Estimating Optimal Weights in Hybrid Recommender Systems

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften

(Dr.-Ing.)

von der KIT-Fakultät für
Wirtschaftswissenschaften
am Karlsruher Institut für Technologie (KIT)

genehmigte

Dissertation

von

M.Sc. Inform.-Wirt. Nicolas Haubner

Referent: Prof. Dr. Thomas Setzer
Korreferent: Prof. Dr. York Sure-Vetter
Tag der mündlichen Prüfung: 26. März 2021

Karlsruhe, 2021

Contents

List of Figures iii

List of Tables v

List of Abbreviations vii

List of Symbols viii

I. Introduction and Foundations 1

1. Introduction 3
- 1.1. Motivation 3
- 1.2. Research Questions 5
- 1.3. Structure of this Thesis 9

2. Foundations 13
- 2.1. Recommender Systems 13
  - 2.1.1. Recommender Systems Paradigms 17
  - 2.1.2. Weighted Hybrid Recommender Systems 26
- 2.2. Robust Combination of Forecasts 28
  - 2.2.1. Statistical Forecast Combination 28
  - 2.2.2. The Bias-Variance Trade-Off in Forecast Combination 30

II. Weighting Schemes for Hybrid Recommender Systems 33

3. Applying Optimal Weight Combination in Hybrid Recommender Systems 35
- 3.1. Motivation and Related Work 35
- 3.2. Methodology 36
- 3.3. Experimental Design 38
  - 3.3.1. Data Set 38
  - 3.3.2. Individual Recommender Systems 39
  - 3.3.3. Evaluation and Benchmarks 41
- 3.4. Results and Discussion 42
  - 3.4.1. Data Preparation 42

3.4.2. Experimental Results 45
3.4.3. Results with Fewer Observations 52
3.5. Conclusion 57

4. Projection of Optimal Weights in Hybrid Recommender Systems 59
4.1. Motivation and Related Work 60
4.2. Methodology 61
4.2.1. Model Assumptions 62
4.2.2. Projection Methodology 62
4.2.3. Improvement Potential 65
4.3. Experimental Design 69
4.4. Empirical Evaluation 70
4.4.1. Data Preparation 71
4.4.2. Experimental Results 71
4.5. Discussion 78
4.6. Conclusion 79

5. Hybrid Recommender Systems for Next Purchase Prediction Based on Optimal Combination Weights 81
5.1. Motivation and Related Work 82
5.2. Methodology 84
5.2.1. Model Assumptions 84
5.2.2. Classifier Weighting Method 84
5.3. Experimental Design 87
5.3.1. Use Case and Data Set 87
5.3.2. Individual Classifiers 88
5.3.3. Evaluation and Benchmarks 90
5.4. Empirical Evaluation 91
5.4.1. Data Preparation 91
5.4.2. Results 92
5.5. Discussion 95
5.6. Conclusion 97

III. Finale 99

6. Conclusion 101
6.1. Summary of Contributions 101
6.2. Future Research 105

Bibliography 107

List of Figures

2.1. Example for the user-item rating matrix Y, containing the first ten users (rows) and items (columns) of the MovieLens 1M data set. . . . 15

2.2. Boxplot of the mean rating (left) and the standard deviation of ratings (right) per user in the MovieLens 1M data set. 20

3.1. Distribution of ratings in the MovieLens 1M data set. 39

3.2. Histogram of the number of ratings per user in the MovieLens 1M data set. 40

3.3. Histogram of the number of ratings per movie in the MovieLens 1M data set. 40

3.4. Histograms of the individual recommender systems' errors. 44

3.5. Comparison of optimal weight (OW) estimate, simple average (SA), individual recommender systems, and modified OW ignoring error correlation for pairwise combinations of recommender systems on the MovieLens 1M data set. 51

3.6. Display analogous to Figure 3.5 for 50% of the training data. Results are similar to the full data set, but the ex-post out-of-sample best weights are closer to the simple average (SA). 55

3.7. Display analogous to Figure 3.5 for 10% of the ratings. The smaller number of observations increases estimation uncertainty for the optimal weight (OW) estimate. 56

4.1. Pseudocode for projection of optimal weights. 63

4.2. Example of the projection approach. In this case, 25% of the available data is used as a holdout set and initial training set, respectively ($s_h = s_t^0 = \frac{s}{4}$). $o = 3$ iterations are run, and finally OW are projected to all s observations. 64

4.3. Sensitivity of the optimal weight to changes (β) in the root mean squared errors of the individual recommender systems, depending on the initial ratio of RMSEs (α) and the error correlation ρ. 66

4.4. Improvement of the root mean squared error (RMSE) of a hybrid recommender system with perfect extrapolation of the optimal weight (OW), depending on the error correlation ρ and the deviation Δw of OW between the training set and the entire data set. 68

