. Now, we can express the Poisson regression model as

$$y_i \approx \exp(\mathbf{x}_i'\mathbf{b}) \ , \tag{2.10}$$

where $\mathbf{b}$ is a vector of regression parameters to be estimated. Furthermore, it is assumed that y_i is Poisson distributed having expectation $E(\exp(\mathbf{x}_i'\mathbf{b}))$. The Poisson regression model can be fitted by maximizing its loglikelihood function, which is often done by an iteratively reweighted least squares algorithm. Besides the model parameters $b_{k\ell}$, also standard errors σ_{kl} can be derived by this algorithm.

2.4.2 Handling Missing Values

Unfortunately, the Poisson regression model cannot cope with missing values in an integrated way. Missings are in general a problem in GLMs and Ibrahim, Chen, Lipsitz, and Herring (2005) recently compared different techniques that can be used to handle missing values in combination with GLMs. One of the best methods (leading to unbiased estimates and reliable standard errors) in their comparison was multiple imputation (MI, see Rubin, 1987). MI methods create a number of 'complete' data sets in which values for originally missing values are drawn from a distribution conditionally on the nonmissing values. These imputed data sets can be created using two different methods: data augmentation (J. L. Schafer & Olsen, 1998) and sampling importance/resampling (King, Honaker, Joseph, & Scheve, 2001). Although both methods lead to results having the same quality, the second method computes these results much faster. Therefore, an algorithm based on the second approach, namely the Amelia algorithm (King et al., 2001) which is available as a package (Honaker, King, & Blackwell, 2008) for the statistical software environment `R`, is used in our approach. For a discussion about how the regression coefficients and standard errors of these imputed datasets can be used to estimate the parameters and standard errors of the Poisson regression model, we refer to Kagie et al. (2008a).

2.4.3 Choosing Weights Using Poisson Regression

There are three reasons why we cannot use the coefficients $b_{k\ell}$ resulting from the Poisson regression model as weights directly. The first reason is that dissimilarity scores and attributes do not have the same scale. Second, uncertainty about the correctness of the coefficient is not taken into account when using $b_{k\ell}$ directly. Although the value of a coefficient can be relatively high, it can still be unimportant. Consider, for example, a dummy attribute having very few 1's and a high coefficient value. Then, this high impact of the coefficient is only applicable to a limited number of items and its total importance is limited. By taking the uncertainty we have into account, we can correct for this. Third, weights should always be equal to or larger than 0, while $b_{k\ell}$ can also be negative.

The first two problems can be overcome by using the t-value

$$t_{k\ell} = \frac{b_{k\ell}}{\sigma_{kl}} \tag{2.11}$$

of coefficient $b_{k\ell}$ as a basis to compute weights. Due to standardization, all $t_{k\ell}$'s are on the same scale and, since these are standardized by division by the corresponding standard error $\sigma_{k\ell}$, uncertainty is taken into account. When we use $|t_{k\ell}|$ instead of $t_{k\ell}$ we guarantee the weights to be larger than or equal to 0.

For numerical attributes, we can map $|t_{k1}|$ directly to a 'pseudo' absolute t-value v_k for attribute k, that is, $v_k = |t_{k1}|$. Then, including a normalization of the weights (for ease of interpretability), we can compute w_k using

$$w_k = \frac{v_k}{\sum_{k'=1}^{K} v_{k'}} \quad . \tag{2.12}$$

For (multi-valued) categorical attributes, we first have to compute v_k using the L_k values of $t_{k\ell}$. This is done by taking the average of the absolute $t_{k\ell}$ values

$$v_k = \frac{1}{L_k} \sum_{\ell=1}^{L_k} |t_{k\ell}| \quad . \tag{2.13}$$

These v_k's can then be used to compute the weights for (multi-valued) categorical attributes using (2.12).

2.4.4 Stepwise Poisson Regression Model

The t-values $t_{k\ell}$ can be compared to a t-distribution to determine whether these values are significantly different from 0. In a so-called stepwise model, this significance testing is used for attribute selection, finally resulting in a model only having significant attributes. The stepwise model approach starts off with a model containing all attributes. Then, each time the most insignificant attribute is deleted from the model, until the model only contains significant attributes. Note that this stepwise approach leads to a different model as when all insignificant attributes are deleted at once, since, due to collinearity, significance of an attribute may change, when another attribute is deleted. When using the stepwise model to determine weights w_k, we consider L_k to be the number of dummies incorporated in the final model. This stepwise model is used in the application shown in Section 2.5.

2.5 E-Commerce Applications

In this section, we describe three working and online available prototypes, two of which implement the product catalog maps based on MDS and NL-PCA and the third prototype being a graphical shopping interface (GSI). Both the MDS based product catalog map and the GSI also have an option to use weights determined by the method described in Section 2.4. All prototypes are using a product catalog of MP3 players that is described next.

The product catalog of MP3 players was made available to us by Compare Group. Compare Group hosts, among other European price comparison sites, the Dutch price comparison site `http://www.vergelijk.nl`. The product catalog used is based on a data base dump of

Table 2.2: Attribute characteristics of the MP3 player data. Only the attributes having less than 50% missing values ares listed. For (multi-valued) categorical attributes only the three most occurring values are shown

Attribute	% Missing Values	Mean
Numerical Attributes		
Price	0.0%	134.54
Height	40.9%	63.65
Width	40.9%	48.87
Weight	44.0%	71.21
Depth	40.9%	16.54
Memory Size	3.1%	8635.30
Battery Life	44.4%	17.26
Categorical Attributes		
Brand	0.0%	Samsung (12.0%), Creative (9.8%),Philips (8.4%)
Radio	32.0%	yes (68.6%), no (31.4%)
Extendable Memory	44.9%	yes (17.7%), no (82.3%)
Equalizer	39.6%	yes (85.3%), no (14.7%)
Screen	30.2%	yes (99.4%), no (0.6%)
Battery Type	44.4%	li-ion (44.8%), 1×AAA (33.6%), li-polymer (20.8%)
Voice Memo	24.4%	yes (81.2%), no (18.8%)
Multi-Valued Categorical Attributes		
Storage Type	42.7%	flash memory (69.0%), hdd (21.7%), sd card (10.1%)
Interface	4.0%	usb (63.4%), usb 2.0 (31.5%), hi-speed usb (7.4%)
Color	38.2%	black (68.3%), white (20.1%), silver (16.5%)
Operating System	31.1%	windows (78.7%), mac os (34.2%), windows xp (32.3%)
Audio Formats	1.8%	MP3 (98.6%), WMA (90.0%), WAV (48.0%)

this site from October 2007. At that moment, there were 225 MP3 players listed on the site. In total, these products were described using 45 attributes of which there were 16 numerical, 20 categorical, and 45 multi-valued categorical. Of these 45 attributes, 26 attributes were missing for more than half of the MP3 players. The other attributes are listed in Table 2.2. Note that this, although it also concerns MP3 players, is a different data set than used in Kagie et al. (2007, 2008b, 2008c). To be able to determine the weights automatically as described in Section 2.4, we matched the product catalog to a clickstream log of the same website logged during a period of two months from July 15 until September 15, 2007.

2.5.1 MDS Based Product Catalog Map Using Attribute Weights

The MDS based product catalog map prototype can be visited online at `http://www.browsingmap.com/mapvis.html`, where it is available as a Java applet. A screenshot of

Table 2.3: Weights determined using stepwise Poisson regression for the MP3 player catalog. Only selected attributes are shown

Attribute	Weight
Brand	0.242
Memory Size	0.176
Audio Formats	0.131
Battery Type	0.106
Width	0.090
Operating System	0.086
Color	0.084
Storage Type	0.084

this prototype is shown in Figure 2.5. The GUI of the prototype mainly consists of a large focus map which gives a detailed view of a part of the complete product catalog map. In this map, a couple of products, in this case 12, are represented by a larger full color image, the other products are visualized as smaller monochrome images. By default, the monochrome images are colored according to the cluster they belong to. Alternatively, the user has the option to color them by product popularity (based on the clickstream data) or by attribute value. The small overview map at the top always shows the complete product catalog map. The user can navigate over the map in several ways. By selecting a rectangular subspace in the focus or overview map, the user can zoom in on a specific part of the map. Zooming is also possible using the mouse wheel. Finally, the user can move over the map using dragging. Since the map itself is static, that is, the coordinates of products are fixed, the computation of the map can be done offline once, decreasing online computation time. Of course, it is straightforward to add traditional search technology, such as performing crisp queries, to the map based interfaces. All remaining products satisfying the constraints can then be displayed using a map. This would require the map to adapt when the product set changes and in this case client computations may be necessary.

The map of the prototype has been made using the attribute weight determination technique described in Section 2.4. These weights were determined using the data and product catalog described above. To be able to impute the missing values, we excluded attributes having more than 50% missing values and categories of (multi-valued) categorical attributes that were observed less than 10 times. The weights determined in this way are shown in Table 2.3. Note that attributes not mentioned in the table have a weight of zero and, hence, have no influence in the MDS procedure. According to our method, Brand and Memory Size are the most important attributes determining popularity of MP3 players and receive the highest weights.

In Figure 2.6, we labeled the product points of both a map made by MDS using equal weights for all attributes and a map made using our weighting approach by their corresponding Brand. The second map was rotated using Procrustean transformation (Green, 1952) to best match the first map. As can be seen in Figure 2.6, the products having the same brand are more clustered in the second map where brand had been made more important.

Figure 2.7a shows the second most important attribute Memory Size. This map shows that

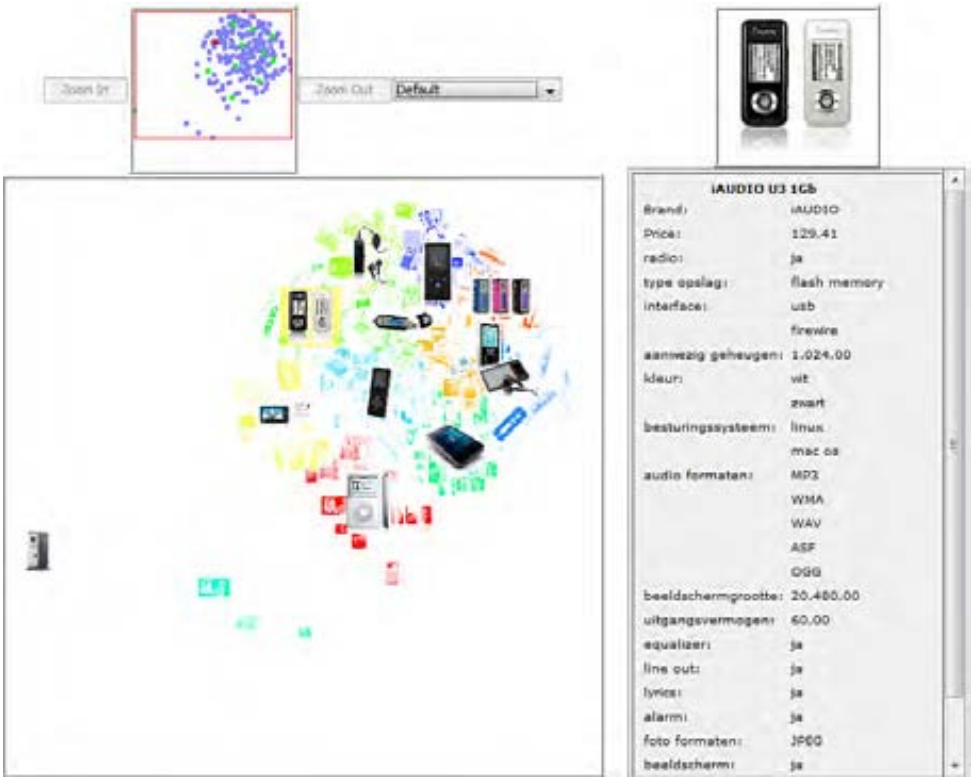
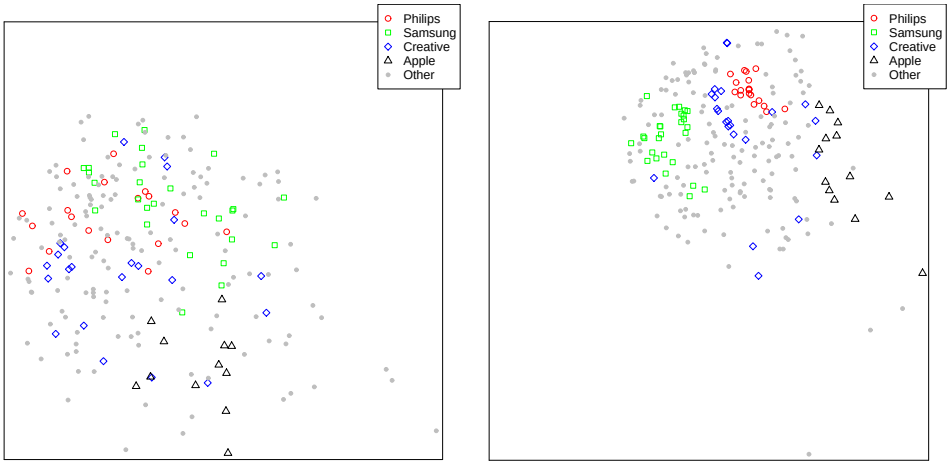


Figure 2.5: Screenshot of the MDS based product catalog map prototype.

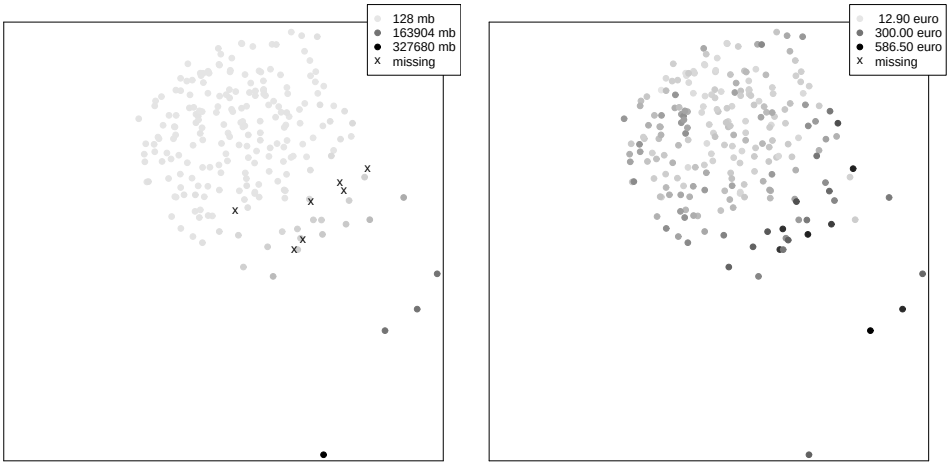
the MP3 players are generally ordered from cheap at top left to expensive at the bottom right in the map. The most striking attribute absent in Table 2.3 and, thus, receiving a weight of zero, was Price. However, when we look at Figure 2.7b where we have made a map labeled by Price, there is a clear pattern in the map. Both effects exist, since Price highly correlates with other features of MP3 players, such as the Memory Size. These features are, hence, the features explaining most of the price that is given to a product.

More generally, we can see that the MDS produces a kind of circular shaped map with some outliers that represent products very different from all other ones. Due to this, large parts of the map are unused. On the other hand, the map provides at first sight a good interpretation, where the important attributes show a clear pattern.



a. Equal weights b. Poisson regression weights

Figure 2.6: Product catalog maps based on MDS labeled by brand.



a. Memory Size b. Price

Figure 2.7: Product catalog maps based on MDS labeled by Memory Size and Price.

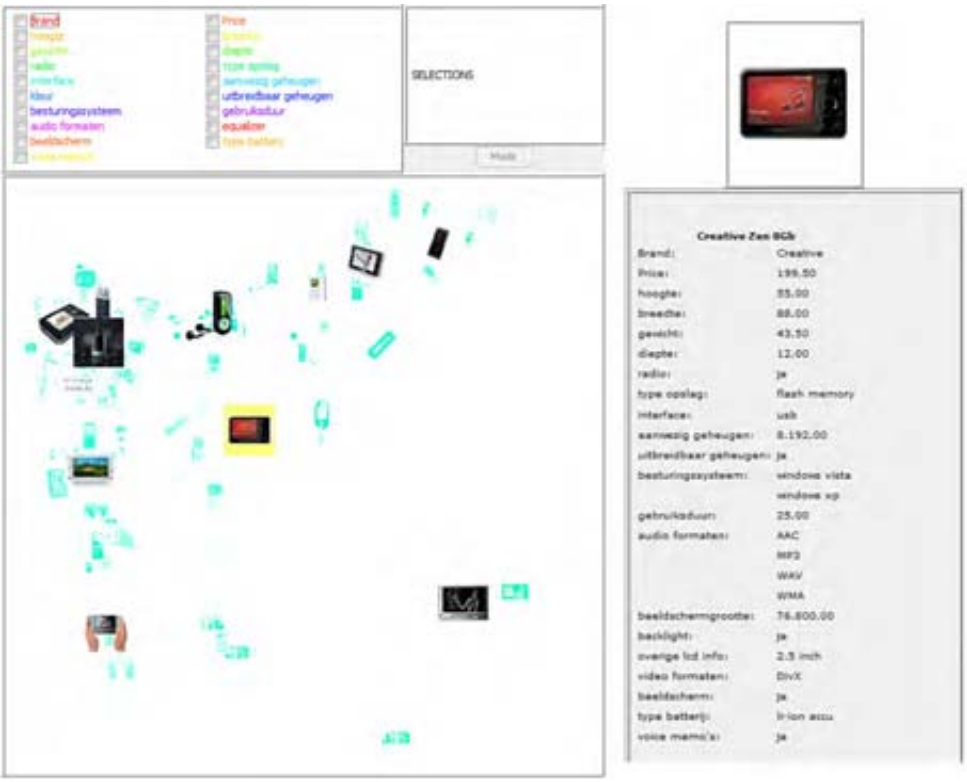


Figure 2.8: Screenshot of the NL-PCA based product catalog map prototype.

2.5.2 NL-PCA Based Product Catalog Map

A screenshot of the prototype using NL-PCA is shown in Figure 2.8. This application is available online at `http://www.browsingmap.com/mapvis.html`. The main part of the GUI is the product catalog map. Initially, this map shows all products. A couple of products, in this case 12, are highlighted using a larger full color image, the other products are represented by a smaller monochrome image. For the initial map, the products that are highlighted are selected using a k-means clustering procedure identical to the one used in the MDS based product catalog map.

Above the product catalog map, there is a list with all product attributes. Each attribute can be selected by clicking on its check box. If one selects a certain attribute, the category points of this attribute are added to the map. The category points of the selected attributes not only help the user to interpret the map, they are also tools to navigate through the map. By clicking on a category point on the map this category is added to the selection list.

The selection list, which is shown to the right of the list of attributes, determines which products are highlighted in the map. The set of highlighted products is determined as follows. For each of the selected attributes, a shown product should belong to at least one of the selected categories.

When the selection list has been adapted by adding or removing a category point, a couple of things are modified in the visualization of the map. The first thing is that all products satisfying the new constraints are colored red, while all other products are colored blue. Furthermore, a number of products is randomly selected from this set to be highlighted.

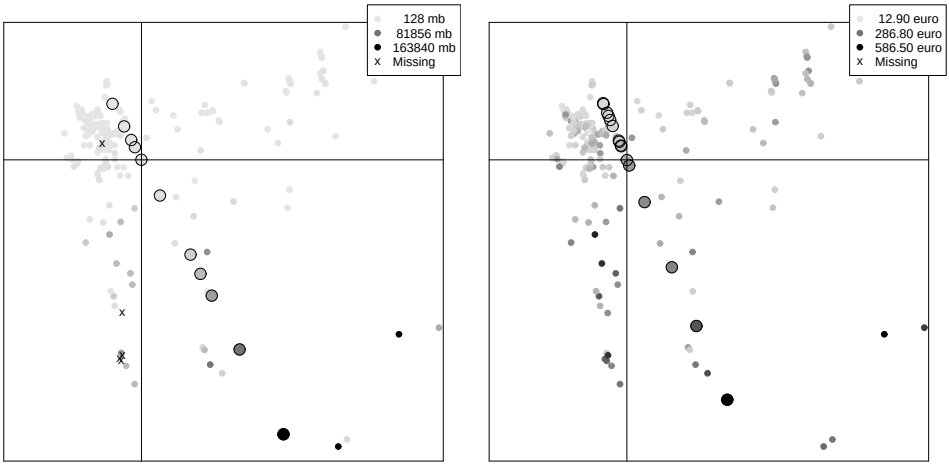
Since a selection will often lead to a subspace of the map, it is also possible to zoom in on this part of the map. However, there is no guarantee that all points in this subspace satisfy all constraints imposed by the selection list. We have chosen not to delete these product points, since these may be interesting to the user. Although these products do not satisfy all the demands of the user, they are very similar to the products that do and may have some appealing characteristics the user until then did not think of.

In both the complete and the zoomed in map, the user can click on the highlighted products to get a full product description of this selected product, which is shown at the right side of the application. However, by moving over both a full color or a monochrome image, a tooltip is shown containing an image of the product and the values of some of its most important attributes. Furthermore, the values for the attributes in the selection list are shown in this tooltip, colored green when they match the preferred category value and red when they do not. Since the GUI is based on a single NL-PCA map, this map too can be computed offline just as the MDS product catalog map.

Since the quality of NL-PCA maps may become quite poor when having a lot of missing values, we removed attributes having more than 50% missing values. Then, we also removed products having more than 50% missing values on this limited set of attributes. This resulted in a set of 189 MP3-players described by 19 attributes, namely the attributes shown in Table 2.2.

In the NL-PCA algorithm, we will treat all numerical attributes as being ordinal, because of two reasons. In the first place, many of the numerical attributes do not have a linear interpretation for users, such as, for example, the memory size. The second advantage of using the ordinal transformation is that due to the underlying monotone regression procedure some adjacent categories can be merged into a single category point. Since a numerical attribute has a lot of categories (i.e. all unique values in the data set), visualizing all these categories may become unclear and selection using these category values may become useless, since a lot of category values only belong to a single object. Using an ordinal transformation this becomes less of a problem, since categories with a small number of objects are often merged with their neighbors.

In Figure 2.9, two biplots are shown resulting from the NL-PCA algorithm. A biplot visualizes both the products labeled by their attribute value and the category points also labeled by their value. Also, the origin is plotted in the biplots. By the design of the NL-PCA method, ordinary products should be close to the origin, while more distinct products should be far away. This also holds for the categories. Since both biplots in Figure 2.9 are plots of numerical attributes, the category points are on a line. Like in the MDS map, also here both Price and Memory Size correlate with each other and are well represented, in this case on the second dimension.



a. Memory Size b. Price

Figure 2.9: Biplots based on NL-PCA labeled by Memory Size and Price. Large circles are category points and small dots represent object points.

Where the second dimension tells something about the memory size and attributes correlating with that, the first dimension seems to separate older from newer products. This can, for instance, be seen in Figure 2.10, where we labeled products by the Interface attributes. Since Interface is a multi-valued categorical attribute, we have chosen to label each product only by a single value, namely the one most frequent in the product catalog. Also, we only show the category points of the three most occurring categories, since all products belong to at least one of them. As said, there seem to be two groups, the older MP3 players supporting USB and the newer ones supporting USB 2.0. For the operating systems attribute one can observe a similar pattern.

More generally, the NL-PCA approach creates a map in which more of the available space is used than in the MDS approach. However, most of the map is used to visualize the special, not that common products, while the ordinary products are cluttered together near the right top corner of the map. Also, the map only shows those products and attributes that do not have too many missing values.

2.6 Summary, Conclusions, and Outlook

In this chapter, we have discussed how product catalogs can be visualized in a map to provide a way in which e-commerce website users may get a better overview of all products being available. In such a map, products that are similar in their attributes should be located close to each other, while dissimilar products should be located in different areas of the map.

In the framework presented in this chapter, two methods have been used to create such prod-

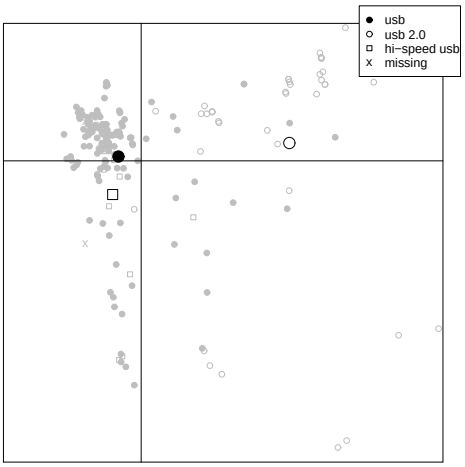


Figure 2.10: Biplot based on NL-PCA labeled by Interface. Large shapes indicate category points and small transparent shapes represent object points.

uct catalog maps: Multidimensional scaling (MDS) and nonlinear principal components analysis (NL-PCA). MDS has the advantage that it is the only method providing a real distance interpretation. Similarity between products is matched as closely as possible to distances in a two dimensional space. We combined MDS with an adapted version of the Gower coefficient being very flexible, since it is able to handle mixed attribute types and missing values. The map made by MDS for the MP3 player application we have shown, seems to have a clear interpretation with a clustering of brands and a important price dimension. The main disadvantage of this map (and MDS in general) is that the map has a circular shape leaving the corners of the map open and positions outliers relative far away from the rest of the map. However, using a weighting scheme emphasizing the small dissimilarities, this may be overcome.

NL-PCA has the advantage that it is the only method that is able to also visualize attribute categories next to the product visualization. These category points can be used to select subsets of products in the map as was shown in our prototype. In general, the interpretation of the NL-PCA map was in line with the interpretation of the MDS map. Although distinct products also take a large part of the map in the NL-PCA approach, the objects are more spread over the map. The main disadvantage of using the NL-PCA method on our product catalog was that we could not visualize all products, because NL-PCA may create poor maps when introducing objects with too many missing values. Another disadvantage is that the dissimilarity between products is not directly mapped to distances as is done in the MDS method. This can be done in NL-PCA by using a different normalization method. However, then interpretation of category points becomes more difficult which may mean that these cannot be used for navigation anymore.

Since users do usually not consider all product attributes to be equally important we have shown a method based on a Poisson regression model, which can determine attribute importance

weights automatically based on counting product popularity in a clickstream log file. Since this method is independent of the visualization technique, it can be used with every technique allowing for weights for attributes and it can even be applied in recommender systems which allow for attribute weights. However, the proposed method has some shortcomings. Weights for categorical attributes are determined in a quite heuristic way and interactions between attributes are ignored. Therefore, we are working on a different way to determine this weights using a more flexible model based on boosted regression trees (Kagie, Van Wezel, & Groenen, 2009).

Map based visualizations might become even stronger when they are integrated with recommendation techniques (Resnick & Varian, 1997). One example of this combination is the Graphical Shopping Interface introduced in Kagie et al. (2008b), that combines MDS based maps with a recommendation by proposing approach (Shimazu, 2002). However, we expect that map based visualization could also be successfully combined with other content- and knowledge-based recommendation techniques, such as critiquing (Reilly, McCarthy, McGinty, & Smyth, 2005).

Besides combinations with other types of recommendation, we think there are some more challenges in product catalog visualization. First of all, since determining which methods provides the best map is a matter of personal taste and subject to the data to be visualized, one could also try different visualization methods, such as independent component analysis (Comon, 1994) or projection pursuit (Friedman & Tukey, 1974). A good idea would be to compare different visualization approaches in a user study. In this study, we used a data set of reasonable size. Using larger product catalogs, for instance, having thousands of products, means that both the algorithms used to create the map as well as the interface itself should be able to cope with these numbers. Besides visualizing a large catalog of a single product type, another challenge might be to create a map containing multiple types of products, for instance, different electronic devices.

Chapter 3

A Graphical Shopping Interface Based on Product Attributes*

3.1 Introduction

In most electronic commerce stores, consumers can choose from an enormous number of different products within a product category. Although one would assume that increased choice is better for consumer satisfaction, the contrary is often the case (Schwartz, 2004). This phenomenon is known as the paradox of choice: a large set of options to choose from makes it more difficult for the consumer to find the product that she prefers most, that is, the product that is most similar to the consumer's ideal product. When the number of choice options increases, consumers often end up choosing an option that is further away from the product they prefer most.

In many product categories, such as real estate and electronics, a consumer has to choose from a heterogeneous range of products with a large amount of product attributes. In most current e-commerce interfaces, the consumer can only search by constraining the values of one or more product attributes, for example, the brand of an MP3 player should be 'Apple', and its memory size should be larger than 2GB. The products that satisfy these constraints are then shown in a list.

A disadvantage of this approach to product search is the use of crisp boundaries, which the consumer may find too strict. For instance, 1.99GB players are not listed with a 2GB boundary, while a consumer may be perfectly happy with a 1.99GB player. Another disadvantage of the traditional 'query-and-list' approach is its inability to cope with substitution effects. Such an effect occurs when a shortcoming on one attribute can be compensated by another attribute. For example, a consumer can be equally happy with a cheaper MP3 player with less memory as with a more expensive one with a larger amount of memory compared to some ideal product specification. Yet, when restricting the search results to cheap MP3 players, the latter one will not be shown.

*This chapter is based on Kagie, Van Wezel, and Groenen (2008b).

An alternative to this approach is to let the consumer describe an ideal product based on her ideal values for the product attributes. Then, products can be listed in order of similarity to the ideal product.

A disadvantage of the usual approach of presenting the products in a list is that no information is given on how similar the selected products are to each other. For example, two products that have almost the same similarity to the ideal product can differ from the ideal product on a completely different set of attributes and thus differ a lot from each other. Therefore, mutual similarities should be incorporated when displaying recommended products to consumers.

In this chapter, we propose a graphical recommender system (GRS) that visualizes the recommended products together with the ideal product in a 2D map using the mutual similarities. To do this, multidimensional scaling (MDS, see Borg & Groenen, 2005) will be used. Since a consumer does not always want to or is able to specify her preferences, we also introduce a graphical shopping interface (GSI) that enables the consumer to navigate through the products in a 2D map. The GSI system presents the user a map of a limited set of products of which the user has to select the product she prefers most. Based on this selection, a new map is constructed with products that are more similar to the selected product than in the previous map, and this process repeats itself.

The main contribution of this chapter is that both the GRS and the GSI combine elements of existing recommendation and query techniques (Chaudhuri & Gravano, 1999; Ricci, Wöber, & Zins, 2005) on one hand with intelligent data visualization techniques (Borg & Groenen, 2005) for creating map based representations of recommendations on the other hand.

The remainder of this chapter is organized as follows. The next section gives a brief overview of related work. In Section 3.3, we give a description of the methodology used with an emphasis on the measure of similarity and MDS. In Section 3.4, the graphical recommender system is introduced and in Section 3.5 we extend the GRS to the GSI. In Section 3.6, an application of both systems on MP3 players is given, followed by an evaluation of our approach. Finally, we give conclusions and recommendations.

3.2 Related Work

J. B. Schafer, Konstan, and Riedl (2001) define recommender systems as: "[Systems that] are used by E-commerce sites to suggest products to their consumers and to provide consumers with information to help them decide which products to purchase" (J. B. Schafer et al., 2001, p. 116). These suggestions can be the same for each consumer, like top overall sellers on a site, but are often dependent on the user's preferences. These preferences can be derived in many ways, for example, using past purchases, navigation behavior, rating systems, or just by asking the user's preferences directly. There is a wide literature on recommender systems. For an overview we refer to Adomavicius and Tuzhilin (2005) and Prasad (2003).

In this chapter, we limit ourselves to the recommender systems where a personalized recommendation is given. Recent overview papers (Adomavicius & Tuzhilin, 2005; Prasad, 2003) make a distinction between three types of recommender systems based on how the recommendation is computed.

- *Content-based or knowledge based recommendation* (Burke, 2000) systems suggest products that are similar to the product(s) the consumer liked in the past.
- *Collaborative filtering* (D. Goldberg, Nichols, Oki, & Terry, 1992) systems recommend products that other people with similar taste bought or liked in the past.
- *Hybrid approaches* (Burke, 2002) combine both content-based and collaborative methods.

Our systems belong to the category of content-based recommender systems. A large number of recommender systems in this group, including our approach, is based on case-based reasoning (CBR) (Lorenzi & Ricci, 2005) and follow the CBR recommendation cycle. In most CBR recommender systems (CBR-RS) the data used is the product catalog, that is, data describing products by their attribute values. The suggestion of products to recommend is then based on a similarity measure between the product descriptions in the catalog and a query given by the user or some sample product. These kind of systems have many similarities with k-nearest neighbor classification (Cover & Hart, 1967) and top-k querying (Ayanso, Goes, & Mehta, 2007; Chaudhuri & Gravano, 1999). Both techniques search for the k most similar items to a given query specification.

One specific type of CBR-RS's are the systems that use *recommendation by proposing* (Shimazu, 2002) or *inspiration seeking* (Ricci et al., 2005). These systems, as does the GSI, present a limited number of sample products to the user of which the user can choose one. Then, based on the selection by the user, new products are recommended and the user can choose a new product. This approach is supported by the idea that people do not have well-defined preferences in advance, but construct these preferences during the search process (Bettman, Luce, & Payne, 1998). Recommender systems implementing this approach are, for instance, ExpertClerk (Shimazu, 2002), the comparison-based recommender system (McGinty & Smyth, 2002), SmartClient (Pu & Kumar, 2004), and DieToRecs (Ricci et al., 2005).

Other graphical applications using 2D maps, so-called inspiration interfaces, are used in the field of industrial design engineering (Keller, 2000; Stappers & Pasman, 1999; Stappers, Pasman, & Groenen, 2000). These applications are used to explore databases in an interactive way. At first, a small set of items is shown in a 2D map. Then, the user can click in any point on the map and a new item that is closest to that point is added to the map. Our GSI differs from these systems by recommending more items at a time and doing so using the similarities and not the distances in the 2D map.

Also, interfaces have been developed based on a single product map (Kagie, Van Wezel, & Groenen, 2007, 2008c) visualizing the complete product space. These maps are made using MDS (Kagie et al., 2007) and nonlinear principal components analysis (Kagie et al., 2008c). In somewhat related fields like news (Ong, Chen, Sung, & Zhu, 2005), the web (Chung, Bonillas, Lain, Xi, & Chen, 2006; Turetken & Sharda, 2004; C. C. Yang, Chen, & Hong, 2003), music playlists (Donaldson, 2007; Van Gulik, Vignoli, & Van der Wetering, 2004), and image browsing (Pečenović, Do, Vetterli, & Pu, 2000) GUI's based on 2D visualizations have been created only using different visualization techniques like self-organizing maps (Kohonen, 2001), treemap (Schneiderman, 1992), and other MDS methods like classical scaling (Torgerson, 1952) and Sammon mapping (Sammon, 1969). There are also a couple of commercial websites using

2D visualizations, such as Browse Goods[1], Liveplasma[2], Musicovery[3], and Newsmap[4]. Also, the popular video website YouTube[5] has recently introduced a feature providing recommendations in a 2D map.

3.3 Methodology

An important part of the GSI is the similarity measure that is used to find cases that are recommended to the user. This similarity measure will be used for the selection of products and for visualizing the similarities between all of the recommended products. The method used for creating these 2D maps is called multidimensional scaling (MDS, see Borg & Groenen, 2005) which is discussed in Section 3.3.1.

To define the measure of similarity between products, we introduce some notation. Consider a data set D, which contains products $\{\mathbf{x}_i\}_1^n$ having K attributes $\mathbf{x}_i = (x_{i1}, x_{i2} \dots x_{iK})$. In most applications, these attributes have mixed types, that is, the attributes can be numerical, binary, or categorical. The most often used (dis)similarity measures, like the Euclidean distance, Pearson's correlation coefficient, and Jaccard's similarity measure, are only suited to handle one of these attribute types.

One similarity measure that can cope with mixed attribute types is the general coefficient of similarity proposed by Gower (1971). Define the similarity s_{ij} between products i and j as the average of the non-missing similarity scores s_{ijk} over the K attributes

$$s_{ij} = \sum_{k=1}^{K} m_{ik} m_{jk} s_{ijk} \Bigg/ \sum_{k=1}^{K} m_{ik} m_{jk} \ , \tag{3.1}$$

where m_{ik} is 0 when the value for attribute k is missing for product i and 1 when it is not missing.

The exact way of computing the similarity score s_{ijk} depends upon the type of attribute. However, Gower proposed that for all types it should have a score of 1 when the objects are completely identical on the attribute and a score of 0 when they are as different as possible. For numerical attributes, s_{ijk} is based on the absolute distance divided by the range, that is,

$$s_{ijk}^N = 1 - \frac{|x_{ik} - x_{jk}|}{\max(\mathbf{x}_k) - \min(\mathbf{x}_k)}, \tag{3.2}$$

where $\mathbf{x}_k$ is a vector containing the values of the k^{th} attribute for all n products. For binary and categorical attributes the similarity score is defined as

$$s_{ijk}^C = 1(x_{ik} = x_{jk}), \tag{3.3}$$

[1] http://www.browsegoods.com
[2] http://www.liveplasma.com
[3] http://www.musicovery.com
[4] http://www.marumushi.com/apps/newsmap/index.cfm
[5] http://www.youtube.com

implying that objects having the same category value get a similarity score of 1 and 0 otherwise.

To use Gower's coefficient of similarity in our system, three adaptation have to be made. First, the similarity has to be transformed to a dissimilarity, so that it can be used in combination with MDS. Second, we want to have the possibility to make some variables more important than others. Therefore, we need to incorporate weights into the coefficient. Third, the influence of categorical and binary attributes on the general coefficient turns out to be too large. The reason for this is that the similarity scores on binary or categorical attributes always have a score of 0 or 1 (that is, totally identical or totally different), whereas the similarity scores on numerical attributes almost always have a value between 0 and 1. Thus, the categorical attributes dominate the similarity measure. There is no reason to assume the categorical attributes are more important than numerical ones and we want to compensate for this. Therefore, we propose the following adaptations.

Both types of dissimilarity scores are normalized to have an average dissimilarity score of 1 between two different objects. Since the dissimilarity between the object and itself (δ_{ii}) is excluded and $\delta_{ij} = \delta_{ji}$, dissimilarities having $i \geq j$ are excluded from the sum without loss of generality. The numerical dissimilarity score becomes

$$\delta_{ijk}^{N} = \frac{|x_{ik} - x_{jk}|}{\left(\sum_{i<j} m_{ik} m_{jk}\right)^{-1} \sum_{i<j} m_{ik} m_{jk} |x_{ik} - x_{jk}|}. \tag{3.4}$$

The categorical dissimilarity score becomes

$$\delta_{ijk}^{C} = \frac{1(x_{ik} \neq x_{jk})}{\left(\sum_{i<j} m_{ik} m_{jk}\right)^{-1} \sum_{i<j} m_{ik} m_{jk} 1(x_{ik} \neq x_{jk})}. \tag{3.5}$$

Let C be the set of categorical attributes and N the set of numerical attributes. Then, the combined dissimilarity measure δ_{ij} is defined as

$$\delta_{ij} = \sqrt{\frac{\sum_{k \in C} w_k m_{ik} m_{jk} \delta_{ijk}^{C} + \sum_{k \in N} w_k m_{ik} m_{jk} \delta_{ijk}^{N}}{\sum_{k=1}^{K} w_k m_{ik} m_{jk}}}. \tag{3.6}$$

Here, a vector with weights $\mathbf{w}$ is incorporated to emphasize attributes differently. Note that these weights are applied to the normalized dissimilarity scores. This makes the effect of weighting independent of the original attribute scale, that is, a certain weight value has the same effect in increasing/decreasing an attribute's importance for all attributes.

In our application, the attribute weights $\mathbf{w}$ must be specified by the user. It is known from marketing literature, however, that setting sensible attribute weights is very difficult, even for experts. Therefore, an ideal solution would be to estimate these attribute weights from choice data. This is a complex issue beyond the scope of this chapter.

Note that we use the weights in a linear fashion that is equal for all products. In his paper, Gower (1971) discussed several other weighting approaches that allow for differential weighting per product, such as hierarchical and result-weighting. In result-weighting, a weight depends

on the specific values of x_{ik} and x_{jk}, implying that the weight is not the same for all products. Hierarchical weighting implies a nested structure of variables, which is not present in our data. Although they can be potentially useful, for reasons of simplicity we do not consider these weighting schemes.

In (3.6), the overall square root is taken, since these dissimilarities can be perfectly represented in a high dimensional Euclidean space, when there are no missing values (Gower, 1971). In this case, the dissimilarities are city-block distances and taking the square root of city block distances makes them Euclidean embeddable. We use (3.6) as dissimilarity measure in the remainder of this chapter.

3.3.1 Multidimensional Scaling

The dissimilarities discussed above are used in the GRS and GSI to create the 2D map where products are represented as points. A statistical technique for doing this is multidimensional scaling (MDS, see Borg & Groenen, 2005). Its aim is to find a low dimensional Euclidean representation such that distances between pairs of points represent the dissimilarities as closely as possible. This objective can be formalized by minimizing the raw Stress function (Kruskal, 1964)

$$\sigma_r(\mathbf{Z}) = \sum_{i<j}(\delta_{ij} - d_{ij}(\mathbf{Z}))^2, \tag{3.7}$$

where the matrix $\mathbf{Z}$ is the $n \times 2$ coordinate matrix representing the n products in two dimensions, δ_{ij} is the dissimilarity between objects i and j forming the symmetric dissimilarity matrix $\mathbf{\Delta}$, and $d_{ij}(\mathbf{Z}) = \left(\sum_{s=1}^{2}(z_{is} - z_{js})^2\right)^{1/2}$ is the Euclidean distance between row points i and j.

To minimize $\sigma_r(\mathbf{Z})$, we use the SMACOF algorithm (De Leeuw, 1988) based on majorization. One of the advantages of this method is that it is reasonably fast and that the iterations yield monotonically improved Stress values and the difference between subsequent coordinate matrices $\mathbf{Z}$ converges to zero (De Leeuw, 1988), which is important when visualizing the iterations to the user by a smooth dynamic GRS and GSI.

3.4 Graphical Recommender System

The first system we introduce is the graphical recommender system (GRS). Based on a specification of an ideal product and importance weights for the different attributes, the GRS recommends a set of products most similar to the ideal product and shows them together with the ideal product in a 2D map to the user.

The input of the GRS is the vector of product attributes of the ideal product, $\mathbf{x}^*$, and the weights of the attributes, $\mathbf{w}$. Furthermore, we use the product catalog, that is, data set D. The implementation of the GRS is as follows.

We start by computing the weighted dissimilarities δ_{i*} between $\mathbf{x}^*$ and all the products $\mathbf{x}_i$ in data set D using the dissimilarity measure introduced in Section 3.3. This step leads to n values δ_{i*}.

Then, $p-1$ products are selected that are most similar to $\mathbf{x}^*$, by sorting the δ_{i*}'s and selecting the first $p-1$ products. We combine $\mathbf{x}^*$ and the $p-1$ selected products in one data set D^*. We use (3.6) again to compute all the mutual dissimilarities of the selected objects and the ideal product and gather the dissimilarities in the symmetric matrix $\mathbf{\Delta}^*$ of size $p \times p$. This dissimilarity matrix $\mathbf{\Delta}^*$ is the input for the multidimensional scaling algorithm discussed in Section 3.3.1. The algorithm returns the $p \times 2$ coordinate matrix $\mathbf{Z}$ which is used to create the 2D map.

3.5 Graphical Shopping Interface

To facilitate selection of products by consumers who do not have a clear idea about what they are looking for, we propose the graphical shopping interface (GSI) . The idea is to let the consumer navigate through the complete product space in steps, where at each step a set of products is represented in a 2D map. In this 2D map, the user can select a product and then a new set of products, including the selected product, is produced and visualized by MDS.

The implementation of the GSI is not straightforward. The reason is that it is not trivial to find a way to recommend products more similar to the selected product without overfitting a single selection. We analyze three different approaches: a random system, a clustering system, and a hierarchical system.