4.5. Development of the root mean squared error (RMSE) of the individual recommender systems depending on the number of training observations s_t (average over ten runs with random allocation of $s_t^{max} \approx 800,000$ observations for training and $s_p \approx 200,000$ for evaluation). 72
4.6. Comparison of the weight estimation approaches in terms of their difference in root mean squared error (RMSE) to the ex-post optimal weight (OW). The values are displayed as a percentage of the difference between the RMSE of the naive forecast ($RMSE_{naive}$) and the ex-post OW ($RMSE_{best}$). 74
4.7. Results for the projection of the optimal weights (OW) estimated for each recommender system. The black curves represent actual OW, the gray curves show the predictions. The vertical dashed line displays s_t^{max}, i.e. predictions beyond this point are out-of-sample. 76
4.8. Results for the projection of the error standard deviation of each recommender system. The lines are analogous to Figure 4.7. 77
4.9. Results for the projection of the error correlation of each pairwise combination of recommender systems. The lines are analogous to Figure 4.7. 78

5.1. Example for prediction vectors and actual outcomes with $s = 3$ observations, $m = 3$ classes, and $k = 2$ classification models. 85
5.2. Example for error matrix based on the predictions in Figure 5.1. . . . 86
5.3. Schematic display of the real-world data set used for evaluation. The proprietary data set stems from a large European telecommunications provider. 87
5.4. Histograms of all individual classifiers' out-of-sample errors using ten equal-width bins. The vertical axes are not labeled in order not to disclose the confidential number of observations in the proprietary data set. 92
5.5. Histograms of all individual classifiers' out-of-sample errors using ten equal-width bins after applying isotonic calibration. The vertical axes are not labeled in order not to disclose the confidential number of observations in the proprietary data set. 93

List of Tables

2.1. Advantages and disadvantages of neighborhood-based collaborative filtering, based on Ricci et al. (2015). Disadvantages with an asterisk (*) are partially mitigated by using item-based instead of user-based NCF. 21

2.2. Advantages and disadvantages of model-based collaborative filtering, based on Ricci et al. (2015). 23

2.3. Advantages and disadvantages of content-based filtering, based on Ricci et al. (2015). 25

2.4. Advantages and disadvantages of demographic filtering, based on Ricci et al. (2015). 26

3.1. Mean errors of the individual recommender systems, together with the standard error of the mean. The mean errors deviate slightly from zero. 43

3.2. Comparison of the root mean squared error of a hybrid recommender system using an optimal weight estimate with and without a Box-Cox transformation. The transformation does not lead to a smaller error. . 44

3.3. Aggregated predictive results of the individual recommender systems and their pairwise combination using the optimal weight (OW) estimate, the modified OW estimate, and a simple average (SA) on the MovieLens 1M data set. 46

3.4. Results analogous to Table 3.3 for 50% of the training data. Principle relationships are similar to the full data set, but the differences are smaller. 53

3.5. Results analogous to Table 3.3 for 10% of the ratings. OW still dominates SA and the modified OW overall, but the gap is smaller due to the higher model variance. 54

4.1. Comparison between root mean squared errors (RMSE) of weighted hybrid recommender systems (RS) using the weighting strategies simple average, naive forecast (weights learned on a training subset), projection of the error covariance, and direct projection of optimal weights. For reference, the RMSEs of the individual RSs are also included. 72

4.2. Comparison between mean absolute error (MAE) of the naive forecast, the covariance projection, and the direct projection in estimating the error standard deviations σ (where applicable), the error correlations ρ (where applicable), and the optimal weights w assigned to each individual recommender system. The experiment was run ten times with random training-holdout-test splits, and all values are averages over those runs, while the standard deviations are displayed in parentheses. 75

5.1. Comparison of performance between a hybrid recommender system using optimal weight (OW) estimation, a simple average (SA) combination, and all individual classifiers. 94

5.2. Results analogous to Table 5.1 after applying isotonic calibration to all individual classifiers. 95

List of Abbreviations

CB	content-based filtering
CF	collaborative filtering
DF	demographic filtering
HRS	hybrid recommender system
MAE	mean absolute error
MCF	model-based collaborative filtering
MSE	mean squared error
NCF	neighborhood-based collaborative filtering
OW	optimal weights
RMSE	root mean squared error
RS	recommender system
SA	simple average
WHRS	weighted hybrid recommender system

List of Symbols

y_{ui}	User u's rating (purchase / interaction) for item i
$\hat{y}_{ui}$	Predicted rating by an RS if y_{ui} is unknown
Y	User-item rating matrix
n	Number of users (rows in Y)
m	Number of items / products (columns in Y)
$\mathbf{y}$	Vector of all known ratings in Y (flattened Y)
s	Number of ratings (observations) in $\mathbf{y}$
Y_t	Matrix of ratings used for fitting individual models (training set)
$\mathbf{y}_t$	Flattened Y_t
s_t	Number of ratings in $\mathbf{y}_t$
Y_h	Matrix of ratings used for estimating out-of-sample errors for deriving OW (holdout set)
$\mathbf{y}_h$	Flattened Y_h
s_h	Number of ratings in $\mathbf{y}_h$
Y_p	Matrix of ratings used for evaluation (test set)
$\mathbf{y}_p$	Flattened Y_p
s_p	Number of ratings in $\mathbf{y}_p$
U	Matrix of user metadata
p_U	Number of attributes (columns) in the user metadata U
I	Matrix of item metadata
p_I	Number of attributes (columns) in the item metadata I
k	Number of RSs combined in the WHRS
$\vec{\mathbf{1}}$	k-dimensional vector of ones
$\hat{\mathbf{y}}_l$	RS l's vector of predictions for ratings in $\mathbf{y}$, $l \in \{1, \ldots, k\}$
$\mathbf{e}_l$	Error vector of RS l with $\mathbf{e}_l = \mathbf{y} - \hat{\mathbf{y}}_l, l \in \{1, \ldots, k\}$
E	Error matrix with $E = (\mathbf{e}_1, \ldots, \mathbf{e}_k)$
Σ_E	Variance-covariance matrix of E
ρ_{lq}	Pearson correlation between $\mathbf{e}_l$ and $\mathbf{e}_q$, $l, q \in \{1, \ldots, k\}$ (for $k = 2$, we simply write ρ)
σ_l	Standard deviation of RS l' error vector, $l \in \{1, \ldots, k\}$
$\mathbf{w}$	Weight vector, $\mathbf{w} \in \mathbb{R}^k$, $\sum_{l=1}^{k} w_l = 1$
w_l	Weight assigned to RS $l \in \{1, \ldots, k\}$
w	For brevity, when $k = 2$, we define $\mathbf{w} = (w, 1 - w)'$
Δw	Change of OW from s_t to s observations: $\Delta w = w_{OW}(s) - w_{OW}(s_t)$
α	Ratio of error standard deviations of two RSs: $\alpha(s_t) = \frac{\sigma_1(s_t)}{\sigma_2(s_t)}$
β	Ratio of α for s to s_t training observations: $\beta(s_t, s) = \frac{\alpha(s)}{\alpha(s_t)}$
o	Number of steps (iterations) in weight projection
s_t^0	Initial number of training observations in weight projection
s_t^{max}	Maximum number of training observations in weight projection

Part I.

Introduction and Foundations

Chapter 1.

Introduction

1.1. Motivation

The information age has brought with it a vast amount of possibilities, be it when shopping online, finding movies to watch, or exploring new music. Often, users are overwhelmed by the sheer amount of options which complicate decision making, even for simple products. Advances in technology have made it possible to collect and process increasing amounts of data, such as customer profiles, activities and interests. Turning this data into actionable insights is not only key to acquiring and retaining customers, but also to providing suitable purchasing recommendations for up- and cross-selling items relevant to and appreciated by existing customers in order to increase customer lifetime values.

Recommender systems (RS) are an integral part of today's digital marketplaces and are used in a wide range of areas, e.g. e-commerce, personalized search engines, banner advertising, music and video streaming, job search, multi-sided matching platforms, social networks, or news article recommendation (Zhang et al., 2011; Malgonde et al., 2020). Some websites are even personalized based on the visitor, not only in terms of content, but also design and user experience. The advantages of using RSs as a decision aid have been shown extensively: For users, RSs are important because they help explore the space of items in order to find new and interesting products, reduce information overload by limiting the number of options, and thus improve users' decision quality (Xu et al., 2014). For companies, they help increase sales, diversify sales by advancing long-tail items which would otherwise be hard to find, and gain an understanding of their customers' needs (Oestreicher-Singer and Sundararajan, 2012). Increasing customer satisfaction by means of useful RSs is beneficial for customer loyalty, either through renewing subscriptions or repeating sales. There is a variety of algorithms for generating recommendations, using different paradigms and input data, each with individual strengths and weaknesses.

The quality of an RS can be measured by a number of different criteria. The most important goals are (e.g. Adomavicius and Tuzhilin, 2005):

- *Accuracy*: The RS should recommend items the given user actually finds interesting, i.e. the predicted rating should be as close as possible to the eventual true rating, or the eventually purchased product should be as high on the recommendation list as possible.
- *Serendipity*: A good RS should be able to positively surprise users, i.e. suggest items which they were not aware of but are interested in.
- *Diversity*: The items in the recommendation list should not be too similar to each other (e.g. the same smartphone in three different colors, or all three movies of a trilogy). Instead, they should be able to reproduce different aspects of the given user's interest.

The focus of this thesis lies on the first criterion. Research has shown that the predictive accuracy of recommendations, i.e. the perceived personalization, is of key importance for customers to adopt an RS as a decision aid, and thus, to purchase recommended items (Komiak and Benbasat, 2006). More accurate RSs increase decision quality and also help companies retain customers (Zhang et al., 2011).