The random system uses random selection to select a small set of products that will be shown to the user out of a larger set of products that are similar to the selected product. The first iteration of the GSI is an initialization iteration. We refer to this iteration as iteration $t = 0$. The input of the user is unknown in this iteration, because the first input of the user will be given after this iteration. Therefore, the product set D_t in this iteration will contain the complete product catalog, that is, $D_0 = D$. Then, p products are selected at random (without replacement) from D_0 and stored in the smaller set D_0^*. Using the dissimilarity metric proposed in Section 3.3, we compute the dissimilarity matrix $\mathbf{\Delta}_0^*$, given D_0^*. With the use of MDS we then create a 2D map $\mathbf{Z}_0$ containing these random selected products and show this to the consumer.

The process starts when the consumer selects one of the shown products. In every iteration, the selected product is treated as the new input $\mathbf{x}_t^*$. Then, we compute the dissimilarities between $\mathbf{x}_t^*$ and all other products in D. Based on these dissimilarities we create a set D_t with the $\max(p-1, \alpha^t n - 1)$ most similar products, where the parameter α with $0 < \alpha \leq 1$ determines how fast the data set selection is decreased each iteration. The smaller set D_t^* shown to the user, consists of product $\mathbf{x}_t^*$ and $p-1$ products that are randomly selected from the D_t. We again compute dissimilarity matrix $\mathbf{\Delta}_t^*$ and create the 2D map $\mathbf{Z}_t$ using MDS. The procedure terminates when D^* does not change anymore. This happens when the size of D^* has decreased to p and the same product is chosen as was chosen in the last iteration.

When we set $\alpha = 1$, the system always returns a complete random selection at each stage and the user's input is almost completely ignored, that is, only the selected product is kept and $p-1$ new random products are positioned in a new 2D map together with the kept product. When α is lower, we have more confidence in the selection of the user, but we also more quickly decrease the variance in D_t. The random system is summarized in Figure 3.1.

A disadvantage of the random system is that is difficult for the user to find outlying products

procedure RANDOM_GSI(D, p, α)
$D_0 = D$.
Generate random $D_0^* \subset D_0$ with size p.
Compute $\mathbf{\Delta}_0^*$ given D_0^* using (3.6).
Compute $\mathbf{Z}_0$ given $\mathbf{\Delta}_0^*$ using MDS.
$t = 0$.
repeat
$t = t + 1$.
Select a product $\mathbf{x}_t^* \in D_{t-1}^*$.
Get $D_t \subset D$ containing $\max(p-1, \alpha^t n - 1)$ products most similar to $\mathbf{x}_t^*$ using (3.6).
Generate random $D_t^* \subset D_t$ with size $p-1$.
$D_t^* = D_t^* \cup \mathbf{x}_t^*$.
Compute $\mathbf{\Delta}_t^*$ given D_t^* using (3.6).
Compute $\mathbf{Z}_t$ given $\mathbf{\Delta}_t^*$ using MDS.
until $D_t^* = D_{t-1}^*$.
end procedure

Figure 3.1: GSI implementation using random selection.

with it. There is only a small probability of selecting such a product in D_0^* and it is likely that this product is not in D_t the second time. For this reason, it can be advantageous to have the products selected in D_t^* represent the different groups of products in D_t. By decreasing the size of D_t each time these groups will become more similar to each other and finally become individual products.

The clustering system is quite similar to the random system, the only difference being that the random selection procedure is replaced by a clustering algorithm. In principle, every clustering algorithm can be used that can cluster a dissimilarity matrix. We use the average linkage method which is a hierarchical clustering method, yielding a complete tree (dendrogram) T of cluster solutions. Since this clustering method is based on a dissimilarity matrix, the dissimilarity matrix $\mathbf{\Delta}_t$ based on D_t is computed first using (3.6). Then, the average linkage algorithm is performed on $\mathbf{\Delta}_t$ resulting in dendrogram T_t. The system only uses the solution with p clusters D_t^c. Each of the p clusters is then represented by one prototypical product in the product set D_t^*. For an easy navigation, the product selected in the previous iteration will always represent the cluster it belongs to. For the other clusters, we determine the product with the smallest total dissimilarity to the other products in the cluster. Define $\mathbf{\Delta}^c$ as the dissimilarity matrix (with elements δ_{ij}^c) between all products in cluster D^c, n_c as the size of this cluster, and i_c as the index of the prototypical product, we can define this ideal index as follows

$$i_c = \arg\min_i \sum_{j=1}^{n_c} \delta_{ij}^c. \tag{3.8}$$

procedure CLUSTERING_GSI(D, p, α)
  $D_0 = D$.
  Compute $\mathbf{\Delta}_0$ given D_0 using (3.6).
  Compute T_t given $\mathbf{\Delta}_0$ using average linkage.
  Find p clustering solution in T_0.
  Determine prototypical products of clusters using (3.8).
  Store prototypical products in D_0^*.
  Compute $\mathbf{\Delta}_0^*$ given D_0^* using (3.6).
  Compute $\mathbf{Z}_0$ given $\mathbf{\Delta}_0^*$ using MDS.
  $t = 0$.
  repeat
    $t = t + 1$.
    Select a product $\mathbf{x}_t^* \in D_{t-1}^*$.
    Get $D_t \subset D$ containing $\max(p-1, \alpha^t n - 1)$ products most similar to $\mathbf{x}_t^*$ using (3.6).
    Compute $\mathbf{\Delta}_t$ given D_t using (3.6).
    Compute T_t given $\mathbf{\Delta}_t$ using average linkage.
    Find p clustering solution in T_t.
    Determine prototypical products of clusters using (3.8).
    Store prototypical products in D_t^*.
    Compute $\mathbf{\Delta}_t^*$ given D_t^* using (3.6).
    Compute $\mathbf{Z}_t$ given $\mathbf{\Delta}_t^*$ using MDS.
  until $D_t^* = D_{t-1}^*$.
end procedure

Figure 3.2: GSI implementation using clustering.

The resulting product set D_t^* is used in the same way as in the random system to compute $\mathbf{\Delta}_t^*$ and $\mathbf{Z}_t$. The clustering system is summarized in Figure 3.2.

Clustering (and especially hierarchical clustering) becomes quite slow as the product space gets larger. Since D_t is interactively selected each time, the clustering has to be done each time as well. Therefore, our third implementation, the hierarchical system, does not create a set D_t each time, but uses only one hierarchical clustering result. We start by applying the average linkage algorithm to the complete product catalog D. To do this, we first compute dissimilarity matrix $\mathbf{\Delta}$ using (3.6). We will use the dendrogram T, created by this clustering algorithm, to navigate through the product space. In the first iteration (the initialization), we start by setting $T_0 = T$. An iteration starts at the root of T_t. We will go down T_t until we find the p cluster solution. If this clustering does not exist, the largest possible clustering solution is chosen which is equal to the number of products in the previous selected cluster. This solution exists of p clusters D_t^c, where each of the p clusters is represented by one prototypical product in the product set D_t^*. The procedure for determining the prototypical products is the same as in the clustering system. Then, the dissimilarity matrix $\mathbf{\Delta}_t^*$ of D_t^* is computed using (3.6) and used as input for the MDS algorithm to compute the 2D representation $\mathbf{Z}_t$. When a product $\mathbf{x}_t^*$ is selected from this map,

procedure HIERARCHICAL_GSI(D, p)
  Compute $\mathbf{\Delta}$ given D using (3.6).
  Compute T given $\mathbf{\Delta}$ using average linkage.
  $T_0 = T$.
  $t = 0$.
  $n_c = size(D)$.
  repeat
    Find $\min(n^c, p)$ clustering solution in T_t.
    Determine prototypical products of clusters using (3.8).
    Store prototypical products in D_t^*.
    Compute $\mathbf{\Delta}_t^*$ given D_t^* using (3.6).
    Compute $\mathbf{Z}_t$ given $\mathbf{\Delta}_t^*$ using MDS.
    Select $\mathbf{x}_t^* \in D_t^*$ en determine cluster D_t^c it represents.
    $n_c = size(D_t^c)$.
    D_t^c is root of T_{t+1}.
    $t = t + 1$.
  until $n^c \leq p$.
end procedure

Figure 3.3: GSI implementation using hierarchical clustering.

the cluster it represents is used as the root of T_{t+1}. The procedure is terminated when a selected cluster only contains a single product. This product is the final recommendation of the system.

Note that there is no convergence parameter α in this approach. Since a cluster is selected every step and products outside this cluster are not considered anymore in the remainder of the recommendation procedure, this approach converges quickly to a recommended product. As a consequence, it may lead to worse recommendations. The hierarchical system is summarized in Figure 3.3.

3.6 A Prototype Application to MP3 Players

In this section, we show a prototype implementing both the GRS and GSI on a data set containing MP3 players. This data set consists of 22 attributes of 321 MP3-players collected from the Dutch website `http://www.kelkoo.nl` during June 2006. The data set is of a mixed type, which means that we have both categorical and numerical attributes and contains several missing values. An overview of these data is given in Table 3.1.

A prototype of our GUI is shown in Figure 3.4. This prototype is available at `http://www.browsingmap.com/gsi.html`. The prototype is implemented as a Java Applet, which means that it can be used in a web environment. The interface uses three tabs that contain a 2D map and some buttons: The Navigate tab implementing the graphical shopping interface (GSI),

Table 3.1: Description of the MP3-player data set. The data set describes 321 MP3-players using 22 product attributes

Categorical Characteristics	Missing	Levels (frequency)
Brand	0	Creative (53), iRiver (25), Samsung (25), Cowon (22), Sony (19), and 47 other brands (207)
Type	11	MP3 Player (254), Multimedia Player (31) USB key (25)
Memory Type	0	Integrated (231), Hard Disc (81), Compact Flash (8), Secure Digital (1)
Radio	9	Yes (170), No (139), Optional (3)
Audio Format	4	MP3 (257), ASF (28), AAC (11), Ogg Vorbis (9), ATRAC3 (5), and 4 other formats (6)
Interface	5	USB 2.0 (242), USB 1.0/1.1 (66), Firewire (6), Bluetooth (1), Parallel (1)
Power Supply	38	AAA x 1 (114), Lithium Ion (101), Lithium Polymeer (45), AA x 1 (17), AAA x 2 (4), Ni Mh (3)
Remote Control	9	No (289), In Cable (13), Wireless (10)
Color	281	White (7), Silver (5), Green (5), Orange (4), Purple (4), Red (4),Pink (4), Black (4),Blue (3)
Headphone	15	Earphone (290), Chain Earphone (8), Clip-on Earphone (2), Earphone With Belt (2), No Earphone (2), Minibelt Earphone (1), Collapsible Earphone (1)

Numerical Characteristics	Missing	Mean	Stand. Dev.
Memory Size (MB)	0	6272.10	13738.00
Screen Size (inch)	264	2.16	1.04
Screen Colors (bits)	0	2.78	5.10
Weight (grams)	66	83.88	84.45
Radio Presets	9	3.06	7.84
Battery Life (hours)	40	18.63	12.56
Signal-to-Noise Ratio (dB)	247	90.92	7.32
Equalizer Presets	0	2.60	2.22
Height (cm)	28	6.95	2.48
Width (cm)	28	5.57	2.82
Depth (cm)	28	2.18	4.29
Screen Resolution (pixels)	246	31415.00	46212.00

Figure 3.4: Screenshot of the graphical user interface of the prototype of the GSI for MP3-players.

the Direct Search tab implementing the graphical recommender system (GRS), and a Saved Products tab to save products in.

In a 2D map, each product is represented by a thumbnail picture. To add a selected product to the shopping basket the user presses the *Save* button represented by a shopping cart in the GSI or the GRS tab. The *Recommend* (or play) button uses the selected product for the next step in the GSI and by pressing the same button in the GRS tab recommendations are given based on the ideal values for the product attributes and weights specified by the user. The ideal product is represented by the symbol $\otimes$ on the GRS map. The products in the Saved Products map are represented using MDS as in the other two maps. With the *Delete* button saved products can be removed from this map. A panel at the right shows the product attributes of a selected product together with a picture of this product.

Furthermore, there is a Preferences tab. In this tab the user can specify the weights that are used in the dissimilarity calculation of the GRS and GSI, that is, how important the attributes are.

Table 3.2: Description of ideal product used in Figure 3.5

Attribute	Value	Attribute	Value
Brand	Samsung	Screen Size	1.8 inch
Type	Multimedia	Screen Colors	18 bits
Memory Type	Hard-disk	Weight	100 grams
Radio	No	Radio Presets	0
Audio Format	MP3	Battery Life	12 hours
Interface	USB2.0	Signal-to-Noise	95 dB
Power Supply	Lithium Ion	Equalizer Presets	5
Remote Control	No	Height	8 cm
Color	Black	Width	4 cm
Headphone	Earphone	Depth	1 cm
Memory Size	20 GB	Screen Resolution	60000 px

Also, the user can deselect certain attributes, which means that their weights are set to 0. When changes are made in this tab, the 2D maps are immediately adapted.

The transition between two steps in the GSI is implemented in a smooth way. After the selection of a product by the user, the new products are added to the map at random positions. Then, the map is optimized using MDS. This optimization is shown to the user. When the optimization has converged, the old products are gradually made less important (using a weighted version of MDS) until they have no influence anymore. Finally, the old products are removed and the map of new products is optimized. This implementation yields smooth visual transitions, which are important for an effective GUI.

Figure 3.5 shows an example of a 2D map created by the GRS. The description of the ideal product is shown in Table 3.2. The MP3-players closest to the ideal product specification are the Samsung YH-820 and the Maxian MP2220 positioned above and below the ideal product respectively. It is perhaps surprising that the distance between these two products is one of the largest in the map. However, when we have a closer look, we see that although both MP3-players are quite similar to our ideal product description, they are quite different from each other. The Samsung YH-820 is a small and light MP3 player with limited memory size and a smaller screen than we wanted. On the other hand, the Maxian has a large screen and memory size, but it is also larger and heavier than our ideal product. The Samsung YH-925 and the 20GB versions of the Cowon iAudio and the Samsung YH-J70 are all MP3-players having a memory of 20GB as we wanted, but having worse screens than the Maxian. Conversely, these players are somewhat smaller and lighter than the Maxian. The 30GB versions of the Cowon iAudio and the Samsung YH-J70 only differ in memory size from the 20GB versions. Therefore, these MP3-players are visualized somewhat farther from the ideal product than the 20GB versions.

Figure 3.5: An example of the GRS.

3.7 Evaluation of the Graphical Shopping Interface

To test our approach, the MP3 players data introduced in the previous section are used. We study the quality of the 2D maps by considering the Stress values. Through a simulation study we evaluate how easily a consumer can find the product she wants using the GSI. Finally, we have conducted a survey among 71 subjects to test the usability of the GSI.

Representing recommended solutions in a 2D map might only be an improvement when the 2D representations are of a sufficient quality. Solutions with a low Stress value represent, at least technically, the products in a good way. One should keep in mind that a low Stress value is merely a necessary condition for a usable system, it is not a sufficient one. Since we also like to compare solutions with a different number of products, Stress is normalized by dividing by the sum of squared dissimilarities, that is,

$$\sigma_n = \frac{\sum_{i<j}(\delta_{ij} - d_{ij}(\mathbf{Z}))^2}{\sum_{i<j}\delta_{ij}^2}. \tag{3.9}$$

This normalized Stress can be interpreted as the proportion of the unexplained sum-of-squares of the dissimilarities (Borg & Groenen, 2005).

To estimate the average quality of a fit in the GSI is practically impossible, because of the interactivity and randomness in the system. However, we can say something about the goodness of the representations in the GRS. Since the user has much freedom in specifying her ideal

Table 3.3: Normalized Stress values for the experiments to determine the quality of the 2D representations in the GRS

p	Mean Normalized Stress
2	$3.36 \cdot 10^{-32}$
3	$3.13 \cdot 10^{-4}$
4	$5.77 \cdot 10^{-3}$
5	$1.21 \cdot 10^{-2}$
6	$1.96 \cdot 10^{-2}$
7	$2.63 \cdot 10^{-2}$
8	$3.16 \cdot 10^{-2}$
9	$3.62 \cdot 10^{-2}$
10	$4.05 \cdot 10^{-2}$

product, there is a very large number of possible plots in practice. We use a leave-one-out method to approximate the quality of the plots. Each time, we pick one product from the data set as the ideal product of the user and we use the other products as our product catalog. All attributes are used to compute the dissimilarities and all weights are set to 1. Then, the $p - 1$ most similar products to this ideal product are selected and a 2D map of these p products is created using MDS. This procedure is repeated, until each product has functioned once as an ideal product description. This overall procedure is done for $p = 2$ until $p = 10$. The results for the average normalized Stress values are shown in Table 3.3.

It is no surprise that solutions with only two products have a Stress of (almost) zero, since there is only one dissimilarity in that case that can always be perfectly scaled on a line. Also the solutions with three points have an average Stress value very close to zero. When we increase p, the Stress values increase, as may be expected because the problem gets more difficult. Since this increase seems almost linear, it is hard to identify the ideal product set size. If one wants near perfect solutions, p should be set to 3, but a map of three products is not very informative. For larger p, the Stress values are still acceptable, even for $p = 10$, 96% of the sum-of-squares in the dissimilarities is explained by the distances in the 2D map. Obviously, the quality of the solutions can be better or worse, when using different product catalogs.

Apart from the quality of the 2D representations, the navigation aspect was tested in the different implementations of the GSI. In an ideal system, the user will always find the product she likes most in a small number of steps. We expect that there will be a trade-off between the number of steps that is necessary to find the product and the probability that the consumer will find the product she likes.

To evaluate the navigation aspect of the different systems in a simulation, some assumptions need to be made about the navigation behavior of the user. First, we assume that the consumer implicitly or explicitly can specify what her ideal product looks like in terms of its attributes. Second, we assume that the user compares products using the same dissimilarity measure as the system uses (using all attributes and all weights set to 1). Finally, it is assumed that in each step the consumer chooses the product that is most similar to the ideal product of the consumer. Note

that we only evaluate the search algorithm in this way and not the complete interface, since the results would have been the same when the results were shown in a list.

We use a leave-one-out procedure to select the ideal product descriptions of the user. A random ideal product description is not used, since such a procedure will create a lot of ideal products that do not exist in reality. Each time, one product is selected as the ideal product of the consumer and all other products are used as the product catalog. We repeat this until every product is left out once. We evaluate the three different implementations (the random, the clustering, and the hierarchical system) with p set to 4, 6, 8, and 10. For the random and clustering system we also vary the parameter α by setting it to the values 0.2, 0.4, 0.6, and 0.8. Before starting a single experiment, we determine which product in the product catalog is most similar to the product we left out. During each step in a single experiment, we use the assumptions above to compute the product the user will select. We stop when the most similar product is in D_t^* that is shown to the user or when the system terminates. A quite similar evaluation procedure was used by McGinty and Smyth (2002).

For the different systems and specifications Tables 3.4, 3.5, and 3.6 show the percentages of success, the percentage of successes during the first five steps of the process, and the average number of steps the system uses before it stops. The random system in Table 3.4 performs better for larger p as expected: the percentage of successes is higher and the average number of steps lower. As the quality of MDS representations reduces with increasing p, it becomes more difficult for the user to get an overview of the product space. Also, there is a trade-off between the number of steps necessary to find a product and the probability to find the product. For example, a random system with $p = 10$ and $\alpha = 0.8$ finds the correct product in 84% of the cases, but needs on average almost 10 steps. In this case, after 5 steps in only 20% of the cases is the correct product found. However, a system with $p = 10$ and $\alpha = 0.4$ has a success after 5 steps in 57% of the cases, but the total success rate is 59%. The average number of steps for this system is also 4.8.

The clustering systems perform overall worse than the random systems and seem, therefore, not to be an alternative. However, the hierarchical systems, especially the one with $p = 10$, show similar performance as the random systems with a small α.

Tables 3.4, 3.5, and 3.6 only showed the success rates, but did not say anything about the cases where the most similar product was not recommended to the user. Therefore, we have also counted how many times the second, third etc. most similar product was recommended. This information is summarized for the random and hierarchical system in Tables 3.7 and 3.8. They show that many misrecommendations of the systems are recommendations of products that are quite similar to the ideal product. Looking at the top 5 of most similar products, these products are recommended in up to 94% of the cases to the consumer. In many systems that desire only a small number of steps, like the hierarchical system, products 2 until 5 are recommended in one fifth of the cases to the user. In many cases, these products have a not much higher dissimilarity than the most similar product and in some cases their dissimilarity is the same. Therefore, recommending one of these products instead of the most similar one, will not make the recommendation much worse.

Since a simulation study only shows limited theoretical results, we also performed a usability

Table 3.4: Results for different specifications of the random system

p	α	Successes	In 5 Steps	Average number of Steps
4	0.2	16.8%	16.8%	5.02
	0.4	29.3%	19.3%	6.38
	0.6	41.7%	10.0%	9.12
	0.8	56.1%	5.9%	15.99
6	0.2	29.3%	28.7%	4.77
	0.4	42.7%	34.9%	5.83
	0.6	52.7%	17.1%	8.01
	0.8	70.1%	10.9%	13.12
8	0.2	43.0%	42.7%	4.34
	0.4	47.0%	43.6%	5.29
	0.6	66.4%	30.5%	6.96
	0.8	80.1%	16.2%	11.22
10	0.2	46.4%	46.4%	4.09
	0.4	58.9%	56.7%	4.82
	0.6	70.4%	37.1%	6.27
	0.8	83.8%	19.9%	9.92

study to get opinions of potential users on the GSI and to find directions for improvement of the GSI. The usability study was carried out by means of a web survey in which the GSI was integrated and was filled out by the respondents in the controlled environment of a behavioral lab at Erasmus University Rotterdam. The 71 respondents who filled in the questionnaire were mainly business or psychology students at Erasmus University Rotterdam.

The respondents were asked to perform two tasks with both the GSI and a traditional interface we developed ourselves. This interface enables a user to restrict attribute values and displays the results in a list. This interface is available at `http://www.browsingmap.com/gsi.html`. In total there were four tasks and these were randomly assigned to the two interfaces for each user. Also, the order in which the users had to use both interfaces was randomized.

In the usability study, we used an implementation of the random GSI system, since this implementation performed best in the simulation study. The system parameters were set to $p = 8$ and $\alpha = 0.5$. During the study the GRS tab was not incorporated in the interface, so that only the GSI could be used to perform the tasks.

After carrying out their shopping tasks, the respondents were asked to rate the satisfaction with the product that was chosen on a 5-point scale. A t-test of these data showed that there was no significant difference in satisfaction between the two interfaces, despite the fact that all users were far more familiar with the traditional interface than with the GSI. The tasks took a little more time when the GSI was used than when the traditional interface was used. We suspect that one reason for this is that users were still learning how to use the GSI.

Furthermore, the survey asked the participants for qualitative feedback regarding the GSI. The weaknesses of the GSI that were reported can be divided in two subsets. The first set of

Table 3.5: Results for different specifications of the clustering system

p	α	Successes	In 5 Steps	Average number of Steps
4	0.2	14.6%	14.6%	4.85
	0.4	20.3%	12.2%	6.68
	0.6	19.3%	8.7%	8.90
	0.8	13.7%	10.0%	7.83
6	0.2	22.1%	21.2%	4.83
	0.4	32.7%	27.1%	6.17
	0.6	44.9%	19.0%	8.30
	0.8	25.9%	11.5%	9.10
8	0.2	31.5%	29.9%	4.57
	0.4	38.9%	35.5%	5.38
	0.6	60.4%	25.6%	7.34
	0.8	45.8%	13.7%	10.20
10	0.2	39.6%	38.9%	4.34
	0.4	50.5%	45.5%	5.04
	0.6	72.6%	29.6%	6.40
	0.8	68.2%	16.2%	11.02

Table 3.6: Results for different specifications of the hierarchical system

p	Successes	In 5 Steps	Average number of Steps
4	47.4%	11.5%	8.03
6	47.7%	18.4%	5.69
8	48.9%	24.6%	5.02
10	52.3%	38.0%	4.10

Table 3.7: Proportions of cases that the ranking of the recommended product was in the specified ranges for the random system

		ranking ranges							
p	α	1	≤ 2	≤ 3	≤ 5	≤ 10	≤ 25	≤ 50	≤ 100
4	0.2	16.8%	24.6%	29.9%	36.5%	47.0%	66.7%	84.4%	95.0%
	0.4	29.3%	36.5%	42.4%	49.5%	60.1%	78.5%	91.0%	99.4%
	0.6	41.7%	50.5%	58.9%	64.2%	75.7%	87.9%	94.4%	100.0%
	0.8	56.1%	67.3%	73.2%	80.4%	89.4%	96.0%	98.4%	99.7%
6	0.2	29.3%	40.8%	45.2%	49.2%	60.4%	75.1%	86.6%	96.0%
	0.4	42.7%	50.5%	57.9%	64.5%	74.1%	88.5%	96.0%	97.8%
	0.6	52.7%	64.8%	69.2%	75.7%	84.1%	91.9%	96.9%	97.8%
	0.8	70.1%	80.1%	82.2%	86.3%	94.1%	97.8%	99.7%	100.0%
8	0.2	43.0%	54.8%	59.5%	64.5%	75.4%	89.1%	95.3%	98.8%
	0.4	47.0%	58.6%	63.9%	71.3%	77.3%	90.0%	94.7%	98.1%
	0.6	66.4%	76.0%	79.4%	84.7%	91.0%	96.6%	97.5%	98.8%
	0.8	80.1%	87.2%	91.0%	93.2%	97.5%	99.4%	99.4%	99.4%
10	0.2	46.4%	56.1%	60.8%	67.0%	79.8%	92.5%	96.6%	99.1%
	0.4	58.9%	70.4%	74.8%	80.4%	91.0%	95.3%	98.8%	100%
	0.6	70.4%	76.0%	81.6%	87.5%	93.2%	97.2%	98.4%	98.8%
	0.8	83.8%	89.7%	92.2%	93.8%	96.0%	98.8%	99.1%	99.1%

Table 3.8: Proportions of cases that the ranking of the recommended product in the specified ranges for the hierarchical system

	ranking ranges							
p	1	≤ 2	≤ 3	≤ 5	≤ 10	≤ 25	≤ 50	≤ 100
4	47.4%	56.4%	64.2%	71.3%	77.6%	85.4%	89.7%	95.6%
6	47.7%	55.1%	62.3%	70.1%	76.0%	85.4%	90.3%	95.6%
8	48.9%	56.4%	62.3%	70.4%	77.3%	86.9%	94.1%	97.5%
10	52.3%	58.6%	67.9%	74.8%	81.6%	90.0%	93.8%	98.8%

weaknesses concerns the perceived complexity of the GSI. For example, some users found it difficult to specify weights on the preferences tab, or found the GSI to be confusing as a whole. We suspect that these weaknesses are partially caused by limited familiarity with the GSI. The other set of weaknesses sometimes mentioned by respondents were caused by functionality that was missing in the evaluated prototype of the GSI. For instance, users missed a way to restrict the search space to a subset of products using a crisp query and the possibility of a text search.

As strengths of the GSI, the respondents reported the following aspects. First of all, many users mentioned that they found the GSI more fun to use than the traditional interface. Another positive aspect of the GSI that was mentioned was that people felt to have a better overview of the presented products, since the GSI repeatedly represents a small number of products in an organized map. Finally, some users liked it that the GSI suggested products that were useful, but previously unknown to the user.

3.8 Conclusions and Discussion

In this chapter, we presented two recommender systems which both use 2D maps to represent products in. Products that are similar to each other, based on their product attributes, are represented close to each other in the 2D representation. The difference between the two systems is that the GRS uses explicit input from the user, whereas the GSI can be used to navigate through the product space. Both were combined in a prototype application for MP3 players.

Some simulation tests were performed to evaluate the quality of the 2D representations and the navigation behavior in the different implementations of the GSI. The first type of test showed that the quality of the 2D representations in the GRS is acceptable at least up to 2D maps with 10 products, but as expected the quality of the representations became less when p was increased. The navigation tests showed that there is a trade-off between the number of steps a user needs to find the best product in a certain implementation of the GSI and the probability that the user will find the best product. Results of the implementation with a clustering in each step were worse than both the random system and the hierarchical system, implying that the clustering method is not good enough to be applied in practice. To be quite certain that the best product eventually will be found, the random system with a high value for α should be preferred. If a high probability of successes in the first few steps is preferred, then one should choose a random system with a low α or a hierarchical system.

A usability study was performed, which showed that people were equally satisfied with the products they chose in four tasks using the GSI as they were when using a traditional interface, with which they were more familiar. Reported weaknesses of the GSI were the relative high complexity and the lack of some functionality. We expect that the experienced complexity is partially caused by the unfamiliarity with the system. The lack of functionality might be solved by integrating parts of more traditional systems in the GSI, such as crisp selection and search options. Strengths of the GSI that were mentioned were the fun-factor, good representation of products, and the recommendation of previously unknown products.

Both systems show that it is possible to combine 2D product representations with recommender systems. However, several possibilities remain for extending and possibly improving

the systems. Besides some practical improvements that were mentioned by users during the usability study, such as the possibility to preselect a subset of products, there are also some other interesting extensions that can be made.

The first extension would be to include user or rating data. The user data could contain user account information, which means that the similarity between users is used in the recommendation process. The ratings could be given directly by the user or can be based on saved or purchased products. When product ratings are incorporated in the recommender system the weights in the dissimilarity measure can be learned by the system for the total consumer population or even for individual consumers. In a CBR-RS for traveling (Arslan, Ricci, Mirzadeh, & Venturini, 2002), for example, a weighting approach based on frequencies of features previously used in a query was used.

The GSI can also be further improved by not only allowing the consumer to select a product, but by allowing the consumer to select any point on the map. With the use of external unfolding (Borg & Groenen, 2005) an invisible product can be found that is closest to the selected point and used in the next iteration. This technique is, for example, used in an application to compare roller skates by Stappers et al. (2000).

Part II

Advances in Recommender Systems

Chapter 4

Determination of Attribute Weights for Recommender Systems Based on Product Popularity*

4.1 Introduction

Over the past fifteen years, a very large number of approaches has been proposed which can be used to recommend products from a product catalog to users. These so-called recommender systems (Resnick & Varian, 1997) can be subdivided in three groups (Adomavicius & Tuzhilin, 2005): Collaborative, content-based, and hybrid recommenders. Where collaborative filtering, the most popular method, recommends products based on similarity of the user's taste with the taste of other users, content-based recommenders use characteristics of products to base their recommendations on. Hybrid recommenders combine both approaches.

Although collaborative recommendation methods seem to be the most popular type in practice and in the literature, content-based methods are far more useful in certain areas of electronic commerce, such as consumer electronics, real estate, and tourism. In these areas, products can usually be described by a well-defined set of attributes and it would be wasteful not to use these attributes when recommending. Moreover, often only very limited choice- or preference data are available, since customers buy these kinds of products infrequently. This makes it impossible to discover and exploit correlation structure in user preferences, as is typically done by collaborative recommenders. Furthermore, using attributes alleviates the cold start item problem: since attribute values are known from the start, new products can immediately be used in the recommendation cycle, whereas collaborative methods have to wait until sufficient co-preference data has been gathered. Finally, attribute-based recommendations open the door for explaining to the user why certain recommendations are given, thus making the recommendation process more transparent. Various sources (see e.g., Herlocker, Konstan, & Riedl, 2000; Sinha & Swearingen,

*This chapter is partly based on Kagie, Van Wezel, and Groenen (2008a).

2002; Tintarev & Masthoff, 2007) suggest that users favor transparent recommendations over nontransparent ones.

Many content-based recommender systems use some type of case-based reasoning or nearest neighbor retrieval (Lorenzi & Ricci, 2005; McSherry, 2002; O'Sullivan, Smyth, & Wilson, 2005). These techniques rely heavily on some dissimilarity measure between different products for their recommendation strategy. Usually, this dissimilarity measure is based on the attributes of the products. However, not all attributes of an product are equally important to the user and, thus, this asymmetry should be reflected in the dissimilarity measure. Therefore, to avoid a mismatch of the system and the perceived similarity, the dissimilarity measure should use a suitable attribute weighting, thus avoiding wrong recommendations by the system.

Although weights are generally specified by experts, some work has been done on recommender systems that automatically learn these weights on a per-user basis, such as systems based on MAUT-based preference models (Jameson, Schäfer, Simons, & Weis, 1995). For example, Schwab, Pohl, and Koychev (2000) learn user specific weights for binary features using significance testing assuming normal distributions. When the user selects items having a specific attribute value more often, that is, there is a significant effect, this attribute got a higher weight. Arslan, Ricci, Mirzadeh, and Venturini (2002) use the number of times an attribute was used in the query of the user to learn these attribute weights. Branting (2004) uses an algorithm that changes weights based on the set of items recommended and the selection of the user of one of these items. Finally, Coyle and Cunningham (2004) compare the final choice of the user with the provided recommendations and learn the attribute weights from that.

All these approaches assume that the user gives the system time to let it learn his/her preferences in one or more sessions. However, in many e-commerce domains, such as durable goods, this is not a realistic assumption, due to the infrequent purchases and subsequent data sparsity mentioned above. Although it might be possible to adapt the weighting approaches above to a nonpersonalized setting so that the use of pooled data solves the data sparsity problem, these approaches remain incapable of handling mixed data (that is, consisting of both numerical and categorical values) and missing values, both of which are usually abundant in electronic commerce product catalogs.

In summary, attribute-based recommendation is relevant because it is useful in e-commerce domains with sparse preference data and rich attribute data, and it allows for more transparency in the recommendation process. It generally employs an attribute-based dissimilarity measure. Since it is likely that some attributes are more relevant than others in users' perceptions of product similarity, the dissimilarity measure used by the recommender should match the perceived attribute importances. Thus, the quest is to find (1) a way for determining attribute weights that match similarity perceived by users from the available catalog and choice data with their many missing values, their mixed attributes and their sparse observations, and (2) a way to evaluate a given set of attribute weights.

In this chapter, we introduce two methods to determine attribute weights for dissimilarity in recommender systems. These methods only rely on some measure of product popularity as 'target' attribute, while the 'input' attributes are the product characteristics. We show how these weighting methods can be used with mixed data and missing values. We then evaluate the

weights that are produced by these methods in two novel ways. The first evaluation method is based on clickstream analysis and the second one is based on a survey.

The first weighting method we discuss is based on Poisson regression (McCullagh & Nelder, 1989; Nelder & Wedderburn, 1972) and was introduced by Kagie, Van Wezel, and Groenen (2008a). It uses multiple imputation (Rubin, 1987) to handle missing values and dummy variables to handle categorical attributes. The weights are determined using t-values of the regression coefficients. The second weighting method is based on boosting (Friedman, 2001) and has some advantages over the Poisson regression method, since it handles missing values and categorical attributes in a more natural way. Also, there is a straightforward method to determine attribute importance for boosting (Friedman, 2001) and it is more flexible.

In the first evaluation, we use clickstream logs of four real life electronic commerce product catalogs, to see which products are often visited together in a session. These co-visits give rise to empirical similarities between product pairs: we assume that products frequently visited together are considered to be similar by users. Therefore, a dissimilarity measure based on attributes should also identify these as similar and thus a vector with attribute weights can be evaluated by considering how well the attribute-based similarities match the observed empirical similarities. To measure this 'match', we introduce a measure called the mean average relative co-occurrence (MARCO).

The second evaluation method is based on a user experiment consisting of an online survey. In this survey, respondents were given a reference product. Then, they were asked to indicate which products they considered to be relevant recommendations given the reference product. The weighted dissimilarity measure should be able to recommend these relevant products. To measure the amount of overlap between the recommendations by the dissimilarity measure and the relevant recommendations, we propose the mean average relative relevance (MARR).

We use the MARCO and MARR measures to evaluate the weights yielded by both the Poisson regression- and the Boosting based weighting methods. The performance of both methods is benchmarked against the naive (but often used) method of weighting each attribute equally.

The remainder of this chapter is organized as follows. In the next section, we introduce the notion of dissimilarity in recommender systems and the dissimilarity measure we use in this chapter. Then, in Section 4.3, we introduce the two weighting approaches based on product popularity. In Section 4.4, these two approaches are applied to four real life electronic commerce product catalogs. Sections 4.5 and 4.6 discuss both evaluation approaches based on clickstream logs and a survey. Finally, we draw conclusions in Section 4.7.

4.2 The Use of Dissimilarities in Recommender Systems

A large group of content-based recommender systems, the so-called case-based recommender systems (O'Sullivan et al., 2005) or case-based reasoning recommender systems (Lorenzi & Ricci, 2005) rely on a dissimilarity measure based on the attributes of the products to provide

recommendations. In general, we can define such a measure as

$$\delta_{ij} = f\left(\sum_{k=1}^{K} w_k \delta_{ijk}\right) \quad , \tag{4.1}$$

where δ_{ij} is the dissimilarity between product i and j, w_k is the attribute weight for attribute k, δ_{ijk} is a dissimilarity score measuring the difference between product i and j on attribute k, and $f(\cdot)$ is a monotone increasing function. Note that often $f(\cdot)$ is the identity function.

In case-based recommender systems, the use of a dissimilarity measure has the advantage that it can be used for recommendation in two different situations. First, when a reference product is available (for example, the product the user is looking at or has selected in some way), recommendation is based on the dissimilarity between this product and other products. A second way to use a dissimilarity for recommendation is when the user specifies a search query in the form of an (incomplete) ideal product specification, we can provide recommendations. These recommendations are based on the dissimilarity between the ideal product specification and the products in the product catalog.

Although our approach to determine attribute weights can be used with every dissimilarity measure that can handle linear weights, it is vital to apply a good dissimilarity measure, that is, one that is close to the user's notion of dissimilarity. In the case of electronic commerce product catalogs, a dissimilarity measure should be able to handle missing values and attributes of mixed type. Often used (dis)similarity measures, like the Euclidean and Hamming distance, Pearson's correlation coefficient, and Jaccard's similarity measure, lack these abilities. Many dissimilarity measures in the domain of knowledge-based recommender systems need domain knowledge (Burke, 2000; Coyle & Cunningham, 2004; Coyle, Doyle, & Cunningham, 2004; McSherry, 2002). We would like to avoid this, such that the approach is more flexible.

To our knowledge, only two dissimilarity measures previously used in electronic commerce applications can handle both mixed attribute types and missing values and do not need any domain knowledge. The first are measures based on the heterogeneous Euclidean overlap metric (HEOM, see Wilson & Martinez, 1997) as used by Arslan et al. (2002); Ricci and Del Missier (2004); Sandvig and Burke (2005). The second is a modified version of the Gower's general coefficient of similarity (Gower, 1971), which has been used by Kagie, Van Wezel, and Groenen (2007, 2008a, 2008b).

HEOM and the adapted Gower coefficient both compute dissimilarity scores for all attributes separately and finally combine these. The computation of these scores differ between attribute types. When HEOM is not able to compare two attribute values, because one or both of them is missing, it will treat them as dissimilar from each other. The adapted Gower coefficient ignores the attribute and computes the dissimilarity based on the nonmissing dissimilarity scores, which is, in our opinion, a better approach. Also, the adapted Gower coefficient has the advantage that the dissimilarity scores are normalized such that each attribute is equally important in the dissimilarity measure. We will use this dissimilarity measure in the remainder of this chapter. For more details on the definition of the adapted Gower coefficient, we refer to Appendix 4.A.

4.3 Determining Attribute Weights

In this section, we will introduce two methods to determine attribute weights based on product popularity, such as sales or pageviews. The first method is based on Poisson regression in which we use multiple imputation to handle missing values. Weights are based on the t-values of the Poisson regression coefficients. We consider a full Poisson regression model and a stepwise model that has a built-in method for attribute selection.

The second method is based on a new boosting algorithm, PoissonTreeBoost, that optimizes Poisson deviance. The relative importance measure is used to determine weights. This method is more flexible than Poisson regression. Also, it has built-in methods to handle categorical attributes and missing values.

In these two methods, the popularity counts for the I products are used as dependent/target attribute vector $\mathbf{y} = (y_1, y_2 \ldots, y_I)$. This target attribute is due to its nature a count variable being discrete and nonnegative and, therefore, different models than ordinary least squares regression models should be used. As independent variables we use, naturally, the attribute vectors $\{\mathbf{x}_i\}_1^I$, that is, $\mathbf{x}_i = (x_{i1}, x_{i2} \ldots, x_{iK})$, with K the number of attributes. The counts in $\mathbf{y}$ and the attributes will be used to determine weights $\{w_k\}_1^K$ for the product attributes in some dissimilarity measure.

4.3.1 Poisson Loglikelihood and Deviance

Models for count data, such as Poisson regression (McCullagh & Nelder, 1989; Nelder & Wedderburn, 1972) which we will use in Section 4.3.2, maximize the loglikelihood of the Poisson distribution

$$\log \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{I} [-\hat{y}_i + y_i \log \hat{y}_i - \log y_i!] \ , \tag{4.2}$$

where $\hat{y}_i$ are the model predictions and $y_i!$ is the factorial of y_i. An alternative approach, which is also used by the Poisson regression tree (Therneau & Atkinson, 1997) and the boosting method we discuss in Section 4.3.3, is minimizing the Poisson deviance, which is defined as

$$\begin{aligned} D(\mathbf{y}, \hat{\mathbf{y}}) &= -2(\log \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) - \log \mathcal{L}(\mathbf{y}, \mathbf{y})) \\ &= 2\left(\sum_{i=1}^{I} \left[y_i \log\left(\frac{y_i}{\hat{y}_i}\right) - (y_i - \hat{y}_i) \right]\right) \\ &= 2\sum_{i=1}^{I} L^{PD}(y_i, \hat{y}_i) \ , \end{aligned} \tag{4.3}$$

which equals minus two times the loglikelihood of the model minus the loglikelihood of a model perfectly fitting the data. In this equation, L^{PD} denotes the Poisson deviance loss function. Note that $D(\mathbf{y}, \hat{\mathbf{y}})$ has the advantage that it is a minimization problem and a deviance of zero represents a perfect model (on the training data).

4.3.2 Poisson Regression

In Kagie et al. (2008a), we introduced one way in which attributes can be determined based on product popularity, namely using a Poisson regression model, which is a member of the generalized linear model (GLM) framework (McCullagh & Nelder, 1989; Nelder & Wedderburn, 1972). To be able to estimate a Poisson regression model on our data, we set up the matrix $\mathbf{X}$ of predictor variables as follows. Categorical attributes are replaced by series of dummy variables as is common in the literature on GLMs. Every categorical attribute is represented by L_k dummies $x_{ik\ell}$, which are 1 for the category where the product belongs to and 0 for all other attributes. To avoid multicollinearity, the last category is not represented by a dummy, so that L_k is the number of different categories for attribute k minus one. For multi-valued categorical attributes the same approach is used, only now all categories are represented by the L_k dummies. For numerical attributes we use the original value. Hence, $x_{ik} = x_{ik1}$ and $L_k = 1$. We collect all $x_{ik\ell}$ for product i in vector $\mathbf{x}_i$. Also, an intercept term x_{i0} is incorporated in this vector, which equals 1 for all products, so that $\mathbf{X}$ has I rows and $(1 + \sum_{k=1}^{K} L_k)$ columns. The Poisson regression model is defined as

$$y_i \approx \exp(\mathbf{x}_i'\mathbf{b}) \ , \tag{4.4}$$

where y_i is a dependent count variable (in our case product popularity or sales) and $\mathbf{b}$ is a $(1 + \sum_{k=1}^{K} L_k)$ by 1 vector of regression coefficients. Additionally, y_i is assumed to have a Poisson distribution having expectation $E(\exp(\mathbf{x}_i'\mathbf{b}))$. The regression coefficients can be estimated in this model by adapting the loglikelihood equation (4.2) to

$$\log \mathcal{L}(\mathbf{b}) = \sum_{i=1}^{I} [-\exp(\mathbf{x}_i'\mathbf{b}) + y_i \mathbf{x}_i'\mathbf{b} - \log y_i!] \ , \tag{4.5}$$

which is often maximized using an iteratively reweighted least squares procedure. This algorithm produces both the model parameters $b_{k\ell}$ and their standard errors σ_{kl}.

A disadvantage of Poisson regression models (and GLMs in general) is that they lack a built-in way to handle with missing values. In a recent paper, Ibrahim, Chen, Lipsitz, and Herring (2005) compared different methods to handle missing values in combination with GLMs and found that multiple imputation (MI, see Rubin, 1987) was among the best methods to be used in this situation. MI methods create a number of 'complete' data sets in which values for originally missing values are drawn from a distribution conditionally on the nonmissing values. There are two methods to create these imputed data sets: Data augmentation (J. L. Schafer & Olsen, 1998) and sampling-importance-resampling (King, Honaker, Joseph, & Scheve, 2001). Both lead to results of identical quality, while the latter does this much faster. Therefore, we use that algorithm, which is available in the Amelia II package (Honaker, King, & Blackwell, 2008) for the statistical software environment R, in our approach. For a more detailed discussion on how regression coefficients and standard errors are computed using MI, we refer to Kagie et al. (2008a).

The weights to be used in the dissimilarity measure are based on the regression coefficients $b_{k\ell}$. However, we cannot use these coefficients directly as weights for several reasons. First, all attributes are on different scales and this also holds for the coefficients. Second, any uncertainty

about the coefficient is not taken into account. Although a coefficient value can be reasonably large, this does not necessarily mean we are also certain about the value of this coefficient. Finally, coefficients can also be negative, while weights should always be positive. The first two problems can be overcome by using the t-value of $b_{ik\ell}$ which is defined as

$$t_{k\ell} = \frac{b_{k\ell}}{\sigma_{k\ell}} \ , \tag{4.6}$$

while the second can be solved by using the absolute value $|t_{k\ell}|$.

However, since we had to use dummy variables for (multi-valued) categorical attributes, the weight for these attributes is based on the average over the corresponding t-values. Hence, we can define a 'pseudo' t-value v_k as

$$v_k = \frac{1}{L_k} \sum_{\ell=1}^{L_k} |t_{k\ell}| \ . \tag{4.7}$$

For numerical attributes v_k just equals the corresponding absolute t-value. Finally, we normalize the weights to have a sum of 1

$$w_k = \frac{v_k}{\sum_{k'=1}^{K} v_{k'}} \ . \tag{4.8}$$

Note that, instead of using t-values other approaches could be taken to determine weights based on a Poisson regression model. For example, one could first normalize all attributes and then use the absolute values of $\mathbf{b}$ to determine the weights. Preliminary experimentation did not show improvements over using (4.7). Therefore, we do not pursue this approach any further.

Besides a Poisson regression model using all attributes, also a stepwise Poisson regression model was discussed in Kagie et al. (2008a) to determine attribute weights. Stepwise models use the statistical significance of the attributes to do attribute selection. Each time the most insignificant attribute (based on the t-values) is deleted from the model specification and a new model is estimated until only significant attributes remain. In our evaluation, we will consider both the stepwise and the complete Poisson regression approach.

4.3.3 Boosting Poisson Deviance

Although the procedure based on Poisson regression is simple to use, it has some drawbacks. Poisson regression models have no integrated way to handle missing values (while product catalogs, in practice, contain a lot of missing values), due to which a computational intensive imputation techniques (see Rubin, 1987) need to be applied. Furthermore, (multi-valued) categorical attributes need to be preprocessed prior to be included into the model. Finally, Poisson regression models cannot handle interaction effects (without dramatically increasing the attribute space or using some kernel method) and, therefore, may possibly miss attributes that are important, but only in combination with other attributes.

Therefore, we introduce a model that has integrated ways to handle missing values, categorical attributes, and interactions and is also able to handle a count variable as target attribute. This

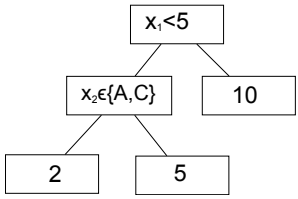


Figure 4.1: Example of a Regression Tree: The root divides the data in two parts using numerical attribute x_1. Instances having a x_1 value smaller than 5 turn left, the other instances right. The instances that have turned right get a y-value of 10, which is the sample mean of training instances belonging to this node. The instances that have turned left are again split using categorical attribute x_2. When x_2 of an instance belongs to category A or C, this instance turns left.

method is based on boosting (Freund & Schapire, 1996, 1997), a method to combine a series of base models, which are often decision trees (Breiman, Friedman, Olshen, & Stone, 1983). First, we briefly introduce these decision trees.

Decision trees are a popular method for both classification and regression, where trees applied to the latter are often called regression trees. Regression trees minimize a squared loss function

$$B = \arg\min_B \sum_{i=1}^{I} L^S(y_i, B(\mathbf{x}_i)) = \arg\min_B \sum_{i=1}^{I} (y_i - B(\mathbf{x}_i))^2 \ , \tag{4.9}$$

where $B(\mathbf{x})$ denotes the prediction of tree B for input vector $\mathbf{x}$ and L^S is the squared loss function. This function is minimized in such a way that this results in a tree with non-terminal nodes containing split criterions and terminal nodes with predicted y-values. An example of such a tree is shown in Figure 4.1. Regression trees have the ability to deal with categorical attributes as is shown in Figure 4.1. In addition, regression trees are able to automatically handle missing values. Different methods exist to handle missing values in decision trees, see for example (Hastie, Tibshirani, & Friedman, 2001; Ripley, 1996). A convenient way to handle them is using surrogate splits (Hastie et al., 2001), introduced by Breiman et al. (1983). We use this method to handle missing values in our application.

A drawback of decision trees is their instability. The implemented model depends heavily on the data set used for model creation, and a small change in the data may have large consequences for the model. Ensemble methods, such as bagging (Breiman, 1996), random forests (Breiman, 2001), and boosting (Freund & Schapire, 1996, 1997), have a stabilizing effect by averaging over a number of decision trees.

Optimizing a squared loss function works well in most regression problems. However, when y is a count variable as in our problem, assuming squared loss is not valid. Therefore, a Poisson regression tree (Therneau & Atkinson, 1997) has been introduced that minimizes Poisson deviance (see (4.3)). However, Poisson regression trees suffer from the same problem as ordinary regression trees: instability. Therefore, we introduce a boosting method minimizing the Poisson deviance.

Boosting is a method to combine multiple models, called base learners, into a single model. All base learners are of the same type and each subsequent model tries to minimize the error made by the previous base learners. Boosting was originally developed for classification problems by Freund and Schapire (1996, 1997). Friedman (2001) developed a framework called Gradient-TreeBoost which can be used to use boosting in combination with all kinds of loss functions, such as squared, absolute, and Huber loss.

GradientTreeBoost (Friedman, 2001) is inspired by gradient based optimization techniques. Similar to these techniques GradientTreeBoost takes steps in the negative gradient direction. Traditional gradient based model fitting does this by refining a parameter vector, and thus indirectly improving the outputs of a model with that parameter vector. In contrast, boosting directly improves the performance of a (composite) model by adding an additional model to it, such that the fit of the composite model on the training data improves.

Often, the base learners are regression trees. These regression trees are ordinary regression trees minimizing a squared loss function, irrespective of the loss function that is minimized by the GradientTreeBoost algorithm. In practice, this means that the base learners are fitted on data sets $\{\mathbf{x}_i, \tilde{y}_i\}_1^I$ in which $\tilde{y}_i$ is a pseudo target replacing the original target y_i. This $\tilde{y}_i$ is determined by the negative gradient of the loss function given the current estimate of y_i (provided by the previous base learners) and the real value of y_i. GradientTreeBoost algorithms have two parameters that need to be set: the number of base learners N and a learning rate parameter ν which indicates how much the algorithm should learn from the base learners. Often, these parameters are determined using cross validation. For a more detailed explanation of GradientTreeBoost, we refer to Friedman (2001).

To create a boosting model for a count variable, we use a new member of GradientTreeBoost family minimizing the Poisson deviance loss function L^{PD} introduced in (4.3). The derivation of this method which we call PoissonTreeBoost is shown in Appendix 4.B.

We introduced the PoissonTreeBoost model as a way to determine weights for attributes, which can subsequently be used in a dissimilarity measure. To specify these weights we need a measure that indicates how large the influence of a certain attribute is on the predictions of the model. For this, we use an importance measure for GradientTreeBoost introduced by Friedman (2001), which is based on the importance measure developed for CART (Breiman et al., 1983). Since this importance measure is applicable to all members of the GradientTreeBoost family (all implementing different loss functions), it is also valid in our case, when using the Poisson deviance loss function. It is based on the quantity (how many times is the attribute used to split) and quality (does it split the data in two almost equally sized parts) of the splits based on a certain attribute. For a detailed description of this importance measure, we refer to Friedman (2001). For numerical and categorical attributes this procedure works fine. For multi-valued categorical attributes (which can have multiple values for one attribute), a binary attribute is incorporated in the boosting model for each category. The weight is then based on the total importance of all these categories.

4.4 Determining Weights for Four Real-Life Product Catalogs

In this section, the two approaches to determine attribute weights for our dissimilarity measure are applied to four product catalogs of an electronic commerce website. We first give a short description of the product catalogs. Then, we compare the performance of the PoissonTreeBoost algorithm with two benchmark models to justify our model choice followed by a description of the weights resulting from our approach. In Sections 4.5 and 4.6, we evaluate these weights in two ways: The first is based on a clickstream analysis, while the second is based on a user experiment.

4.4.1 Data

Both the product catalogs and the clickstream log files were made available to us by the Dutch internet company 'Compare Group' that hosts, among other European price comparison sites, the Dutch price comparison site `http://www.vergelijk.nl`. The product catalogs used are based on a database dump of this site (October 2007). The log files are used to count how many times users clicked on a link to an internet shop to buy a certain product, which is called an 'outclick'. We counted the outclicks during two months from July 15 until September 15, 2007. These outclicks are used as product popularity in this chapter, but other measures for product popularity can be used as long as they are count variables, such as, for example, sales or pageviews. We have used product catalogs containing four types of products: MP3 players, Digital Cameras, Notebooks, and Microwave Ovens. Some characteristics of these catalogs are described in Table 4.1. As can be seen in this table all these product catalogs have about the same size, which is a typical size for catalogs of these kinds of products. All have a reasonable high number of attributes of which a substantial number are (multi-valued) categorical and a lot of these attributes contain many missing values.

4.4.2 Model Performance

In this section we compare the model fits of our regression models, both in-sample and out-of-sample. It is worth repeating that the aim of this chapter is not to find a well-fitting model but rather to find attribute weights that match user-perceived attribute importances. This requires a different evaluation than considering model fit, and we will turn to that in Section 4.5 and 4.6. However, it is reasonable to assume that a model that fits the popularity data well gives rise to an attribute weight vector that closely resembles user-perceived attribute importances. To investigate this hypothesis, we included the current section on model performance.

On all of the four product catalogs we have fitted the Poisson regression models and the PoissonTreeBoost algorithm. For the Poisson regression model, we use the same procedure as in Kagie et al. (2008a). To be able to estimate the missing values by multiple imputation, we delete attributes having more than 50% missing values and dummies with less than 10 ones. Furthermore, we used 25 imputations of the data. As with all members of the GradientTreeBoost family,

Table 4.1: Description of the product catalogs used in this chapter

Product Catalog	MP3 Players	Digital Cameras	Notebooks	Microwave Ovens
Number of Products (I)	228	283	206	221
Number of Outclicks per Product				
Mean	109	321	72	23
Median	14	78	16	9
Maximum	5652	7871	1417	295
Number of Attributes				
Numerical	16	21	14	14
Categorical	20	12	10	14
Multi-valued categorical	9	7	4	2
Total (K)	45	40	28	30
Missing Values				
Percentage Missing	63%	39%	51%	39%

three parameters need to be set in the PoissonTreeBoost algorithm: the number of base learners N, the learning rate ν, and the size of the regression trees used. Following Friedman (2001), we set the learning rate ν to 0.1. Lowering the learning rate (and correspondingly increasing the number of base learners) did not lead to better results. We chose to use decision trees having a maximum depth of 6. This lead to better results than when using smaller trees, while using larger trees did not substantially improve results. The number of base learners was determined using cross validation. Note that, since the GradientTreeBoost algorithm does not depend on multiple imputation, we can use all attributes for this algorithm.

To test the accuracy of these models, we compared them to each other and to a single Poisson regression tree model (Therneau & Atkinson, 1997). The Poisson regression tree, for which we have used the Poisson option of the `rpart` package, is included to see whether our models fit the data better than a single tree. For the Poisson regression model, we only show results for the complete model, since in general accuracy does not differ much between a complete and a stepwise model.

To determine the in-sample and out-of-sample performance of the three algorithms on the four product catalogs, we divided the data 100 times randomly in a training (90%) and test set (10%), which is a common way to test model performance in boosting literature (see e.g. (Breiman, 1998)). The in-sample performance is averaged over the 100 training sets, while out-of-sample performance is averaged over the test sets. Performance is measured by Deviance

$$Deviance = \frac{1}{I}\sum_{i=1}^{I}\left[y_i \log\left(\frac{y_i}{\hat{y}_i}\right) - (y_i - \hat{y}_i)\right] \tag{4.10}$$

and the mean absolute error (MAE).

As can been seen in Table 4.2, the PoissonTreeBoost algorithm is much better than the other two algorithms in minimizing both the deviance and MAE on the training data. On three data

Table 4.2: Performance of the different models on training and test data

	MP3 Players				Digital Cameras			
	Training		Test		Training		Test	
Method	Deviance	MAE	Deviance	MAE	Deviance	MAE	Deviance	MAE
Poisson regression	39.45	44.39	69.20	77.10	116.55	221.57	390.74	416.95
Poisson regression tree	24.02	42.42	83.47	94.68	249.76	299.97	407.71	390.66
PoissonTreeBoost	2.02	14.64	52.16	78.16	33.06	109.47	302.79	328.43
	Notebooks				Microwave Ovens			
Poisson regression	21.57	41.62	35.26	53.18	9.50	16.26	17.67	21.08
Poisson regression tree	50.42	63.46	60.43	68.21	8.16	14.72	14.96	20.51
PoissonTreeBoost	9.14	26.40	56.13	66.34	1.57	6.32	11.84	18.17

sets this also lead to a lower deviance on the test data, what in two cases is accompanied by a lower MAE than both other algorithms. Only on the MP3 player data, the MAE of the Poisson regression model is slightly lower. The product catalog on which the PoissonTreeBoost algorithm, despite a better in-sample performance, had both a higher out-of-sample deviance and MAE, is the notebooks catalog. In general, the boosting approach fits the data better than the Poisson regression model. However, more evaluation is needed to see whether this also leads to better weights on the data sets.

4.4.3 Determined Weights

Table 4.3 shows the ten attributes getting the highest weights for the MP3 player catalog according to the three methods: stepwise Poisson regression, complete Poisson regression, and PoissonTreeBoost. Although some attributes are considered to be important by all three methods (Brand, Memory Size, and Color are in all three top 10's), there are quite some differences between the weights determined by the different methods. There is high correlation between many attributes in the data and the methods then have a preference for different attributes. For instance, boosting is somewhat biased to categorical attributes having a lot of values, since they have a higher probability to provide a good split in a tree. For the other three product catalogs similar patterns exist. In general, the difference in weights seems to be the largest between PoissonTreeBoost and both Poisson regression models.

4.5 Clickstream Evaluation

Remember that our aim was to find a weighted attribute-based dissimilarity measure which matches perceived (subjective) dissimilarities as closely as possible. The performance evaluation in Section 4.4.2 is no proper evaluation of the degree in which this aim was achieved because it merely considers goodness-of-fit. So, it is clear that we need an evaluation method that takes user perceptions into account. We performed two such evaluations: one using a clickstream analysis

Table 4.3: Weights of the ten most important attributes (out of in total 45 attributes)for the MP3 player catalog using PoissonTreeBoost and a stepwise and complete Poisson regression model

	Stepwise Poisson Regr.		Complete Poisson Regr.		PoissonTreeBoost	
	Attribute	Weight	Attribute	Weight	Attribute	Weight
1.	Brand	.242	Memory Size	.125	Color	.220
2.	Memory Size	.176	Brand	.119	Audio Formats	.186
3.	Audio Formats	.131	Voice Memo	.093	Brand	.090
4.	Battery Type	.106	Height	.090	Interfaces	.086
5.	Width	.090	Depth	.081	Photo Formats	.080
6.	Operating System	.086	Color	.057	Weight	.049
7.	Color	.084	Extendable Memory	.052	Price	.042
8.	Memory Type	.084	Operating System	.049	Memory Size	.035
9.			Width	.048	Height	.030
10.			Battery Type	.045	Memory Type	.026

and another one based on a user experiment. We first describe the clickstream evaluation, while the user evaluation is deferred to Section 4.6.

Both evaluation methods are based on evaluation of top P recommendations of products given that the user is looking at another product (which we will call the reference product) in the product catalog. Such a list could one describe as: "When you are interested in this product, you would maybe also like these products". In the clickstream analysis, we determine good recommendations as products that are relatively often visited together in a session with the reference product. Then, we determine the overall quality of a top P of recommendations as the average of the relative co-occurrence of the products in this top P with the reference product. Finally, we average over all reference products, that are all products in the product catalog, to determine the overall quality of the recommendations. These top P's are determined in this evaluation using the different weighting schemes discussed earlier. However, this evaluation approach can be used to evaluate all kind of dissimilarity measures and other types of recommendations algorithms.

4.5.1 Mean Average Relative Co-Occurrence

In more detail, we looked for sessions in the clickstream log in which two or more products out of one of the four product catalogs were visited. We defined a visit of a product as a visit of the details page of a product. Since people generally look for a relative specific type of product, products that a user visits during a session are probably considered to be similar by the user. Generalizing this, we can assume that products that are visited together in many sessions should be products that are considered to be similar by users in general. Therefore, we counted in the clickstream log for each pair of products in the catalog how often they co-occurred in sessions, which resulted in a product co-occurrence matrix $\mathbf{C}$.

Consider the following hypothetical co-occurrence matrix

$$\mathbf{C} = \begin{pmatrix} 20 & 10 & 2 & 7 \\ 10 & 17 & 3 & 6 \\ 2 & 3 & 6 & 4 \\ 7 & 6 & 4 & 10 \end{pmatrix} , \tag{4.11}$$

where on the diagonal the total occurrence of a product is reported. We normalized this matrix by dividing each row i by the number of times product i was visited in a session (sessions in which only one product was visited were not taken into account), such that position ij in the matrix contains the relative occurrence of product j given that product i was visited in a session. Hence, we create a relative co-occurrence matrix $\mathbf{R}$ having elements $r_{ij} = c_{ij}/c_{ii}$. For the example, this results in the following matrix

$$\mathbf{R} = \begin{pmatrix} 1.00 & 0.50 & 0.10 & 0.35 \\ 0.59 & 1.00 & 0.18 & 0.35 \\ 0.33 & 0.50 & 1.00 & 0.67 \\ 0.70 & 0.60 & 0.40 & 1.00 \end{pmatrix} . \tag{4.12}$$

Since we assume that a high relative co-occurrence of two products implies that these two products are considered to be similar by users, we can use this matrix to evaluate a dissimilarity measure in the following way. First, we use the dissimilarity measure that we want to evaluate to compute a dissimilarity matrix between the products in the catalog and we transform the dissimilarity values into their row-wise ranks. For instance, assume that this yields

$$\mathbf{D} = \begin{pmatrix} 0 & 1 & 3 & 2 \\ 1 & 0 & 2 & 3 \\ 3 & 2 & 0 & 1 \\ 2 & 3 & 1 & 0 \end{pmatrix} . \tag{4.13}$$

In this example, $\mathbf{d}_4$ indicates that for product 4, the most similar product is product 3, followed by product 1 and product 2. Hence, a case-based recommender based on the distance measure used to construct $\mathbf{d}$ would recommend products in this order to a user that bought, or has otherwise shown interest in, product 4.

Supposedly, the recommendation of the top ranked product 3 is good if this product has a high session co-occurrence with product 4, as indicated by the value in matrix $\mathbf{R}$. In general, under the assumption that relative co-occurrence is a good measure for perceived similarity, a high average value of the relative co-occurrences for the top-ranked recommendations for all reference products is desirable. (The average is over all reference products.)

Since this average depends on the ranks in matrix $\mathbf{D}$, and these depend on the dissimilarity measure used, the average value can be used to evaluate dissimilarity measures. We term this measure it MARCO, which is an acronym for Mean Average Relative Co-Occurrence. Instead of taking the average over all top-1-ranked recommendations, one can also take the average over all recommendations with rank P and lower. In this case, the average relative co-occurrence is

computed over all best, 2-nd best, ..., P-th best recommendations for each reference product. We call the corresponding measure $MARCO_P$.

In the example, the value for $MARCO_1$ is

$$MARCO_1 = \frac{.50 + .59 + .67 + .40}{4} = .54 \ . \tag{4.14}$$

and the value for $MARCO_2$ is

$$\begin{aligned} MARCO_2 &= \frac{(.50 + .35) + (.59 + .18) + (.67 + .50) + (.40 + .70)}{8} \\ &= .49 \ . \end{aligned} \tag{4.15}$$

Expressed as an equation, $MARCO_P$ is

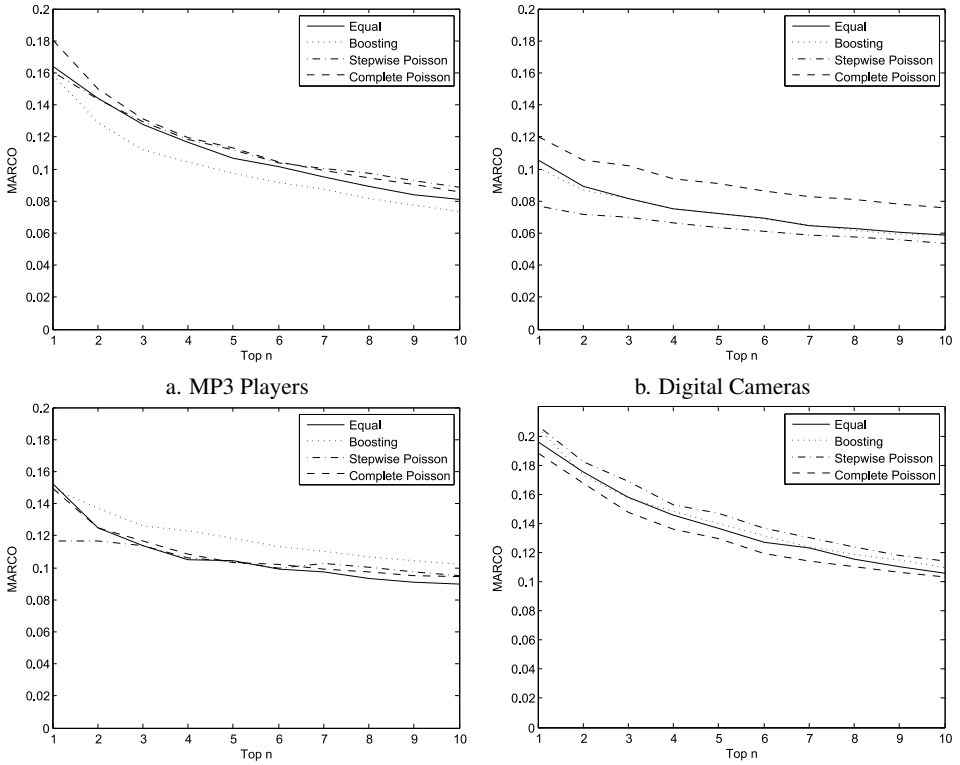
$$\begin{aligned} MARCO_P &= \frac{1}{I \times P} \sum_{i=1}^{I} \sum_{p=2}^{P+1} r_{i,\text{rankindex}(i,p)} \\ &= \frac{1}{I \times P} \sum_{i=1}^{I} \left(c_{ii}^{-1} \sum_{p=2}^{P+1} c_{i,\text{rankindex}(i,p)} \right) \end{aligned} \tag{4.16}$$

where rankindex(i, p) is the index of the p-th most similar product for reference product i. Note that $1 \leq P \leq I - 1$, and in the summation p starts at 2 so that relative co-occurrences of products with themselves are omitted.

Note that the MARCO is always between 0 and 1, where higher values indicate a higher recommendation quality. An optimal MARCO value could be obtained by directly use co-occurrences to provide recommendations. However, co-occurrences cannot be used when recommending new products or recommending products given a search query, which are situations in which (weighted) dissimilarity measures based on product attributes can be used. Additionally, although we use co-occurrences here as measure of evaluation, they may not be present in other systems in which a measure of popularity is present.

4.5.2 Evaluation Results

We applied this approach on the four product catalogs described earlier and tested four dissimilarity weighting methods: equal weights, PoissonTreeBoost, and stepwise and complete Poisson regression. The approaches were applied to the adapted Gower coefficient measure (see Appendix 4.A). The MARCO for different top P's from top 1 up to top 10 for four product catalogs is shown in Figure 4.2. As can be seen in Figure 4.2a, both Poisson model based dissimilarity measures generally perform better than the dissimilarity with equal weights on the MP3 player catalog, while the boosting approach does worse. In fact, the complete Poisson regression model is overall the best model followed by stepwise Poisson regression for this data set. Also for the digital camera catalog of which the MARCO plot is shown in Figure 4.2b, the complete Poisson model based dissimilarity measure does best. However for this data set, there is not much difference between using equal weights or weights determined by boosting and the stepwise Poisson



a. MP3 Players b. Digital Cameras

c. Notebooks d. Microwave Ovens

Figure 4.2: Mean average relative co-occurrence (MARCO) for top 1 until 10 for the different weighting methods.

regression performs worse than all other methods. The MARCO plot for the notebook catalog is shown in Figure 4.2c. For the top 3 and higher all weighting methods perform better than the equal weights approach. On this catalog boosting performs best, followed by both Poisson regression approaches. Equal weights only perform well when recommending a very small number of products. Finally, the plot for the microwave oven catalog is shown in Figure 4.2d. For this catalog the stepwise Poisson regression method is generally the best performing method followed by boosting both performing better than the method using equal weights. The complete Poisson regression model based dissimilarity measure performs worst on these data.

We also tested whether methods performed significantly better than using equal weights over all products in a catalog. To do so, we used a one sided paired t-test on the average relative co-occurrence of top 10's testing whether the model based method performs better than the equal weighted dissimilarity measure. The p-values of these tests are given in Table 4.4. On all product

Table 4.4: p-Values of one-sided paired t-tests testing whether a weighting scheme performs better than using equal weights on a product catalog

Product Catalog	Stepwise Poisson Regr.	Complete Poisson Regr.	PoissonTreeBoost
MP3 Player	**0.000**	**0.002**	1.000
Digital Camera	0.999	**0.000**	0.645
Notebook	0.054	0.079	**0.000**
Microwave Oven	**0.000**	0.854	0.051

catalogs there is at least one method performing significantly better (at a 0.05 significance level) than the equal weighted dissimilarity. However, which method this is, is different among the product catalogs. Hence, although weighting of dissimilarity based on product popularity may improve recommendations which method should be used changes among product catalogs.

4.6 User Experiment

We also evaluated the different weighting methods in an user experiment in the form of an online survey. In this survey, we collected data on which products users found to be relevant recommendations when they were looking at another product. These results could then be used to evaluate the different weighting methods.

4.6.1 Online Survey

People were asked to participate in this experiment via the Vergelijk.nl newsletter. In total, 70 people completed the survey. This experiment consisted of an online survey having three parts. In the first part, we asked participant to do the following. We gave them a randomly selected product, which they should treat as a reference product of which they wanted to see the details. Then, we also provided six other randomly selected products that served as potentially relevant products with respect to the reference product. The respondents were asked to select those products they thought to be a relevant recommendation for the reference product. A screenshot of this task is shown in Figure 4.3. We repeated this procedure twice for each reference product. Every respondent had to perform this task for three reference products. Thus, per reference product, a maximum of twelve relevant products can be chosen by a respondent. Subsequently, we asked the respondents directly how important they judged the different attributes of the products on a five point scale. To avoid any order effect, the order in which the attributes were listed was also randomized per respondent.

For this experiment, we used the microwave oven catalog. However, to make the exercise manageable for the participants, we only selected seven attributes, that were, the seven attributes having the least number of missing values: Brand, Price, Color, Type, Volume, Max. Power, and Model. Only these seven attributes are shown to the participant. We also constrained the product catalog used in the experiment to the 25 most popular microwave ovens that did not have any

Figure 4.3: Screenshot of the online survey.

missing value on one of these seven attributes. This was done, so that we could obtain relatively many observations for all considered products for the first part of the experiment.

The four weighting approaches were applied to only these seven attributes, which also means that the weights were determined using models only having these seven attributes as input. The four weighting schemes we evaluate are: equal weights for the seven attributes, PoissonTree-Boost, a complete Poisson regression model, and one using the average importance stated by the respondents (normalized to have sum of one). The weights of these four methods are shown in Table 4.5. The last column contains average weights specified by the users themselves in the survey.

4.6.2 Mean Average Relative Relevance

To evaluate the different weighting schemes, we used a similar approach as taken in the click-stream analysis in Section 4.5. We again evaluate top P's given a reference product. Only now, we define a good recommendation as a product that is considered to be a relevant recommendation by the respondents given the reference product.

Therefore, we first counted how many times each product j was considered to be relevant given a reference product i and store this in a matrix $\mathbf{A}$. Contrary to the co-occurrence matrix used in the previous section, this matrix is not symmetric. Another difference with the co-

Table 4.5: Relative weights of the different weighting schemes used in the experiment

Attribute	Equal Weights	PoissonTreeBoost	Poisson regression	Average User Importance
Color	0.143	0.265	0.145	0.121
Brand	0.143	0.237	0.128	0.124
Price	0.143	0.191	0.238	0.167
Type	0.143	0.129	0.259	0.128
Volume	0.143	0.102	0.120	0.158
Max. Power	0.143	0.048	0.059	0.164
Model	0.143	0.030	0.051	0.139

occurrence matrix in the previous section is that the number of times a combination of a reference product and a potential relevant product is presented to a respondent is random. Therefore, we also counted how often each product j was shown to a respondent given reference product i and store this in a matrix $\mathbf{B}$. By dividing these two numbers (that is a_{ij}/b_{ij}), we get the relative relevance of products j given reference product i. By doing this for all products, a 25×25 matrix $\mathbf{R}$ is obtained. Then, a 25×25 dissimilarity matrix $\mathbf{\Delta}$ is computed for each of the four weighting schemes using the weights shown in Table 4.5 using the adapted Gower coefficient described in Appendix 4.A. Note that Table 4.5 contains relative weights that sum to one. Similar to as was done in the clickstream evaluation, we use the rankorders in a row i of $\mathbf{\Delta}$ to order the relative relevances in $\mathbf{r}_i$. Based on this ordering, we can compute top the average relative relevance for different top P's using $\mathbf{R}$ and average this over all 25 products in a similar way as in (4.16). To acknowledge the difference in constructing $\mathbf{R}$, we will refer to this measure as the mean average relative relevance (MARR), defined as

$$\begin{aligned} MARR_P &= \frac{1}{I \times P} \sum_{i=1}^{I} \sum_{p=2}^{P+1} r_{i,\text{rankindex}(i,p)} \\ &= \frac{1}{I \times P} \sum_{i=1}^{I} \sum_{p=2}^{P+1} \frac{a_{i,\text{rankindex}(i,p)}}{b_{i,\text{rankindex}(i,p)}} \end{aligned} \tag{4.17}$$

where rankindex(i, p) is the index of the p-th most similar product for reference product i.

4.6.3 Evaluation Results

Figure 4.4 shows the MARR for the four weighting schemes on the user experiment data. As can be seen in this plot, both model-based weighting schemes did not perform well in this experiment, since they did worse than the equal weighted dissimilarity measure. The dissimilarity measure using the weights based on the average importance according to users only performs slightly better than the equal weights.

It seems that respondents acted differently than in the clickstream evaluation. The limited number of attributes used in the experiment may have been a factor why the performance of the

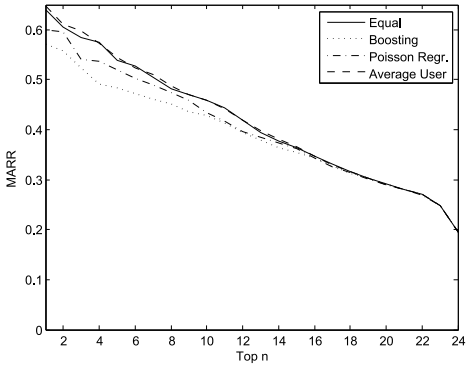


Figure 4.4: Mean average relative relevance (MARR) of four weighting schemes.

model based methods is quite poor. One thing the weighting methods can do quite well is giving very unimportant attributes very low or zero weights. However, these seven attributes were all considered relatively important attributes by the users, since they were have a mean value higher than 3 on a 5 point scale.

4.7 Summary, Conclusions & Discussion

Finding accurate attribute weights for case-based recommender systems is relevant because it may help to increase both accuracy and transparency of recommendations. In this chapter, we have attempted to find methods for both identifying attribute weights, as well as for evaluating a set of attribute weights.

We have presented two methods to determine attribute weights, both methods modeling product popularity as a function of attribute values. The first method was based on a Poisson regression model, while the second method was based on a new boosting algorithm minimizing Poisson deviance. The latter model has the advantage that it has built-in methods to handle categorical attributes and missing values, which are common in e-commerce data. Also, it fits the data better, since it is more flexible and also able to handle interaction effects.

The methods and their resulting attribute weights were evaluated in a number of ways. The first evaluation used was aimed at model performance in terms of goodness-of-fit. In this evaluation, we used data from four real life commercial product catalogs with a corresponding clickstream log, and we evaluated how well the Poisson regression and boosting models could predict the number of 'outclicks' on a product. The latter model performed better in general, both in- and out-of-sample.

The second and third evaluations were aimed at determining how well dissimilarity measures using the resulting attribute weights matched user-perceived product dissimilarities. To this end, we extracted attribute weights from the the Poisson and Boosting models. We used those weights

in the adapted Gower coefficient, and we developed two methods for determining how well the weighted Gower coefficient matches the user perceived similarities.

The first of these analyses was based on clickstream data. We developed a new measure, called MARCO, for assessing the quality of the weights in the Gower coefficient. Surprisingly, the only catalog on which the boosting based weights significantly outperformed the other models was the notebook catalog, which was also the only catalog on which the out-of-sample performance of boosting was *worse* than of Poisson regression. Hence, prediction performance of a method seems not to be a good predictor of whether a method provides good weights. Reasons for this might be the correlation between attributes and the bias of models for certain types of attributes.

Furthermore, in the evaluation on clickstream data, there was at least one model-based weighting scheme for each product catalog that was significantly better than using equal weights. This implies that attribute weighting based on product popularity improves recommendations. However, at this stage, it is not clear which model should be used in which situation and this is a topic for future research. Also, it may be the case that there exists another model based on product popularity that could improve recommendations on all product catalogs.

The second evaluation approach we used was a user experiment in the form of an online survey. Surprisingly, in this evaluation, the model-based weighting approaches performed worse than using equal weights, implying that users choose products differently in an experimental setup than on a real website. On one hand this result can be driven by the fact that the website setup leads users to certain products. On the other hand people may compare products in a different way during an experiment than they would do in real life. Another reason why the weighting models did not perform well in the user experiment, is that the number of attributes used was limited to only seven relative important attributes. The true contribution of the weighting methods may be the mere selection of these important attributes and not so much the difference in weights between these important attributes.

We see several directions in which this research can be extended or used. First of all, our results are inconclusive with respect to the single best method for determining attribute weights. Improving upon the proposed evaluation method, e.g. by deriving new features from the clickstream data, may alleviate this problem.

Both the clickstream analysis and the user experiment framework are set up in such a way that they can be used for evaluation of various types of (dis)similarity measures and weighting schemes. Such evaluations may potentially lead to a combination of a dissimilarity measure and weighting scheme that outperforms other approaches in both evaluations.

Also, we would like to study the use of the attribute weights in our previous work on map based recommender systems (Kagie et al., 2007, 2008b). We hope that the usability of these systems will benefit from attribute weights. Finally, we think that the PoissonTreeBoost model can be a competitive alternative for Poisson regression models in different application fields. We believe that the weighting approaches combined with the dissimilarity measure can be a powerful addition for recommender systems.

4.A Adapted Gower Coefficient

In the adapted Gower coefficient framework (Kagie et al., 2007, 2008a, 2008b), the dissimilarity δ_{ij} between products i and j is defined as the square root of the weighted average of nonmissing dissimilarity scores δ_{ijk} on the K attributes

$$\delta_{ij} = \sqrt{\frac{\sum_{k=1}^{K} w_k m_{ik} m_{jk} \delta_{ijk}}{\sum_{k=1}^{K} w_k m_{ik} m_{jk}}} \ , \tag{4.18}$$

in which the w_k's are the weights for the different dissimilarity scores and, hence, for the different attributes. The binary indicator m_{ik} has a value of 1 when attribute k is nonmissing for product i. The weights w_k specify how important the different attributes are in the computation of the dissimilarity measure and, hence, in the application. In Section 4.3, we discuss two methods to determine these weights.

The computation of the dissimilarity scores δ_{ijk} in (4.18) is dependent on the type of the attribute. For numerical attributes, the dissimilarity score δ_{ijk} is the normalized absolute distance

$$\delta_{ijk}^{N} = \frac{|x_{ik} - x_{jk}|}{\left(\sum_{i<j} m_{ik} m_{jk}\right)^{-1} \sum_{i<j} m_{ik} m_{jk} |x_{ik} - x_{jk}|} \ , \tag{4.19}$$

where $x_i k$ is the attribute value of product i for attribute k. For categorical attributes, the dissimilarity score δ_{ijk} is defined as

$$\delta_{ijk}^{C} = \frac{1(x_{ik} \neq x_{jk})}{\left(\sum_{i<j} m_{ik} m_{jk}\right)^{-1} \sum_{i<j} m_{ik} m_{jk} 1(x_{ik} \neq x_{jk})} \ , \tag{4.20}$$

where $1()$ is the indicator function returning a value of 1 when the condition is true and 0 otherwise.

However, in many product catalogs, as will also be the case in the catalogs used in this chapter, a third type of attributes exists, which we call multi-valued categorical attributes. Where a product can have only one value for a categorical attribute such as, for example, its brand, it can have multiple values for a multi-valued categorical attribute. For instance, an MP3 player can have an attribute called 'supported audio formats', which can contain the values MP3 and WMA at the same time.

In recommender systems, the products are often compared to a query and not to each other. When a user, for example, queries for an MP3 player that supports the audio formats MP3 and WMA, it does not matter for this user that a product also supports other audio formats, but when MP3 or WMA is not supported it does matter. Therefore, we use the following dissimilarity score for multi-valued categorical attributes in our framework

$$\delta_{ijk}^{M} = \frac{\sum_{s=1}^{S} 1(x_{iks} \notin x_{jk})}{\left(\sum_{i \neq j} m_{ik} m_{jk}\right)^{-1} \sum_{i \neq j} m_{ik} m_{jk} (\sum_{s=1}^{S} 1(x_{iks} \notin x_{jk}))} \ , \tag{4.21}$$

where x_{iks} denotes one of the in total S values product x_i has for attribute x_{ik}. Note that we normalize over all dissimilarities where $i \neq j$, since the dissimilarity score is not symmetric anymore.

4.B Derivation of PoissonTreeBoost

The PoissonTreeBoost algorithm can be derived by applying the GradientTreeBoost (Friedman, 2001) framework to optimize the Poisson deviance loss function in (4.3) . Boosting estimates target values as a sum of the predictions of the base learners

$$F_N(\mathbf{x}) = F_0 + \sum_{n=1}^{N} B_n(\mathbf{x}) \ . \tag{4.22}$$

The initial estimate F_0 can be determined by

$$\begin{aligned} F_0(\mathbf{x}) &= \arg\min_{\rho} \sum_{i=1}^{I} L(y_i, \rho) \\ &= \arg\min_{\rho} \left(\sum_{i=1}^{I} \left[y_i \log\left(\frac{y_i}{\rho}\right) - (y_i - \rho) \right] \right) \\ &= \frac{1}{I} \sum_{i=1}^{I} y_i \ , \end{aligned} \tag{4.23}$$

which implies that F_0 should equal the mean values of y. Then, we have to determine pseudo targets $\tilde{y}_i$ on which a base learner, that is, a regression tree, should be fitted by determining the negative gradient direction

$$\begin{aligned} \tilde{y}_i &= -\frac{\partial L(y_i, F_{n-1}(\mathbf{x}_i))}{\partial F_{n-1}(\mathbf{x}_i)} \\ &= -\frac{\partial \left[y_i \log\left(\frac{y_i}{F_{n-1}(\mathbf{x}_i)}\right) - (y_i - F_{n-1}(\mathbf{x}_i)) \right]}{\partial F_{n-1}(\mathbf{x}_i)} \\ &= \frac{y_i}{F_{n-1}(\mathbf{x}_i)} - 1 \ . \end{aligned} \tag{4.24}$$

On these pseudo targets we train an ordinary regression tree using a squared loss function. Finally, we have to replace the values of the terminal nodes by

$$\begin{aligned} \gamma_{pn} &= \arg\min_{\gamma} \sum_{\mathbf{x}_i \in R_{pn}} L(y_i, F_{n-1}(\mathbf{x}_i) + \gamma) \\ &= \arg\min_{\gamma} \sum_{\mathbf{x}_i \in R_{pn}} \left(y_i \log\left(\frac{y_i}{F_{n-1}(\mathbf{x}_i) + \gamma}\right) - (y_i - (F_{n-1}(\mathbf{x}_i) + \gamma)) \right) \ . \end{aligned} \tag{4.25}$$

procedure POISSONTREEBOOST($\{\mathbf{x}_i, y_i\}_1^I, N, \nu$)
$F_0(\mathbf{x}) = \frac{1}{I}\sum_{i=1}^{I} y_i$
for $n = 1$ to N **do**
$\left\{\tilde{y}_i = \frac{y_i}{F_{n-1}(\mathbf{x}_i)} - 1\right\}_1^I$
Train tree B_n having terminal nodes $\{R_{pn}\}$ using $\{\mathbf{x}_i; \tilde{y}_i\}_1^I$
$\gamma_{pn} = \arg\min_\gamma \sum_{\mathbf{x}_i \in R_{pn}} \left(y_i \log\left(\frac{y_i}{F_{n-1}(\mathbf{x}_i)+\gamma}\right) - (y_i - (F_{n-1}(\mathbf{x}_i) + \gamma))\right)$
$F_n(\mathbf{x}) = F_{n-1}(\mathbf{x}) + \nu \sum_{p=1}^{P} \gamma_{pn} 1(\mathbf{x} \in R_{pn})$
end for
end procedure

Figure 4.5: PoissonTreeBoost algorithm.

Unfortunately, this minimum can often not be found analytically and therefore a line search in each node needs to be performed. Finally, we add our new base learner, a regression tree with terminal nodes R_{pn}, which have values γ_{pn}, to the boosting model

$$F_n(\mathbf{x}) = F_{n-1}(\mathbf{x}) + \nu \sum_{p=1}^{P} \gamma_{pn} 1(\mathbf{x} \in R_{pn}) \ . \tag{4.26}$$

Note that there is also a shrinkage parameter ν incorporated in the GradientTreeBoost framework which is known as the learning rate. Setting the learning rate smaller than 1 implies that the results of a base learner are only partially incorporated in the model. Setting the learning rate ν very small, that is $\nu \leq 0.1$ leads to the best results (Friedman, 2001; Hastie et al., 2001). The algorithm is summarized in Figure 4.5.

Chapter 5

Including Item Characteristics in the Probabilistic Latent Semantic Analysis Model for Collaborative Filtering*

5.1 Introduction

Nowadays, a lot of electronic commerce companies assist their customers online in buying their product using recommender systems (Resnick & Varian, 1997). In general, two different types of these systems can be identified.

A large group of recommender systems, so-called collaborative filtering systems (D. Goldberg, Nichols, Oki, & Terry, 1992), recommend items based on similar preferences between users. When customers (henceforth called users) A and B liked the same products (henceforth called items) in the past, collaborative systems assume that a good item to recommend to A would be an item that user B appreciated, but which A is unfamiliar with. Often, this idea is incorporated in the collaborative filtering method by modelling communities of users having similar taste. A popular collaborative filtering algorithm with a strong probabilistic foundation that uses this idea is the probabilistic latent semantic analysis model for collaborative filtering (pLSA-CF, see Hofmann, 2004). A disadvantage of these collaborative type recommender systems is that they neglect item characteristics, such as genre and keywords for a movie, when making recommendations, whereas these may carry useful information. Another disadvantage is that handling new items is difficult, since new items have not been appreciated by anyone yet. This is called the 'cold start item problem'.