Increasing the predictive accuracy of an RS can be achieved in several ways. One could (a) collect more input data or use additional data sources, (b) improve or extend the algorithm which is applied in the system, (c) tune the hyper-parameters of the algorithm, such as regularization parameters, or (d) use a different, more accurate algorithm which might be computationally more expensive or less intuitive and interpretable.

In addition to and independent of the former approaches, another frequently-used means of increasing the accuracy and robustness of RSs, and the focus of this thesis, is the combination of multiple different prediction algorithms. It has been shown that the combination of two or more RS approaches into a hybrid RS (HRS) can improve predictive accuracy. This is well in line with findings when combining statistical forecasts or human judges (social forecasts), where various studies have shown that combining statistical forecasts can improve forecast accuracy (e.g. Bates and Granger, 1969; Clemen, 1989; Makridakis and Hibon, 2000) or judgmental forecasts (e.g. Hill, 1982; Gigone and Hastie, 1997; Mannes et al., 2014). HRSs have been shown to increase decision quality and satisfaction with the system, compared to using only single recommendation methods such as collaborative or content-based approaches (Xiao and Benbasat, 2007).

Different approaches to combine RSs into a hybrid have been proposed. Burke (2002) classifies them into weighted, switching, mixed, feature combination, cascade, feature augmentation, and meta-level. This thesis concentrates on the first mentioned hybridization technique, weighted HRSs (WHRS), where multiple different RS algorithms are run independently and their predictions are combined using a weighted average or other aggregation functions. While WHRSs have often been shown to increase accuracy and robustness over their individual components, there is little prior research on the selection of the weight vector for maximizing the combination's accuracy. Chapter 2 describes this research gap in more detail, based on a review of the existing literature.

Based on the importance of maximizing an RS's accuracy and the lack of research on the selection of weights in HRSs, this thesis proposes and evaluates analytical weighting strategies with the goal of maximizing the accuracy and robustness of the WHRS, both for predicting ratings, such as in a movie streaming site, as well as recommending products to purchase, such as in an e-commerce context. The introduced weighting approaches build on as well as extend techniques from the area of statistical forecast combination. The foundations of both HRSs and forecast combination are reviewed in Chapter 2.

The following section formulates the research questions which are investigated in this thesis.

1.2. Research Questions

The goal of this thesis is the development of robust analytical weighting schemes for WHRSs, both for regression settings, e.g. rating predictions, and classification settings, e.g. purchase predictions. Therefore, three research questions are addressed in the thesis. In this section, all research questions are formulated and motivated based on existing approaches and their limitations. Note that a more detailed review of prior research is provided in Chapter 2.

The combination of multiple RS algorithms by aggregating their predictions using a weighted average has been shown to improve accuracy and robustness over the individual methods. However, little research has been dedicated to the optimization of the accuracy of a WHRS by selection of the weight vector. Existing research in WHRSs mostly uses equal weights or compares a limited number of weights manually.

In the statistical forecasting literature, a large body of research exists on the combination of forecasts and the estimation of OW. Methods have been developed which can be shown to minimize the mean squared error (MSE) of a combined forecast in-sample, given certain assumptions. The weights are calculated based on the variance-covariance matrix of the individual forecasts' error vectors. However, those methods are often outperformed out-of-sample by more robust weighting strategies such as a simple average (SA), where all forecasts are assigned equal weights. This phenomenon is called the "forecast combination puzzle". Apart from applying an SA, the usage of a weighting scheme which ignores the error correlations between forecasts, i.e. assumes correlations of 0, has been reported to lead to lower out-of-sample errors than estimating OW from the covariance matrix as-is.

The first research question, which is investigated in Chapter 3, therefore deals with the development of a novel weighting scheme for WHRSs based on OW estimation in statistical forecast combination:

RQ 1 *Development of an analytical weighting scheme for the pairwise combination of RS algorithms*

(a) *Given the assumptions of OW, does a weighting scheme for a WHRS with two individual RSs based on OW estimation lead to lower errors than each individual RS as well as an SA combination on an RS benchmark data set?*

(b) *Can the weighting scheme based on OW estimation lead to lower errors than the more robust modified OW estimate ignoring error correlations on the benchmark data set?*

(c) *Does the forecast combination puzzle, i.e. OW being outperformed by more robust weighting schemes, occur in the combination of RSs on the benchmark data set, and if so, for which number of ratings?*

However, there are often more than two RS algorithms, each with its own benefits and shortcomings, which can be combined. E.g., collaborative filtering (CF) is a well-established RS paradigm which estimates future ratings by a given user from past ratings by the same as well as other users. Still, a lot of times there is more meaningful input data which can be used to generate recommendations, but CF cannot trivially exploit that input data. Content-based filtering (CB) is able to compute predictions based on the recommended items' metadata and their similarities, in combination with the user ratings. Demographic filtering (DF), on the other hand, processes metadata about the users of the system, such as demographic variables, and their similarities. Together with users' past ratings, it generates recommendations for future interests. There are more RS paradigms, and for each of them, there exist potentially multiple algorithms using different statistical models. Therefore, the combination of more than two RS algorithms bears the potential of boosting accuracy and robustness to a significantly higher level than a pairwise combination.