In contrast, content-based recommender systems (Pazzani & Billsus, 2007), use only characteristics of the items to provide a recommendation, and recommend items that are similar, based on these characteristics, to items that the user liked in the past. The advantage that this approach has over collaborative methods, is that it is also able to recommend items that are not yet rated by any user, since item characteristics are known upfront. A disadvantage is that such systems

*This chapter is based on Kagie, Van der Loos, and Van Wezel (2009)

only use earlier ratings from the active user when providing a recommendation, whereas ratings by other users are discarded. (The active user is the user receiving the recommendations.)

To combine some of the strengths of both approaches, hybrid recommender systems have emerged (Burke, 2002), which integrate collaborative and content-based techniques. Section 5.2 discusses these systems in more detail. In this chapter, we introduce a new hybrid recommendation approach which exploits the idea that communities of users exist which find the same item characteristics important to like or dislike a product.

The basis of our method is pLSA-CF (Hofmann, 2004) mentioned above. pLSA-CF attempts to find 'hidden' user communities that rate items similarly. For each community, pLSA-CF estimates expectations of the ratings for all items. For a specific user, the expected rating for an item is computed as a weighted average over the community ratings for that item. The weighting coefficients, which are also estimated, express to what degree a user belongs to each community. We will briefly describe the pLSA-CF model in Section 5.3.1 below.

We extend pLSA-CF by conditioning the expected ratings on item characteristics. This is achieved using a separate linear regression function that models the relationship between item characteristics and ratings for each user community. For each user, an individual regression function can then be constructed by taking a weighted average over the the community regression functions. This idea borrows from, but is not identical to so-called clusterwise linear regression (CLR, see DeSarbo & Cron, 1988). We discuss CLR and our extension to pLSA-CF in Sections 5.3.2 and 5.3.3 below.

Our model, which we call the latent class regression recommender system (LCR-RS), is evaluated on a movie recommendation data set. This data set was constructed by combining two well-known databases: The Netflix database[1], which contains user ratings of movies, and the IMDb database[2], which contains movie characteristics. We compare prediction performance of our model with a set of standard models, pLSA-CF, and another hybrid recommender system, namely Content-Boosted Collaborative Filtering (CB-CF, see Melville, Mooney, & Nagarajan, 2002). Although our method does not outperform collaborative filtering methods on these data (but it does outperform the content-based methods and CB-CF), it is competitive to these methods. However, our method has other advantages over those methods.

There are a number of advantages that our method has over pLSA-CF: (1) Since the regression coefficients are shared among items, it contains far fewer parameters, making it less susceptible to overfitting; (2) The regression coefficients add to the interpretability since they can help in explaining *why* the system thinks that items are highly/poorly rated; (3) Since items are recommended based on characteristics, our method does not suffer from the cold-start item problem, which we show in an experiment. We think that these advantages may compensate the slightly worse performance of our model.

The remainder of this chapter is organized as follows. In the next section, we discuss recommender systems with an emphasis on hybrid recommendation approaches. Section 5.3 describes pLSA-CF, CLR, and LCR-RS. In Section 5.4, we describe the movie data that is used in the

[1] `http://www.netflix.com`

[2] `http://www.imdb.com`

experiments that are shown in Section 5.5. Finally, we draw conclusions and give directions for future research in Section 5.6.

5.2 Recommender Systems

As mentioned in the introduction, there exist three types of recommender systems (Adomavicius & Tuzhilin, 2005; Prasad, 2003): Collaborative filtering (CF, see D. Goldberg et al., 1992), content-based (Pazzani & Billsus, 2007), and hybrid systems (Burke, 2002), combining these latter two approaches.

Adomavicius and Tuzhilin (2005) and Breese, Heckerman, and Kadie (1998) make a distinction between two types of collaborative filtering systems. The first type consists of the memory-based recommenders, which use the entire user-item ratings matrix each time recommendations are computed. Often, these methods compute similarities between users based on their rating behavior and then apply some nearest neighbor approach to provide recommendations, an idea first applied in the GroupLens system (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994). Similarity between items based on the ratings they received can also be used. This technique is called item-based CF (Sarwar, Karypis, Konstan, & Riedl, 2001) and is applied at e.g., Amazon.com (Linden, Smith, & York, 2003). Model-based systems on the other hand, fit a model which provides the recommendations. The ratings matrix is only used during model fitting, not during deployment. Popular model-based collaborative recommenders include EigenTaste (K. Goldberg, Roeder, Gupta, & Perkins, 2001), latent Dirichlet allocation (Blei, Ng, & Jordan, 2003), and probabilistic latent semantic analysis for collaborative filtering (Hofmann, 2004), which is the basis of our algorithm.

Recently, the development of collaborative filtering algorithms has been boosted by the Netflix Prize[3]. Netflix is an online company that provides a DVD movie rental service. To provide their customers with personalized DVD recommendations, their web site makes use of a recommender system called Cinematch. Netflix has organized the Netflix Prize, to challenge the public to improve the prediction accuracy of Cinematch with ten percent. For this contest a data set was made available containing over 100 million movie ratings on a 5 point scale, of which we will use a subset.[4] In the contest, algorithms based on matrix factorization (Koren, Bell, & Volinsky, 2009; Takács, Pilászy, Németh, & Tikk, 2008; Zhou, Wilkinson, Schreiber, & Pan, 2008) are performing best.

Content-based recommenders make recommendations based on the characteristics of items that the user has liked in the past. Also in this kind of systems, which originate from information retrieval, nearest neighbor methods have been very popular. When the items at hand are text documents, the term frequency - inverse document frequency is often used to construct the characteristics vector of the items (Adomavicius & Tuzhilin, 2005). Besides the nearest neighbor

[3] `http://www.netflixprize.com`

[4] Since the Netflix data does not contain item attributes, but merely ratings, we combine the Netflix dataset with another dataset, containing movie attributes. See Section 5.4.

method, (Pazzani & Billsus, 1997) show that also other classifiers as naive Bayes, decision trees, and neural networks can be used as content-based recommender systems.

Hybrid recommender systems combine content-based and collaborative filtering methods. (Adomavicius & Tuzhilin, 2005) made a distinction between four types of hybrid combinations of content-based and collaborative filtering systems.

The most basic hybrid recommender systems combine two separate recommender systems, a content-based and a collaborative filtering system. The final recommendation is then based on a linear combination (Claypool et al., 1999) or a voting scheme (Pazzani, 1999). Also, a quality metric can be used to select the best prediction of the systems as is done by, for instance, (Billsus & Pazzani, 2000) and (Tran & Cohen, 2000).

Another type of hybrid recommenders includes content-based characteristics to collaborative filtering methods. A way in which this can be done, is, for instance, by augmenting the vector of user ratings with additional ratings, that are computed using a content-based method as is done by Melville et al. (2002). Similar approaches were taken by Balabanović and Shoham (1997), Good et al. (1999), and Pazzani (1999). Soboroff and Nicholas (1999) took the opposite approach by adding collaborative characteristics, using latent semantic indexing (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990), to a content-based recommender system.

Finally, there is a group of hybrid recommender systems in which content-based and collaborative filtering are combined into one model. This has be done, for example, using rule-based classifiers (Basu, Hirsh, & Cohen, 1998) or Bayesian mixed-effect regression models (Ansari, Essegaier, & Kohli, 2000; Condliff, Lewis, Madigan, & Posse, 1999). Popescul, Ungar, Pennock, and Lawrence (2001) and Schein, Popescul, Ungar, and Pennock (2002) integrated item characteristics (words contained in a document) in the original probabilistic latent semantic analysis model for binary targets (observed – not observed) (Hofmann, 1999, 2001).

In our method, which we discuss in detail in Section 5.3.3, we integrate both methods in a different way. Where in collaborative filtering systems similar users are determined by similar rating patterns, we define similar users as users who find the same characteristics of an item important and, thus, incorporate content-based information in the model. Since our approach is an extension of the probabilistic latent semantic analysis model for collaborative filtering (pLSA-CF, see Hofmann, 2004), we first discuss this model in Section 5.3.1.

5.2.1 Explanations in Recommender Systems

Although providing good recommendations to users is essential for a recommender system, users are more likely to accept the recommendations of the system, when it gives a good explanation why it gives certain recommendations (Herlocker, Konstan, & Riedl, 2000). In an overview of explanation strategies in recommender systems, Tintarev and Masthoff (2007) distinguish seven aims for explanations in recommender systems. In our approach, we will focus on the aim of transparency, that is, explaining why a certain recommendation is given by the system, an approach which is, for example, taken by McSherry (2005). In a small user study, Sinha and Swearingen (2002) showed that users liked transparent recommender systems over nontranspar-

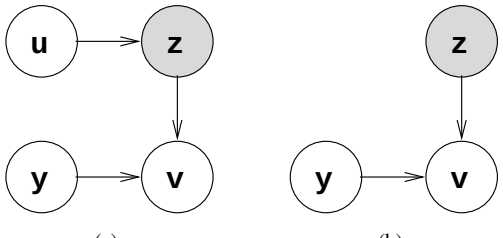


Figure 5.1: Graphical model representations of 5.1(a): pLSA-CF model and 5.1(b): Gaussian mixture model.

ent ones. In Section 5.3.3, we show how our system can be used to provide transparency on how it provides its recommendations.

5.3 Models

This section describes the latent class regression recommender system (LCR-RS). Before describing LCR-RS itself, we first describe the models it is built on: the pLSA-CF model for collaborative filtering and the CLR model for clusterwise regression.

5.3.1 Probabilistic Latent Semantic Analysis for Collaborative Filtering

pLSA-CF, introduced in Hofmann (2004), models unobserved (latent) classes of users which give similar ratings to items. To discuss this model more formally we introduce some notation. Consider a set of items $\mathcal{Y} = \{y_1, \ldots, y_n\}$, a set of users $\mathcal{U} = \{u_1, \ldots, u_m\}$, and a set of numerical ratings $\mathcal{V}$. A typical collaborative data set contains N entries $\langle u, y, v \rangle$, that is, ratings v of a certain item y by a certain user u. However, this set of entries is not complete, since by far not all items are rated by each user.

pLSA-CF assumes that there is also a set $\mathcal{Z} = \{z_1, \ldots, z_K\}$ of latent classes of users with similar preferences. This dependency structure is depicted in Figure 5.1. Intuitively, these classes can be interpreted as communities of users with similar preferences. (An example could be the 'Trekkies', interested in science fiction.) Since these classes are unobserved we do not know to which class a user belongs. However, we do know that the fact that a user belongs to a certain class influences his ratings. Although class memberships are unobserved, we can still estimate class membership probabilities using the EM algorithm (Dempster, Laird, & Rubin, 1977), as we will see shortly.

First, consider the joint probability density function, including the hidden variable z

$$P(u, v, y, z) = P(u)P(y)P(z|u)P(v|z, y) \tag{5.1}$$

and the marginal density, obtained by summing over all possible values of z

$$P(u, v, y) = P(u)P(y) \sum_{k=1}^{K} P(z_k|u)P(v|z_k, y) \ . \tag{5.2}$$

$P(u)$ and $P(y)$ are constants, depending on the number of votes by from user u and for item y. They will be discarded henceforth. $P(z|u)$ denotes the class membership probabilities for user u. For each k, $P(z_k|u)$ represents the probability that user u will behave according to class z_k when making a vote. $P(z_k|u)$ is just a scalar and $\sum_{k=1}^{K} P(z_k|u) = 1$. In the pLSA-CF model, $P(v|z_k, y)$ is taken to be Gaussian

$$P(v|z_k, y) = P(v|\mu_{k,y}, \sigma_{k,y}) = \mathcal{N}(v|\mu_{k,y}, \sigma_{k,y}) = \frac{1}{\sqrt{2\pi}\sigma_{k,y}} \exp\left[-\frac{(v - \mu_{k,y})^2}{2\sigma_{k,y}^2}\right] \ . \tag{5.3}$$

This Gaussian distribution, with a class/item specific mean and standard-deviation, specifies the distribution of votes given that a user votes for item y and behaves according to class z_k. In total there are $K \times (m + 2n)$ parameters in the model. These parameters are collectively denoted θ.

The joint density thus becomes

$$P(u, v, y, z|\theta) \propto \prod_{k=1}^{K} P(z_k|u)^{I(z=z_k)} \mathcal{N}(v|\mu_{k,y}, \sigma_{k,y})^{I(z=z_k)} \ , \tag{5.4}$$

where $I()$ denotes the indicator function returning 1 if its argument is true, 0 otherwise. The marginal density (for the observed data) is

$$P(u, v, y|\theta) \propto \sum_{k=1}^{K} P(z_k|u)\mathcal{N}(v|\mu_{k,y}, \sigma_{k,y}) \ . \tag{5.5}$$

Although this seems similar to a Gaussian mixture model, there is an important difference: The mixture proportions are user-dependent. A GMM lacks this dependency, as can be seen from Figure 5.1.

Based on the observed data, the log-likelihood is

$$L(\theta) = \sum_{\langle u,v,y \rangle} \log \left\{ \sum_{k=1}^{K} P(z_k|u)\mathcal{N}(v|\mu_{k,y}, \sigma_{k,y}) \right\} \ , \tag{5.6}$$

where we omitted the constant representing $P(u)P(y)$ since it is irrelevant to optimization. This function should be optimized with respect to the parameters θ. Unfortunately, this is difficult because of the summation inside the logarithm.

Consider instead the complete data log-likelihood (again omitting the constant)

$$L^c(\theta) = \sum_{\langle u,v,y,z \rangle} \sum_{k=1}^{K} I(z = z_k) \left\{ \log P(z_k|u) + \log \mathcal{N}(v|\mu_{k,y}, \sigma_{k,y}) \right\} \ . \tag{5.7}$$

This function is fairly easy to optimize, because the logarithm appears inside the summation. Unfortunately, the z are unobserved, so direct optimization of this likelihood is impossible since the values of the indicator function are unknown. We can, however, compute the expected value of the indicator function for each k, given certain values for the parameters θ and observation $\langle u, v, y\rangle$. This would then amount to the posterior probability that $\langle u, v, y\rangle$ was generated by class k given the parameters θ. This is done in the expectation step of the EM-algorithm, where we estimate $E[I(z = z_k)|u, y, v, \theta] = P(z_k|u, y, v, \theta)$ (Hofmann, 2004)

$$P(z_k|u, y, v, \theta) = \frac{P(z_k|u)P(v|\mu_{k,y}, \sigma_{k,y})}{\sum_{k'=1}^{K} P(z_{k'}|u)P(v|\mu_{k',y}, \sigma_{k',y})} \quad . \tag{5.8}$$

These expectations can now be plugged into (5.7) giving the expected likelihood

$$E_z[L^c(\theta)] = \sum_{\langle u,v,y\rangle} \sum_{k=1}^{K} P(z_k|u, y, v, \theta) \left\{\log P(z_k|u) + \log \mathcal{N}(v|\mu_{k,y}, \sigma_{k,y})\right\} \quad . \tag{5.9}$$

This function is then minimized w.r.t. θ in the minimization step of the EM algorithm. This leads to the following update equations for the user-specific mixing weights $P(z_k|u)$ (Hofmann, 2004)

$$P(z_k|u)^{\text{new}} = \frac{\sum_{\langle u',y,v\rangle:u'=u} P(z_k|u', y, v, \theta^{\text{old}})}{|\{\langle u', y, v\rangle : u' = u\}|} \tag{5.10}$$

and for $\mu_{k,y}$ and $\sigma_{k,y}$

$$\mu_{k,y}^{\text{new}} = \frac{\sum_{\langle u,y',v\rangle:y'=y} vP(z_k|u, y', v, \theta^{\text{old}})}{\sum_{\langle u,y',v\rangle:y'=y} P(z|u, y', v, \theta^{\text{old}})} \tag{5.11}$$

$$\sigma_{k,y}^{2,\text{new}} = \frac{\sum_{\langle u,y',v\rangle:y'=y} (v - \mu_{k,y'}^{\text{new}})^2 P(z_k|u, y', v, \theta^{\text{old}})}{\sum_{\langle u,y',v\rangle:y'=y} P(z_k|u, y', v, \theta^{\text{old}})} \quad . \tag{5.12}$$

These updated estimates for $P(z_k|u)$, $\mu_{k,y}$, and $\sigma_{k,y}$ are then plugged into (5.8) and the process continues until convergence, i.e., until the expected log-likelihood (5.9) does not increase anymore. It can be shown see e.g., Bishop (2006) that this end-point is also a local optimum of the observed data likelihood (5.6).

It is worth mentioning that (5.10) does not depend on the normality assumption (5.3), but rather, it holds whenever the joint distribution is decomposable according to Figure 5.1(a). See Hofmann (2004) for details.

Notice that (5.10) cannot be applied when a user has not cast any votes yet. This can easily be overcome by e.g., adding a term a/K to the numerator and adding a term a to the denominator, for a suitable integer a. This corresponds to assigning equal prior class probabilities to users. The larger a, the slower the actual class probabilities for a user will be learned from ratings data. Alternatively, the constants can be constructed such that prior class membership probabilities for new users match average observed class membership probabilities probabilities over the users with sufficient votes. This would solve the cold start user problem.

procedure pLSA_CF(entries $\langle u, y, v \rangle$,K)

▷ Initialization

for all u **do**
Generate random $P(z_k|u)$ such that $\sum_k P(z_k|u) = 1$.
end for
for all y, z_k **do**
Generate random $\mu_{k,y}$.
Generate random $\sigma_{k,y}$.
end for
Set $err = \infty$.

▷ EM-loop

repeat
$prevErr = err$.
for all $\langle u, y, v \rangle, z_k$ **do**
Compute $P(z_k|u, y, v)$ using (5.8).
end for
for all u, z_k **do**
Compute $P(z_k|u)$ using (5.10).
end for
for all y, z_k **do**
Compute μ_{y,z_k} using (5.11).
Compute σ_{y,z_k} using (5.12).
end for
Compute $err = -E[L^c(\theta)]$ using (5.9).
until $prevErr - err < \epsilon$
end procedure

Figure 5.2: pLSA-CF algorithm.

After the model has been trained, we can simply predict a new rating as

$$v_{u,y} = \sum_{k=1}^{K} P(z_k|u)\mu_{k,y} \quad . \tag{5.13}$$

A drawback of the pLSA-CF model is that, like all CF methods, it is unable to predict ratings for items not rated by any user in the system. Therefore, we extend the pLSA-CF model using item characteristics to solve this cold-start item problem. The pLSA-CF model is summarized in Figure 5.2.

5.3.2 Clusterwise Linear Regression

Often, we can represent a certain item by a vector $\mathbf{x}_y$ containing a set of characteristics (such as keywords and genres in the case of movies). A simple way to build a content-based recommender system, is to estimate a linear regression model. When we specify $\mathbf{v}_u$ as the set of ratings of user u and matrix $\mathbf{X}_u$ as the matrix containing the characteristic vectors of the items rated by user u,

we can estimate user specific regression weights by

$$\mathbf{b}_u = (\mathbf{X}_u'\mathbf{X}_u)^{-1}\mathbf{X}_u'\mathbf{v} \ . \tag{5.14}$$

A rating for an item unrated by user u then can simply be predicted by

$$E[v|u, y] = \mathbf{x}_y'\mathbf{b}_u \ . \tag{5.15}$$

Using this simple content-based method we could easily predict ratings for items that have not been rated yet, since we only need the characteristics of the new item. Moreover, it would also be possible to explain why certain recommendations are given using the regression weights. Clearly, these two properties make the regression approach worth considering.

Constructing user-specific regressions may be problematic since we need at least a substantial number of ratings from a user if we want to estimate this user's regression coefficients. As an alternative, one could 'pool' all ratings generated by all user for all items, and estimate a 'population-regression' with this pooled data set. Such a single model is likely to perform poorly, since people have varying preferences.

Hence, an intermediate solution would be not to specify a regression for each user separately or for the total population at once, but to create regression models for different groups of users having similar preferences. We will discuss such a method below in Section 5.3.3. First we discuss method called clusterwise linear regression (CLR) proposed by DeSarbo and Cron (1988). This method, although not directly suitable for the recommendation problem, forms the basis for the method in Section 5.3.3.

CLR and its extensions are very popular in various areas of marketing, such as conjoint analysis (DeSarbo, Wedel, Vriens, & Ramaswamy, 1992; Vriens, Wedel, & Wilms, 1996) and estimating marketing mix elasticities (Ramaswamy, DeSarbo, Reibstein, & Robinson, 1993). Especially the connection between conjoint analysis and item recommendation is very clear. In conjoint analysis subjects rate potential products which are described by a limited set of product characteristics. Then a model is estimated that explains which product characteristics are important to make a product popular. When a model based on CLR is used, one can differentiate between market segments. Besides in marketing, models based on CLR were also successfully applied in other fields of business research as accounting (Larcker & Richardson, 2004), finance (Pennings & Garcia, 2004), and strategic management (DeSarbo, Di Benedetto, Song, & Sinha, 2005). But also in other research disciplines as psychology (Kriegler & Zimprich, 2006) and climate research (Camargo, Robertson, Gaffney, Smyth, & Ghil, 2007), CLR has been used.

In the original CLR the dependent variable (in our case the rating variable v) is modeled as a mix of K conditional univariate densities with cluster-specific regression coefficients and standard deviations

$$P(v|y) = \sum_{k=1}^{K} \lambda_k \mathcal{N}(v|\mathbf{x}_y'\mathbf{b}_k, \sigma_k) \ , \tag{5.16}$$

where the mixture proportions are constrained to be positive $\forall k : 0 \leq \lambda_k$ and sum up to unity ($\sum_k \lambda_k = 1$). Moreover, the cluster variances are constrained to be larger than zero, $\forall k : \sigma_k^2 > 0$, since otherwise the likelihood function is unbounded and consistent estimates are not possible (DeSarbo & Cron, 1988).

CLR is optimized using the EM-algorithm (Dempster et al., 1977). In the E-step we estimate the posterior probabilities for each vote v that this vote belongs to class z_k

$$P(z_k|v,y,\theta) = \frac{\lambda_k \mathcal{N}(v|\mathbf{x}_y'\mathbf{b}_k,\sigma_k)}{\sum_{k'=1}^{K}\lambda_{k'}\mathcal{N}(v|\mathbf{x}_y'\mathbf{b}_{k'},\sigma_{k'})} \ , \tag{5.17}$$

where y denotes the item that vote v concerned. Next, in the M-step, we compute the cluster-weights λ_k

$$\lambda_k = \frac{\sum_{\langle y,v\rangle} P(z_k|y,v,\theta)}{N} \ . \tag{5.18}$$

Here, the summation is over all item/vote pairs in the dataset. In the rest of the M-step, we estimate the cluster-specific regression weights $\mathbf{b}_k$ and variances σ_k^2 using weighted least squares (WLS). Following Ramaswamy et al. (1993), the estimate for $\mathbf{b}_k$ is given by

$$\begin{aligned} \mathbf{b}_k^{\text{new}} &= (\mathbf{X}'\mathbf{P}_k\mathbf{X})^{-1}\mathbf{X}'\mathbf{P}_k\mathbf{v} \ , \\ &\text{where } \mathbf{P}_k = \text{diag}(P(z_k|v,y,\theta^{\text{old}})) \end{aligned} \tag{5.19}$$

and matrix $\mathbf{X}$ contains the characteristics for each item y of all item/value pairs $\langle y,v\rangle$ and vector $\mathbf{v}$ contains the corresponding rating. Following DeSarbo and Cron (1988), the estimate for σ_k^2 is

$$\sigma_k^{2,\text{new}} = \frac{\sum_{\langle y,v\rangle} P(z_k|y,v,\theta^{\text{old}})(v-\mathbf{x}_y'\mathbf{b}_k)^2}{\sum_{\langle y,v\rangle} P(z_k|y,v,\theta^{\text{old}})} \ . \tag{5.20}$$

Although CLR provides us with a way to model cluster-specific regression coefficients, it has some shortcomings due to which it cannot be directly used for our problem: Contrary to the pLSA-CF model, the class memberships do not depend on the user. Thus, latent classes in this model lack a clear interpretation in terms of user communities. In fact, CLR is an extension of a Gaussian mixture model, which also lacks that ability (see Figure 5.1). Although we can compute these user communities in CLR afterwards, the model is not optimized for these user communities. Therefore, we integrate the idea of cluster-specific regression coefficients in the pLSA-CF framework with its user specific mixing weights.

5.3.3 Latent Class Regression Recommender System

As mentioned above our method, which we call the latent class regression recommender system (LCR-RS), integrates the idea of CLR into the pLSA-CF framework. Informally, this implies the following changes w.r.t. pLSA-CF: we replace item means $\mu_{y,k}$ for each latent class z_k by one vector of regression coefficients $\mathbf{b}_k$. Also, the item-specific standard deviations $\sigma_{y,k}$ for each class z_k are replaced by one class-specific standard deviation σ_{z_k}. Hence, the number of parameters becomes independent of the number of items. When there are many items, this could lead to a substantial reduction.

Therefore, we adapt the Gaussian density function (5.3) to

$$P(v|z_k,y) = P(v|\mathbf{x}_y'\mathbf{b}_k,\sigma_k) = \mathcal{N}(v|\mathbf{x}_y'\mathbf{b}_k,\sigma_k) \ , \tag{5.21}$$

where $\mathbf{x}_y$ is a vector containing the characteristics of item y. Again, the model is fitted with EM. The E-step then becomes

$$P(z_k|u,y,v,\theta^{\text{old}}) = \frac{P(z_k|u)^{\text{old}} P(v|\mathbf{x}_y'\mathbf{b}_k^{\text{old}}, \sigma_k^{\text{old}})}{\sum_{k'=1}^{K} P(z_{k'}|u)^{\text{old}} P(v|\mathbf{x}_y'\mathbf{b}_k^{\text{old}}, \sigma_{k'}^{\text{old}})} \ . \tag{5.22}$$

In the M-step, we first compute new values for the user specific mixture weights $P(z_k|u)$. Because of the dependence on u (as in Figure 5.1(a)), this can be done using (5.10) from the pLSA-CF model. Here, the difference w.r.t. CLR becomes clear, where general cluster weights were computed using (5.18), rather than user-specific ones.

In the remainder of the M-step we compute new values for the regression coefficients $\mathbf{b}_k$, and the standard deviations σ_k. The $\mathbf{b}_k$ are computed using WLS

$$\begin{aligned} \mathbf{b}_k^{\text{new}} &= (\mathbf{X}'\mathbf{P}_k\mathbf{X})^{-1}\mathbf{X}'\mathbf{P}_k\mathbf{v} \ , \\ \text{where } \mathbf{P}_k &= \text{diag}(P(z_k|u,y,v,\theta^{\text{old}})) \ , \end{aligned} \tag{5.23}$$

where $\text{diag}(P(z|u,y,v,\theta^{\text{old}}))$ is a diagonal matrix containing probabilities $P(z|u,y,v,\theta^{\text{old}})$ for all entries $\langle u,y,v\rangle$, $\mathbf{v}$ contains the ratings v of all entries, and $\mathbf{X}$ contains for all entries the characteristics of the corresponding item y. Finally, we estimate σ_k for each latent class

$$\sigma_k^{2,\text{new}} = \frac{\sum_{\langle u,y,v\rangle}(v - \mathbf{x}_y\mathbf{b}_k)^2 P(z_k|u,y,v,\theta^{\text{old}})}{\sum_{\langle u,y,v\rangle} P(z_k|u,y,v,\theta^{\text{old}})} \ . \tag{5.24}$$

Note that the model can be optimized by letting the σ_k go to 0, which is undesirable. In CLR (DeSarbo & Cron, 1988), therefore, σ_k is replaced by a small value when it is lower than a threshold value. pLSA-CF and LCR-RS use another way to overcome this problem, namely early stopping. Instead of stopping the optimization process when the loglikelihood has converged, the algorithm is stopped when the mean absolute error on an independent validation set stops to improve. Due to the early stopping, the optimization is stopped before the σ_k's start to decrease to zero and make the model uninformative, since the error on the validation set has increased. The algorithm is summarized in Figure 5.3.

After the optimization, we can use the model to predict the rating for a certain item y by user u

$$E[v|u,y] = \sum_{k=1}^{K} P(z_k|u)\mathbf{x}_y'\mathbf{b}_k \ . \tag{5.25}$$

Notice that we can also use this equation to predict ratings for items that have not yet been rated by any user as long as we have the characteristics of this item. Also, the computation of a rating is very fast, that is, of order $O(Kp)$, where K is the number of classes and p the number of characteristics.

As mentioned in the introduction, LCR-RS can also easily be used to explain the recommendations. This can be done by creating the personal regression vector $\mathbf{b}_u$ in the following way

$$\mathbf{b}_u = \sum_{k=1}^{K} P(z_k|u)\mathbf{b}_k \ . \tag{5.26}$$

procedure LCR_RS(entries $\langle u, y, v\rangle$,$\mathbf{X}$,K)

▷ Initialization

for all u **do**
Generate random $P(z_k|u)$ such that $\sum_k P(z_k|u) = 1$.
end for
for all y, z_k **do**
Generate random $\mathbf{b}_k$.
Generate random σ_k.
end for
Set $err = \infty$.

▷ EM-loop

repeat
$prevErr = err$.
for all $\langle u, y, v\rangle, k$ **do**
Compute $P(z_k|u, y, v)$ using (5.22).
end for
for all u, z_k **do**
Compute $P(z_k|u)$ using (5.10).
end for
for all y, z_k **do**
Compute $\mathbf{b}_k$ using (5.23).
Compute σ_k using (5.24).
end for
Compute $err = -E[L^c(\theta)]$ using (5.9).
until $prevErr - err < \epsilon$
end procedure

Figure 5.3: LCR-RS algorithm.

Then, using the product characteristics vector $\mathbf{x}_y$ of a highly rated product, the coefficients of the regression vector can be multiplied by the values of the characteristics. The product is then an identifier of how important that characteristic was in the recommendation. Note that when all characteristics are binary, that is have a value of 0 or 1, this multiplication is not necessary and one only has to sort the coefficients of the characteristics with a value of 1. In the same way, we can also provide insights in the user communities.

Another problem that may occur in our model and other recommender systems based on numerical ratings is commensurability. This problem may occur, since although some users may have identical item preferences, they express them using a different rating pattern. For instance, some users give very extreme ratings, where others hardly use the extreme values. Also, the average (neutral) rating may differ among users. In theory, this problem can be solved in our model using the latent classes. A latent class, then contains the users with similar taste and a similar rating pattern. However, this may require a complex model with many classes. Therefore, we use the same approach as Hofmann (2004), that is, we first compute normalized ratings v'. Ratings v' are standardized by subtracting the mean user rating and dividing by the user specific

standard deviation

$$v' = \frac{v - \mu_u}{\sigma_u} \ . \tag{5.27}$$

This standard deviation is smoothed in the following way suggested by Hofmann (2004)

$$\sigma_u = \sqrt{\frac{\sum_{\langle u=u',y,v\rangle}(v - \mu_u)^2 + 5\bar{\sigma}^2}{N_u + 5}} \ , \tag{5.28}$$

where μ_u is the user specific mean, $\bar{\sigma}^2$ the overall standard deviation, and N_u is the number of ratings done by the user.

Finally, note that the number of parameters in the LCR-RS model is $K \times (m + p + 1)$. (p denotes the number of characteristics in the $\mathbf{x}$ vector.) When there are many items, this is typically much smaller than the number of parameters in the pLSA-CF model, since the regression weights and the variances within each latent class are shared among all items rather than being item-specific. Thus, the LCR-RS model is likely to be less prone to overfitting.

5.4 Data

We evaluate our new algorithm on a data set concerning movie ratings. Since there is no publicly available data set containing both movie ratings and movie characteristics, we combine two publicly available data sets for this purpose.

The first data set we use is the Netflix data set discussed in Section 5.2. However, the Netflix data set does not contain characteristics about the movies except the title and the year of release. Therefore, we combined it with the IMDb data set. The Internet Movie Database (IMDb – See `http://www.imdb.com`) is a web site that contains information about approximately 900,000 titles (movies, TV series, etc.). It offers information such as plots, keywords that describe a title, genre classification, and participating actors and actresses. The IMDb database is available for download from the website.

When we combined both data sets and only include ratings corresponding to a movie also described in the IMDb data set, we were left with approximately 80 million ratings by about 480,000 users of 9,620 movies. Since it is practically impossible to work and experiment with such a large data set, we use a subset of these data in our evaluation.

To construct this reduced data set, first users with more than 50 ratings were selected. Next a random sample of .25% was taken from the selected users. The ratings of these sampled users form the foundation of the reduced data set. As characteristics of the movies we used dummy variables for each genre and for the 100 most often used keywords. This results in a data set of 191,059 ratings by 741 users of 6,305 movies. On average the users have rated 258 movies and a movie is on average rated by 30 users. The 6,305 movies have on average 2.50 genres and 28.2 keywords (of the 100) connected to them. We provide more statistics on the item characteristics in Table 5.1. The table shows both the number and percentage of occurrence both per rating in the data set and per item in the data set. One should especially note that there are several genres, such as Adult, Reality-TV, and News, which have only a very small number of ratings

and items connected to them. Therefore, results for these genres may be unreliable and regression coefficients may be insignificant.

5.5 Experiments & Results

In this section, we first describe the experiment setup used to evaluate the LCR-RS. Then, we discuss the findings of these experiments and, finally, we give an interpretation of the parameters of the LCR-RS model and discuss how these can be used to explain the recommendation to the user.

5.5.1 Experiment Setup

To evaluate the performance of the LCR-RS, we compared the prediction performance of LCR-RS to several other recommendation methods on the data described in the previous section.

The first method we compared with is a simple baseline method, which was also used in Hofmann (2004). The prediction for an item $\langle u, y \rangle$ is simply the mean rating for item y in the training data.

We also used a simple memory-based collaborative and three simple content-based recommendation methods as comparison. The collaborative method is the Pearson correlation method. And as content-based methods we use two approaches based on linear regression. The first approach uses a single linear regression model on the complete data set and provides the same predictions for all users. The second approach fits a linear regression model for each user, only using the items rated by this user as input. For users that do not have rated enough items to be able to train a linear regression model, the global linear regression model is used for prediction. Finally, we also tested a Nearest Neighbor algorithm (Cover & Hart, 1967) using the Hamming distance as distance measure. We report the results of using 10 neighbors, since this lead to the best results.

The final CF benchmark model we use is the original pLSA-CF model of Hofmann (2004). Since the LCR-RS model uses a lot less parameters (informally, the LCR-RS model estimates the $\mu_{y,z}$ for each latent class z by a linear function), we do not expect the LCR-RS model to improve the predictions of pLSA-CF. However, LCR-RS has the advantage that it can also predict ratings for new items.

Finally, we also included a hybrid recommender system for comparison. The method included in the experiment is Content-Boosted Collaborative Filtering (CB-CF, see Melville et al., 2002), which first fills the user-item rating matrix using a content-based method (in our case 10-Nearest Neighbor) and then performs Pearson Correlation CF on this complete matrix.

We compare the models using the data described in the previous section. We divide the data in a training and test set by randomly selecting 10% of the rated items per user (with a minimum of 1) and adding these to the test set. All other data are used in the training set. Both pLSA-CF and LCR-RS also need a validation set to stop the algorithm. This validation set is extracted from the training set in the same way as the test set was extracted from the complete data set.

Table 5.1: Item Characteristics of the data set

Characteristic	per rating #	per rating %	per item #	per item %
Intercept	191.059	100,00%	6.305	100,00%
Genres				
Comedy	84.844	44,41%	2.310	36,66%
Animation	4.553	2,38%	209	3,32%
Family	14.888	7,79%	550	8,73%
Biography	8.386	4,39%	239	3,79%
Sci-Fi	19.238	10,07%	504	8,00%
Mystery	17.021	8,91%	449	7,12%
Musical	4.557	2,39%	256	4,06%
Thriller	61.383	32,13%	1.451	23,02%
Film-Noir	388	0,20%	50	0,79%
Adult	62	0,03%	13	0,21%
Romance	45.530	23,83%	1.200	19,04%
Sport	7.668	4,01%	179	2,84%
Drama	108.071	56,56%	3.497	55,49%
Adventure	33.434	17,50%	751	11,92%
Horror	13.631	7,13%	626	9,93%
Western	3.509	1,84%	165	2,62%
Action	49.939	26,14%	1.129	17,91%
Documentary	2.032	1,06%	190	3,01%
Fantasy	21.297	11,15%	474	7,52%
War	9.078	4,75%	271	4,30%
Short	175	0,09%	29	0,46%
Reality-TV	11	0,01%	2	0,03%
History	4.588	2,40%	136	2,16%
Music	5.246	2,75%	163	2,59%
Crime	36.925	19,33%	921	14,61%
News	154	0,08%	1	0,02%

Characteristic	per rating #	per rating %	per item #	per item %
Keywords				
blockbuster	53.957	28,24%	437	6,93%
murder	51.462	26,94%	1.206	19,14%
title-spoken-by-char.	41.820	21,89%	510	8,09%
based-on-novel	41.707	21,83%	1.399	22,20%
char.-name-in-title	36.899	19,31%	996	15,80%
twist-in-the-end	31.177	16,32%	463	7,35%
independent-film	32.998	17,27%	1.821	28,90%
police	30.310	15,86%	685	10,87%
helicopter	28.847	15,10%	452	7,17%
friendship	28.453	14,89%	468	7,43%
flashback-sequence	28.046	14,68%	476	7,55%
love	27.120	14,19%	508	8,06%
father-son-rel.	26.151	13,69%	433	6,87%
violence	25.354	13,27%	482	7,65%
gun	25.000	13,08%	393	6,24%
marriage	23.904	12,51%	489	7,76%
death	23.364	12,23%	462	7,33%
new-york-city	22.703	11,88%	404	6,41%
revenge	22.900	11,99%	441	7,00%
father-daughter-rel.	22.143	11,59%	337	5,35%
dog	22.165	11,60%	422	6,70%
blood	22.466	11,76%	409	6,49%
vulgarity	22.496	11,77%	314	4,98%
hospital	21.231	11,11%	397	6,30%
mother-son-rel.	20.176	10,56%	320	5,08%

We repeat this approach in total 20 times with different random seeds. As measure for prediction error, we use the mean absolute error

$$MAE = \frac{\sum_{i=1}^{N^*} |\tilde{v}_i - v_i|}{N^*} \quad , \tag{5.29}$$

where N^* is the size of the test set and $\tilde{v}_i$ is the predicted rating and v_i the given rating. Besides as measure for the prediction error, the MAE is also used as measure on the validation set for stopping the pLSA-CF and LCR-RS algorithms.

This approach is very similar to the one used in Hofmann (2004) and has the advantage that it is more reliable than using a single test set (as is, for example, done in the Netflix prize), since the 20 fold replication reduces the variance of the estimated MAE. Using the MAE as evaluation metric is very popular in recommender system research, see e.g. Breese et al. (1998),Herlocker, Konstan, Borchers, and Riedl (1999),Herlocker, Konstan, Terveen, and Riedl (2004),Hofmann (2004), and Shardanand and Maes (1995), and has the advantages over mean squared error related measures that its interpretation is more clear and that it does not emphasize the larger errors.

Following Hofmann (2004), we implemented tempered EM (Hofmann, 2001) in pLSA-CF in which (5.8) is replaced by

$$P(z_k|u, y, v, \theta) = \frac{(P(z_k|u)P(v|\mu_{k,y}, \sigma_{k,y}))^\beta}{\sum_{k'}(P(z_{k'}|u)P(v|\mu_{k',y}, \sigma_{k',y}))^\beta} \quad , \tag{5.30}$$

where β is a tempering parameter used for regularization and, therefore, avoiding overfitting when set to a value < 1. In our experiments, setting β to .95 lead to the best results for pLSA-CF. Unfortunately, using tempered EM did not lead to improved results for LCR-RS.

For both the pLSA-CF and LCR-RS model we need to determine the number of classes K we use in the model. Often, information criteria, such as the Aikake (Aikake, 1974) or Bayes information criterion (Schwarz, 1978) are used to determine the number of classes when a model is based on loglikelihood maximization as are pLSA-CF and LCR-RS. However, due to the very large number of parameters in both models and the early stopping approach used, these criteria seem to give unreliable results in our case. Therefore, we determined the number of classes used for pLSA-CF using out-of-sample testing, that is, we repeated the evaluation method described above for different K. The average MAE over the 20 runs for these K are shown in Figure 5.4. As can be seen setting $K = 40$ lead to the lowest error and, therefore, we choose to use 40 classes for pLSA-CF in this chapter. Since using the same approach for LCR-RS is too computationally demanding, we use the same number of classes, that is, $K = 40$, for LCR-RS.

5.5.2 Experiment Results

Table 5.2 shows the test MAE's (averaged over 20 runs) for the different recommendation methods. As can be seen 5 out of 7 methods have a higher accuracy than the baseline method. Only the two content based methods based on linear regression perform worse. Also, the third pure content based method, the nearest neighbor method performs not that good, only improving the baseline by six percent. On the other hand, the collaborative methods were able to improve the

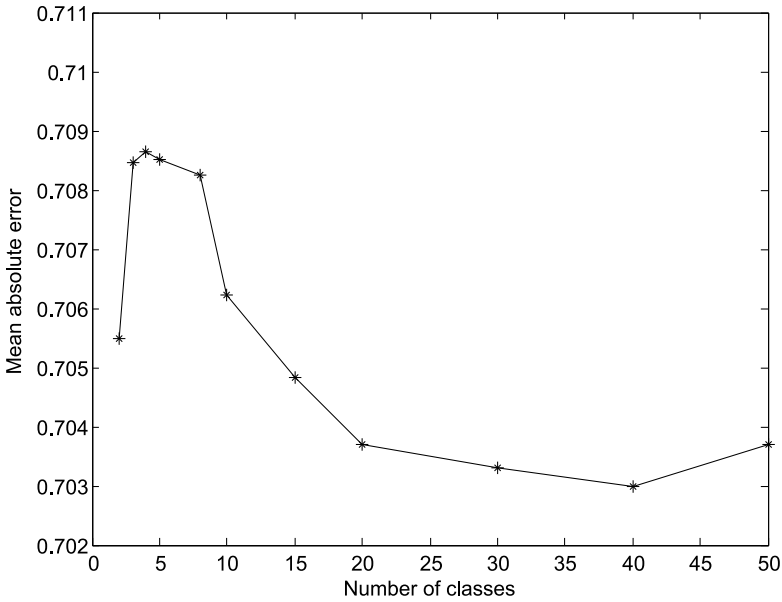


Figure 5.4: Average out-of-sample mean absolute error for different numbers of classes for the pLSA-CF model.

Table 5.2: Prediction accuracy of the different recommendation methods

Method	MAE	Improvement
Baseline	0.8309	–
Collaborative Filtering		
Pearson	0.7083	14.8%
pLSA-CF	0.7030	15.4%
Content Based Filtering		
Linear Regression	0.8671	-4.4%
User Based Linear Regression	0.9731	-17.1%
10 Nearest Neighbor	0.7782	6.3%
Hybrid Recommender Systems		
Content Boosted CF	0.7473	10.1%
LCR-RS	0.7254	12.7%

accuracy compared to the baseline by 15%. Although pLSA-CF performed somewhat better, the difference with the Pearson correlation is very small. The error made by LCR-RS is much smaller than the error made by the content-based methods, although it is somewhat higher than the error of the collaborative methods as we expected. CB-CF also performs better than the content-based methods, but worse than LCR-RS. It is remarkable that, although linear regression itself seems to be a poor performing content-based model, it works well in our framework. When comparing LCR-RS to the collaborative methods, LCR-RS only performs 3% worse than these methods. However, only LCR-RS (and Content Boosted CF) has the ability to provide recommendations of new items. Furthermore, LCR-RS is the only one of these methods that is able to explain its recommendations.

5.5.3 Cold Start Item Problem

As stated in the introduction, one of our hypotheses is that LCR-RS is better than traditional CF algorithms in predicting ratings for items that have received only a few ratings, or none at all. To evaluate whether this is really the case we have set up a simulation experiment using the data described earlier.