In addition, conventional methods for learning OW usually split the available data into a training and a holdout set or apply cross-validation in order to estimate out-of-sample errors required for computing the error covariance matrix. After OW have been estimated, all individual algorithms are re-fitted on the entire available data set in order to maximize their performance. Their predictions on new, unseen data are then combined using the OW estimated when the models had been fitted on a subset of the data. Typically, an RS (or any statistical or machine learning model) increases in accuracy with the number of training observations it is fitted on. Since OW directly depends on the individual models' errors and their correlations, which change when switching from the subset to the entire data set, OW estimated this way can be biased.

In Chapter 4, a novel weight projection approach is proposed which can be applied to more than two RSs at a time and mitigates the aforementioned bias in estimating OW by measuring error covariances and OW for increasing numbers of training observations and extrapolating them to the full size of the available data set. The research question addressed in that chapter is the following:

RQ 2 *Development of an extended weight learning approach for WHRSs using projection*

(a) *How does the accuracy of an OW combination on the benchmark data set develop in relation to the individual RSs when the number of RSs in the WHRS is increased from 2 to 4?*

(b) *Assuming perfect extrapolation, what is the performance improvement potential of a weight projection approach over a conventional OW estimate?*

(c) *Does the projection of error covariances or OW lead to a lower error than the conventional OW estimate on the benchmark data set?*

The first two research questions focus on the combination of RSs in a regression context. Typically, this is used in an explicit feedback setting when user ratings, e.g. for unseen movies, are to be predicted and those items with the highest predicted ratings are recommended. However, another important RS task, especially in e-commerce, is the prediction of purchase probabilities for recommending fitting products when there is no explicit feedback, but only past purchases, i.e. implicit feedback, available. Especially when the input data for a given user does not contain past purchases for that same user, but only a user profile consisting of different attributes such as demographic and behavioral variables, classification algorithms from machine learning can be used to recommend to a user products which similar users bought, i.e. the classifiers serve as DF algorithms.

While for regression settings, there exist, albeit few, approaches for selecting the weight vector in a WHRS, there is even less prior research for the combination of classifiers into a WHRS. Therefore, in Chapter 5, a weight learning method for classifiers based on OW is introduced. The approach combines the probability estimates by the individual classifiers using a weighted average. It aims at minimizing the Brier score, the classification equivalent of the MSE, in order to maximize the accuracy score of the WHRS. It is also investigated whether the calibration of probabilty estimates before the combination step improves the weighting approach. The research question tackled in that chapter, and the third and final one of this thesis, is therefore the following:

RQ 3 *Development of a weight learning approach for WHRSs in a classification setting*

(a) *Can the estimation of OW for the combination of regressors into a WHRS be adapted to be used for the combination of classifiers?*

(b) *Does the minimization of the Brier score of a WHRS comprised of classifying algorithms lead to an increase of the accuracy score over all individual classifiers as well as an SA combination?*

(c) *Does the calibration of the classifiers' probability estimates lead to an improvement in estimating OW in a WHRS?*

Based on the derived research questions, this thesis makes contributions to the literature on the selection of weight vectors in WHRSs as well as the general literature on the estimation of OW for the combination of models in both regression and classification.

1.3. Structure of this Thesis

In order to introduce the developed weighting strategies for WHRSs and embed them into the existing literature, this thesis is structured into three parts. The first part motivates the need for the methods proposed in this thesis based on existing approaches and their shortcomings. The current Chapter 1 discusses the importance of accurate RSs and the role that WHRSs play in maximizing accuracy. Based on the lack of prior work on the selection of weights in WHRSs, three research questions are established which guide the thesis. Chapter 2 reviews related work and the foundations of the weighting approaches proposed in this thesis. The literature review provides background on the goals and challenges of RSs, prior work on HRSs, and foundations of statistical forecast combination on which the proposed methods are based.

The second part of this thesis deals with the introduction of novel weighting schemes for WHRSs and their evaluation on real-world data sets. In Chapter 3, a method for selecting weights in a WHRS comprised of $k = 2$ individual RSs is proposed. The approach is based on statistical forecast combination and adapts a model by Bates and Granger (1969) to be used in a rating prediction RS. The novel weighting scheme is evaluated on the MovieLens 1M data set (Harper and Konstan, 2015), where it is shown to outperform the individual RSs, an SA combination, as well as a more robust weighting scheme which ignores the correlation of the individual RSs' error vectors. Results on subsets of the data using 50% and 10% of the ratings, respectively, indicate that the forecast combination puzzle starts to appear with a decreasing number of observations.

Chapter 4 introduces a weighting technique which extends the one from Chapter 3 in two aspects: (a) It is able to estimate OW for the combination of $k > 2$ individual RSs, thus making it possible to combine e.g. CF, CB, and DF approaches, and (b) more importantly, it mitigates the problem of biased OW estimates due to the learning of OW on subsets of available data. For that purpose, error covariances and OW are measured and stored for increasing numbers of training observations and finally, the series is extrapolated to the full size of the available data set using a supervised learning model. Like in Chapter 3, the MovieLens 1M data set is used for empirical evaluation of a combination of $k = 4$ individual RSs, namely model-based CF (MCF), neighborhood-based CF (NCF), CB, and DF. Results show that (a) combining more than two individual RSs increases the performance boost of the combination substantially, and even though the OW estimation now has higher degrees of freedom, SA and all individual RSs are still outperformed significantly, and (b) the weight projection leads to even smaller errors compared to the conventional approach of estimating OW on a subset of the available data.

In Chapter 5, a weighting scheme for WHRS in a classification context is proposed. The approach is based on the one from Chapter 3, however it is used to combine the probabilistic predictions of classification models used for next purchase predictions from user profiles. The Brier score is used as a classification equivalent of the MSE and OW are estimated which minimize the Brier score of a WHRS with $k = 7$ classifiers. The approach is evaluated on a real-world data set from a large European telecommunications provider, where it significantly outperforms all individual classifiers as well as an SA combination. The minimization of the Brier score is shown to lead to substantial improvements in the accuracy score of the ensemble.

Finally, the third part concludes and summarizes the contributions of this thesis. Possible directions for interesting future research to extend the methods proposed in the thesis are named.

Extracts of this work have already been published or are currently under review. Chapter 3 is based on Haubner and Setzer (2020) and contains insights, models, evaluations, and textual paragraphs from that publication. Chapter 4 is based on a manuscript currently under review at a scientific journal and contains insights, models, evaluations and textual paragraphs from that manuscript. Chapter 5 is based on Haubner and Setzer (2021) and contains insights, models, evaluations and textual paragraphs from that publication. Parts I and III contain paragraphs from all those publications or manuscripts. They are extended and discussed in more detail.

Chapter 2.

Foundations

This chapter provides a selective introduction of concepts and topics underlying this thesis. As the overall theme is the derivation of accurate and robust weighting schemes for HRSs, the foundations are subdivided into two sections: While Section 2.1 reviews the literature on RSs with a focus on WHRSs, Section 2.2 summarizes statistical forecast combination and its connection to the bias-variance trade-off.

2.1. Recommender Systems

This section reviews the literature in the RS field most relevant to the methods proposed in the thesis. We start by introducing the basics of RSs, i.e. the problem formulation, input and output, and evaluation. Section 2.1.1 describes the most prominent paradigms and algorithms used for deriving recommendations. Finally, Section 2.1.2 delves into the different strategies for combining individual RSs into an HRS, focusing on the main topic of this thesis, i.e. WHRSs.

RSs are software systems which estimate users' interests for products based on past transactions and other inputs, and recommend the items with the highest predicted interest. By limiting the number of options, pre-selecting a set of interesting items, and displaying trade-offs between product properties, they can reduce information overload and the effort of product search as well as improve users' decision quality (Xiao and Benbasat, 2007; Xu et al., 2014). For companies, RSs can increase sales and help market long-tail items which would otherwise be hard to find. RSs are nowadays used by, among others, e-commerce sites, digital marketing systems, social networks, and streaming platforms, where their advantages have been shown extensively (Xiao and Benbasat, 2007; Zhang et al., 2011).

The input to every recommendation algorithm is a set of n users, a set of m items, and a set of s interactions, or feedback, between users and items. There is a distinction between explicit and implicit feedback: Explicit feedback is the one which a user consciously submits to the vendor, e.g. rating a product on a five-star Likert scale or clicking thumbs-up or -down. Implicit feedback, on the other hand, is raised by the vendor through user actions which are not primarily intended as an indication of interest or feedback. Examples for implicit feedback are the purchase of a product, a page view or click, a search query, or the percentage of a movie which a user watched (e.g. Adomavicius and Tuzhilin, 2005).

The interactions between users and items are typically stored in a user-item matrix, which is denoted by

$$Y = \begin{pmatrix} y_{11} & \cdots & y_{1m} \\ \dots & \dots & \dots \\ y_{n1} & \cdots & y_{nm} \end{pmatrix} \in \{1, \dots, 5\}^{(n \times m)}$$

in this thesis. In the matrix Y, each row represents one user, and each column represents one item. For a specific user $u \in \{1, \dots, n\}$ and a specific item $i \in \{1, \dots, m\}$, the matrix entry y_{ui} contains user u's interaction with item i, if any interaction has taken place. Throughout this thesis, unless explicitly stated otherwise, explicit feedback in the form of five-star Likert scale ratings is assumed. Therefore, Y will also be referred to as the rating matrix. Note that not all entries of the matrix are specified, since users have not rated or interacted with all available items in the system. Typically, the user-item matrix Y is very sparse in RS contexts (e.g. Ricci et al., 2015). The missing entries are the ones an RS attempts to predict.