We first randomly selected 500 items that were rated between 50 and 150 times in the original data set. In our experiment, we will vary the number of ratings available in the training data for these items. Also, we only test how well prediction performance is on these 500 items. Therefore, we left out 10% of the original number of ratings for these items for testing. Note that all ratings of the other items (not in the set of 500 randomly selected items) are included for training. In this way, the training data is still quite balanced and comparable to the original data. In the experiments, we varied the number of training ratings for the selected 500 items to be (all) 1, 5, 10, 20, or 50.

Figure 5.5 shows the resulting MAE for the Baseline, Pearson Correlation, pLSA-CF, and the hybrid recommenders CB-CF and LCR-RS. As can be seen all CF methods have difficulties with

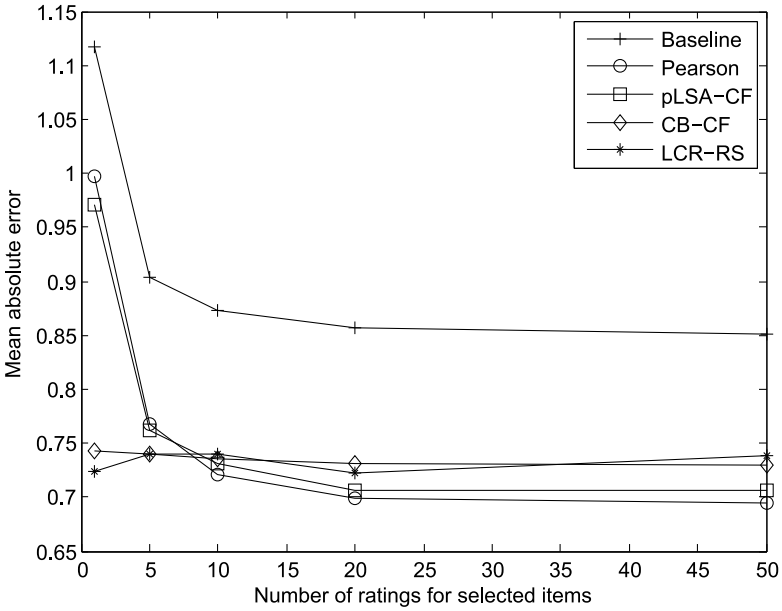


Figure 5.5: Mean absolute prediction error for selected items that were constraint to have 1, 5, 10, 20, or 50 in the training set.

giving reliable recommendations for items with only a few observations. Both hybrid methods perform much better when items have been rated less than 10 times. Although LCR-RS performs slightly worse than Pearson Correlation and pLSA-CF when enough ratings are available, it is much better in predicting ratings for relatively new items. The slight variation in errors for the LCR-RS method, which sometimes leads to higher errors while there are more observations (due to which it sometimes performs better and sometimes performs worse than CB-CF), seems to be caused by bad initializations of the LCR-RS algorithm. Therefore, results of LCR-RS may be improved by choosing a smarter initialization, for instance, using the final result of pLSA-CF. However, we do consider this beyond the scope of this chapter and a line for future research.

5.5.4 Model Interpretation

An extra advantage of the LCR-RS is that it provides an explanation of the recommendations given. Furthermore, the model can be used to get a better insight in the user communities.

For each latent class, which can be seen as some user community, the LCR-RS model contains a class specific regression vector, indicating which characteristics have a high positive or negative effect on the predicted rating. Table 5.3 shows the regression coefficients for two classes of our model for the 10 most occurring genres and keywords. As can be seen in this example, movie preferences can differ a lot among user communities. For example, for these two classes

Table 5.3: Regression coefficients for two user communities

Characteristic	Class 14	Class 15
Intercept	0.9545	0.5255
Drama	0.0889	0.0929
Comedy	-0.0774	-0.0237
Thriller	-0.0919	0.0248
Romance	-0.0403	0.0095
Action	-0.0981	-0.0403
Crime	-0.0354	-0.1090
Horror	-0.2306	-0.4084
Family	-0.0494	-0.0200
Sci-Fi	**-0.1251**	**0.0635**
Fantasy	**-0.0001**	**0.1348**
independent-film	0.0232	-0.0342
based-on-novel	0.0257	0.1692
murder	0.0116	0.0429
char.-name-in-title	0.0145	0.0390
police	-0.0239	-0.0057
title-spoken-by-char.	0.0438	-0.0215
love	0.0756	-0.0367
marriage	0.0124	0.0905
violence	-0.0449	0.0991
flashback-sequence	0.0172	-0.0181

one can see that users belonging to class 15 consider it to be quite positive that a movie belongs to the science fiction or fantasy genre, while people in class 14 do not seem to prefer these movies.

Interpretation can be quite complicated at times, especially if some characteristics are correlated which each other. For instance, the genre romance and the keyword love, will occur quite a lot together. When looking at the regression coefficients for both classes for these characteristics, we see that the one has a negative coefficient for love and a positive one for romance, where for the other class it is opposite. Therefore, it is quite difficult to see which community would like these kind of movies more. When one wants to overcome this problem of correlated characteristics one could use feature selection or extraction methods. However, although this may increase the usefulness of the explanations, it might also reduce prediction performance.

To gain somewhat more insight in the user communities in class 14 and 15 of the model, we have estimated ratings for all movies in the data based on the regression coefficients of these classes and ordered the recommendations based on these predicted ratings. In Table 5.4, we show the recommendation rank for ten well-known movies for both classes. We see that the order is often in line with the results shown in Table 5.3, such as the higher ranks for the Sci-Fi movie 'The Matrix' and action movie 'Die Hard' in class 15. However, sometimes movies get a very

Table 5.4: Recommendation order per user community for ten selected movies

Movie Title	Class 14		Class 15	
	Abs	Rel	Abs	Rel
Bridget Jones' Diary	572	7	131	1
Die Hard	1568	9	190	4
Harry Potter and the Sorcerer's Stone	55	3	220	5
I Know What You Did Last Summer	5745	10	2167	8
Reservoir Dogs	16	1	1364	7
Saving Private Ryan	38	2	352	6
Seven	125	4	3051	9
The Matrix	1459	8	147	2
Titanic	290	5	6265	10
Unforgiven	474	6	188	3

Table 5.5: Regression coefficients of most important positive and negative characteristics for user 369

Positive Characteristics	Coefficient	Negative Characteristics	Coefficient
News	0.4785	Adult	-0.2894
Short	0.3920	Horror	-0.2282
Documentary	0.2425	Sci-Fi	-0.0907
sequel	0.2176	Film-Noir	-0.0820
famous-score	0.1868	male-nudity	-0.0809

high or low rating due to a not that common characteristic not shown in Table 5.3. This might for instance be the case for the movie 'Reservoir Dogs' in class 14.

The same regression coefficients that were shown for the user communities can be computed for individual users using (5.26). We computed these individual regression coefficients for a random user, namely user 369. For this user, the most important positive and negative characteristics are listed in Table 5.5. As can be seen, the model expects that the user would like to see news, short movies, and documentaries, while he would not like adult, horror, and science-fiction movies.

These kind of regression coefficients can be used for explanation in, for example, the following way. Top recommendation for this user is the movie 'Schindler's List'. With respect to this movie, we can determine which characteristics have the most influence on this rating, that is, we look for characteristics that have a value of 1 for this movie and then look at the highest coefficients for these characteristics. These coefficients for the five most important positive and negative characteristics that determine the rating for user 369 for the movie 'Schindler's List' are shown in Table 5.6. Note that the most important positive characteristics have a far more higher coefficient than the negative ones, since this is a movie that the user is expected to like. Based on these coefficients a recommender system should be able to provide an explanation of the recommendation made, that should look something like

Table 5.6: Regression coefficients of most important positive and negative characteristics for User 369 that determine the rating for the movie 'Schindler's List'

Positive Characteristics	Coefficient	Negative Characteristics	Coefficient
famous-score	0.1868	sex	-0.0584
blockbuster	0.1801	female-nudity	-0.0438
History	0.1042	premarital-sex	-0.0409
redemption	0.1005	rescue	-0.0275
Drama	0.0802	blood	-0.0202

> We think you would like the movie **Schindler's List**, because it has a **famous score** and is considered to be a **blockbuster**. Also, the movie is about **History**, **redemption**, and it belongs to the **Drama** genre.
>
> However, there are also some things that you may not like about the movie. The movie contains **sex** and **female nudity**. Also, the movie contains **premarital sex**, a **rescue**, and **bloody** scenes.

As we said earlier, we think that the explanations of the system might be improved by using some feature selection or extraction method. Not only will such a method eliminate correlating characteristics, also characteristics that do not occur a lot and are therefore insignificant might be excluded. As can be seen in Table 5.5, some of these characteristics now have high coefficients in the model, while these are probably highly unreliable due to the limited number of positive examples they are based on.

5.6 Conclusions

In this chapter, we presented a new hybrid recommendation approach. This approach is based on the idea that there are communities of users that find the same characteristics important to like or dislike a product. To construct this model, which we call the latent-class regression recommender system (LCR-RS), we extended Hofmann's probabilistic latent semantic analysis model for collaborative filtering (pLSA-CF, see Hofmann, 2001). This extension is based on clusterwise regression (DeSarbo & Cron, 1988) and leads to a model containing community specific regression vectors. Advantages of this model are that it is able to recommend new items and provides a framework to explains its recommendations, while it is also able to use the ratings of all users.

We applied this approach on a data set containing movie ratings and movie characteristics (genres and keywords). On these data, prediction performance of LCR-RS was slightly worse than collaborative methods, such as Pearson correlation and pLSA-CF, but better than content-based methods and the hybrid CB-CF method. Although performance of LCR-RS is somewhat worse than the collaborative methods, this method has in favor that it can recommend new items and provides a way to explain its recommendations, which was also illustrated for the movie

data. In an experiment, we showed that LCR-RS performs better in recommending relatively new items than collaborative filtering methods.

We see some directions to future research based on the experiments with the pLSA-CF algorithm. We experienced that LCR-RS benefits from a good initialization. Therefore, we think performance of LCR-RS may be improved by a better initialization strategy, such as, initialization using pLSA-CF. Also, we expect that explanations of the LCR-RS might be improved using feature extraction or selection methods excluding correlated and insignificant characteristics. This can maybe done by integrating stepwise regression methods or lasso regression (Tibshirani, 1996). Furthermore, the LCR-RS model might be adapted to model other type of user evaluations such as a binary 'like - don't like' evaluation. This can be done using generalized linear models (McCullagh & Nelder, 1989), such as logistic or probit regression, in combination with latent classes (Wedel & DeSarbo, 1995).

Part III

Visualization of Large Data Sets

Chapter 6

Multidimensional Scaling for Large Data Sets*

6.1 Introduction

Multidimensional scaling (MDS, see Borg & Groenen, 2005) techniques aim to represent measurements of dissimilarity between pairs of objects as distances in a low dimensional Euclidean space. This leads to a representation in which similar objects are located close to each other and dissimilar objects far apart.

There are many different approaches to MDS, e.g. Torgersen's Principal Coordinate Analysis (Torgerson, 1952; Gower, 1966), Kruskal's Nonmetric MDS (Kruskal, 1964), Sammon Mapping (Sammon, 1969), and SMACOF (De Leeuw, 1977a, 1988; De Leeuw & Heiser, 1977). However, all these methods become very slow or not practically applicable when applied to large data sets due to their $O(n^2)$ complexity, since adding one object leads to an extra n dissimilarities. To show the rise in CPU time for various n, consider the Proxscal program in SPSS, one of the fastest algorithms currently available. Figure 6.1 shows the CPU times of generated dissimilarity data sets as a function of the number of objects n. The largest problem that could be handled had $n = 2,500$. It is obvious from this plot, that even with more memory available, CPU time will become prohibitive, even for reasonably sized MDS problems with $n = 10,000$.

The purpose of this chapter is to make large scale MDS feasible for, say, up to $n = 100,000$. We study the inherent problems that may occur when analyzing that many dissimilarities and propose practical and algorithmic solutions for it. We focus on one-step algorithms that estimate all object coordinates in one-step and do not discuss two-step approaches in which first an MDS is performed on a small subset of points and in a second step these points are kept fixed and the remaining points are fitted to these fixed points. The main reason is that this latter approach only fits the remaining points to the fixed small subset of points and totally ignores the relations between these remaining points.

A part of these kind of algorithms is based on the early work of Chalmers (1996) in which he

*This chapter is joint work with Patrick J.F. Groenen and Michael W. Trosset.

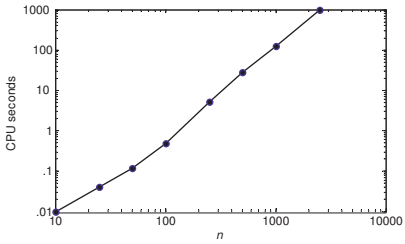


Figure 6.1: CPU times of the least-squares MDS program Proxscal available in SPSS as a function of the number of objects n.

introduced a stochastic adaptation of the original metric MDS model (also called Spring model, see Eades, 1984). This algorithm tries to find neighboring points in a stochastic way and only uses these in the optimization. Morrison, Ross, and Chalmers (2003) and Jourdan and Melançon (2004) reduce the computational complexity of this model by only using subsets of the dissimilarity matrix and fitting other dissimilarities in this final solution.

Another group of algorithms is based on classical MDS (Torgerson, 1952). Landmark MDS (Silva & Tenenbaum, 2003, 2004) first selects a number of landmark points and computes an classical MDS on these point. Then, using distance based triangulation the other points are fitted into the solution. Platt (2005) showed that Landmark MDS outperformed the older, but similar, methods FastMap (Faloutsos & Lin, 1995) and MetricMap (Wang et al., 1999). T. Yang, Liu, McMillan, and Wang (2006) and Brandes and Pich (2007) and introduced relative similar methods called FastMDS and PivotMDS. Finally, Lee and Choi (2009) introduced a method combining multiple landmark MDS solutions and mapped them in a single space.

An important contribution of this chapter lies in the realization that the efficiency of the MDS majorization algorithm can be drastically improved on large data sets when only a small subset of the $n(n-1)/2$ possible dissimilarities is used, where n is the number of objects. The assumption is that many relations between the points are redundant and that a small fraction is sufficient to show the main relations in the data. Some data sets are sparse by their nature. For example, when visually representing large scale social networks, most pairs of objects will not be directly connected, thereby yielding a very sparse matrix. Even when all dissimilarity data are available, it often will be enough to only select a (very) small sample of all dissimilarities to be able to make a useful MDS map.

In large scale MDS, special care needs to be taken for the choice of the dissimilarity measure. Irrespective of the sparseness, when n is large, MDS solutions turn out to be dominated by the most frequent dissimilarity value. As a consequence, the dissimilarity measure should have adequate spread and residual weights can be used to down weight the importance of the more frequent dissimilarities so that an informative MDS solution can be obtained.

An important advantage of using sparse data is that memory needed to store the dissimilarities becomes much more efficient as it is dependent only on the number of nonmissing dissimilarities. In addition, we propose a large scale MDS algorithm that exploits the sparseness of the data and is based on iterative majorization. Two main properties of this algorithm are (1) the complexity of the operations in each iteration is of the order of the number of nonmissing dissimilarities and (2) in each iteration the least-squares MDS loss function is reduced until convergence is reached.

A potential drawback of sparse dissimilarities is that unconnected subsets of objects may occur. If this is so, then each subset may as well be analyzed by a separate MDS analysis, one for each subset. We provide a simple algorithm that identifies such subsets of unconnected objects.

Thus, in this chapter, we discuss the steps needed to successfully obtain a large scale MDS solution.

The remainder of this chapter is structured as follows. In the next section, we will shortly introduce the SMACOF algorithm. Then, in Section 6.3 we will show how sparseness and a specific design can be used for large scale MDS. In Section 6.4, we introduce our new algorithm for large scale MDS. A short discussion on how to choose dissimilarities and weighting schemes is given in Section 6.5. Then, in Section 6.6, we show two applications of the large Scale MDS algorithm, one on a artificial data in the shape of a Swiss roll and one on an empirical Thesaurus data set. Finally, we present our conclusions and discuss some open issues.

6.2 MDS by Iterative Majorization

When we let δ_{ij} denote the dissimilarity of objects i and j and $\mathbf{X}$ be an $n \times p$ matrix containing the coordinates of the objects in a p dimensional space, MDS can be formalized by optimizing the raw stress criterion

$$\sigma_r = \sum_{i<j} w_{ij}(\delta_{ij} - d_{ij}(\mathbf{X}))^2 \ , \tag{6.1}$$

where $d_{ij}(\mathbf{X})$ denotes the Euclidean distance between the coordinates of objects i and j in low dimensional space

$$d_{ij}(\mathbf{X}) = \sqrt{\sum_{s=1}^{p}(x_{is} - x_{js})^2} \ . \tag{6.2}$$

In (6.1), $w_{ij} \geq 0$ is a weight belonging to δ_{ij}. These residual weights are especially needed for large data sets to handle missing dissimilarities. They can also be used to (de)emphasize errors of certain object pairs.

Kearsley, Tapia, and Trosset (1998) showed that one should use second-derivative information (as in Newton's method) for minimizing this criterion to achieve fast (quadratic) convergence. However, for extremely large n, we do not expect to find extremely accurate solutions, nor can we afford the Hessian, gradient, and function evaluations and manipulations that Newton's method uses to find them simply because of limitations in memory and computer power. Thus, although Kearsley et al. (1998) observed that the SMACOF algorithm (De Leeuw, 1977a,

1988; De Leeuw & Heiser, 1977), which is based on iterative majorization, exhibited slow (linear) convergence (De Leeuw, 1988), it has other features that are admirably suited to our present concerns.

We can rewrite the weighted raw stress criterion in the following way

$$\begin{aligned}\sigma_r &= \sum_{i<j} w_{ij}(\delta_{ij} - d_{ij}(\mathbf{X}))^2 \\ &= \sum_{i<j} w_{ij}\delta_{ij}^2 + \sum_{i<j} w_{ij}d_{ij}^2(\mathbf{X}) - 2\sum_{i<j} w_{ij}\delta_{ij}d_{ij}(\mathbf{X}) \\ &= \eta_\delta^2 + \text{tr } \mathbf{X}'\mathbf{V}\mathbf{X} - 2 \text{ tr } \mathbf{X}'\mathbf{B}(\mathbf{X})\mathbf{X} \ ,\end{aligned} \tag{6.3}$$

where

$$\mathbf{V} = \sum_{i<j} w_{ij}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)' \tag{6.4}$$

having elements

$$v_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j \\ \sum_{j,j\neq i} w_{ij} & \text{if } i = j \end{cases} \tag{6.5}$$

and

$$\mathbf{B}(\mathbf{X}) = \sum_{i<j} \frac{w_{ij}\delta_{ij}}{d_{ij}(\mathbf{X})}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)' \tag{6.6}$$

having elements

$$b_{ij}(\mathbf{X}) = \begin{cases} -w_{ij}\delta_{ij}/d_{ij}(\mathbf{X}) & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) > 0, \\ 0 & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) = 0, \\ \sum_{j,j\neq i} -b_{ij} & \text{if } i = j. \end{cases} \tag{6.7}$$

Both $\mathbf{V}$ and $\mathbf{B}(\mathbf{X})$ are $n \times n$ matrices. The first part of (6.3) does not depend on $\mathbf{X}$; so η_δ^2 is constant. The second part, tr $\mathbf{X}'\mathbf{V}\mathbf{X}$, is quadratic in $\mathbf{X}$, which is easy to handle. De Leeuw (1988) showed that the last part can be majorized by

$$-2\text{tr } \mathbf{X}'\mathbf{B}(\mathbf{X})\mathbf{X} \leq -2 \text{ tr } \mathbf{X}'\mathbf{B}(\mathbf{Z})\mathbf{Z} \tag{6.8}$$

with equality when $\mathbf{X} = \mathbf{Z}$. Therefore, it holds that

$$\sigma_r(\mathbf{X}) \leq \mu(\mathbf{X}|\mathbf{Z}) = \eta_\delta^2 + \text{ tr } \mathbf{X}'\mathbf{V}\mathbf{X} - 2 \text{ tr } \mathbf{X}'\mathbf{B}(\mathbf{Z})\mathbf{Z} \tag{6.9}$$

and $\mu(\mathbf{X}|\mathbf{Z})$ is a majorizing function of $\sigma_r(\mathbf{X})$. The minimizer of $\mu(\mathbf{X}|\mathbf{Z})$ is

$$\mathbf{X}^* = \mathbf{V}^+\mathbf{B}(\mathbf{Z})\mathbf{Z} \ , \tag{6.10}$$

where $\mathbf{V}^+$ is the Moore-Penrose inverse of $\mathbf{V}$. Replacing $\mathbf{Z}$ by the estimates of $\mathbf{X}$ in the previous iteration, we get the update rule for the iterative majorization algorithm

$$\mathbf{X}^{[k]} = \mathbf{V}^+\mathbf{B}(\mathbf{X}^{[k-1]})\mathbf{X}^{[k-1]} \ , \tag{6.11}$$

procedure SMACOF($\boldsymbol{\Delta}$,$\mathbf{W}$,p)
  $\mathbf{V} = \sum_{i<j} w_{ij}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)'$.
  Compute $\mathbf{V}^+$ as Moore-Penrose inverse of $\mathbf{V}$.
  $k = 0$
  Generate random $n \times p$ matrix $\mathbf{X}^{[0]}$.
  $\sigma_r^{[0]}(\mathbf{X}^{[0]}) = \sum_{i<j} w_{ij}(\delta_{ij} - d_{ij}(\mathbf{X}^{[0]}))^2$.
  repeat
    $k = k + 1$.
    $\mathbf{B}(\mathbf{X}^{[k-1]}) = \sum_{i<j} \frac{w_{ij}\delta_{ij}}{d_{ij}(\mathbf{X}^{[k-1]})}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)'$.
    $\mathbf{Y} = \mathbf{B}(\mathbf{X}^{[k-1]})\mathbf{X}^{[k-1]}$.
    $\mathbf{X}^{[k]} = \mathbf{V}^+\mathbf{Y}$.
    $\sigma_r^{[k]}(\mathbf{X}^{[k]}) = \sum_{i<j} w_{ij}(\delta_{ij} - d_{ij}(\mathbf{X}^{[k]}))^2$.
  until $(\sigma_r^{[k-1]} - \sigma_r^{[k]}) < \epsilon$.
end procedure

Figure 6.2: SMACOF algorithm.

the so-called Guttman transform.

Note that $\mathbf{V}^+$ only needs to be computed once. In the case that all $w_{ij} = 1$, $\mathbf{V}^+ = n^{-1}(\mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}')$. Since $\mathbf{B}(\mathbf{X}^{[k-1]})\mathbf{X}^{[k-1]}$ is already centered, this simplifies the update (6.11) to

$$\mathbf{X}^{[k]} = n^{-1}\mathbf{B}(\mathbf{X}^{[k-1]})\mathbf{X}^{[k-1]} \quad . \tag{6.12}$$

When not all weights are equal to one, computing $\mathbf{V}^+$ is of $O(n^3)$ complexity. The other computations in the update step are of lower complexity, but have to be recomputed each iteration. Computing $\mathbf{Y} = \mathbf{B}(\mathbf{X}^{[k-1]})\mathbf{X}^{[k-1]}$ is of order n^2. Finally, the multiplication $\mathbf{X}^{[k]} = \mathbf{V}^+\mathbf{Y}$ has a complexity of $O(n^2p)$. The SMACOF algorithm is summarized in Figure 6.2.

6.3 Sparse Data

When dealing with large data sets often a lot of mutual dissimilarities may be missing, that is, these dissimilarities have a weight of 0. In such cases, computation and data storage in the SMACOF algorithm can be made much more efficient. Instead of storing all dissimilarities in some dissimilarity matrix $\boldsymbol{\Delta}$, we create a vector $\boldsymbol{\delta}$ with items δ_r only containing the nonmissing dissimilarities having $i < j$. Furthermore, we define two vectors $\mathbf{i}$ and $\mathbf{j}$ to contain the indices i and j of these dissimilarities, such that $\delta_r = \delta_{i_r j_r}$. This results in that we only have to store $3n^*$ numbers (the indices in $\mathbf{i}$ and $\mathbf{j}$ and dissimilarities in $\boldsymbol{\delta}$ for all nonmissing dissimilarities) , where $n^* \leq \frac{1}{2}n(n-1)$ is the number of nonmissing dissimilarities, instead of storing the n^2 elements of matrix $\boldsymbol{\Delta}$. When weights of nonmissing dissimilarities are not constrained to be either 0 or 1, we also create a n^*-length vector $\mathbf{w}$ with items w_r. We also use such vectors in all computations and storage of the matrices $\mathbf{V}$ and $\mathbf{B}(\mathbf{X}^{[k-1]})$ resulting in the vectors $\mathbf{v}$ and $\mathbf{b}(\mathbf{X}^{k-1]})$.

First, we compute the items of vector $\mathbf{v}$. For all nonmissing dissimilarities we compute

$$v_r = -w_r \quad . \tag{6.13}$$

Furthermore, we have to compute the diagonal elements of the original matrix $\mathbf{V}$ and store these in n-length vector $\mathbf{v}^D$, which can be done using the following procedure

$\mathbf{v}^D = \mathbf{0}$
for $r = 1$ to $r = n^*$ **do**
  $v^D_{i_r} = v^D_{i_r} + w_r \quad .$
  $v^D_{j_r} = v^D_{j_r} + w_r \quad .$
end for

The computation of $\mathbf{v}$ only requires $2n^*$ computations and has as size of n^*. Vector $\mathbf{v}^D$ has n elements and can be computed using n^* computations. However, since we need the pseudo inverse of $\mathbf{V}$ in our computations, which is not sparse, we still have to store, and to do computations, with the $n \times n$ matrix $\mathbf{V}^+$. Also, the computation of $\mathbf{V}^+$ still is of $O(n^3)$ complexity.

We can also make a vector $\mathbf{b}(\mathbf{X}^{[k-1]})$ instead of using matrix $\mathbf{B}(\mathbf{X}^{[k-1]})$. For all nonmissing dissimilarities, we compute

$$b_r(\mathbf{X}) = \frac{-w_r \delta_r}{d_r(\mathbf{X})} \quad . \tag{6.14}$$

Since we can express the diagonal elements of $\mathbf{B}(\mathbf{X}^{[k-1]})$ in terms of its off-diagonal elements, we do not need to compute these separately as we will see later. To compute $\mathbf{b}(\mathbf{X}^{[k-1]})$ we need n^* computations and need to store n^* values. Since we only need to compute (6.14) for nonmissing dissimilarities, we also only need to compute the distances corresponding with these nonmissing dissimilarities. Therefore, we use $d_r(\mathbf{X})$ instead of $d_{ij}(\mathbf{X})$, which is defined as

$$d_r(\mathbf{X}) = \sqrt{\sum_{s=1}^{p} (x_{i_r s} - x_{j_r s})^2} \quad . \tag{6.15}$$

The computation $\mathbf{Y} = \mathbf{B}(\mathbf{X}^{[k-1]})\mathbf{X}^{[k-1]}$ leads to a nonsparse matrix $\mathbf{Y}$, also in the case of a sparse $\mathbf{B}(\mathbf{X})$. However, the computation of the elements of $\mathbf{Y}$ can be done more efficiently. Note that $\mathbf{X}$ is a nonsparse matrix of size $n \times p$. Also, we use the fact that the diagonal elements of $\mathbf{B}(\mathbf{X})$ can be expressed as $b_{ii} = \sum_{j,j \neq i} -b_{ij}$. This leads to the following procedure to compute $\mathbf{Y}$

$\mathbf{Y} = \mathbf{0}$
for $r = 1$ to $r = n^*$ **do**
  for $s = 1$ to $s = p$ **do**
    $y_{i_r s} = y_{i_r s} + b_r(x_{i_r s} - x_{j_r s}) \quad .$
    $y_{j_r s} = y_{j_r s} + b_r(x_{j_r s} - x_{i_r s}) \quad .$
  end for
end for

When using this way of computation only $2n^*p$ computations are needed instead of the n^2p computations in the nonsparse SMACOF algorithm.

Since both $\mathbf{V}^+$ and $\mathbf{Y}$ are nonsparse matrices, we cannot use the same approach to this step of the algorithm. Therefore, this step remains to have $O(n^2p)$ complexity. To reduce the computational complexity of this step, we propose a new MDS algorithm for large sparse data sets in Section 6.4. First, we discuss in the next section how a specific data design can be used to reduce computational complexity of the algorithm. This efficient version of the SMACOF algorithm for sparse data sets is summarized in Figure 6.3.

6.3.1 Structured Data Design

In a simulation study, Spence and Domoney (1974) showed that it is possible to recover MDS distances quite accurately using only a small subset of the original dissimilarities, that is, all other dissimilarities are given a weight of 0.

When we select these dissimilarities according to a specific design, Gower and Groenen (1991) show that $\mathbf{V}^+$ can be computed in linear time, that is, has $O(n)$ complexity. This design is imposed on the weight matrix $\mathbf{W}$, where only a small subset of the weights will be larger than 0. The design we impose on $\mathbf{W}$ is called the symmetric circulant design which is shown in Figure 6.4 and has the following properties. Matrix $\mathbf{W}$ is circulant when it holds that for every row $\mathbf{w}_{i+1}$ it holds that this row is equal to row $\mathbf{w}_i$, except that all elements are shifted one position to the right and the final element is put in the first position. The circulant $\mathbf{W}$ is also symmetric, when it holds that $w_{ij} = w_{i(n-j+2)}$ for all $i, j = 1, 2, \ldots, n$.

Assuming that $\mathbf{W}$ is symmetric circulant and that row $\mathbf{w}_1$ has q values $w_{1j} \neq 0$, every row $\mathbf{w}_i$ has exactly q values $w_{ij} \neq 0$ meaning that every object is connected to exactly q other objects.

Using the fact that the matrix $\mathbf{W}$ is symmetric circulant, the complexity of some computations in the iterative majorization algorithm is reduced. Computing $\mathbf{V}^+$ can now be done in $O(n)$ as shown by Gower and Groenen (1991). Computing $\mathbf{Y} = \mathbf{B}(\mathbf{X}^{[k-1]})\mathbf{X}^{[k-1]}$ (using the efficient SMACOF algorithm) is still of order $O(n^*)$, only now $n^* = qn$, which is linear in n. Only the computation of $\mathbf{X}^{[k]} = \mathbf{V}^+\mathbf{Y}$ still is not linear in n having a complexity of $O(n^2p)$. Therefore, we will introduce a new iterative majorization algorithm for Large Scale MDS in the next section, which has overall $O(n)$ complexity, when using a symmetric circulant design and $O(n^*)$ complexity in general.

6.3.2 Unconnected Subsets

When the data is sparse by itself another problem may occur: the data may consists of unconnected subsets of objects. In that case, as the relative position of one subset of points to another is undefined, separate MDS analyses should be performed for each subset. Here we discuss a simple algorithm that determines the unconnected subsets in the weights matrix. This algorithm should always be used before a Large Scale MDS analysis, unless the subsets are known in advance through construction of the data or when a design is used with known connectedness properties, such as the symmetric circulant design. This algorithm for subset retrieval is very

procedure EFFICIENTSMACOF($\boldsymbol{\delta}$,**i**,**j**,**w**,n,p)
  Compute $\mathbf{v}$:
  for $r = 1$ to $r = n^*$ **do**
    $v_r = -w_r \ .$
  end for

  Compute $\mathbf{v}^D$:
  $\mathbf{v}^D = \mathbf{0}$
  for $r = 1$ to $r = n^*$ **do**
    $v^D_{i_r} = v^D_{i_r} + w_r \ .$
    $v^D_{j_r} = v^D_{j_r} + w_r \ .$
  end for

  Compute $\mathbf{V}^+$ using $\mathbf{v}$ and $\mathbf{v}^D$.
  $k = 0$
  Generate random $n \times p$ matrix $\mathbf{X}^{[0]}$.
  $\sigma_r^{[0]}(\mathbf{X}^{[0]}) = \sum_r w_r(\delta_r - d_r(\mathbf{X}^{[0]}))^2$.

  repeat
    $k = k + 1$.
    Compute $\mathbf{b}$:
    for $r = 1$ to $r = n^*$ **do**
      $b_r(\mathbf{X}) = -w_r\delta_r/d_r(\mathbf{X}) \ .$
    end for

    Compute $\mathbf{Y}$:
    $\mathbf{Y} = \mathbf{0}$
    for $r = 1$ to $r = n^*$ **do**
      for $s = 1$ to $s = p$ **do**
        $y_{i_rs} = y_{i_rs} + b_r(x_{i_rs} - x_{j_rs}) \ .$
        $y_{j_rs} = y_{j_rs} + b_r(x_{j_rs} - x_{i_rs}) \ .$
      end for
    end for

    $\mathbf{X}^{[k]} = \mathbf{V}^+\mathbf{Y}$.
    $\sigma_r^{[k]}(\mathbf{X}^{[k]}) = \sum_r w_r(\delta_r - d_r(\mathbf{X}^{[k]}))^2$.
  until $(\sigma_r^{[k-1]} - \sigma_r^{[k]}) < \epsilon$.
end procedure

Figure 6.3: Efficient SMACOF algorithm.

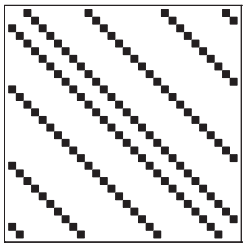


Figure 6.4: Symmetric circulant design.

similar to algorithms used to analyze the connectivity of graphs, such as breadth first and depth first search (Cormen, Leiserson, Rivest, & Stein, 2009).

Input for the subset retrieval algorithm are the vectors $\mathbf{i}$ and $\mathbf{j}$, that are the indexes i and j of the nonmissing dissimilarities. Furthermore, we initialize the algorithm with an n-length vector $\mathbf{s}$ containing the subset numbers of items i. At the start, we assume that each item belongs to its own subset, that is, $s_i = i$. Then, we start to loop over the nonmissing dissimilarities. For each i_r and j_r we retrieve the corresponding subset numbers s_{i_r} and s_{j_r}. If $s_{i_r} \neq s_{j_r}$, we set the largest subset number to the value of the smallest subset number, such that from now on, items i_r and j_r are in the same subset. Note that all the items that are in the subset with the larger subset number need to be changed as well to the smaller subset number. After having analyzed all nonmissing i, j-pairs, we can count the number of unique subset numbers in $\mathbf{s}$ and this is the number of unconnected subsets in the data. Note that it is easy to identify subsets of size one, that is, a single item in $\mathbf{s}$. This approach is summarized in Figure 6.5.

6.4 Large Scale MDS

In the previous section we showed that we can reduce the complexity of the algorithm, when we use only a subset of the data. However, one step in the algorithm, computing $\mathbf{X}^{[k]} = \mathbf{V}^{+}\mathbf{Y}$, still has a quadratic complexity, since it involves a multiplication of an $n \times n$ with an $n \times p$ matrix. To avoid this matrix multiplication, we propose a new majorization step to majorize the second part of (6.3), that is, tr $\mathbf{X}'\mathbf{V}\mathbf{X}$, such that we only need to premultiply $\mathbf{Y}$ with a diagonal matrix, resulting in $O(np)$ complexity.

It can be proven that $\mathbf{V} - 2\mathbf{V}_D$ is negative semi definite, where $\mathbf{V}_D = \text{diag}(\mathbf{V})$. Therefore, it holds that

$$\text{tr } (\mathbf{X} - \mathbf{Z})'[\mathbf{V} - 2\mathbf{V}_D](\mathbf{X} - \mathbf{Z}) \leq 0. \tag{6.16}$$

Using this inequality, we can derive that

$$\text{tr } \mathbf{X}'\mathbf{V}\mathbf{X} \leq 2 \text{ tr } \mathbf{X}'\mathbf{V}_D\mathbf{X} - 2 \text{ tr } \mathbf{X}'[2\mathbf{V}_D - \mathbf{V}]\mathbf{Z} + c, \tag{6.17}$$

where c is some constant not dependent on $\mathbf{X}$. With this new majorization step we can ob-

```
procedure UNCONNECTEDSUBSETS(i,j,n)
    Initialize subset vector s:
    for i = 1 to i = n do
        s_i = i .
    end for
    for r = 1 to r = n* do
        if s_{i_r} ≠ s_{j_r} then
            a = min(s_{i_r}, s_{j_r}) .
            b = max(s_{i_r}, s_{j_r})
            for i = 1 to i = n do
                if s_i = b then
                    s_i = a .
                end if
            end for
        end if
    end for
    return s .
end procedure
```

Figure 6.5: Algorithm to retrieve unconnected subsets.

tain a different majorization function $\mu_{LS}(\mathbf{X}|\mathbf{Z})$, originally suggested by Groenen, Heiser, and Meulman (1999)

$$\begin{aligned}\sigma_r(\mathbf{X}) &\leq \mu(\mathbf{X}|\mathbf{Z}) = \eta_\delta^2 + \text{ tr } \mathbf{X}'\mathbf{V}\mathbf{X} - 2 \text{ tr } \mathbf{X}'\mathbf{B}(\mathbf{Z})\mathbf{Z} \\ &\leq \mu_{LS}(\mathbf{X}|\mathbf{Z}) = c + 2 \text{ tr } \mathbf{X}'\mathbf{V}_D\mathbf{X} \\ &\qquad -2 \text{ tr } \mathbf{X}'[2\mathbf{V}_D - \mathbf{V}]\mathbf{Z} - 2 \text{ tr } \mathbf{X}'\mathbf{B}(\mathbf{Z})\mathbf{Z} \qquad (6.18) \\ &= c + 2 \text{ tr } \mathbf{X}'\mathbf{V}_D\mathbf{X} - 2 \text{ tr } \mathbf{X}'[2\mathbf{V}_D - \mathbf{V} + \mathbf{B}(\mathbf{Z})]\mathbf{Z}. \qquad (6.19)\end{aligned}$$

The minimizer of our new majorization function $\mu_{LS}(\mathbf{X}|\mathbf{Z})$ is

$$\mathbf{X}^* = \mathbf{Z} + \frac{1}{2}\mathbf{V}_D^{-1}[\mathbf{B}(\mathbf{Z}) - \mathbf{V}]\mathbf{Z}, \qquad (6.20)$$

leading to the update

$$\mathbf{X}^{[k]} = \mathbf{X}^{[k-1]} + \frac{1}{2}\mathbf{V}_D^{-1}[\mathbf{B}(\mathbf{X}^{[k-1]}) - \mathbf{V}]\mathbf{X}^{[k-1]}. \qquad (6.21)$$

Because $\mu_{LS}(\mathbf{X}|\mathbf{Z}) \geq \mu(\mathbf{X}|\mathbf{Z})$, minimizing the Large Scale MDS majorization function $\mu_{LS}(\mathbf{X}|\mathbf{Z})$ seems to be a less efficient optimization strategy than minimizing the original SMACOF majorization function $\mu(\mathbf{X}|\mathbf{Z})$, because it typically would require a lot more iterations.

However, the iterations of the Large Scale MDS algorithm are substantially less expensive than the ones in the original SMACOF algorithm even when using a symmetric circulant design.

We use the same kind of computations as in the efficient version of SMACOF to compute $\mathbf{v}$ and $\mathbf{v}^D$. Then, $\mathbf{v}^*$ is computed as a sparse version of $\mathbf{V}_D^{-1}$. However, these steps are easily expressed in terms of $\mathbf{w}$ and we compute $\mathbf{v}^*$ using this procedure

$\mathbf{v}^D = \mathbf{0}$
for $r = 1$ to $r = n^*$ **do**
  $v^D_{i_r} = v^D_{i_r} + w_r \ .$
  $v^D_{j_r} = v^D_{j_r} + w_r \ .$
end for
for $i = 1$ to $i = n$ **do**
  $v^*_i = 1/v^D_i \ .$
end for

Hence, we only need to compute and store the n diagonal elements of $\mathbf{v}^*$. The computational complexity of computing $\mathbf{v}^*$ is $O(n^*)$.

Also, $\mathbf{B}(\mathbf{X}^{[k-1]})$ can still be computed in the same way, that is, using (6.14). Before we can compute $\mathbf{Y}$, we have to compute $\mathbf{U} = \mathbf{B}(\mathbf{X}^{[k-1]}) - \mathbf{V}$. Since $\mathbf{B}(\mathbf{X}^{[k-1]})$ and $\mathbf{V}$ are both sparse matrices with the same elements and the same structure of the diagonal elements, we can compute a vector $\mathbf{u}$ for the nonmissing dissimilarities

$$u_r = b_r - v_r = b_r + w_r \ . \tag{6.22}$$

Hence, also this step has $O(n^*)$ complexity.

The computation of $\mathbf{Y} = \mathbf{U}\mathbf{X}^{[k-1]}$ is very similar to the computation of $\mathbf{Y}$ in the original sparse SMACOF algorithm (Figure 6.3, lines 27 to 33). Only, instead of multiplying by the elements b_r, we now multiply by the elements u_r. So also the complexity stays the same, that is, a complexity of $O(n^*p)$. Premultiplying $\mathbf{Y}$ with $\mathbf{V}_D^{-1}$, can be done using $\mathbf{v}^*$

$$t_{is} = v^*_i y_{is} \ , \tag{6.23}$$

where $\mathbf{T} = \mathbf{V}_D^{-1}\mathbf{Y}$. It is easy to see that this step now has a complexity of $O(np)$ instead of the $O(n^2p)$ complexity of the computation of $\mathbf{X}^{[k]} = \mathbf{V}^+\mathbf{Y}$ in the original SMACOF algorithm.

Multiplying $\mathbf{T}$ by $1/2$ and adding this to $\mathbf{X}^{[k-1]}$ are both standard matrix computations that require $O(np)$ computations. The complete Large Scale MDS algorithm is summarized in Figure 6.6.

6.5 Choosing Dissimilarity Measures and Weights in Large Scale MDS

In practice, large MDS solutions turn out to be dominated by the most frequent dissimilarity value. That is, the mode of the distribution of the dissimilarities tends to dominate the solution.

procedure LARGESCALEMDS($\boldsymbol{\delta}$,**i**,**j**,**w**,n,p)
  Compute $\mathbf{v}^*$:
  $\mathbf{v}^D = \mathbf{0}$
  for $r = 1$ to $r = n^*$ **do**
    $v^D_{i_r} = v^D_{i_r} + w_r$.
    $v^D_{j_r} = v^D_{j_r} + w_r$.
  end for
  for $i = 1$ to $i = n$ **do**
    $v^*_i = 1/v^D_i$.
  end for
  $k = 0$
  Generate random $n \times p$ matrix $\mathbf{X}^{[0]}$.
  $\sigma_r^{[0]}(\mathbf{X}^{[0]}) = \sum_r w_r(\delta_r - d_r(\mathbf{X}^{[0]}))^2$.
  repeat
    $k = k + 1$.
    Compute $\mathbf{U}$:
    for $r = 1$ to $r = n^*$ **do**
      $u_r = -w_r\delta_r/d_r(\mathbf{X}) + w_r$.
    end for

    Compute $\mathbf{Y}$:
    $\mathbf{Y} = \mathbf{0}$.
    for $r = 1$ to $r = n^*$ **do**
      for $s = 1$ to $s = p$ **do**
        $y_{i_r s} = y_{i_r s} + u_r(x_{i_r s} - x_{j_r s})$.
        $y_{j_r s} = y_{j_r s} + u_r(x_{j_r s} - x_{i_r s})$.
      end for
    end for

    Compute $\mathbf{T}$:
    for $i = 1$ to $i = n$ **do**
      for $s = 1$ to $s = p$ **do**
        $t_{is} = v^*_i y_{is}$.
      end for
    end for
    $\mathbf{X}^{[k]} = \mathbf{X}^{[k-1]} + \frac{1}{2}\mathbf{T}$.
    $\sigma_r^{[k]}(\mathbf{X}^{[k]}) = \sum_r w_r(\delta_r - d_r(\mathbf{X}^{[k]}))^2$.
  until $(\sigma_r^{[k-1]} - \sigma_r^{[k]}) < \epsilon$.
end procedure

Figure 6.6: Large Scale MDS algorithm.

Such solutions in 2D are characterized by a circle shape with points uniformly distributed in the circle. The reason for this effect lies in the behavior of the least squares loss function for constant dissimilarities. Buja, Logan, Reeds, and Shepp (1994) have proved that if all dissimilarities have the same value (and none is missing) then least-squares MDS yields in 2D points in a concentric circle, in 1D points equally spaced on a line, and in 3D or higher, points evenly distributed in a hypersphere. Practical experience has shown that large data sets contain a large number of equal valued dissimilarities tend to dominate the solution and yield the uninformative constant dissimilarity type solution. To handle this problem and retrieve meaningful large scale MDS solutions, it is essential to de-emphasize these dissimilarities choosing an appropriate dissimilarity measure and weighting scheme.

We show an example of this problem on the Edinburgh Associative Thesaurus (EAT)[1] data set. The purpose is to create a map in which similar terms are located close to each other, where we define similar terms as terms that are relatively often associated with each other. The EAT data set is based on an empirical experiment held between 1968 and 1971 under undergraduate students on British universities in which subjects were asked to give a word (response) that came in their minds first given a stimulus word. The data set we use contains counts on how often a certain response was given to a stimulus. However, to be able to perform MDS, we symmetrized the data, such that a cell c_{ij} contains the number of times term i was the response to stimulus term j plus the number of times term j was the response to stimulus i. We call this number c_{ij} the number of associations between i and j. Note that, this number of associations has the same properties as co-occurrence counts in many other applications. In total, there are 23,219 terms in this data set. However, there are only 325,060 nonzero association counts between terms, meaning that only 0.1% of the data is nonzero.

In Figure 6.7, we show a histogram of the distribution of nonzero association counts. It is clear that the vast majority of the counts have a value of 1. When we use a simple strategy to create dissimilarities, that is, we define the dissimilarity as one over the association count and set all nonzero weights to 1, we get a Large Scale MDS solution as shown in Figure 6.8a. The Shepard diagram of this solution is shown in Figure 6.8b. This solution clearly shows a circle with many points densely distributed within the circle and somewhat less dense outside the circle. This outer region of the circle shape contains only terms with very few nonzero associations with other terms. Therefore, the positions of these terms are relatively free. Also, the Shepard diagram shows that there is hardly any relation between distances in the solution and the original dissimilarities as each of the distinct dissimilarity values is represented by the whole range of distances. As a consequence, this solution is totally uninformative.

In the next section we show how weighting can help to retrieve sensible information out of this Large Scale MDS solution for the Thesaurus data.

[1] `http://www.eat.rl.ac.uk`

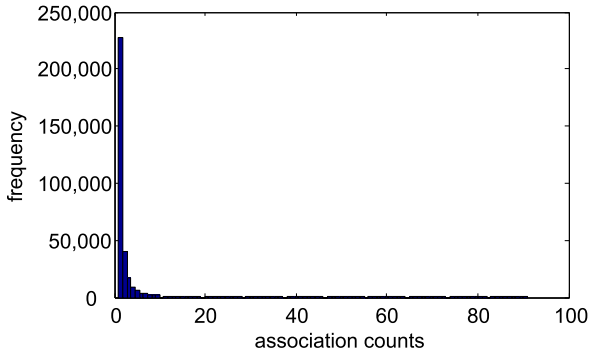
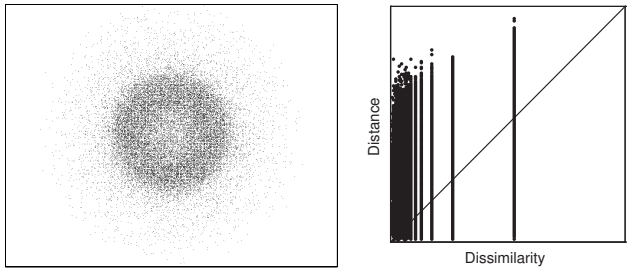


Figure 6.7: Histogram showing the distribution of association counts in the Thesaurus data set.



a. Solution b. Shepard diagram

Figure 6.8: Solution and Shepard diagram for the Thesaurus data set using one over association counts as dissimilarity measure.

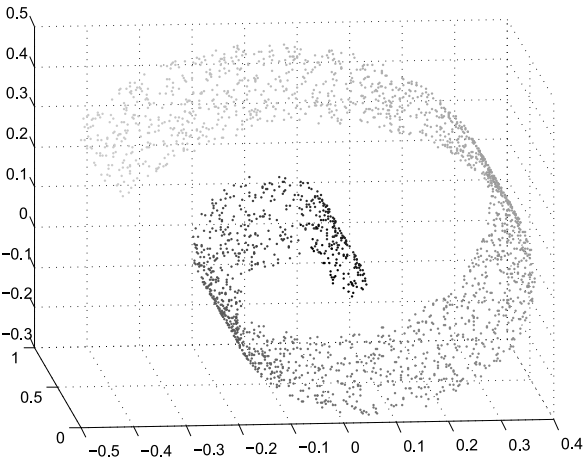


Figure 6.9: A Swiss Roll of 2500 data points.

6.6 Applications

In this section we show applications on two data sets of the Large Scale MDS algorithm. The first application deals with revealing the intrinsic dimensionality of a Swiss roll. In the second application, discussed in Section 6.6.2, we visualize a, very large but sparse, thesaurus data set.

6.6.1 Swiss Roll Data

The Swiss roll is a 3D shape, shown in Figure 6.9, which encompasses a 2D nonlinear manifold. The Swiss roll has been used before as an MDS application in Tenenbaum (1998) and Tenenbaum, Silva, and Langford (2000), where the Isomap algorithm was introduced. They used their algorithm to unroll the Swiss roll, that is, to recover the 2D manifold. A problem in unrolling the Swiss roll using ordinary MDS is that Euclidean distances between points far away from each other on the manifold do not correspond to the distance between these points along the manifold, but are much smaller. While Tenenbaum (1998) and Tenenbaum et al. (2000) estimate a full dissimilarity matrix, measuring distances along the manifold, using local distances and a shortest path algorithm, another strategy to unroll the Swiss roll by MDS is to select only the smallest distances in the 3D space as the dissimilarities and declare the rest missing, since we are sure that these are correctly on the manifold. This results in a sparse MDS problem.

In our application, we use 2500 points uniformly distributed over the Swiss roll, see Figure 6.9. In our application, we will only use the 2.5% smallest, that is, 78,094 of the original 3,123,750, 3D Euclidean distances which leads to a sparse dissimilarity matrix and, hence, it might be beneficial to use the Large Scale MDS algorithm.

We performed this task with both the original SMACOF algorithm and the Large Scale MDS

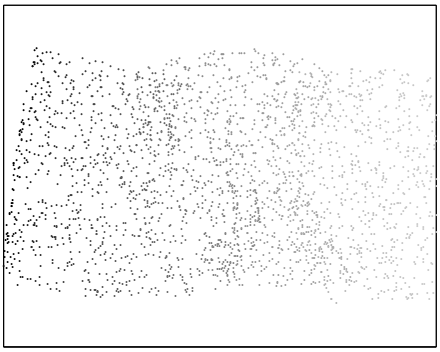


Figure 6.10: Large Scale MDS solution for the Swiss roll only using the 2.5% smallest dissimilarities.

algorithm. The SMACOF algorithm was programmed in Matlab and did not make use of the sparse structure of the matrix in any way. The Large Scale MDS algorithm was also implemented in Matlab, although the most computations are done using linked Fortran code to make efficient use of the algorithm in Figure 6.6. For both algorithms we made use of the same random start configuration. The SMACOF solution has a normalized Stress of 0.0002 and converged after 14 iterations. In total, it took 856 CPU seconds before the algorithm converged. The Large Scale MDS solution, shown in Figure 6.6.1, has a normalized Stress of 0.0120 which is higher than the SMACOF solution, but still low. Also, the algorithm needed 2500 iterations taking only 37 CPU seconds. Consequently, the Large Scale MDS algorithm is more than 20 times faster than the SMACOF algorithm, yielding both approximately the same solution. The present analysis can be repeated for larger n. Then, larger speed gains will be obtained for the Large Scale MDS algorithm compared to standard SMACOF.

6.6.2 Thesaurus Data Set

In Section 6.5 we argued that for large scale dissimilarities as the Thesaurus data set it is essential to choose a proper dissimilarity measure and weighting scheme to be able to create a meaningful visualization. A wide variety of (dis)similarity measures is available to compute dissimilarity from co-occurrence data (see e.g., Gower & Legendre, 1986). Tobler and Wineberg (1971) proposed to use the gravity model in such circumstances, which has also been used by, e.g., Zielman and Heiser (1993) and Groenen and Heiser (1996). It is directly derived from the relation between gravity forces, the masses of large objects (such as moon and earth) and their distances. Using this gravity model, we can define the dissimilarity between two terms as

$$\delta_{ij} = \frac{o_i o_j}{c_{ij}} \ , \tag{6.24}$$

where o_i is the total number of occurrences of term i and c_{ij} is the co-occurrence between objects i and j. The original gravity model defines the dissimilarity as the square root of (6.24). However,

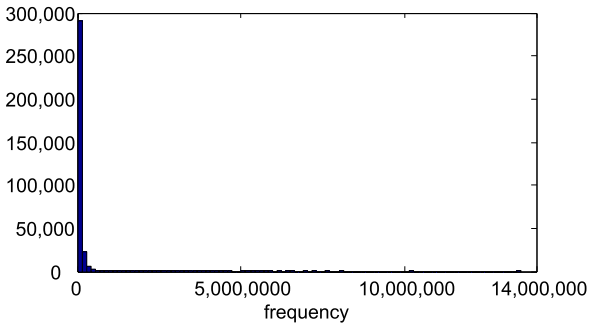


Figure 6.11: Histogram showing the distribution of the dissimilarities created using the gravity model.

we do not do this in this application as the resulting map turns out more pronounced without taking the square root and thereby yielding a clearer interpretation. In the gravity model, the largest dissimilarity scores are given between very frequently occurring terms that co-occur very infrequently. When $c_{ij} = 0$, dissimilarity δ_{ij} is not defined in the gravity model and considered as missing. In our case, with many unconnected words a very sparse dissimilarity matrix is obtained. Note that the inverse of (6.24) equals the proximity index (Rip & Courtial, 1984) which is a popular similarity measure used in informetrics for, for example, co-authorship and co-citation visualization.

The histogram in Figure 6.11 shows that the vast majority of the dissimilarities obtained by the gravity model out the Thesaurus data are very small. In Section 6.5, we concluded that such data sets yield uninformative solutions. However, with an appropriate weighting scheme, an informative solution can be obtained. One solution to avoid this large number of rather uninteresting dissimilarities dominating the solution is to de-emphasize them and emphasize the more interesting larger dissimilarities. Therefore, high weights should be given to the larger dissimilarities. A method frequently used to do this is power weighting (see, e.g. Buja & Swayne, 2002; Borg & Groenen, 2005; Buja et al., 2008), in which the weights used are a power transformation of the dissimilarities

$$w_{ij} = \delta_{ij}^{\rho} \quad . \tag{6.25}$$

In case of sparse data, the power method is only used for the nonmissing dissimilarities, the missing dissimilarities remain to have a weight of zero.

Figure 6.12 shows the resulting 2D solutions and Shepard diagrams of the Large Scale MDS algorithm for different values of ρ. The five largest dissimilarities are shown by lines in the plots and a star in the Shepard diagrams. For $\rho = 0$ (that is, with constant weights), the Shepard diagram shows that the smaller dissimilarities are better represented than the larger ones. Since these small dissimilarities dominate the larger dissimilarities, the resulting solution with $\rho = 0$ is has the uninformative circular shape with some outliers. Also visible in the Shepard diagram is that the large dissimilarities are represented by distances that are too small. When looking at the solutions with higher ρ, their shapes become more distinct from a circle and their Shepard

Table 6.1: Frequently occurring terms in the thesaurus and terms in their neighborhood in the map

Frequently occuring term	Terms in neighborhood
Food	Lamb, Beef, Weight, Orange
Sex	Bed, Girl, Flesh
Water	Ocean, Sand, Burn, Bottles
Money	Birthday, Success, Industry
Beer	Opener, Cocktail, Belly
Work	Mechanics, Waitress, Dustman, Cleaning
Horse	Steeple, Chariot, Transport

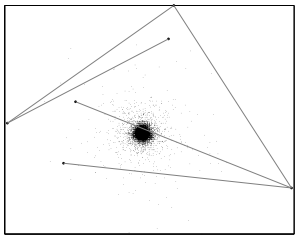
diagrams show that the larger dissimilarities are fitted better. The solution with $\rho = 5$ allows the larger dissimilarities to be fitted almost perfectly, although at the cost of a worse representation of the smaller dissimilarities. Note that this is fine as small dissimilarities are less important. Therefore, we set ρ to 5 in our application.

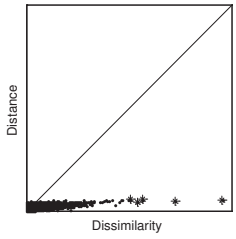
We made a map of the thesaurus data using the Large Scale MDS algorithm and the setup provided above. The Large Scale MDS algorithm was implemented in Matlab, although the most computations are done using linked Fortran code to make efficient use of the algorithm in Figure 6.6. The algorithm needed 389 iterations that took in total CPU 65 seconds to converge to a solution with a normalized stress of 0.00085. The resulting map is shown in Figures 6.12g and 6.13 and Figure 6.14 zooms in on the top right corner of this map. To avoid cluttering we only show labels of frequently occurring terms, where the font size is proportional to the frequency of the terms. To interpret the details of a map of this size a static map as is shown in Figure 6.13 is usually not sufficient. Therefore, interactive software is needed which gives the ability to zoom in on specific parts of the map.

The map can be interpreted in the following way. There are a number of very frequently occurring terms (food, sex, horse, work, beer, money, water) that all are located at the borders of the map, which all seem to attract some terms to them that have something to do with these terms. In Figure 6.14, we zoom in on the area near the frequently occurring term food. This figure shows clearly that most of the words in this area have something to do with food. Most of them are items that can be eaten, such as cheese, meat, orange, and egg. However, terms like weight, hot, tongue, and teeth are terms associated with food in other ways. These terms and some other frequently occurring terms are given in Table 6.1 together with terms found in their neighborhood.

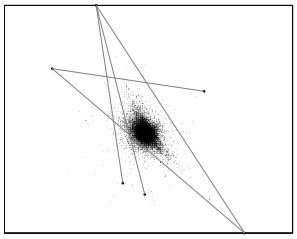
6.7 Conclusions

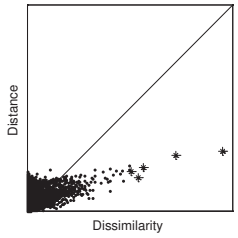
This chapter has introduced the Large Scale MDS algorithm that is able to perform multidimensional scaling (MDS) on large sparse data sets in linear time of the number of nonmissing dissimilarities. When the data are not sparse by nature, they can be made sparse by a design (such as the symmetric circulant design) or by randomly setting parts of the data as missing.



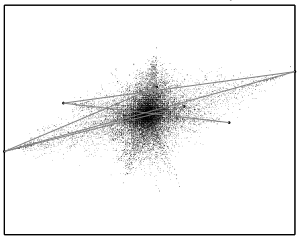


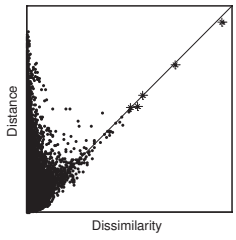
a. Visualization with $\rho = 0$ b. Shepard diagram with $\rho = 0$



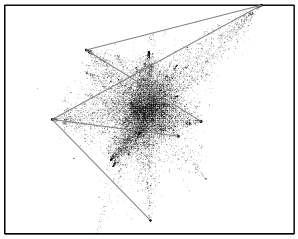


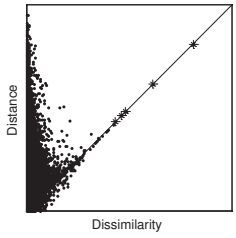
c. Visualization with $\rho = 1$ d. Shepard diagram with $\rho = 1$





e. Visualization with $\rho = 3$ f. Shepard diagram with $\rho = 3$





g. Visualization with $\rho = 5$ h. Shepard diagram with $\rho = 5$

Figure 6.12: Resulting visualizations and Shepard diagrams for four different power weighting schemes on the Thesaurus data set. Five largest distances are shown by lines in visualizations and stars in the Shepard diagrams.

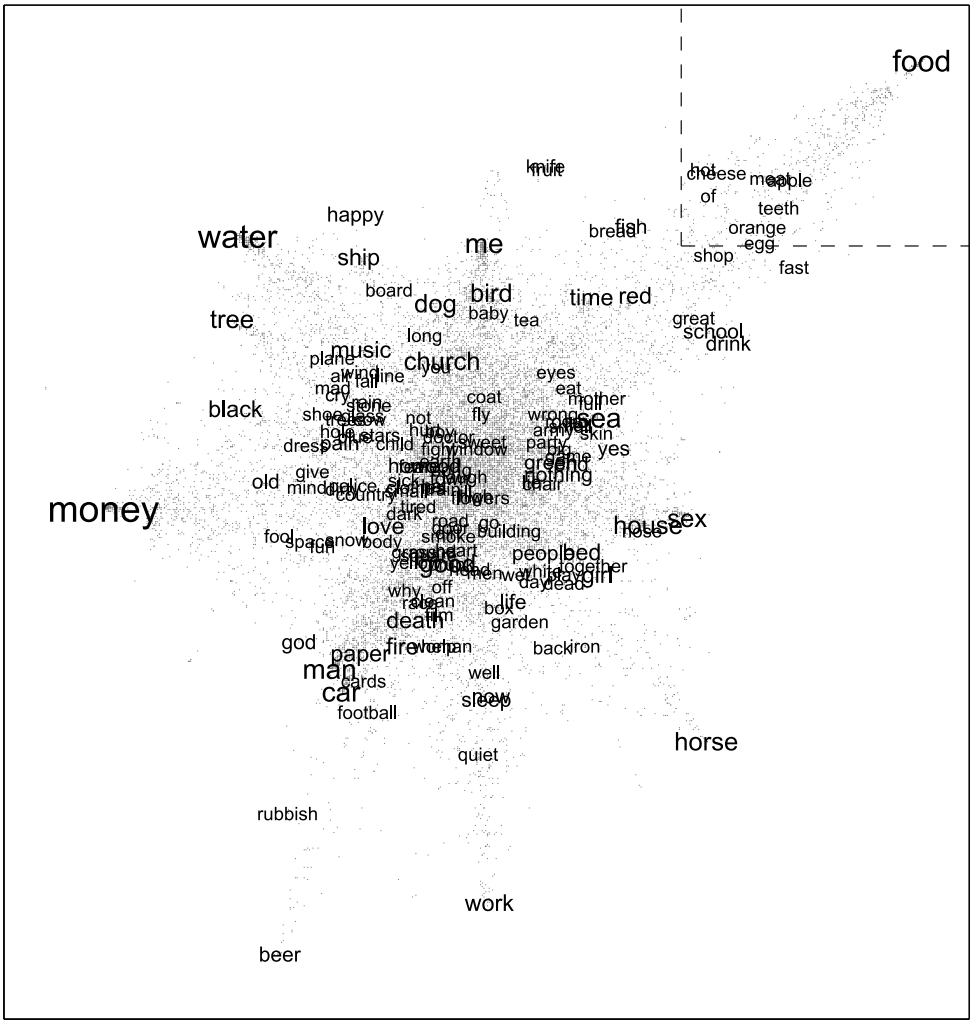


Figure 6.13: Map of the thesaurus data set.

Figure 6.14: Part of Map of the thesaurus data set near the frequently occurring term food.

Furthermore, we addressed the problems that occur when doing MDS on large sparse dissimilarity matrices. A consequence of the sparseness is that unconnected subsets of objects may be present in the dissimilarity matrix. If so, then separate MDS solutions should be obtained for each subset. A simple algorithm is presented that is able to find these subsets. Another problem is that in large data sets the most frequent dissimilarity value may dominate the solution. Since these are often not the most interesting dissimilarities, a proper dissimilarity measure and weighting scheme of the residuals must be found to solve this issue.

In one applications, we showed that the Large Scale MDS algorithm is able to unroll a Swiss roll in only a fraction of the time needed by the standard SMACOF algorithm. In a second application, we used Large Scale MDS to visualize a Thesaurus. We showed it is essential to choose a proper dissimilarity measure and weighting scheme for these data to yield a interpretable visualization.

In MDS, often transformations of the dissimilarities are jointly estimated with the coordinates. For example, in ordinal MDS an optimal ordinal transformation is sought that is monotone with the original dissimilarities. If the number of unique dissimilarity values is not too large, then the Large Scale MDS approach could be easily extended to deal with such transformations. However, if the number of different values becomes large, then the update step of the transformation in each iteration will become computationally too hard to be of practical use. We believe that for large scale MDS, it may be better not use such transformations.

A further extension of the Large Scale MDS algorithm might be to include so-called repulsive forces (L. Chen & Buja, 2009). Often, there are some subsets of objects that are only very loosely connected to other subsets, for example, by only two or one dissimilarities. If so, then the position of points with respect to each other is not uniquely determined. Two subsets only connected by a single dissimilarity will place the two connecting points at a distance that matches exactly the dissimilarity. However, each of these subsets can rotate freely around the connecting points without affecting the loss of the solution. In the case of two connecting dissimilarities, again the distances of the connecting points will match their dissimilarity exactly. In 2D, each subset can be mirrored in the line through the connecting points and in 3D the subsets can be freely rotated around this line. To solve such indeterminacies, repulsive forces for unconnected points can be included L. Chen and Buja (2009). However, their approach is not efficient as the computational efficiency of the large scale MDS algorithm is lost. A new algorithm should be developed that can handle repulsive forces but is still computationally efficient.

Chapter 7

Discussion

In this chapter, we will first give a summary of the main findings for each chapter. Then, we discuss some directions for future research based on the methods discussed in this thesis.

7.1 Summary and Discussion of the Main Findings

This thesis focused on two approaches to improve online shopping interfaces: product catalog maps and recommender systems. Several different topics were discussed in Chapter 2 up to 6. Below, we give the main findings for each chapter.

In Chapter 2, two shopping interfaces were developed based on a map based representation of a product catalog. In these maps, similar products in terms of their characteristics are generally located close to each other. Two visualization methods were used to develop these map based interfaces: multidimensional scaling (MDS) and nonlinear principal components analysis (NL-PCA). Also, we introduced a method to determine attribute weights based on overall product popularity. This method was used to improve the map of the MDS based interface. In an application on a real life MP3 player product catalog, we showed prototype interfaces for both methods and we showed that both methods lead to well interpretable maps.

In Chapter 3, another online shopping environment was developed combining product catalog maps made using MDS with product recommendation. This online shopping environment consists of two map based shopping interfaces, both using a very flexible dissimilarity measure that is used both as input for MDS and for recommendation. In the first interface, the graphical recommender system (GRS), an ideal product specification is plotted together with some recommendations similar to this ideal product in a map. The second interface, the graphical shopping interface (GSI), builds on the idea of recommendation by proposing: first, the system presents the user with a map of a small number of products of which the user can select the product she likes most; then, based on this product a new map with more similar products to the chosen product is created and the process can be repeated. We developed three different search approaches to be used in the GSI: the random, the clustering, and the hierarchical system. We showed a prototype interface both implementing the GRS and GSI on an MP3 player product catalog. This data set was also used to evaluate the map quality in the GRS. The quality of the maps measured

in Stress in the GRS was high. Also, we used these data to evaluate the three search algorithms for the GSI. The experiments showed that the random system should be preferred and this search approach worked well. In a usability study among 71 participants, it was shown that the major weakness of the GSI is its perceived complexity experienced by users. An important positive feature is that users found the GSI fun to use.

In Chapter 4, we introduced two methods to determine attribute importance. The assumption is that attributes that are considered to be important by users are those that can distinguish popular products from non popular products. The first method is based on Poisson regression and the second method is based on a novel boosting algorithm minimizing Poisson deviance. Both methods were used to determine attribute weights for a dissimilarity measure which in turn is used for product recommendation. Both methods were evaluated using a clickstream from a price comparison website and a user experiment. Four real life product catalogs were used in the clickstream evaluation. For each of the four product catalogs at least one of the weighting methods significantly outperformed the use of an unweighted dissimilarity measure. In the user experiment, the unweighted dissimilarity measure outperformed both weighting approaches. One possible reason for these contradicting findings might be that only a small number of relatively important attributes was considered whereas the main strength of the weighting methods is to de-emphasize unimportant attributes.

In Chapter 5, a novel hybrid recommender system, called latent class regression recommender system (LCR-RS), was introduced, that combines collaborative filtering and content-based recommendation. This system is an extension of the probabilistic latent semantic analysis model for collaborative filtering (pLSA-CF). This method was combined with ideas from cluster-wise linear regression such that item characteristics can be used in the algorithm. This approach has two major advantages: the system can explain its recommendations and new items can be recommended by the system. On a movie data set that is a combination of the Netflix and IMDb databases, LCR-RS was compared to some other methods including the original pLSA-CF method. Overall, LCR-RS performed slightly worse than pLSA-CF and other collaborative filtering algorithms, but it was much better in predicting ratings for relatively new items. Finally, we showed how LCR-RS can be used to explain its recommendations.

In Chapter 6, an MDS algorithm was developed that can be applied to large sparse dissimilarity matrices. Instead of the quadratic complexity in the number of items of ordinary MDS methods, this new method has a complexity that is linear in the number of nonmissing dissimilarities. This is much faster when having a large number of missing dissimilarities. We showed how a large nonsparse dissimilarity matrix can be made sparse using a symmetric circulant design. Also, two other issues that are important when dealing with large sparse dissimilarity matrices were addressed. First, we addressed the issue that sparse dissimilarity matrices may consist of a number of unconnected subsets and introduced a simple algorithm to retrieve these subsets. Secondly, we paid attention to the issue that the most frequent dissimilarity value tends to dominate large scale MDS solutions. Using an empirical Thesaurus data set containing similarities between 23,000 words, we showed that it is essential to choose a proper dissimilarity measure and weighting scheme to overcome this. In a second example, we used the large scale MDS algorithm to unroll the points in a Swiss roll manifold by using only the smallest dissimi-

larities. The large scale MDS algorithm lead to a high quality solution only taking a fraction of the time original MDS needed.

7.2 Directions for Future Research

During the research projects resulting in thesis several ideas came up for possible improvements in the field of online shopping interfaces.

The key factor in developing a successful online shopping interface is a high usability. Although recommendation or visualization techniques may be very advanced, when users do not understand the system and it is too complex to use, these advanced techniques are worthless. Therefore, both product catalog map based interfaces introduced in Chapter 2 should be tested in such a usability study. Such a study should discover the strengths and weaknesses of each system so that the most promising could be used to develop further and whether there are relatively small adaptations that can further improve the system. In such a usability study, our approach could also be compared to other map based user interfaces that have recently been proposed in related fields, such as web search result visualization (see e.g., Turetken & Sharda, 2004, Chung et al., 2006, and Chung, 2008), online news visualization (see e.g., Ong et al., 2005), image library visualization (see e.g., Pečenović et al., 2000), and music library visualization (see e.g., Van Gulik et al., 2004 and Donaldson, 2007). These user interfaces are based on different visualization techniques such as self-organizing maps and treemaps. A good comparison is only possible after studying how easy it is to apply these approaches to product catalog visualization.

For the graphical shopping interface (GSI), we performed such a usability study which showed that users found this system rather complex to use. Although we think that this will become less of a problem when users have time to get acquainted to the system, users will often not take this time in a real online shopping environment. We expect that users like systems based on a large static map better, since their complexity is lower. However, this needs to be tested.

Using weights that match the notion of users might dramatically improve usability of product catalog maps, since the products are then more frequently located on the position expected by the user. We introduced two methods to determine such weights based on product popularity and evaluated these in a recommender system context. Since we obtained mixed results in this study, more research is needed to find a method that consistently leads to better weights. Also, it would be nice when the weighting methods are tested in a product catalog map context.

User satisfaction is also very important in recommender system applications. The recommender system providing the best recommendations is not per se the recommender system having the highest user satisfaction (Herlocker et al., 2000). For example, users tend to trust the recommendations of a system more when a meaningful explanation is provided (Sinha & Swearingen, 2002). The latent class regression recommender system (LCR-RS) we proposed in Chapter 5 is able to give such recommendations, without losing much recommendation quality. In contrast to other recommender systems that are able to give explanations (see Tintarev & Masthoff, 2007, for an overview), LCR-RS is a hybrid recommender system both using user community and item characteristics data. However, we did not test the quality of these explanations given by LCR-RS in a user experiment. For such an experiment, a real life recommender system is needed with

an active pool of users, since ratings are needed to train the system and the users who gave the ratings need to evaluate whether the explanations of the system match their taste.

Besides more user testing, several other steps need to be taken before real life implementation of the systems discussed in this thesis will be possible. For the map based interfaces, there are, a number of technical aspects that need to be taken care of. The Java development environment used is this thesis may not be the best alternative to implement this system in practice, due to the slow load times of Java applets and the fact that a plug-in is required. Hence, a client side platform that lacks these shortcomings, such as Javascript of Flash, may be more suitable for implementing a production quality user interface. Furthermore, making the system applicable to a multitude of different product catalogs would require the implementation of a layer that ensures that the map-construction logic receives data it can work with, i.e., data of sufficient quality and in the right format. Such a layer is not trivial, since it should implement data cleaning, and it should interface with various types of database systems, such as relational databases and XML databases.

An important choice that needs to be made for practical implementation is in what form the map based interface will be integrated in the website, since this poses different requirements to the implemented system. When the system will be implemented as a gadget besides a traditional interface, it should primarily be fun to use. Then, dynamic visualizations and advanced graphics are important. On the other hand, when a map based system is implemented to replace the original interface, it is important to reduce the complexity of the interface to make it easy to understand and use. Furthermore, the interface should then also implement more traditional mechanisms such as constraint based searching. In fact, the interface has to be fully integrated in the website. This integration would give rise to a number of (research) problems, e.g.: what is the best way to integrate a recommended system with a map-based system; and: how should we represent special offers in such a context?

Also for the LCR-RS some further steps should to be taken before practical implementation is possible. The current algorithm is too slow and requires too much from the computer's memory to be able to handle the large databases used in practice. The main bottleneck in this algorithm is the weighted least squares solution that must be obtained in each iteration for each latent class. A fast algorithm to estimate this solution would make the system substantially faster. Furthermore, an approach should be found to retrieve textual explanations from the regression coefficients LCR-RS returns. Otherwise, the explanations will not be understood by the users and, hence, the explanation system will be worthless.

Large scale MDS is a technique that may become very useful when the idea of product catalog maps is applied to larger product catalogs. It would then be very interesting to use collaborative data, such as co-purchases or rating correlation. Since not all products are purchased or rated together this will lead to a large, but sparse (dis)similarity matrix and, thus, a perfect problem to perform large scale MDS on. Applications areas interesting to use this approach for are, for example, books, music, and movies. An extension to large scale MDS that may improve these kind of maps is the inclusion of repulsive forces (L. Chen & Buja, 2009) to better locate very loosely connected subsets with respect to each other. In the before mentioned application domains one could think of different genres being these subsets.

Above, we gave several directions in which the methodology discussed in this thesis may be further improved. However, we think that the methods discussed can form a basis for new promising online shopping interfaces both in research as in practice.

Nederlandse Samenvatting (Summary in Dutch)

Gedurende de laatste twee decennia is het internet snel een belangrijk medium geworden om informatie te vinden, sociale contacten te onderhouden en om online te winkelen. Dit laatste heeft een aantal voordelen ten opzichte van traditioneel winkelen. Ten eerste zijn producten vaak goedkoper via internet en bieden internetwinkels een breder assortiment. Ook kan de consument op het internet winkelen op het tijdstip dat hij zelf wil en dat zonder het verlaten van zijn eigen stoel.

Daarentegen hebben de huidige internetwinkels nog twee belangrijke nadelen ten opzichte van 'echte' winkels. Het eerste nadeel is dat producten online vaak veel moeilijker te vinden zijn. Meestal bieden internetwinkels twee manieren om producten te zoeken: via het ingeven van een zoekterm of door het selecteren van een bepaalde categorie producten. Beide methoden geven vaak lange lijsten met producten terug waar de gebruiker dan doorheen moet scrollen. Het tweede nadeel is dat er geen verkoper is om de consument te helpen met zijn aankoopbeslissingen. Om dit probleem gedeeltelijk te overkomen zijn aanbevelingssystemen ontwikkeld die de consument producten aanbevelen die hij misschien interessant vindt en daarmee de verkoper gedeeltelijk vervangen. In dit proefschrift zullen we op beide van deze nadelen ingaan.

Dit proefschrift uit drie delen. In deel I, dat bestaat uit twee hoofdstukken, ligt de nadruk op het gebruik van productcataloguskaarten om online winkelomgevingen te verbeteren. Deel II richt zich op aanbevelingssystemen en bestaat ook uit twee hoofdstukken. Ten slotte bevat deel III één hoofdstuk over een algoritme voor het visualiseren van grote datasets. Drie van deze hoofdstukken zijn gepubliceerd of geaccepteerd voor publicatie in de wetenschappelijke literatuur. De twee andere hoofdstukken staan klaar om opgestuurd te worden. Hieronder geven we een korte omschrijving van elk hoofdstuk.

In hoofdstuk 2 ontwikkelen we twee online winkelomgevingen die gebaseerd zijn op een tweedimensionale kaart van een productcatalogus. Op deze kaarten staan, in het algemeen, producten die op elkaar lijken op basis van hun eigenschappen dicht bij elkaar. We gebruiken twee verschillende visualisatiemethoden om deze winkelongevingen te maken. De eerste is gebaseerd op meerdimensionale schaling (Borg & Groenen, 2005), terwijl de tweede gebruik maakt van niet-lineaire principale componenten analyse (Linting et al., 2007). Daarnaast introduceren we een methode om te bepalen welke eigenschappen van een product belangrijk gevonden worden door gebruikers. Deze methode kan worden gebruikt om gewichten te berekenen die gebruikt kunnen worden voor het maken van de kaarten. We tonen toepassingen van

beide winkelomgevingen op een productcatalogus van MP3-spelers. Hoofdstuk 2 is gebaseerd op Kagie et al. (2010) en is geaccepteerd voor publicatie in het *Recommender Systems Handbook*.

In hoofdstuk 3 ontwikkelen we een online winkelomgeving waarin we het idee van de productcatologuskaarten uit hoofdstuk 2 combineren met productaanbevelingen. Daarvoor introduceren we eerst een flexibele ongelijkheidsmaat die zowel voor meerdimensionale schaling als voor het aanbevelen van producten gebruikt kan worden. Daarna introduceren we twee aanbevelingssystemen die gebruik maken van tweedimensionale kaarten. Het *Graphical Recommender System* visualiseert een ideaal-productspecificatie samen met een aantal aanbevelingen van producten dat hierop lijkt. De *Graphical Shopping Interface* (GSI) combineert het gebruik van kaarten met het *recommending by proposing* principe (Shimazu, 2002), waarin gebruikers in een beperkt aantal stappen naar het product van hun gading navigeren. We tonen prototypen van deze twee systemen die gebruik maken van een productcatalogus van MP3-spelers. We testen zowel de kwaliteit van de kaarten als het achterliggende zoekalgoritme van de GSI in een simulatiestudie. Ook testen we de kwaliteit van de GSI in een gebruikersstudie. Hoofdstuk 3 is gebaseerd op Kagie et al. (2008b) en is gepubliceerd in *Decision Support Systems*.

In hoofdstuk 4 gaan we dieper in op het bepalen van belangrijke producteigenschappen dat ook al behandeld is in hoofdstuk 2. Twee modellen worden besproken die bepalen wat belangrijke producteigenschappen zijn op basis van een maat voor productpopulariteit, zoals het aantal verkopen. De aanname is dat producteigenschappen die in staat zijn om populaire producten te onderscheiden van impopulaire producten, eigenschappen zijn die door gebruikers belangrijk worden gevonden. De eerste methode is gebaseerd op Poisson regressie en de tweede methode is gebaseerd op een nieuw boostingalgoritme dat een Poisson foutfunctie minimaliseert. Beide modellen worden geëvalueerd met gebruik van twee nieuwe evaluatiemethoden, waarin de wegingsmodellen worden vergeleken met een ongewogen ongelijkheidsmaat. De eerste methode is gebaseerd op een clickstream analyse. We gebruiken vier productcatalogi (MP3-spelers, digitale camera's, notebooks en magnetrons) in deze evaluatie. De tweede methode is gebaseerd op een gebruikersexperiment waarvoor we de magnetroncatalogus gebruiken. Hoofdstuk 4 is gedeeltelijk gebaseerd op Kagie et al. (2008a) dat gepubliceerd is in de *Proceedings of the 2nd ACM Conference on Recommender Systems*.

In hoofdstuk 5 introduceren we een nieuw hybride aanbevelingsalgoritme dat *collaborative filtering* en content-gebaseerde aanbevelen combineert. De basis van dit systeem is *probabilistic latent semantic analysis for collaborative filtering* (pLSA-CF) ontwikkeld door Hofmann (2004). Een belangrijk nadeel van deze methode is dat objecteigenschappen niet door deze methode gebruikt worden. Wij combineren pLSA-CF met ideeën uit *clusterwise linear regression* (DeSarbo & Cron, 1988), zodat we objecteigenschappen in het model kunnen integreren. Dit heeft twee belangrijke voordelen: ten eerste kunnen we de regressiecoëfficienten die voor de objecteigenschappen geschat zijn gebruiken om de gebruiker uit te leggen waarom een bepaalde aanbeveling gegeven is; ten tweede kunnen we met gebruik van deze nieuwe methode ook objecten aanbevelen die (relatief) nieuw zijn in het systeem. De prestaties van onze methode op een filmdataset zijn bijna hetzelfde als de originele pLSA-CF methode. Echter, onze methode presteert wel veel beter op relatief nieuwe objecten. Ook laten we op deze dataset zien hoe onze methode gebruikt

kan worden om aanbevelingen uit te leggen. Hoofdstuk 5 is gebaseerd op Kagie, Van der Loos, and Van Wezel (2009) en is gepubliceerd in *AI Communications*.

In hoofdstuk 6 ontwikkelen we een algoritme voor meerdimensionale schaling (MDS) op grote ijle (sparse) ongelijkheidsmatrices. Normale MDS-algoritmes hebben een kwadratische complexiteit in het aantal objecten, terwijl wij een methode voorstellen die lineair is in het aantal niet-ontbrekende ongelijkheden wat veel sneller is wanneer de ongelijkheidsmatrix heel ijl is. Ook laten we zien hoe een ongelijkheidsmatrix ijl gemaakt kan worden met behulp van een specifiek design, wanneer deze dat niet van nature is. Daarnaast behandelen we twee andere zaken die belangrijk zijn wanneer men gebruik maakt van grote ijle ongelijkheidsmatrices. Ten eerste kunnen ijle data bestaan uit een aantal onverbonden subsets, zodat een aparte MDS-oplossing verkregen moet worden voor elk van deze subsets. Ten tweede blijken MDS-oplossingen voor grote datasets gedomineerd te worden door de meest voorkomende ongelijkheidswaarde. Daarom is het essentieel om de juiste combinatie van ongelijkheidsmaat en wegingsschema te gebruiken. We laten zien hoe dat gedaan kan worden. We gebruiken ons MDS-algoritme voor grote datasets voor twee toepassingen. Ten eerste gebruiken we het om een *Swiss roll* te ontrollen door alleen de kleinste ongelijkheden te gebruiken. Ons algoritme geeft een oplossing van hoge kwaliteit in slechts een fractie van de tijd die het originele MDS-algoritme nodig heeft. Ten tweede passen we ons algoritme toe op een Thesaurus dataset om de gelijkenissen tussen 23.000 worden te tonen. Patrick J.F. Groenen en Michael W. Trosset zijn co-auteurs van hoofdstuk 6.

Bibliography

Adomavicius, G., & Tuzhilin, A. (2005). Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749.

Aikake, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*(6), 716–723.

Alhoniemi, E., Himberg, J., Parviainen, J., & Vesanto, J. (1999). SOM toolbox 2.0 for Matlab 5 [Computer software manual]. Helsinki, Finland: Helsinki University of Technology. (Available from `http://www.cis.hut.fi/projects/somtoolbox/`)

Ansari, A., Essegaier, S., & Kohli, R. (2000). Internet recommendation systems. *Journal of Marketing Research*, *37*(3), 363–375.

Arslan, B., Ricci, F., Mirzadeh, N., & Venturini, A. (2002). A dynamic approach to feature weighting. *Management Information Systems*, *6*, 999–1008.

Ayanso, A., Goes, P. B., & Mehta, K. (2007). A practical approach for efficiently answering top-k relational queries. *Decision Support Systems*, *44*, 326–349.

Balabanović, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Communications of the ACM*, *40*(3), 66–72.

Basu, C., Hirsh, H., & Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the 15th/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence* (pp. 714–720). Menlo Park, CA, USA: AAAI Press.

Bederson, B. B., Schneiderman, B., & Wattenberg, M. (2002). Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. *ACM Transactions on Graphics*, *21*(4), 833–854.

Bettman, J. R., Luce, M. F., & Payne, J. W. (1998). Constructive consumer choice processes. *Journal of Consumer Research*, *25*(3), 187–217.

Billsus, D., & Pazzani, M. J. (2000). User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, *10*, 147–180.

Bishop, C. M. (2006). *Pattern recognition and machine learning.* New York, NY, USA: Springer.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine*

Learning Research, *3*, 993–1022.

Borg, I., & Groenen, P. J. F. (2005). *Modern multidimensional scaling* (2nd ed.). New York: Springer.

Brandes, U., & Pich, C. (2007). Eigensolver methods for progressive multidimensional scaling of large data. *Lecture Notes on Computer Science*, *4372*, 42–53.

Branting, L. K. (2004). Learning feature weights from customer return-set selections. *Knowledge and Information Systems*, *6*, 188–202.

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence* (pp. 43–52). San Francisco, CA, USA: Morgan Kaufmann.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L. (1998). Arcing classifiers. *Annals of Statistics*, *26*(2), 801–824.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Breiman, L., Friedman, J. H., Olshen, R., & Stone, C. (1983). *Classification and regression trees*. New York: Chapman & Hall.

Bruls, M., Huizing, K., & Van Wijk, J. J. (2000). Squarified treemaps. In *Proceedings of Joint Eurographics and IEEE TCVG Symposium on Visualization* (pp. 33–42).

Buja, A., Logan, B. F., Reeds, J. A., & Shepp, L. A. (1994). Inequalities and positive-definite functions arising from a problem in multidimensional scaling. *Annals of Statistics*, *22*(1), 406–438.

Buja, A., & Swayne, D. F. (2002). Visualization methodology for multidimensional scaling. *Journal of Classification*, *19*, 7–43.

Buja, A., Swayne, D. F., Littman, M. L., Dean, N., Hofmann, H., & Chen, L. (2008). Data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics*, *17*(2), 444–472.

Burke, R. (2000). Knowledge based recommender systems. In J. E. Daily, A. Kent, & H. Lancour (Eds.), *Encyclopedia of library and information science* (Vols. 69, Supplement 32). New York, NY, USA: Marcel Dekker.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, *12*, 331–370.

Camargo, S. J., Robertson, A. W., Gaffney, S. J., Smyth, P., & Ghil, M. (2007). Cluster analysis of typhoon tracks. part I. general properties. *Journal of Climate*, *20*(14), 3635–3653.

Chalmers, M. (1996). A linear iteration time layout algorithm for visualising high-dimensional data. In R. Yagel & G. M. Nielson (Eds.), *Proceedings of the 7th Conference on Visualization* (pp. 127–132). Los Alamitos, CA, USA: IEEE Computer Society Press.

Chaudhuri, S., & Gravano, L. (1999). Evaluating top-k selection queries. In M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, & M. L. Brodie (Eds.), *Proceedings of the 25th VLDB Conference* (pp. 397–410). San Francisco, CA, USA: Morgan Kaufmann.

Chen, H., Houston, A. L., Sewell, R. R., & Schatz, B. R. (1998). Internet browsing and searching:

User evaluations of category map and concept space techniques. *Journal of the American Society for Information Science*, *49*(7), 582–603.

Chen, L., & Buja, A. (2009). Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, *104*(485), 209–220.

Chung, W. (2008). Web searching in a multilingual world. *Communications of the ACM*, *51*(5), 32–40.