Figure 2.1 displays an example of such a rating matrix. It contains the first ten users (rows) and items (columns) of the MovieLens 1M data set (Harper and Konstan, 2015), a well-known RS benchmark data set which is used for evaluation purposes in Chapters 3 and 4 and will be used throughout the thesis for demonstration purposes. The data set is described in more detail later. When referring to the MovieLens data set, “item” and “movie” will be used synonymously, since the items in this data set are movies.

This example already shows some of the challenges which have to be overcome

$$\begin{array}{r|cccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & 5 & & & & & & & & & \\
2 & & & & & & & & & & \\
3 & & & & & & & & & & \\
4 & & & & & & & & & & \\
5 & & & & & & 2 & & & & \\
6 & 4 & & & & & & & & & \\
7 & & & & & & 4 & & & & \\
8 & 4 & & & 3 & & & & & & \\
9 & 5 & & & & & & & & & \\
10 & 5 & 5 & & & & & 4 & & &
\end{array}$$

Figure 2.1.: Example for the user-item rating matrix Y, containing the first ten users (rows) and items (columns) of the MovieLens 1M data set.

when deploying RSs: First, there are far more unknown, i.e. missing, than known ratings in the matrix. In the figure, they are displayed as empty matrix entries. This is known as *sparsity* and can cause problems for certain RS algorithms, especially those which use neighborhood-based approaches. Second, there are users and items with no ratings. This is only a small sample of the data set. However, assuming that this was the entire data set, those could be new users who just signed up to the system and new movies which were just released, respectively. Many RS algorithms have problems recommending new items or creating recommendations for new users. This is called the *cold-start* problem. Those and some other challenges will be adressed in more detail later.

Certain algorithms make use of additional input data apart from the rating matrix. One such data source is metadata about users and items. E.g., a user can be described by demographic or behavioral properties, and an item can be represented by its price, product group, size, etc. This metadata is also stored in one matrix, respectively. Since the metadata can contain mixed data types such as text, categorical, numerical, ordinal, or binary variables, it needs to be transformed into a numerical representation in order to be used by the algorithms. It is assumed that those transformations have been applied and that the metadata is stored in two matrices containing real numbers: The user metadata is denoted by $U \in \mathbb{R}^{n \times p_U}$, and the item metadata by $I \in \mathbb{R}^{m \times p_I}$, where p_U represents the number of variables in the user metadata after transformation, and p_I analogous for the items.

There are further possible dimensions of input — such as time, location, user-generated tags, or connections between users in a social network — which can be used by some RS algorithms, but those algorithms are not applied in this thesis and therefore the focus lies on Y, U, and I as inputs to an RS.

The outputs of a recommendation algorithm are predictions $\hat{y}_{ui} \in \mathbb{R}$ for a given user u's interest toward item i, where y_{ui} is unknown. For five-star ratings, the predicted rating lies between 1 and 5, but does not have to be an integer. The goal of the RS is not to predict the exact rating, but rather to compare the estimated utility of different items in order to identify the most interesting items. For a given user, unknown items can now be sorted decreasingly by the predicted ratings. Optionally, further modifications can be applied, such as excluding or promoting certain items. Finally, a recommendation list is created which contains e.g. the top one, five, or ten most interesting items for the given user. This list can then be applied for personalized advertisements, e.g. displayed to the user on a website, sent in an e-mail, or used by a call-center agent (e.g. Aggarwal, 2016).

An RS's quality relates to criteria such as serendipity, diversity, and accuracy. Serendipity denotes the ability of an RS to suggest items that a given user was not aware of, but finds interesting. Diversity refers to the composition of recommendations. Instead of suggesting several very similar items, a good RS should be able to cater to the different interests of a given user. Finally, an accurate RS makes recommendations which fit user needs, such that the products are then taken by users with high probability, e.g. a customer ultimately purchases suggested products or watches suggested movies (e.g. Adomavicius and Tuzhilin, 2005). For regression problems such as rating prediction, common quality metrics in RS research are the root mean squared error (RMSE) and the mean absolute error (MAE). For classification problems, such as next purchase prediction, the accuracy score, recall, precision, or mean rank of the eventually purchased product can be used.

Several approaches exist to derive prediction scores from available data, each using different types of input data and applying specific analytical algorithms. The most common paradigms, along with their benefits and shortcomings, are reviewed in the following.

2.1.1. Recommender Systems Paradigms

To make utility predictions, various paradigms exist that use different analytical methods or consider different input data. Amongst the most prominent paradigms are CF, CB, and DF. CF can be further subdivided into NCF and MCF. For each paradigm, there is again a multitude of different algorithms available. While there are other methods like e.g. knowledge-based RSs or context-aware RSs, the focus of the following review lies on the four paradigms mentioned, since those are the ones combined in the HRSs in Chapters 3 and 4. RS challenges like the cold-start problem or sparsity are discussed along the way.