Chung, W., Bonillas, A., Lain, G., Xi, W., & Chen, H. (2006). Supporting non-English Web searching: An experiment on the Spanish business and the Arabic medical intelligence portals. *Decision Support Systems*, *42*, 1697–1714.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., & Sartin, M. (1999). Combining content-based and collaborative filtering in an online newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation.*

Comon, P. (1994). Independent component analysis, a new concept? *Signal Processing*, *36*(3), 287–314.

Condliff, M., Lewis, D., Madigan, D., & Posse, C. (1999). Bayesian mixed-effects models for recommender systems. In *Proceedings of ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation.*

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). Cambridge, MA, USA: MIT Press.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27.

Coyle, L., & Cunningham, P. (2004). Improving recommendation rankings by learning personal feature weights. *Lecture Notes in Computer Science*, *3155*, 560–572.

Coyle, L., Doyle, D., & Cunningham, P. (2004). Representing similarity for CBR in XML. *Lecture Notes in Computer Science*, *3155*, 119–127.

De Leeuw, J. (1977a). Applications of convex analysis to multidimensional scaling. In J. R. Barra, F. Brodeau, G. Romier, & B. van Cutsem (Eds.), *Recent developments in statistics* (pp. 133–145). Amsterdam, Netherlands: North-Holland Publishing Company.

De Leeuw, J. (1977b). Correctness of Kruskal's algorithms for monotone regression with ties. *Psychometrika*, *42*(1), 141–144.

De Leeuw, J. (1988). Convergence of the majorization method for multidimensional scaling. *Journal of Classification*, *5*, 163–180.

De Leeuw, J., & Heiser, W. J. (1977). Convergence of correction-matrix algorithms for multidimensional scaling. In J. C. Lingoes, E. E. Roskam, & I. Borg (Eds.), *Geometric representations of relational data* (pp. 735–752). Ann Arbor, MI, USA: Mathesis.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the E-M-algorithm. *Journal of the Royal Statistical Society Series B*, *39*, 1–38.

DeSarbo, W. S., & Cron, W. L. (1988). A maximum likelihood methodology for clusterwise linear regression. *Journal of Classification*, *5*(2), 249–282.

DeSarbo, W. S., Di Benedetto, C. A., Song, M., & Sinha, I. (2005). Revisiting the miles and snow strategic framework: Uncovering interrelationships between strategic types, capabilities, environmental uncertainty, and firm performance. *Strategic Management Journal*, *26*(1), 47–74.

DeSarbo, W. S., Wedel, M., Vriens, M., & Ramaswamy, V. (1992). Latent metric conjoint analysis. *Marketing Letters*, *3*(3), 273–288.

Donaldson, J. (2007). Music recommendation mapping and interface based on structural network entropy. In V. Oria, A. Elmagarmid, F. Lochovsky, & Y. Saygin (Eds.), *Proceedings of the 23rd International Conference on Data Engineering Workshops* (pp. 811–817). Los Alamitos, CA, USA: IEEE Computer Society Press.

Eades, P. (1984). A heuristic for graph drawing. *Congressus Numerantium*, *42*.

Faloutsos, C., & Lin, K. (1995). FastMap: A fast algorithm for indexing, datamining and visualization. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 163–174). New York, NY, USA: ACM Press.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In L. Saitta (Ed.), *Proceedings of the 13th International Conference on Machine Learning* (pp. 148–156). San Francisco, CA, USA: Morgan Kaufmann.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Friedman, J. H. (2001). Greedy function aproximation: A gradient boosting machine. *Annals of Statistics*, *29*(5), 1189–1232.

Friedman, J. H., & Tukey, J. W. (1974). A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, *22*, 881–890.

Gifi, A. (1990). *Nonlinear multivariate analysis*. Chichester, UK: Wiley.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61–70.

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, *4*, 133–151.

Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B. M., Herlocker, J. L., et al. (1999). Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the 16th National/ 11th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence* (pp. 439–446). Menlo Park, CA, USA: AAAI Press.

Gower, J. C. (1966). Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, *53*, 325–338.

Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, *27*, 857–874.

Gower, J. C., & Groenen, P. J. F. (1991). Applications of the Modified Leverrier-Faddeev algorithm for the construction of explicit matrix spectral decompositions and inverses. *Utilitas Mathematica*, *40*, 51–64.

Gower, J. C., & Hand, D. J. (1996). *Biplots*. London, UK: Chapman & Hall.

Gower, J. C., & Legendre, P. (1986). Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, *3*, 5–48.

Green, B. F. (1952). The orthogonal approximation of an oblique structure in factor analysis. *Psychometrika*, *17*, 429–440.

Groenen, P. J. F., & Heiser, W. J. (1996). The tunneling method for global optimization in multidimensional scaling. *Psychometrika*, *61*, 529–550.

Groenen, P. J. F., Heiser, W. J., & Meulman, J. J. (1999). Global optimization in a least-squares multidimensional scaling by distance smoothing. *Journal of Classification*, *16*, 225–254.

Hartigan, J. A., & Wong, M. A. (1979). A k-means clustering algorithm. *Applied Statistics*, *28*, 100–108.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning*. New York, NY, USA: Springer.

Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. T. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 230–237). New York, NY, USA: ACM Press.

Herlocker, J. L., Konstan, J. A., & Riedl, J. T. (2000). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work* (pp. 241–250). New York, NY, USA: ACM Press.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, *22*, 5–53.

Hicklin, J. (2007). Treemap for Matlab [Computer software manual]. Natick, MA, USA: The Mathworks. (Available from `http://www.mathworks.com/matlabcentral/fileexchange/17192`)

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence* (pp. 289–296). San Francisco, CA, USA: Morgan Kaufmann.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, *42*, 177–196.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, *22*, 89–115.

Honaker, J., King, G., & Blackwell, M. (2008). Amelia II: A program for missing data [Computer software manual]. Cambridge, MA, USA: Harvard University. (R package version

1.1-27, available from `http://gking.harvard.edu/amelia`)

Ibrahim, J. G., Chen, M.-H., Lipsitz, S. R., & Herring, A. H. (2005). Missing-data methods for generalized linear models: A comparative review. *Journal of the American Statistical Association*, *100*(469), 332–346.

Jameson, A., Schäfer, R., Simons, J., & Weis, T. (1995). Adaptive provision of evaluation-oriented information: Tasks and techniques. In *Proceedings of the 15th international joint conference on artificial intelligence* (pp. 1886–1893). San Mateo, CA, USA: Morgan Kaufmann.

Jolliffe, I. (2002). *Principal components analysis* (2nd ed.). New York, NY, USA: Springer.

Jourdan, F., & Melançon, G. (2004). Multiscale hybrid MDS. In *Proceedings of the 8th International Conference on Information Visualisation* (pp. 388–393). Los Alamitos, CA: IEEE Computer Society Press.

Kagie, M., Van der Loos, M., & Van Wezel, M. (2009). Including item characteristics in the probabilistic latent semantic analysis model for collaborative filtering. *AI Communications*, *22*(4), 249–265.

Kagie, M., Van Wezel, M., & Groenen, P. J. F. (2007). Online shopping using a two dimensional product map. *Lecture Notes in Computer Science*, *4655*, 89–98.

Kagie, M., Van Wezel, M., & Groenen, P. J. F. (2008a). Choosing attribute weights for item dissimilarity using clickstream data with an application to a product catalog map. In *Proceedings of the 2nd ACM Conference on Recommender Systems* (pp. 195–202). New York, NY, USA: ACM Press.

Kagie, M., Van Wezel, M., & Groenen, P. J. F. (2008b). A graphical shopping interface based on product attributes. *Decision Support Systems*, *46*(1), 265–276.

Kagie, M., Van Wezel, M., & Groenen, P. J. F. (2008c). An online shopping interface based on a joint product and attribute category map. In *Proceedings of IUI Workshop on Recommendation and Collaboration ReColl 2008.*

Kagie, M., Van Wezel, M., & Groenen, P. J. F. (2009). *Determination of attribute weights for recommender systems based on product popularity* (Tech. Rep. No. ERS-2009-022). Rotterdam, The Netherlands: Erasmus University Rotterdam.

Kagie, M., Van Wezel, M., & Groenen, P. J. F. (2010). Map based visualization of product catalog maps. In P. B. Kantor, F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender systems handbook.* Springer.

Kearsley, A. J., Tapia, R. A., & Trosset, M. W. (1998). The solution of metric STRESS and SSTRESS problems in multidimensional scaling using Newton's method. *Computational Statistics*, *13*(3), 369–396.

Keller, I. (2000). MDS-i for 1 to 1 e-commerce: A position paper. In *Proceedings of the CHI 2000 Workshop on 1-to-1 E-commerce.*

King, G., Honaker, J., Joseph, A., & Scheve, K. (2001). Analyzing incomplete political science data: An alternative algorithm for multiple imputation. *American Political Science Review*,

95(1), 49–69.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464–1480.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, *21*, 1–6.

Kohonen, T. (2001). *Self-organizing maps* (3rd ed.). New York, NY, USA: Springer.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer*, *42*(8), 30–37.

Kriegler, M., & Zimprich, D. (2006). Predictors of cognitive complaints in older adults: A mixture regression approach. *European Journal of Ageing*, *2*(1), 13–23.

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, *29*(1), 1–27.

Larcker, D. F., & Richardson, S. A. (2004). Fees paid to audit firms, accrual choices, and corporate governance. *Journal of Accounting Research*, *42*(3), 625–658.

Lee, S., & Choi, S. (2009). Landmark MDS ensemble. *Pattern Recognition*, *42*, 2045–2053.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, *7*(1), 76–80.

Linting, M., Meulman, J. J., Groenen, P. J. F., & Van der Kooij, A. J. (2007). Nonlinear principal components analysis: Introduction and application. *Psychological Methods*, *12*(3), 336–358.

Lorenzi, F., & Ricci, F. (2005). Case-based recommender systems: A unifying view. *Lecture Notes in Computer Science*, *3169*, 89–113.

McCullagh, P., & Nelder, J. A. (1989). *Generalized linear models* (2nd ed.). Boca Raton, FL, USA: Chapman & Hall.

McGinty, L., & Smyth, B. (2002). Comparison-based recommendation. *Lecture Notes in Computer Science*, *2416*, 731–737.

McSherry, D. (2002). A generalised approach to similarity-based retrieval in recommender systems. *Artificial Intelligence Review*, *18*, 309–341.

McSherry, D. (2005). Explanation in recommender systems. *Artificial Intelligence Review*, *24*, 179–197.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the 18th National Conference on Artificial Intelligence* (pp. 187–192). Menlo Park, CA, USA: AAAI Press.

Meulman, J. J., & Heiser, W. J. (2007). SPSS categories 15 [Computer software manual]. Chicago, IL, USA: SPSS Inc..

Michailidis, G., & De Leeuw, J. (1998). The Gifi system of descriptive multivariate analysis. *Statistical Science*, *13*(4), 307–336.

Morrison, A., Ross, G., & Chalmers, M. (2003). Fast multidimensional scaling through sampling, springs and interpolation. *Information Visualization*, *2*(1), 68–77.

Nelder, J. A., & Wedderburn, R. W. M. (1972). Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, *135*(3), 370–384.

Ong, T.-H., Chen, H., Sung, W., & Zhu, B. (2005). Newsmap: A knowledge map for online news. *Decision Support Systems*, *39*, 583–597.

O'Sullivan, D., Smyth, B., & Wilson, D. C. (2005). Understanding case-based recommendation: A similarity knowledge perspective. *International Journal on Artificial Intelligence Tools*, *14*(1–2), 215–232.

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, *13*, 393–408.

Pazzani, M. J., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, *27*, 313–331.

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation system. *Lecture Notes in Computer Science*, *4321*, 325-341.

Pennings, J. M., & Garcia, P. (2004). Hedging behavior in small and medium-sized enterprises: The role of unobserved heterogeneity. *Journal of Banking and Finance*, *28*(5), 951–978.

Pečenović, Z., Do, M. N., Vetterli, M., & Pu, P. H. Z. (2000). Integrated browsing and searching of large image collections. *Lecture Notes in Computer Science*, *1929*, 173–206.

Platt, J. C. (2005). FastMap, MetricMap, and landmark MDS are all Nyström algorithms. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics* (pp. 261–268).

Popescul, A., Ungar, L. H., Pennock, D. M., & Lawrence, S. (2001). Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence* (pp. 437–444). San Francisco, CA, USA: Morgan Kaufmann.

Prasad, B. (2003). Intelligent techniques for e-commerce. *Journal of Electronic Commerce Research*, *4*(2), 65–71.

Pu, P. H. Z., & Kumar, P. (2004). Evaluating example-based search tools. In *Proceedings of the 5th ACM Conference on Electronic Commerce* (pp. 208–217). New York, NY, USA: ACM Press.

Ramaswamy, V., DeSarbo, W. S., Reibstein, D. J., & Robinson, W. T. (1993). An emperical pooling approach for estimating marketing mix elasticities with PIMS data. *Marketing Science*, *12*(1), 103–124.

Reilly, J., McCarthy, K., McGinty, L., & Smyth, B. (2005). Tweaking critiquing. *Knowledge-Based Systems*, *18*(4–5), 143–151.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. T. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work* (pp. 175–186). New York, NY, USA: ACM Press.

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, *40*(3), 56–58.

Ricci, F., & Del Missier, F. (2004). Supporting travel decision making throug personalized

recommendation. In C.-M. Karat, J. O. Blom, & J. Karat (Eds.), *Designing personalized user experiences in ecommerce* (pp. 231–251). Netherlands: Kluwer Academic Press.

Ricci, F., Wöber, K., & Zins, A. (2005). Recommendation by collaborative browsing. In A. J. Frew (Ed.), *Information and communication technologies in tourism 2005* (pp. 172–182). Vienna, Austria: Springer.

Rip, A., & Courtial, J.-P. (1984). Co-word maps of biotechnology: An example of cognitive scientometrics. *Scientometrics*, *6*(6), 381–400.

Ripley, D. B. (1996). *Pattern recognition and neural networks*. Cambridge, UK: Cambridge University Press.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. New York, NY, USA: Wiley.

Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, *18*(5), 401–409.

Sandvig, J., & Burke, R. (2005). *Aacorn: A CBR recommender for academic advising* (Tech. Rep. No. TR05-15). Chicago, IL, USA: DePaul University.

Sarwar, B. M., Karypis, G., Konstan, J. A., & Riedl, J. T. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (pp. 285–295). New York, NY, USA: ACM Press.

Schafer, J. B., Konstan, J. A., & Riedl, J. T. (2001). E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, *5*, 115–153.

Schafer, J. L., & Olsen, M. K. (1998). Multiple imputation for multivariate missing-data problems: A data analyst's perspective. *Multivariate Behavioral Research*, *33*(4), 545–571.

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 253–260). New York, NY, USA: ACM Press.

Schneiderman, B. (1992). Tree visualizations with tree-maps: 2-d space filling approach. *ACM Transactions on Graphics*, *11*(1), 92–99.

Schwab, I., Pohl, W., & Koychev, I. (2000). Learning to recommend from positive evidence. In *Proceedings of the 5th International Conference on Intelligent User Interfaces* (pp. 241–246). New York, NY, USA: ACM Press.

Schwartz, B. (2004). *The paradox of choice: Why more is less*. New York, NY, USA: HarperCollins.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, *6*(2), 461–464.

Shardanand, U., & Maes, P. (1995). Social information filtering algorithms for automating "word of mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 210–217). New York, NY, USA: ACM Press/Addison-Wesley.

Shimazu, H. (2002). ExpertClerk: A conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops. *Artificial Intelligence Review*, *18*, 223–244.

Silva, V. de, & Tenenbaum, J. B. (2003). Global versus local methods in nonlinear dimension reduction. In *Advances in Neural Information Processing Systems 15* (pp. 721–728). Cambridge, MA, USA: MIT Press.

Silva, V. de, & Tenenbaum, J. B. (2004). *Sparse multidimensional scaling using landmark points* (Tech. Rep.). Stanford, CA, USA: Stanford University.

Sinha, R., & Swearingen, K. (2002). The role of transparency in recommender systems. In *CHI '02 Extended Abstracts on human Factors in Computing Systems* (pp. 830–831). New York, NY, USA: ACM Press.

Soboroff, I., & Nicholas, C. (1999). Combining content and collaboration in text filtering. In *Proceedings of IJCAI Workshop Machine Learning for Information Filtering.*

Spence, I., & Domoney, D. W. (1974). Single subject incomplete designs for nonmetric multidimensional scaling. *Psychometrika*, *39*, 469–490.

Stappers, P. J., & Pasman, G. (1999). Exploring a database through interactive visualised similarity scaling. In M. W. Altom & M. G. Williams (Eds.), *CHI '99 Extended Abstracts on human Factors in Computing Systems* (pp. 184–185). New York, NY, USA: ACM Press.

Stappers, P. J., Pasman, G., & Groenen, P. J. F. (2000). Exploring databases for taste or inspiration with interactive multi-dimensional scaling. In *Proceedings IEA 2000 / HFES 2000, Ergonomics for the New Millennium* (pp. 3-575–3-578). Santa Monica, CA, USA: The Human Factors Society of the USA.

Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2008). Matrix factorization and neighbor based algorithms for the Netflix prize problem. In *Proceedings of the 2008 ACM Conference on Recommender Systems* (pp. 267–274). New York, NY, USA: ACM Press.

Tenenbaum, J. B. (1998). Mapping a manifold of perceptual observations. In *Advances in Neural Information Processing Systems 10* (pp. 682–688). Cambridge, MA, USA: MIT Press.

Tenenbaum, J. B., Silva, V. de, & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Therneau, T. M., & Atkinson, E. J. (1997). *An introduction to recursive partitioning using the RPART routines* (Tech. Rep. No. 61). Rochester, MN, USA: Mayo Foundation.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267–288.

Tintarev, N., & Masthoff, J. (2007). A survey of explanations in recommender systems. In V. Oria, A. Elmagarmid, F. Lochovsky, & Y. Saygin (Eds.), *Proceedings of the 23rd International Conference on Data Engineering Workshops* (pp. 801–810). Los Alamitos, CA, USA: IEEE Computer Society Press.

Tobler, W., & Wineberg, S. (1971). A Cappadocian speculation. *Nature*, *231*, 39–41.

Torgerson, W. S. (1952). Multidimensional scaling: I. Theory and method. *Psychometrika*, *17*, 401–419.

Tran, T., & Cohen, R. (2000). Hybrid recommender systems for electronic commerce. In *Proceedings of Knowledge-Based Electronic Markets, Papers from the AAAI Workshop.*

Menlo Park, CA, USA: AAAI Press.

Turetken, O., & Sharda, R. (2004). Developement of a fisheye-based information search processing aid (FISPA) for managing information overload in the web environment. *Decision Support Systems*, *37*, 415–434.

Van Gulik, R., Vignoli, F., & Van der Wetering, H. (2004). Mapping music in the palm of your hand, explore and discover your collection. In *Proceedings of the 5th International Conference on Music Information Retrieval.*

Vriens, M., Wedel, M., & Wilms, T. (1996). Metric conjoint segmentation methods: A monte carlo comparison. *Journal of Marketing Research*, *33*(1), 73–85.

Wang, J. T. L., Wang, X., Lin, K. I., Shasha, D., Shapiro, B. A., & Zhang, K. (1999). Evaluating a class of distance-mapping algorithms for data mining and clustering. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 307–311). New York, NY, USA: ACM Press.

Wedel, M., & DeSarbo, W. S. (1995). A mixture likelihood approach for generalized linear models. *Journal of Classification*, *12*(1), 21–55.

Wilson, D. R., & Martinez, T. R. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, *6*, 1–34.

Yang, C. C., Chen, H., & Hong, K. (2003). Visualization of large category map for Internet browsing. *Decision Support Systems*, *35*, 89–102.

Yang, T., Liu, J., McMillan, L., & Wang, W. (2006). A fast approximation to multidimensional scaling. In *Proceedings of the ECCV Workshop on Computation Intensive Methods for Computer Vision.*

Zhou, Y., Wilkinson, D., Schreiber, R., & Pan, R. (2008). Large-scale parallel collaborative filtering for the Netflix prize. *Lecture Notes in Computer Science*, *5034*, 337–348.

Zielman, B., & Heiser, W. J. (1993). Analysis of asymmetry by a slide-vector. *Psychometrika*, *58*, 101–114.

Author Index

Adomavicius, G., 7, 38, 61, 87, 88, 143
Aikake, H., 100, 143
Alhoniemi, E., 16, 143
Ansari, A., 88, 143
Arslan, B., 57, 62, 64, 143
Atkinson, E. J., 65, 68, 71, 152
Ayanso, A., 39, 143

Balabanović, M., 88, 143
Basu, C., 88, 143
Bederson, B. B., 17, 143
Bell, R., 87, 149
Bergstrom, P., 87, 150
Bettman, J. R., 39, 143
Billsus, D., 85, 87, 88, 143, 150
Bishop, C. M., 91, 143
Blackwell, M., 26, 66, 147
Blei, D. M., 87, 143
Bonillas, A., 15, 39, 145
Borchers, A., 100, 146, 147
Borg, I., 3, 8, 14, 18, 38, 40, 42, 50, 57, 111, 127, 139, 144
Brandes, U., 112, 144
Branting, L. K., 62, 144
Breese, J. S., 87, 100, 144
Breiman, L., 68, 69, 71, 144
Bruls, M., 17, 144
Buja, A., 123, 127, 132, 136, 144, 145
Burke, R., 39, 64, 86, 87, 144, 151

Camargo, S. J., 93, 144
Chalmers, M., 111, 112, 144, 149
Chaudhuri, S., 38, 39, 144
Chen, H., 15, 39, 144, 145, 150, 153
Chen, L., 132, 136, 144, 145
Chen, M.-H., 26, 66, 148
Choi, S., 112, 149
Chung, W., 15, 39, 135, 145
Claypool, M., 88, 145
Cohen, R., 88, 152
Cohen, W., 88, 143
Comon, P., 36, 145
Condliff, M., 88, 145
Cormen, T. H., 119, 145
Courtial, J.-P., 127, 151
Cover, T., 39, 98, 145
Coyle, L., 62, 64, 145
Cron, W. L., 9, 86, 93–95, 106, 140, 146
Cunningham, P., 62, 64, 145

De Leeuw, J., 14, 18, 20, 23, 24, 42, 111, 113, 114, 145, 149
Dean, N., 144
Deerwester, S., 88, 145
Del Missier, F., 64, 150
Dempster, A. P., 89, 94, 146
DeSarbo, W. S., 9, 86, 93–95, 106, 107, 140, 146, 150, 153
Di Benedetto, C. A., 93, 146
Do, M. N., 18, 39, 150
Domoney, D. W., 117, 152
Donaldson, J., 18, 39, 135, 146
Doyle, D., 64, 145
Dumais, S. T., 88, 145

Eades, P., 112, 146
Essegaier, S., 88, 143

Faloutsos, C., 112, 146
Freund, Y., 68, 69, 146

Friedman, J. H., 36, 63, 68, 69, 71, 83, 84, 144, 146, 147
Furnas, G. W., 88, 145

Gaffney, S. J., 93, 144
Garcia, P., 93, 150
Ghil, M., 93, 144
Gifi, A., 14, 20, 24, 146
Goes, P. B., 39, 143
Gokhale, A., 145
Goldberg, D., 39, 85, 87, 146
Goldberg, K., 87, 146
Good, N., 88, 146
Gower, J. C., 5, 21, 24, 40–42, 64, 111, 117, 126, 146, 147
Gravano, L., 38, 39, 144
Green, B. F., 29, 147
Groenen, P. J. F., 3, 8, 9, 13, 14, 18, 21, 36–40, 42, 50, 57, 61, 63, 64, 111, 117, 120, 126, 127, 139, 144, 147–149, 152
Gupta, D., 87, 146

Hand, D. J., 24, 147
Harshman, R., 88, 145
Hart, P., 39, 98, 145
Hartigan, J. A., 23, 147
Hastie, T., 68, 84, 147
Heckerman, D., 87, 144
Heiser, W. J., 19, 20, 111, 114, 120, 126, 145, 147, 149, 153
Herlocker, J. L., 61, 88, 100, 135, 146, 147
Herring, A. H., 26, 66, 148
Hicklin, J., 18, 147
Himberg, J., 16, 143
Hirsh, H., 88, 143
Hofmann, H., 144
Hofmann, T., 9, 85–89, 91, 96–98, 100, 106, 140, 147
Honaker, J., 26, 66, 147, 148
Hong, K., 15, 39, 153
Houston, A. L., 15, 144
Huizing, K., 17, 144

Iacovou, N., 87, 150
Ibrahim, J. G., 26, 66, 148

Jameson, A., 62, 148
Jolliffe, I., 3, 148
Jordan, M. I., 87, 143
Joseph, A., 26, 66, 148
Jourdan, F., 112, 148

Kadie, C., 87, 144
Kagie, M., 8, 9, 13, 14, 20–22, 25, 26, 28, 36, 37, 39, 61, 63, 64, 66, 67, 70, 81, 82, 85, 140, 141, 148
Karypis, G., 87, 151
Kearsley, A. J., 113, 148
Keller, I., 39, 148
King, G., 26, 66, 147, 148
Kohli, R., 88, 143
Kohonen, T., 15, 16, 39, 149
Konstan, J. A., 5, 38, 61, 87, 88, 100, 146, 147, 151
Koren, Y., 87, 149
Koychev, I., 62, 151
Kriegler, M., 93, 149
Kruskal, J. B., 18, 22, 42, 111, 149
Kumar, P., 39, 150

Lain, G., 15, 39, 145
Laird, N. M., 89, 146
Landauer, T. K., 88, 145
Langford, J. C., 125, 152
Larcker, D. F., 93, 149
Lawrence, S., 88, 150
Lee, S., 112, 149
Legendre, P., 126, 147
Leiserson, C. E., 119, 145
Lewis, D., 88, 145
Lin, K., 112, 146
Lin, K. I., 153
Linden, G., 87, 149
Linting, M., 8, 14, 20, 24, 139, 149
Lipsitz, S. R., 26, 66, 148
Littman, M. L., 144
Liu, J., 112, 153

Logan, B. F., 123, 144
Lorenzi, F., 39, 62, 63, 149
Luce, M. F., 39, 143

Madigan, D., 88, 145
Maes, P., 100, 151
Martinez, T. R., 64, 153
Masthoff, J., 62, 88, 135, 152
McCarthy, K., 36, 150
McCullagh, P., 14, 25, 63, 65, 66, 107, 149
McGinty, L., 36, 39, 52, 149, 150
McMillan, L., 112, 153
McSherry, D., 62, 64, 88, 149
Mehta, K., 39, 143
Melançon, G., 112, 148
Melville, P., 86, 88, 98, 149
Meulman, J. J., 8, 14, 19, 20, 120, 147, 149
Michailidis, G., 14, 20, 24, 149
Miranda, T., 145
Mirzadeh, N., 57, 62, 143
Mooney, R. J., 86, 149
Morrison, A., 112, 149
Murnikov, P., 145

Nagarajan, R., 86, 149
Nelder, J. A., 14, 25, 63, 65, 66, 107, 149
Németh, B., 87, 152
Netes, D., 145
Ng, A. Y., 87, 143
Nicholas, C., 88, 152
Nichols, D., 39, 85, 146

Oki, B. M., 39, 85, 146
Olsen, M. K., 26, 66, 151
Olshen, R., 68, 144
Ong, T.-H., 15, 39, 135, 150
O'Sullivan, D., 62, 63, 150

Pan, R., 87, 153
Parviainen, J., 16, 143
Pasman, G., 18, 39, 152
Payne, J. W., 39, 143
Pazzani, M. J., 85, 87, 88, 143, 150
Pennings, J. M., 93, 150
Pennock, D. M., 88, 150, 151
Perkins, C., 87, 146
Pečenović, Z., 18, 39, 135, 150
Pich, C., 112, 144
Pilászy, I., 87, 152
Platt, J. C., 112, 150
Pohl, W., 62, 151
Popescul, A., 88, 150, 151
Posse, C., 88, 145
Prasad, B., 38, 87, 150
Pu, P. H. Z., 18, 39, 150

Ramaswamy, V., 93, 94, 146, 150
Reeds, J. A., 123, 144
Reibstein, D. J., 93, 150
Reilly, J., 36, 150
Resnick, P., 2, 36, 61, 85, 87, 150
Ricci, F., 38, 39, 57, 62–64, 143, 149–151
Richardson, S. A., 93, 149
Riedl, J. T., 5, 38, 61, 87, 88, 100, 147, 150, 151
Rip, A., 127, 151
Ripley, D. B., 68, 151
Rivest, R. L., 119, 145
Robertson, A. W., 93, 144
Robinson, W. T., 93, 150
Roeder, T., 87, 146
Ross, G., 112, 149
Rubin, D. B., 26, 63, 66, 67, 89, 146, 151

Sammon, J. W., 18, 39, 111, 151
Sandvig, J., 64, 151
Sartin, M., 145
Sarwar, B. M., 87, 146, 151
Schafer, J. B., 5, 6, 38, 146, 151
Schafer, J. L., 26, 66, 151
Schäfer, R., 62, 148
Schapire, R. E., 68, 69, 146
Schatz, B. R., 15, 144
Schein, A. I., 88, 151
Scheve, K., 26, 66, 148
Schneiderman, B., 16, 17, 39, 143, 151
Schreiber, R., 87, 153

Schwab, I., 62, 151
Schwartz, B., 37, 151
Schwarz, G., 100, 151
Sewell, R. R., 15, 144
Shapiro, B. A., 153
Sharda, R., 16, 39, 135, 153
Shardanand, U., 100, 151
Shasha, D., 153
Shepp, L. A., 123, 144
Shimazu, H., 8, 36, 39, 140, 151
Shoham, Y., 88, 143
Silva, V. de, 112, 125, 152
Simons, J., 62, 148
Sinha, I., 93, 146
Sinha, R., 61, 88, 135, 152
Smith, B., 87, 149
Smyth, B., 36, 39, 52, 62, 149, 150
Smyth, P., 93, 144
Soboroff, I., 88, 152
Song, M., 93, 146
Spence, I., 117, 152
Stappers, P. J., 18, 39, 57, 152
Stein, C., 119, 145
Stone, C., 68, 144
Suchak, M., 87, 150
Sung, W., 15, 39, 150
Swayne, D. F., 127, 144
Swearingen, K., 61, 88, 135, 152

Takács, G., 87, 152
Tapia, R. A., 113, 148
Tenenbaum, J. B., 112, 125, 152
Terry, D., 39, 85, 146
Terveen, L. G., 100, 147
Therneau, T. M., 65, 68, 71, 152
Tibshirani, R., 68, 107, 147, 152
Tikk, D., 87, 152
Tintarev, N., 62, 88, 135, 152
Tobler, W., 126, 152
Torgerson, W. S., 18, 39, 111, 112, 152
Tran, T., 88, 152
Trosset, M. W., 113, 148
Tukey, J. W., 36, 146
Turetken, O., 16, 39, 135, 153
Tuzhilin, A., 7, 38, 61, 87, 88, 143

Ungar, L. H., 88, 150, 151

Van der Kooij, A. J., 8, 14, 149
Van der Loos, M., 9, 85, 141, 148
Van der Wetering, H., 15, 39, 153
Van Gulik, R., 15, 39, 135, 153
Van Wezel, M., 8, 9, 13, 14, 21, 36, 37, 39, 61, 63, 64, 85, 141, 148
Van Wijk, J. J., 17, 144
Varian, H. R., 2, 36, 61, 85, 150
Venturini, A., 57, 62, 143
Vesanto, J., 16, 143
Vetterli, M., 18, 39, 150
Vignoli, F., 15, 39, 153
Volinsky, C., 87, 149
Vriens, M., 93, 146, 153

Wang, J. T. L., 112, 153
Wang, W., 112, 153
Wang, X., 153
Wattenberg, M., 17, 143
Wedderburn, R. W. M., 14, 25, 63, 65, 66, 149
Wedel, M., 93, 107, 146, 153
Weis, T., 62, 148
Wilkinson, D., 87, 153
Wilms, T., 93, 153
Wilson, D. C., 62, 150
Wilson, D. R., 64, 153
Wineberg, S., 126, 152
Wöber, K., 38, 151
Wong, M. A., 23, 147

Xi, W., 15, 39, 145

Yang, C. C., 15, 39, 153
Yang, T., 112, 153
York, J., 87, 149

Zhang, K., 153
Zhou, Y., 87, 153
Zhu, B., 15, 39, 150

Zielman, B., 126, 153
Zimprich, D., 93, 149
Zins, A., 38, 151

Curriculum Vitae

Martijn Kagie (1982) obtained his master's degree in Informatics and Economics from Erasmus University Rotterdam in November 2005. In January 2006, he started his PhD research. His research interests are in electronic commerce research, recommender systems, data visualization, and online shopping interfaces. His research has been presented at several international conferences and has resulted in several papers of which five have been published or accepted for publication in *AI Communications*, *Decision Support Systems*, *Intelligent Systems in Accounting, Finance and Management*, *Lecture Notes in Computer Science*, and the *Recommender Systems Handbook*.

ERASMUS RESEARCH INSTITUTE OF MANAGEMENT (ERIM)

ERIM PH.D. SERIES

RESEARCH IN MANAGEMENT

ERIM Electronic Series Portal: http://hdl.handle.net/1765/1

Agatz, N.A.H., *Demand Management in E-Fulfillment*, Promotor: Prof.dr.ir. J.A.E.E. van Nunen, EPS-2009-163-LIS, ISBN: 978-90-5892-200-7, http://hdl.handle.net/1765/15425

Althuizen, N.A.P., *Analogical Reasoning as a Decision Support Principle for Weakly Structured Marketing Problems*, Promotor: Prof.dr.ir. B. Wierenga, EPS-2006-095-MKT, ISBN: 90-5892-129-8, http://hdl.handle.net/1765/8190

Alvarez, H.L., *Distributed Collaborative Learning Communities Enabled by Information Communication Technology*, Promotor: Prof.dr. K. Kumar, EPS-2006-080-LIS, ISBN: 90-5892-112-3, http://hdl.handle.net/1765/7830

Appelman, J.H., *Governance of Global Interorganizational Tourism Networks: Changing Forms of Co-ordination between the Travel Agency and Aviation Sector*, Promotors: Prof.dr. F.M. Go & Prof.dr. B. Nooteboom, EPS-2004-036-MKT, ISBN: 90-5892-060-7, http://hdl.handle.net/1765/1199

Asperen, E. van, *Essays on Port, Container, and Bulk Chemical Logistics Optimization*, Promotor: Prof.dr.ir. R. Dekker, EPS-2009-181-LIS, ISBN: 978-90-5892-222-9, http://hdl.handle.net/1765/1

Assem, M.J. van den, *Deal or No Deal? Decision Making under Risk in a Large-Stake TV Game Show and Related Experiments*, Promotor: Prof.dr. J. Spronk, EPS-2008-138-F&A, ISBN: 978-90-5892-173-4, http://hdl.handle.net/1765/13566

Baquero, G, *On Hedge Fund Performance, Capital Flows and Investor Psychology*, Promotor: Prof.dr. M.J.C.M. Verbeek, EPS-2006-094-F&A, ISBN: 90-5892-131-X, http://hdl.handle.net/1765/8192

Berens, G., *Corporate Branding: The Development of Corporate Associations and their Influence on Stakeholder Reactions*, Promotor: Prof.dr. C.B.M. van Riel, EPS-2004-039-ORG, ISBN: 90-5892-065-8, http://hdl.handle.net/1765/1273

Berghe, D.A.F. van den, *Working Across Borders: Multinational Enterprises and the Internationalization of Employment*, Promotors: Prof.dr. R.J.M. van Tulder & Prof.dr. E.J.J. Schenk, EPS-2003-029-ORG, ISBN: 90-5892-05-34, http://hdl.handle.net/1765/1041

Berghman, L.A., *Strategic Innovation Capacity: A Mixed Method Study on Deliberate Strategic Learning Mechanisms*, Promotor: Prof.dr. P. Mattyssens, EPS-2006-087-MKT, ISBN: 90-5892-120-4, http://hdl.handle.net/1765/7991

Bijman, W.J.J., *Essays on Agricultural Co-operatives: Governance Structure in Fruit and Vegetable Chains*, Promotor: Prof.dr. G.W.J. Hendrikse, EPS-2002-015-ORG, ISBN: 90-5892-024-0, http://hdl.handle.net/1765/867

Bispo, A., *Labour Market Segmentation: An investigation into the Dutch hospitality industry*, Promotors: Prof.dr. G.H.M. Evers & Prof.dr. A.R. Thurik, EPS-2007-108-ORG, ISBN: 90-5892-136-9, http://hdl.handle.net/1765/10283

Blindenbach-Driessen, F., *Innovation Management in Project-Based Firms*, Promotor: Prof.dr. S.L. van de Velde, EPS-2006-082-LIS, ISBN: 90-5892-110-7, http://hdl.handle.net/1765/7828

Boer, C.A., *Distributed Simulation in Industry*, Promotors: Prof.dr. A. de Bruin & Prof.dr.ir. A. Verbraeck, EPS-2005-065-LIS, ISBN: 90-5892-093-3, http://hdl.handle.net/1765/6925

Boer, N.I., *Knowledge Sharing within Organizations: A situated and Relational Perspective*, Promotor: Prof.dr. K. Kumar, EPS-2005-060-LIS, ISBN: 90-5892-086-0, http://hdl.handle.net/1765/6770

Boer-Sorbán, K., *Agent-Based Simulation of Financial Markets: A modular, Continuous-Time Approach*, Promotor: Prof.dr. A. de Bruin, EPS-2008-119-LIS, ISBN: 90-5892-155-0, http://hdl.handle.net/1765/10870

Boon, C.T., *HRM and Fit: Survival of the Fittest!?*, Promotors: Prof.dr. J. Paauwe & Prof.dr. D.N. den Hartog, EPS-2008-129-ORG, ISBN: 978-90-5892-162-8, http://hdl.handle.net/1765/12606

Braun, E., *City Marketing: Towards an Integrated Approach*, Promotor: Prof.dr. L. van den Berg, EPS-2008-142-MKT, ISBN: 978-90-5892-180-2, http://hdl.handle.net/1765/13694

Brito, M.P. de, *Managing Reverse Logistics or Reversing Logistics Management?* Promotors: Prof.dr.ir. R. Dekker & Prof.dr. M. B. M. de Koster, EPS-2004-035-LIS, ISBN: 90-5892-058-5, http://hdl.handle.net/1765/1132

Brohm, R., *Polycentric Order in Organizations: A Dialogue between Michael Polanyi and IT-Consultants on Knowledge, Morality, and Organization*, Promotors: Prof.dr. G. W. J. Hendrikse & Prof.dr. H. K. Letiche, EPS-2005-063-ORG, ISBN: 90-5892-095-X, http://hdl.handle.net/1765/6911

Brumme, W.-H., *Manufacturing Capability Switching in the High-Tech Electronics Technology Life Cycle*, Promotors: Prof.dr.ir. J.A.E.E. van Nunen & Prof.dr.ir. L.N. Van Wassenhove, EPS-2008-126-LIS, ISBN: 978-90-5892-150-5, http://hdl.handle.net/1765/12103

Budiono, D.P., The Analysis of Mutual Fund Performance: Evidence from U.S. Equity Mutual Funds, Promotor: Prof.dr.ir. M.J.C.M. Verbeek, EPS-2010-185-F&A, ISBN: 978-90-5892-224-3, http://hdl.handle.net/1765/1

Burgers, J.H., *Managing Corporate Venturing: Multilevel Studies on Project Autonomy, Integration, Knowledge Relatedness, and Phases in the New Business Development Process*, Promotors: Prof.dr.ir. F.A.J. Van den Bosch & Prof.dr. H.W. Volberda, EPS-2008-136-STR, ISBN: 978-90-5892-174-1, http://hdl.handle.net/1765/13484

Campbell, R.A.J., *Rethinking Risk in International Financial Markets*, Promotor: Prof.dr. C.G. Koedijk, EPS-2001-005-F&A, ISBN: 90-5892-008-9, http://hdl.handle.net/1765/306

Chen, C.-M., *Evaluation and Design of Supply Chain Operations Using DEA*, Promotor: Prof.dr. J.A.E.E. van Nunen, EPS-2009-172-LIS, ISBN: 978-90-5892-209-0, http://hdl.handle.net/1765/1

Chen, H., *Individual Mobile Communication Services and Tariffs*, Promotor: Prof.dr. L.F.J.M. Pau, EPS-2008-123-LIS, ISBN: 90-5892-158-1, http://hdl.handle.net/1765/11141

Chen, Y., *Labour Flexibility in China's Companies: An Empirical Study*, Promotors: Prof.dr. A. Buitendam & Prof.dr. B. Krug, EPS-2001-006-ORG, ISBN: 90-5892-012-7, http://hdl.handle.net/1765/307

Damen, F.J.A., *Taking the Lead: The Role of Affect in Leadership Effectiveness*, Promotor: Prof.dr. D.L. van Knippenberg, EPS-2007-107-ORG, http://hdl.handle.net/1765/10282

Daniševská, P., *Empirical Studies on Financial Intermediation and Corporate Policies*, Promotor: Prof.dr. C.G. Koedijk, EPS-2004-044-F&A, ISBN: 90-5892-070-4, http://hdl.handle.net/1765/1518

Delporte-Vermeiren, D.J.E., *Improving the Flexibility and Profitability of ICT-enabled Business Networks: An Assessment Method and Tool*, Promotors: Prof. mr. dr. P.H.M. Vervest & Prof.dr.ir. H.W.G.M. van Heck, EPS-2003-020-LIS, ISBN: 90-5892-040-2, http://hdl.handle.net/1765/359

Derwall, J.M.M., *The Economic Virtues of SRI and CSR*, Promotor: Prof.dr. C.G. Koedijk, EPS-2007-101-F&A, ISBN: 90-5892-132-8, http://hdl.handle.net/1765/8986

Diepen, M. van, *Dynamics and Competition in Charitable Giving*, Promotor: Prof.dr. Ph.H.B.F. Franses, EPS-2009-159-MKT, ISBN: 978-90-5892-188-8, http://hdl.handle.net/1765/14526

Dietz, H.M.S., *Managing (Sales)People towards Perfomance: HR Strategy, Leadership & Teamwork*, Promotor: Prof.dr. G.W.J. Hendrikse, EPS-2009-168-ORG, ISBN: 978-90-5892-210-6, http://hdl.handle.net/1765/1

Dijksterhuis, M., *Organizational Dynamics of Cognition and Action in the Changing Dutch and US Banking Industries*, Promotors: Prof.dr.ir. F.A.J. van den Bosch & Prof.dr. H.W. Volberda, EPS-2003-026-STR, ISBN: 90-5892-048-8, http://hdl.handle.net/1765/1037

Eijk, A.R. van der, *Behind Networks: Knowledge Transfer, Favor Exchange and Performance*, Promotors: Prof.dr. S.L. van de Velde & Prof.dr.drs. W.A. Dolfsma, EPS-2009-161-LIS, ISBN: 978-90-5892-190-1, http://hdl.handle.net/1765/14613

Elstak, M.N., *Flipping the Identity Coin: The Comparative Effect of Perceived, Projected and Desired Organizational Identity on Organizational Identification and Desired Behavior*, Promotor: Prof.dr. C.B.M. van Riel, EPS-2008-117-ORG, ISBN: 90-5892-148-2, http://hdl.handle.net/1765/10723

Erken, H.P.G., *Productivity, R&D and Entrepreneurship*, Promotor: Prof.dr. A.R. Thurik, EPS-2008-147-ORG, ISBN: 978-90-5892-179-6, http://hdl.handle.net/1765/14004

Fenema, P.C. van, *Coordination and Control of Globally Distributed Software Projects*, Promotor: Prof.dr. K. Kumar, EPS-2002-019-LIS, ISBN: 90-5892-030-5, http://hdl.handle.net/1765/360

Fleischmann, M., *Quantitative Models for Reverse Logistics*, Promotors: Prof.dr.ir. J.A.E.E. van Nunen & Prof.dr.ir. R. Dekker, EPS-2000-002-LIS, ISBN: 35-4041-711-7, http://hdl.handle.net/1765/1044

Flier, B., *Strategic Renewal of European Financial Incumbents: Coevolution of Environmental Selection, Institutional Effects, and Managerial Intentionality*, Promotors: Prof.dr.ir. F.A.J. van den Bosch & Prof.dr. H.W. Volberda, EPS-2003-033-STR, ISBN: 90-5892-055-0, http://hdl.handle.net/1765/1071

Fok, D., *Advanced Econometric Marketing Models*, Promotor: Prof.dr. Ph.H.B.F. Franses, EPS-2003-027-MKT, ISBN: 90-5892-049-6, http://hdl.handle.net/1765/1035

Ganzaroli, A., *Creating Trust between Local and Global Systems*, Promotors: Prof.dr. K. Kumar & Prof.dr. R.M. Lee, EPS-2002-018-LIS, ISBN: 90-5892-031-3, http://hdl.handle.net/1765/361

Gertsen, H.F.M., *Riding a Tiger without Being Eaten: How Companies and Analysts Tame Financial Restatements and Influence Corporate Reputation*, Promotor: Prof.dr. C.B.M. van Riel, EPS-2009-171-ORG, ISBN: 90-5892-214-4, http://hdl.handle.net/1765/1

Gilsing, V.A., *Exploration, Exploitation and Co-evolution in Innovation Networks*, Promotors: Prof.dr. B. Nooteboom & Prof.dr. J.P.M. Groenewegen, EPS-2003-032-ORG, ISBN: 90-5892-054-2, http://hdl.handle.net/1765/1040

Gijsbers, G.W., *Agricultural Innovation in Asia: Drivers, Paradigms and Performance*, Promotor: Prof.dr. R.J.M. van Tulder, EPS-2009-156-ORG, ISBN: 978-90-5892-191-8, http://hdl.handle.net/1765/14524

Gong, Y., *Stochastic Modelling and Analysis of Warehouse Operations*, Promotors: Prof.dr. M.B.M. de Koster & Prof.dr. S.L. van de Velde, EPS-2009-180-LIS, ISBN: 978-90-5892-219-9, http://hdl.handle.net/1765/1

Govers, R., *Virtual Tourism Destination Image: Glocal Identities Constructed, Perceived and Experienced*, Promotors: Prof.dr. F.M. Go & Prof.dr. K. Kumar, EPS-2005-069-MKT, ISBN: 90-5892-107-7, http://hdl.handle.net/1765/6981

Graaf, G. de, *Tractable Morality: Customer Discourses of Bankers, Veterinarians and Charity Workers*, Promotors: Prof.dr. F. Leijnse & Prof.dr. T. van Willigenburg, EPS-2003-031-ORG, ISBN: 90-5892-051-8, http://hdl.handle.net/1765/1038

Greeven, M.J., *Innovation in an Uncertain Institutional Environment: Private Software Entrepreneurs in Hangzhou, China*, Promotor: Prof.dr. B. Krug, EPS-2009-164-ORG, ISBN: 978-90-5892-202-1, http://hdl.handle.net/1765/15426

Groot, E.A. de, *Essays on Economic Cycles*, Promotors: Prof.dr. Ph.H.B.F. Franses & Prof.dr. H.R. Commandeur, EPS-2006-091-MKT, ISBN: 90-5892-123-9, http://hdl.handle.net/1765/8216

Guenster, N.K., *Investment Strategies Based on Social Responsibility and Bubbles*, Promotor: Prof.dr. C.G. Koedijk, EPS-2008-175-F&A, ISBN: 978-90-5892-206-9, http://hdl.handle.net/1765/1

Gutkowska, A.B., *Essays on the Dynamic Portfolio Choice*, Promotor: Prof.dr. A.C.F. Vorst, EPS-2006-085-F&A, ISBN: 90-5892-118-2, http://hdl.handle.net/1765/7994

Hagemeijer, R.E., *The Unmasking of the Other*, Promotors: Prof.dr. S.J. Magala & Prof.dr. H.K. Letiche, EPS-2005-068-ORG, ISBN: 90-5892-097-6, http://hdl.handle.net/1765/6963

Hakimi, N.A, *Leader Empowering Behaviour: The Leader's Perspective: Understanding the Motivation behind Leader Empowering Behaviour*, Promotor: Prof.dr. D.L. van Knippenberg, EPS-2010-184-ORG, http://hdl.handle.net/1765/1

Halderen, M.D. van, *Organizational Identity Expressiveness and Perception Management: Principles for Expressing the Organizational Identity in Order to Manage the Perceptions and Behavioral Reactions of External Stakeholders*, Promotor: Prof.dr. S.B.M. van Riel, EPS-2008-122-ORG, ISBN: 90-5892-153-6, http://hdl.handle.net/1765/10872

Hartigh, E. den, *Increasing Returns and Firm Performance: An Empirical Study*, Promotor: Prof.dr. H.R. Commandeur, EPS-2005-067-STR, ISBN: 90-5892-098-4, http://hdl.handle.net/1765/6939

Hermans. J.M., *ICT in Information Services; Use and Deployment of the Dutch Securities Trade, 1860-1970*, Promotor: Prof.dr. drs. F.H.A. Janszen, EPS-2004-046-ORG, ISBN 90-5892-072-0, http://hdl.handle.net/1765/1793

Hessels, S.J.A., *International Entrepreneurship: Value Creation Across National Borders*, Promotor: Prof.dr. A.R. Thurik, EPS-2008-144-ORG, ISBN: 978-90-5892-181-9, http://hdl.handle.net/1765/13942

Heugens, P.P.M.A.R., *Strategic Issues Management: Implications for Corporate Performance*, Promotors: Prof.dr.ir. F.A.J. van den Bosch & Prof.dr. C.B.M. van Riel, EPS-2001-007-STR, ISBN: 90-5892-009-9, http://hdl.handle.net/1765/358

Heuvel, W. van den, *The Economic Lot-Sizing Problem: New Results and Extensions*, Promotor: Prof.dr. A.P.L. Wagelmans, EPS-2006-093-LIS, ISBN: 90-5892-124-7, http://hdl.handle.net/1765/1805

Hoedemaekers, C.M.W., *Performance, Pinned down: A Lacanian Analysis of Subjectivity at Work*, Promotors: Prof.dr. S. Magala & Prof.dr. D.H. den Hartog, EPS-2008-121-ORG, ISBN: 90-5892-156-7, http://hdl.handle.net/1765/10871

Hooghiemstra, R., *The Construction of Reality: Cultural Differences in Self-serving Behaviour in Accounting Narratives*, Promotors: Prof.dr. L.G. van der Tas RA & Prof.dr. A.Th.H. Pruyn, EPS-2003-025-F&A, ISBN: 90-5892-047-X, http://hdl.handle.net/1765/871

Hu, Y., *Essays on the Governance of Agricultural Products: Cooperatives and Contract Farming*, Promotors: Prof.dr. G.W.J. Hendrkse & Prof.dr. B. Krug, EPS-2007-113-ORG, ISBN: 90-5892-145-1, http://hdl.handle.net/1765/10535

Huij, J.J., *New Insights into Mutual Funds: Performance and Family Strategies*, Promotor: Prof.dr. M.C.J.M. Verbeek, EPS-2007-099-F&A, ISBN: 90-5892-134-4, http://hdl.handle.net/1765/9398

Huurman, C.I., *Dealing with Electricity Prices*, Promotor: Prof.dr. C.D. Koedijk, EPS-2007-098-F&A, ISBN: 90-5892-130-1, http://hdl.handle.net/1765/9399

Iastrebova, K, *Manager's Information Overload: The Impact of Coping Strategies on Decision-Making Performance*, Promotor: Prof.dr. H.G. van Dissel, EPS-2006-077-LIS, ISBN: 90-5892-111-5, http://hdl.handle.net/1765/7329

Iwaarden, J.D. van, *Changing Quality Controls: The Effects of Increasing Product Variety and Shortening Product Life Cycles*, Promotors: Prof.dr. B.G. Dale & Prof.dr. A.R.T. Williams, EPS-2006-084-ORG, ISBN: 90-5892-117-4, http://hdl.handle.net/1765/7992

Jansen, J.J.P., *Ambidextrous Organizations*, Promotors: Prof.dr.ir. F.A.J. Van den Bosch & Prof.dr. H.W. Volberda, EPS-2005-055-STR, ISBN: 90-5892-081-X, http://hdl.handle.net/1765/6774

Jaspers, F.P.H., *Organizing Systemic Innovation*, Promotor: Prof.dr.ir. J.C.M. van den Ende, EPS-2009-160-ORG, ISBN: 978-90-5892-197-), http://hdl.handle.net/1765/14974

Jennen, M.G.J., *Empirical Essays on Office Market Dynamics*, Promotors: Prof.dr. C.G. Koedijk & Prof.dr. D. Brounen, EPS-2008-140-F&A, ISBN: 978-90-5892-176-5, http://hdl.handle.net/1765/13692

Jiang, T., *Capital Structure Determinants and Governance Structure Variety in Franchising*, Promotors: Prof.dr. G. Hendrikse & Prof.dr. A. de Jong, EPS-2009-158-F&A, ISBN: 978-90-5892-199-4, http://hdl.handle.net/1765/14975

Jiao, T., *Essays in Financial Accounting*, Promotor: Prof.dr. G.M.H. Mertens, EPS-2009-176-F&A, ISBN: 978-90-5892-211-3, http://hdl.handle.net/1765/1

Jong, C. de, *Dealing with Derivatives: Studies on the Role, Informational Content and Pricing of Financial Derivatives*, Promotor: Prof.dr. C.G. Koedijk, EPS-2003-023-F&A, ISBN: 90-5892-043-7, http://hdl.handle.net/1765/1043

Kaa, G. van, *Standard Battles for Complex Systems: Empirical Research on the Home Network*, Promotors: Prof.dr.ir. J. van den Ende & Prof.dr.ir. H.W.G.M. van Heck, EPS-2009-166-ORG, ISBN: 978-90-5892-205-2, http://hdl.handle.net/1765/1

Keizer, A.B., *The Changing Logic of Japanese Employment Practices: A Firm-Level Analysis of Four Industries*, Promotors: Prof.dr. J.A. Stam & Prof.dr. J.P.M. Groenewegen, EPS-2005-057-ORG, ISBN: 90-5892-087-9, http://hdl.handle.net/1765/6667

Kijkuit, R.C., *Social Networks in the Front End: The Organizational Life of an Idea*, Promotor: Prof.dr. B. Nooteboom, EPS-2007-104-ORG, ISBN: 90-5892-137-6, http://hdl.handle.net/1765/10074

Kippers, J., *Empirical Studies on Cash Payments*, Promotor: Prof.dr. Ph.H.B.F. Franses, EPS-2004-043-F&A, ISBN: 90-5892-069-0, http://hdl.handle.net/1765/1520

Klein, M.H., *Poverty Alleviation through Sustainable Strategic Business Models: Essays on Poverty Alleviation as a Business Strategy*, Promotor: Prof.dr. H.R. Commandeur, EPS-2008-135-STR, ISBN: 978-90-5892-168-0, http://hdl.handle.net/1765/13482

Knapp, S., *The Econometrics of Maritime Safety: Recommendations to Enhance Safety at Sea*, Promotor: Prof.dr. Ph.H.B.F. Franses, EPS-2007-096-ORG, ISBN: 90-5892-127-1, http://hdl.handle.net/1765/7913

Kole, E., *On Crises, Crashes and Comovements*, Promotors: Prof.dr. C.G. Koedijk & Prof.dr. M.J.C.M. Verbeek, EPS-2006-083-F&A, ISBN: 90-5892-114-X, http://hdl.handle.net/1765/7829

Kooij-de Bode, J.M., *Distributed Information and Group Decision-Making: Effects of Diversity and Affect*, Promotor: Prof.dr. D.L. van Knippenberg, EPS-2007-115-ORG, http://hdl.handle.net/1765/10722

Koppius, O.R., *Information Architecture and Electronic Market Performance*, Promotors: Prof.dr. P.H.M. Vervest & Prof.dr.ir. H.W.G.M. van Heck, EPS-2002-013-LIS, ISBN: 90-5892-023-2, http://hdl.handle.net/1765/921

Kotlarsky, J., *Management of Globally Distributed Component-Based Software Development Projects*, Promotor: Prof.dr. K. Kumar, EPS-2005-059-LIS, ISBN: 90-5892-088-7, http://hdl.handle.net/1765/6772

Krauth, E.I., *Real-Time Planning Support: A Task-Technology Fit Perspective*, Promotors: Prof.dr. S.L. van de Velde & Prof.dr. J. van Hillegersberg, EPS-2008-155-LIS, ISBN: 978-90-5892-193-2, http://hdl.handle.net/1765/14521

Kuilman, J., *The Re-Emergence of Foreign Banks in Shanghai: An Ecological Analysis*, Promotor: Prof.dr. B. Krug, EPS-2005-066-ORG, ISBN: 90-5892-096-8, http://hdl.handle.net/1765/6926

Kwee, Z., *Investigating Three Key Principles of Sustained Strategic Renewal: A Longitudinal Study of Long-Lived Firms*, Promotors: Prof.dr.ir. F.A.J. Van den Bosch & Prof.dr. H.W. Volberda, EPS-2009-174-STR, ISBN: 90-5892-212-0, http://hdl.handle.net/1765/1

Langen, P.W. de, *The Performance of Seaport Clusters: A Framework to Analyze Cluster Performance and an Application to the Seaport Clusters of Durban, Rotterdam and the Lower Mississippi*, Promotors: Prof.dr. B. Nooteboom & Prof. drs. H.W.H. Welters, EPS-2004-034-LIS, ISBN: 90-5892-056-9, http://hdl.handle.net/1765/1133

Le Anh, T., *Intelligent Control of Vehicle-Based Internal Transport Systems*, Promotors: Prof.dr. M.B.M. de Koster & Prof.dr.ir. R. Dekker, EPS-2005-051-LIS, ISBN: 90-5892-079-8, http://hdl.handle.net/1765/6554

Le-Duc, T., *Design and Control of Efficient Order Picking Processes*, Promotor: Prof.dr. M.B.M. de Koster, EPS-2005-064-LIS, ISBN: 90-5892-094-1, http://hdl.handle.net/1765/6910

Leeuwen, E.P. van, *Recovered-Resource Dependent Industries and the Strategic Renewal of Incumbent Firm: A Multi-Level Study of Recovered Resource Dependence Management and Strategic Renewal in the European Paper and Board Industry*, Promotors: Prof.dr.ir. F.A.J. Van den Bosch & Prof.dr. H.W. Volberda, EPS-2007-109-STR, ISBN: 90-5892-140-6, http://hdl.handle.net/1765/10183

Lentink, R.M., *Algorithmic Decision Support for Shunt Planning*, Promotors: Prof.dr. L.G. Kroon & Prof.dr.ir. J.A.E.E. van Nunen, EPS-2006-073-LIS, ISBN: 90-5892-104-2, http://hdl.handle.net/1765/7328

Li, T., *Informedness and Customer-Centric Revenue Management*, Promotors: Prof.dr. P.H.M. Vervest & Prof.dr.ir. H.W.G.M. van Heck, EPS-2009-146-LIS, ISBN: 978-90-5892-195-6, http://hdl.handle.net/1765/14525

Liang, G., *New Competition: Foreign Direct Investment and Industrial Development in China*, Promotor: Prof.dr. R.J.M. van Tulder, EPS-2004-047-ORG, ISBN: 90-5892-073-9, http://hdl.handle.net/1765/1795

Liere, D.W. van, *Network Horizon and the Dynamics of Network Positions: A Multi-Method Multi-Level Longitudinal Study of Interfirm Networks*, Promotor: Prof.dr. P.H.M. Vervest, EPS-2007-105-LIS, ISBN: 90-5892-139-0, http://hdl.handle.net/1765/10181

Loef, J., *Incongruity between Ads and Consumer Expectations of Advertising*, Promotors: Prof.dr. W.F. van Raaij & Prof.dr. G. Antonides, EPS-2002-017-MKT, ISBN: 90-5892-028-3, http://hdl.handle.net/1765/869

Maeseneire, W., de, *Essays on Firm Valuation and Value Appropriation*, Promotor: Prof.dr. J.T.J. Smit, EPS-2005-053-F&A, ISBN: 90-5892-082-8, http://hdl.handle.net/1765/6768

Londoño, M. del Pilar, *Institutional Arrangements that Affect Free Trade Agreements: Economic Rationality Versus Interest Groups*, Promotors: Prof.dr. H.E. Haralambides & Prof.dr. J.F. Francois, EPS-2006-078-LIS, ISBN: 90-5892-108-5, http://hdl.handle.net/1765/7578

Maas, A.A., van der, *Strategy Implementation in a Small Island Context: An Integrative Framework*, Promotor: Prof.dr. H.G. van Dissel, EPS-2008-127-LIS, ISBN: 978-90-5892-160-4, http://hdl.handle.net/1765/12278

Maas, K.E.G., *Corporate Social Performance: From Output Measurement to Impact Measurement*, Promotor: Prof.dr. H.R. Commandeur, EPS-2009-182-STR, ISBN: 978-90-5892-225-0, http://hdl.handle.net/1765/1

Maeseneire, W., de, *Essays on Firm Valuation and Value Appropriation*, Promotor: Prof.dr. J.T.J. Smit, EPS-2005-053-F&A, ISBN: 90-5892-082-8, http://hdl.handle.net/1765/6768

Mandele, L.M., van der, *Leadership and the Inflection Point: A Longitudinal Perspective*, Promotors: Prof.dr. H.W. Volberda & Prof.dr. H.R. Commandeur, EPS-2004-042-STR, ISBN: 90-5892-067-4, http://hdl.handle.net/1765/1302

Meer, J.R. van der, *Operational Control of Internal Transport*, Promotors: Prof.dr. M.B.M. de Koster & Prof.dr.ir. R. Dekker, EPS-2000-001-LIS, ISBN: 90-5892-004-6, http://hdl.handle.net/1765/859

Mentink, A., *Essays on Corporate Bonds*, Promotor: Prof.dr. A.C.F. Vorst, EPS-2005-070-F&A, ISBN: 90-5892-100-X, http://hdl.handle.net/1765/7121

Meyer, R.J.H., *Mapping the Mind of the Strategist: A Quantitative Methodology for Measuring the Strategic Beliefs of Executives*, Promotor: Prof.dr. R.J.M. van Tulder, EPS-2007-106-ORG, ISBN: 978-90-5892-141-3, http://hdl.handle.net/1765/10182

Miltenburg, P.R., *Effects of Modular Sourcing on Manufacturing Flexibility in the Automotive Industry: A Study among German OEMs*, Promotors: Prof.dr. J. Paauwe & Prof.dr. H.R. Commandeur, EPS-2003-030-ORG, ISBN: 90-5892-052-6, http://hdl.handle.net/1765/1039

Moerman, G.A., *Empirical Studies on Asset Pricing and Banking in the Euro Area*, Promotor: Prof.dr. C.G. Koedijk, EPS-2005-058-F&A, ISBN: 90-5892-090-9, http://hdl.handle.net/1765/6666

Moitra, D., *Globalization of R&D: Leveraging Offshoring for Innovative Capability and Organizational Flexibility*, Promotor: Prof.dr. K. Kumar, EPS-2008-150-LIS, ISBN: 978-90-5892-184-0, http://hdl.handle.net/1765/14081

Mol, M.M., *Outsourcing, Supplier-relations and Internationalisation: Global Source Strategy as a Chinese Puzzle*, Promotor: Prof.dr. R.J.M. van Tulder, EPS-2001-010-ORG, ISBN: 90-5892-014-3, http://hdl.handle.net/1765/355

Mom, T.J.M., *Managers' Exploration and Exploitation Activities: The Influence of Organizational Factors and Knowledge Inflows*, Promotors: Prof.dr.ir. F.A.J. Van den Bosch & Prof.dr. H.W. Volberda, EPS-2006-079-STR, ISBN: 90-5892-116-6, http://hdl.handle.net/1765

Moonen, J.M., *Multi-Agent Systems for Transportation Planning and Coordination*, Promotors: Prof.dr. J. van Hillegersberg & Prof.dr. S.L. van de Velde, EPS-2009-177-LIS, ISBN: 978-90-5892-216-8, http://hdl.handle.net/1

Mulder, A., *Government Dilemmas in the Private Provision of Public Goods*, Promotor: Prof.dr. R.J.M. van Tulder, EPS-2004-045-ORG, ISBN: 90-5892-071-2, http://hdl.handle.net/1765/1790

Muller, A.R., *The Rise of Regionalism: Core Company Strategies Under The Second Wave of Integration*, Promotor: Prof.dr. R.J.M. van Tulder, EPS-2004-038-ORG, ISBN: 90-5892-062-3, http://hdl.handle.net/1765/1272

Nalbantov G.I., *Essays on Some Recent Penalization Methods with Applications in Finance and Marketing*, Promotor: Prof. dr P.J.F. Groenen, EPS-2008-132-F&A, ISBN: 978-90-5892-166-6, http://hdl.handle.net/1765/13319

Nederveen Pieterse, A., *Goal Orientation in Teams: The Role of Diversity*, Promotor: Prof.dr. D.L. van Knippenberg, EPS-2009-162-ORG, http://hdl.handle.net/1765/15240

Nguyen, T.T., *Capital Structure, Strategic Competition, and Governance*, Promotor: Prof.dr. A. de Jong, EPS-2008-148-F&A, ISBN: 90-5892-178-9, http://hdl.handle.net/1765/14005

Niesten, E.M.M.I., *Regulation, Governance and Adaptation: Governance Transformations in the Dutch and French Liberalizing Electricity Industries*, Promotors: Prof.dr. A. Jolink & Prof.dr. J.P.M. Groenewegen, EPS-2009-170-ORG, ISBN: 978-90-5892-208-3, http://hdl.handle.net/1765/1

Nieuwenboer, N.A. den, *Seeing the Shadow of the Self*, Promotor: Prof.dr. S.P. Kaptein, EPS-2008-151-ORG, ISBN: 978-90-5892-182-6, http://hdl.handle.net/1765/14223

Ning, H., *Hierarchical Portfolio Management: Theory and Applications*, Promotor: Prof.dr. J. Spronk, EPS-2007-118-F&A, ISBN: 90-5892-152-9, http://hdl.handle.net/1765/10868

Noeverman, J., *Management Control Systems, Evaluative Style, and Behaviour: Exploring the Concept and Behavioural Consequences of Evaluative Style*, Promotors: Prof.dr. E.G.J. Vosselman & Prof.dr. A.R.T. Williams, EPS-2007-120-F&A, ISBN: 90-5892-151-2, http://hdl.handle.net/1765/10869

Nuijten, I., *Servant Leadership: Paradox or Diamond in the Rough? A Multidimensional Measure and Empirical Evidence*, Promotor: Prof.dr. D.L. van Knippenberg, EPS-2009-183-ORG, http://hdl.handle.net/1765/1

Oosterhout, J., van, *The Quest for Legitimacy: On Authority and Responsibility in Governance*, Promotors: Prof.dr. T. van Willigenburg & Prof.mr. H.R. van Gunsteren, EPS-2002-012-ORG, ISBN: 90-5892-022-4, http://hdl.handle.net/1765/362

Oostrum, J.M., van, *Applying Mathematical Models to Surgical Patient Planning*, Promotor: Prof.dr. A.P.M. Wagelmans, EPS-2009-179-LIS, ISBN: 978-90-5892-217-5, http://hdl.handle.net/1765/1

Paape, L., *Corporate Governance: The Impact on the Role, Position, and Scope of Services of the Internal Audit Function*, Promotors: Prof.dr. G.J. van der Pijl & Prof.dr. H. Commandeur, EPS-2007-111-MKT, ISBN: 90-5892-143-7, http://hdl.handle.net/1765/10417

Pak, K., *Revenue Management: New Features and Models*, Promotor: Prof.dr.ir. R. Dekker, EPS-2005-061-LIS, ISBN: 90-5892-092-5, http://hdl.handle.net/1765/362/6771

Pattikawa, L.H, *Innovation in the Pharmaceutical Industry: Evidence from Drug Introduction in the U.S.*, Promotors: Prof.dr. H.R.Commandeur, EPS-2007-102-MKT, ISBN: 90-5892-135-2, http://hdl.handle.net/1765/9626

Peeters, L.W.P., *Cyclic Railway Timetable Optimization*, Promotors: Prof.dr. L.G. Kroon & Prof.dr.ir. J.A.E.E. van Nunen, EPS-2003-022-LIS, ISBN: 90-5892-042-9, http://hdl.handle.net/1765/429

Pietersz, R., *Pricing Models for Bermudan-style Interest Rate Derivatives*, Promotors: Prof.dr. A.A.J. Pelsser & Prof.dr. A.C.F. Vorst, EPS-2005-071-F&A, ISBN: 90-5892-099-2, http://hdl.handle.net/1765/7122

Poel, A.M. van der, *Empirical Essays in Corporate Finance and Financial Reporting*, Promotors: Prof.dr. A. de Jong & Prof.dr. G.M.H. Mertens, EPS-2007-133-F&A, ISBN: 978-90-5892-165-9, http://hdl.handle.net/1765/13320

Popova, V., *Knowledge Discovery and Monotonicity*, Promotor: Prof.dr. A. de Bruin, EPS-2004-037-LIS, ISBN: 90-5892-061-5, http://hdl.handle.net/1765/1201

Pouchkarev, I., *Performance Evaluation of Constrained Portfolios*, Promotors: Prof.dr. J. Spronk & Dr. W.G.P.M. Hallerbach, EPS-2005-052-F&A, ISBN: 90-5892-083-6, http://hdl.handle.net/1765/6731

Prins, R., *Modeling Consumer Adoption and Usage of Value-Added Mobile Services*, Promotors: Prof.dr. Ph.H.B.F. Franses & Prof.dr. P.C. Verhoef, EPS-2008-128-MKT, ISBN: 978/90-5892-161-1, http://hdl.handle.net/1765/12461

Puvanasvari Ratnasingam, P., *Interorganizational Trust in Business to Business E-Commerce*, Promotors: Prof.dr. K. Kumar & Prof.dr. H.G. van Dissel, EPS-2001-009-LIS, ISBN: 90-5892-017-8, http://hdl.handle.net/1765/356

Quak, H.J., *Sustainability of Urban Freight Transport: Retail Distribution and Local Regulation in Cities*, Promotor: Prof.dr. M.B.M. de Koster, EPS-2008-124-LIS, ISBN: 978-90-5892-154-3, http://hdl.handle.net/1765/11990

Quariguasi Frota Neto, J., *Eco-efficient Supply Chains for Electrical and Electronic Products*, Promotors: Prof.dr.ir. J.A.E.E. van Nunen & Prof.dr.ir. H.W.G.M. van Heck, EPS-2008-152-LIS, ISBN: 978-90-5892-192-5, http://hdl.handle.net/1765/14785

Radkevitch, U.L, *Online Reverse Auction for Procurement of Services*, Promotor: Prof.dr.ir. H.W.G.M. van Heck, EPS-2008-137-LIS, ISBN: 978-90-5892-171-0, http://hdl.handle.net/1765/13497

Rinsum, M. van, *Performance Measurement and Managerial Time Orientation*, Promotor: Prof.dr. F.G.H. Hartmann, EPS-2006-088-F&A, ISBN: 90-5892-121-2, http://hdl.handle.net/1765/7993

Roelofsen, E.M., *The Role of Analyst Conference Calls in Capital Markets*, Promotors: Prof.dr. G.M.H. Mertens & Prof.dr. L.G. van der Tas RA, EPS-2010-190-F&A, ISBN: 978-90-5892-228-1, http://hdl.handle.net/1765/1

Romero Morales, D., *Optimization Problems in Supply Chain Management*, Promotors: Prof.dr.ir. J.A.E.E. van Nunen & Dr. H.E. Romeijn, EPS-2000-003-LIS, ISBN: 90-9014078-6, http://hdl.handle.net/1765/865

Roodbergen, K.J., *Layout and Routing Methods for Warehouses*, Promotors: Prof.dr. M.B.M. de Koster & Prof.dr.ir. J.A.E.E. van Nunen, EPS-2001-004-LIS, ISBN: 90-5892-005-4, http://hdl.handle.net/1765/861

Rook, L., *Imitation in Creative Task Performance*, Promotor: Prof.dr. D.L. van Knippenberg, EPS-2008-125-ORG, http://hdl.handle.net/1765/11555

Rosmalen, J. van, *Segmentation and Dimension Reduction: Exploratory and Model-Based Approaches*, Promotor: Prof.dr. P.J.F. Groenen, EPS-2009-165-MKT, ISBN: 978-90-5892-201-4, http://hdl.handle.net/1765/15536

Rus, D., *The Dark Side of Leadership: Exploring the Psychology of Leader Self-serving Behavior*, Promotor: Prof.dr. D.L. van Knippenberg, EPS-2009-178-ORG, http://hdl.handle.net/1765/1

Samii, R., *Leveraging Logistics Partnerships: Lessons from Humanitarian Organizations*, Promotors: Prof.dr.ir. J.A.E.E. van Nunen & Prof.dr.ir. L.N. Van Wassenhove, EPS-2008-153-LIS, ISBN: 978-90-5892-186-4, http://hdl.handle.net/1765/14519

Schaik, D. van, *M&A in Japan: An Analysis of Merger Waves and Hostile Takeovers*, Promotors: Prof.dr. J. Spronk & Prof.dr. J.P.M. Groenewegen, EPS-2008-141-F&A, ISBN: 978-90-5892-169-7, http://hdl.handle.net/1765/13693

Schauten, M.B.J., *Valuation, Capital Structure Decisions and the Cost of Capital*, Promotors: Prof.dr. J. Spronk & Prof.dr. D. van Dijk, EPS-2008-134-F&A, ISBN: 978-90-5892-172-7, http://hdl.handle.net/1765/13480

Schramade, W.L.J., *Corporate Bonds Issuers*, Promotor: Prof.dr. A. De Jong, EPS-2006-092-F&A, ISBN: 90-5892-125-5, http://hdl.handle.net/1765/8191

Schweizer, T.S., *An Individual Psychology of Novelty-Seeking, Creativity and Innovation*, Promotor: Prof.dr. R.J.M. van Tulder, EPS-2004-048-ORG, ISBN: 90-5892-077-1, http://hdl.handle.net/1765/1818

Six, F.E., *Trust and Trouble: Building Interpersonal Trust Within Organizations*, Promotors: Prof.dr. B. Nooteboom & Prof.dr. A.M. Sorge, EPS-2004-040-ORG, ISBN: 90-5892-064-X, http://hdl.handle.net/1765/1271

Slager, A.M.H., *Banking across Borders*, Promotors: Prof.dr. R.J.M. van Tulder & Prof.dr. D.M.N. van Wensveen, EPS-2004-041-ORG, ISBN: 90-5892-066–6, http://hdl.handle.net/1765/1301

Sloot, L., *Understanding Consumer Reactions to Assortment Unavailability*, Promotors: Prof.dr. H.R. Commandeur, Prof.dr. E. Peelen & Prof.dr. P.C. Verhoef, EPS-2006-074-MKT, ISBN: 90-5892-102-6, http://hdl.handle.net/1765/7438

Smit, W., *Market Information Sharing in Channel Relationships: Its Nature, Antecedents and Consequences*, Promotors: Prof.dr.ir. G.H. van Bruggen & Prof.dr.ir. B. Wierenga, EPS-2006-076-MKT, ISBN: 90-5892-106-9, http://hdl.handle.net/1765/7327

Sonnenberg, M., *The Signalling Effect of HRM on Psychological Contracts of Employees: A Multi-level Perspective*, Promotor: Prof.dr. J. Paauwe, EPS-2006-086-ORG, ISBN: 90-5892-119-0, http://hdl.handle.net/1765/7995

Speklé, R.F., *Beyond Generics: A closer Look at Hybrid and Hierarchical Governance*, Promotor: Prof.dr. M.A. van Hoepen RA, EPS-2001-008-F&A, ISBN: 90-5892-011-9, http://hdl.handle.net/1765/357

Srour, F.J., *Dissecting Drayage: An Examination of Structure, Information, and Control in Drayage Operations*, Promotor: Prof.dr. S.L. van de Velde, EPS-2010-186-LIS, http://hdl.handle.net/1765/1

Stam, D.A., *Managing Dreams and Ambitions: A Psychological Analysis of Vision Communication*, Promotor: Prof.dr. D.L. van Knippenberg, EPS-2008-149-ORG, http://hdl.handle.net/1765/14080

Stienstra, M., *Strategic Renewal in Regulatory Environments: How Inter- and Intra-organisational Institutional Forces Influence European Energy Incumbent Firms*, Promotors: Prof.dr.ir. F.A.J. Van den Bosch & Prof.dr. H.W. Volberda, EPS-2008-145-STR, ISBN: 978-90-5892-184-0, http://hdl.handle.net/1765/13943

Sweldens, S.T.L.R., *Evaluative Conditioning 2.0: Direct versus Associative Transfer of Affect to Brands*, Promotor: Prof.dr. S.M.J. van Osselaer, EPS-2009-167-MKT, ISBN: 978-90-5892-204-5, http://hdl.handle.net/1765/1

Szkudlarek, B.A., *Spinning the Web of Reentry: [Re]connecting reentry training theory and practice*, Promotor: Prof.dr. S.J. Magala, EPS-2008-143-ORG, ISBN: 978-90-5892-177-2, http://hdl.handle.net/1765/13695

Teunter, L.H., *Analysis of Sales Promotion Effects on Household Purchase Behavior*, Promotors: Prof.dr.ir. B. Wierenga & Prof.dr. T. Kloek, EPS-2002-016-MKT, ISBN: 90-5892-029-1, http://hdl.handle.net/1765/868

Tims, B., *Empirical Studies on Exchange Rate Puzzles and Volatility*, Promotor: Prof.dr. C.G. Koedijk, EPS-2006-089-F&A, ISBN: 90-5892-113-1, http://hdl.handle.net/1765/8066

Tuk, M.A., *Is Friendship Silent When Money Talks? How Consumers Respond to Word-of-Mouth Marketing*, Promotors: Prof.dr.ir. A. Smidts & Prof.dr. D.H.J. Wigboldus, EPS-2008-130-MKT, ISBN: 978-90-5892-164-2, http://hdl.handle.net/1765/12702

Valck, K. de, *Virtual Communities of Consumption: Networks of Consumer Knowledge and Companionship*, Promotors: Prof.dr.ir. G.H. van Bruggen & Prof.dr.ir. B. Wierenga, EPS-2005-050-MKT, ISBN: 90-5892-078-X, http://hdl.handle.net/1765/6663

Valk, W. van der, *Buyer-Seller Interaction Patterns During Ongoing Service Exchange*, Promotors: Prof.dr. J.Y.F. Wynstra & Prof.dr.ir. B. Axelsson, EPS-2007-116-MKT, ISBN: 90-5892-146-8, http://hdl.handle.net/1765/10856

Verheul, I., *Is There a (Fe)male Approach? Understanding Gender Differences in Entrepreneurship*, Prof.dr. A.R. Thurik, EPS-2005-054-ORG, ISBN: 90-5892-080-1, http://hdl.handle.net/1765/2005

Verwijmeren, P., *Empirical Essays on Debt, Equity, and Convertible Securities*, Promotors: Prof.dr. A. de Jong & Prof.dr. M.J.C.M. Verbeek, EPS-2009-154-F&A, ISBN: 978-90-5892-187-1, http://hdl.handle.net/1765/14312

Vis, I.F.A., *Planning and Control Concepts for Material Handling Systems*, Promotors: Prof.dr. M.B.M. de Koster & Prof.dr.ir. R. Dekker, EPS-2002-014-LIS, ISBN: 90-5892-021-6, http://hdl.handle.net/1765/866

Vlaar, P.W.L., *Making Sense of Formalization in Interorganizational Relationships: Beyond Coordination and Control*, Promotors: Prof.dr.ir. F.A.J. Van den Bosch & Prof.dr. H.W. Volberda, EPS-2006-075-STR, ISBN 90-5892-103-4, http://hdl.handle.net/1765/7326

Vliet, P. van, *Downside Risk and Empirical Asset Pricing*, Promotor: Prof.dr. G.T. Post, EPS-2004-049-F&A, ISBN: 90-5892-07-55, http://hdl.handle.net/1765/1819

Vlist, P. van der, *Synchronizing the Retail Supply Chain*, Promotors: Prof.dr.ir. J.A.E.E. van Nunen & Prof.dr. A.G. de Kok, EPS-2007-110-LIS, ISBN: 90-5892-142-0, http://hdl.handle.net/1765/10418

Vries-van Ketel E. de, *How Assortment Variety Affects Assortment Attractiveness: A Consumer Perspective*, Promotors: Prof.dr. G.H. van Bruggen & Prof.dr.ir. A. Smidts, EPS-2006-072-MKT, ISBN: 90-5892-101-8, http://hdl.handle.net/1765/7193

Vromans, M.J.C.M., *Reliability of Railway Systems*, Promotors: Prof.dr. L.G. Kroon, Prof.dr.ir. R. Dekker & Prof.dr.ir. J.A.E.E. van Nunen, EPS-2005-062-LIS, ISBN: 90-5892-089-5, http://hdl.handle.net/1765/6773

Vroomen, B.L.K., *The Effects of the Internet, Recommendation Quality and Decision Strategies on Consumer Choice*, Promotor: Prof.dr. Ph.H.B.F. Franses, EPS-2006-090-MKT, ISBN: 90-5892-122-0, http://hdl.handle.net/1765/8067

Waal, T. de, *Processing of Erroneous and Unsafe Data*, Promotor: Prof.dr.ir. R. Dekker, EPS-2003-024-LIS, ISBN: 90-5892-045-3, http://hdl.handle.net/1765/870

Waard, E.J. de, *Engaging Environmental Turbulence: Organizational Determinants for Repetitive Quick and Adequate Responses*, Promotors: Prof.dr. H.W. Volberda & Prof.dr. J. Soeters, EPS-2010-189-STR, ISBN: 978-90-5892-229-8, http://hdl.handle.net/1765/1

Wall, R.S., *Netscape: Cities and Global Corporate Networks*, Promotor: Prof.dr. G.A. van der Knaap, EPS-2009-169-ORG, ISBN: 978-90-5892-207-6, http://hdl.handle.net/1765/1

Watkins Fassler, K., *Macroeconomic Crisis and Firm Performance*, Promotors: Prof.dr. J. Spronk & Prof.dr. D.J. van Dijk, EPS-2007-103-F&A, ISBN: 90-5892-138-3, http://hdl.handle.net/1765/10065

Weerdt, N.P. van der, *Organizational Flexibility for Hypercompetitive Markets: Empirical Evidence of the Composition and Context Specificity of Dynamic Capabilities and Organization Design Parameters*, Promotor: Prof.dr. H.W. Volberda, EPS-2009-173-STR, ISBN: 978-90-5892-215-1, http://hdl.handle.net/1765/1

Wennekers, A.R.M., *Entrepreneurship at Country Level: Economic and Non-Economic Determinants*, Promotor: Prof.dr. A.R. Thurik, EPS-2006-81-ORG, ISBN: 90-5892-115-8, http://hdl.handle.net/1765/7982

Wielemaker, M.W., *Managing Initiatives: A Synthesis of the Conditioning and Knowledge-Creating View*, Promotors: Prof.dr. H.W. Volberda & Prof.dr. C.W.F. Baden-Fuller, EPS-2003-28-STR, ISBN: 90-5892-050-X, http://hdl.handle.net/1765/1042

Wijk, R.A.J.L. van, *Organizing Knowledge in Internal Networks: A Multilevel Study*, Promotor: Prof.dr.ir. F.A.J. van den Bosch, EPS-2003-021-STR, ISBN: 90-5892-039-9, http://hdl.handle.net/1765/347

Wolters, M.J.J., *The Business of Modularity and the Modularity of Business*, Promotors: Prof. mr. dr. P.H.M. Vervest & Prof.dr.ir. H.W.G.M. van Heck, EPS-2002-011-LIS, ISBN: 90-5892-020-8, http://hdl.handle.net/1765/920

Wubben, M.J.J., *Social Functions of Emotions in Social Dilemmas*, Promotors: Prof.dr. D. De Cremer & Prof.dr. E. van Dijk, EPS-2009-187-ORG, http://hdl.handle.net/1765/1

Xu, Y., *Empirical Essays on the Stock Returns, Risk Management, and Liquidity Creation of Banks*, Promotor: Prof.dr. M.J.C.M. Verbeek, EPS-2010-188-F&A, http://hdl.handle.net/1765/1

Yang, J., *Towards the Restructuring and Co-ordination Mechanisms for the Architecture of Chinese Transport Logistics*, Promotor: Prof.dr. H.E. Harlambides, EPS-2009-157-LIS, ISBN: 978-90-5892-198-7, http://hdl.handle.net/1765/14527

Yu, M., *Enhancing Warehouse Perfromance by Efficient Order Picking*, Promotor: Prof.dr. M.B.M. de Koster, EPS-2008-139-LIS, ISBN: 978-90-5892-167-3, http://hdl.handle.net/1765/13691

Zhang, X., *Strategizing of Foreign Firms in China: An Institution-based Perspective*, Promotor: Prof.dr. B. Krug, EPS-2007-114-ORG, ISBN: 90-5892-147-5, http://hdl.handle.net/1765/10721

Zhu, Z., *Essays on China's Tax System*, Promotors: Prof.dr. B. Krug & Prof.dr. G.W.J. Hendrikse, EPS-2007-112-ORG, ISBN: 90-5892-144-4, http://hdl.handle.net/1765/10502

Zwart, G.J. de, *Empirical Studies on Financial Markets: Private Equity, Corporate Bonds and Emerging Markets*, Promotors: Prof.dr. M.J.C.M. Verbeek & Prof.dr. D.J.C. van Dijk, EPS-2008-131-F&A, ISBN: 978-90-5892-163-5, http://hdl.handle.net/1765/12703

Erasmus Research Institute of Management - ERIM

ADVANCES IN ONLINE SHOPPING INTERFACES

PRODUCT CATALOG MAPS AND RECOMMENDER SYSTEMS

Over the past two decades the internet has rapidly become an important medium to retrieve information, maintain social contacts, and to do online shopping. The latter has some important advantages over traditional shopping. Products are often cheaper on the internet, internet companies sell a wider collection of products and consumers can buy items whenever they like without leaving their homes. On the other hand, the current state of online shops still has two major disadvantages over 'real' shops: Products are often much harder to find than in traditional shops and there are no salesmen to advise the customers.

In this thesis, we address both these disadvantages. We introduce and evaluate several new user interfaces for online shops that are based on representing products in maps instead of lists to user, such that products are easier to find. In these maps similar products are located close to each other. To create these maps, statistical techniques such as multi-dimensional scaling are used. Furthermore, we combine these maps with recommender systems to address the second disadvantage and to help the user in finding the product best suiting her needs. Also, we introduce a recommender system that is able to explain the recommendations it gives to users. We think that the methods discussed in this thesis can form a basis for new promising online shopping interfaces both in research as in practice.

ERIM

The Erasmus Research Institute of Management (ERIM) is the Research School (Onderzoekschool) in the field of management of the Erasmus University Rotterdam. The founding participants of ERIM are Rotterdam School of Management (RSM), and the Erasmus School of Economics (ESE). ERIM was founded in 1999 and is officially accredited by the Royal Netherlands Academy of Arts and Sciences (KNAW). The research undertaken by ERIM is focused on the management of the firm in its environment, its intra- and interfirm relations, and its business processes in their interdependent connections.

The objective of ERIM is to carry out first rate research in management, and to offer an advanced doctoral programme in Research in Management. Within ERIM, over three hundred senior researchers and PhD candidates are active in the different research programmes. From a variety of academic backgrounds and expertises, the ERIM community is united in striving for excellence and working at the forefront of creating new business knowledge.

Design & layout: B&T Ontwerp en advies (www.b-en-t.nl) Print: Haveka (www.haveka.nl)

ERIM PhD Series

Research in Management

Erasmus Research Institute of Management - **ERIM**
Rotterdam School of Management (RSM)
Erasmus School of Economics (ESE)
P.O. Box 1738, 3000 DR Rotterdam
The Netherlands

Tel. +31 10 408 11 82
Fax +31 10 408 96 40
E-mail info@erim.eur.nl
Internet www.erim.eur.nl