Neighborhood-Based Collaborative Filtering (NCF)

NCF estimates utility by finding statistical neighbors of a user with respect to past ratings, and then recommends items the neighbors also liked. The assumption is that users who behaved in a similar way in the past will also behave in a similar way in the future (e.g. Ning et al., 2015). The only input to the NCF algorithm is the rating matrix Y; no user or item metadata is used. Note that there is user- and item based NCF, and both work in a very similar way. User-based NCF calculates pairwise similarities between users (rows in Y) and identifies a set of users as neighbors to a given user. Item-based NCF, on the other hand, calculates pairwise similarities between items (columns in Y), and identifies neighboring items to a given item. Since the two are analogous in their computations, just with rows and columns exchanged, the review focuses on user-based NCF. For more information on item-based NCF, see e.g. Sarwar et al. (2001).

The term “collaborative filtering” was coined by Goldberg et al. (1992). In their paper, the authors present a mail filtering system called *Tapestry* which records users’ reactions while reading a mail, i.e. positive or negative explicit feedback. Users can then set up personalized filters, in which they can specify, for instance, that they only want to receive mails which were rated positively by certain known colleagues. However, this system still relied on personal relationships in order to identify similar users, and also did not aggregate ratings.

Those problems were addressed by Resnick et al. (1994), who proposed *GroupLens*, a collaborative newsgroup RS. The system is based on newsgroup browsers with an integrated five-star rating functionality for explicit feedback. In contrast to Tapestry, GroupLens is able to identify a neighborhood of similar users to a given user without the requirement that the given user has to personally know those neighbors. Statistical similarities are calculated based on past ratings of the same items, and new items recommended based on their aggregated ratings by a given user's neighbors. This basic NCF approach is still used in many settings today due to its effectiveness and intuitive understandability. The research group which developed GroupLens shortly afterwards deployed a movie recommendation site called MovieLens, which is still running[1]. This system is where the above-mentioned MovieLens 1M data set (Harper and Konstan, 2015) stems from.

As mentioned above, NCF applies a row-wise k-nearest-neighbors approach on the user-item matrix Y. k is a hyper-parameter of the algorithm which needs to be set in advance[2]. For each pair of users u and v in Y, their pairwise similarity w_{uv} is calculated based on their respective rating (row) vector, using a similarity metric such as the cosine similarity. For a given user u, the k users with the highest similarity to u form the set of neighbors $N_k(u)$. For a given item i, the rating is then estimated as an average of the ratings for i given by the neighbors (e.g. Ning et al., 2015):

$$\hat{y}_{ui} = \frac{1}{k} \sum_{v \in N_k(u)} y_{vi} \tag{2.1}$$

This is the simplest form of NCF and already quite effective. However, the simple average over all neighbors' ratings ignores the fact that the similarities between user u and the neighbors $N_k(u)$ are not all identical, i.e. even amongst the neighborhood, there are users with more similar taste to u than others. Therefore, a more sophisticated variant of NCF uses a weighted average over the neighbors' ratings, weighted by the respective similarity to the given user, instead of an equal weights average (e.g. Ning et al., 2015):

[1] `https://movielens.org/`

[2] Please note that k denotes the number of neighbors in NCF only in this subsection. Throughout the thesis, k denotes the number of RSs combined into an HRS unless specified otherwise. Similarly, w denotes the similarity between two users in this subsection, but in the thesis in general, w denotes the weight used in the WHRS.

$$\hat{y}_{ui} = \frac{\sum_{v \in N_k(u)} w_{uv} y_{vi}}{\sum_{v \in N_k(u)} w_{uv}} \tag{2.2}$$

The pairwise similarities now play a double role in estimating future ratings: They are used (a) for selecting the neighborhood of a given user, containing the most similar users, and (b) for weighting the average over the neighbors' past ratings in order to prioritize those users who behaved most similarly to the given user in the past.

However, there is still a problem with this further developed form of NCF: When it comes to assigning a rating to an item, often each user has an individual scale. E.g., for users who mostly give five stars, a four-star rating is relatively negative, while for users who rate items with an average of three stars, a four-star rating is relatively positive (e.g. Ning et al., 2015). The left side of Figure 2.2 shows an example of this phenomenon: It contains a boxplot of each user's arithmetic mean rating in the MovieLens 1M data set, which was introduced earlier. The data set contains ratings by $6,040$ users, and their scales are quite spread out, as the plot clearly shows, over the entire range of possible ratings.

Therefore, calculating a weighted average over the neighbors' absolute ratings can introduce a bias in the estimated rating. An often-used solution is to instead use a weighted average over the neighbors' deviation from their average rating, i.e. their centered ratings. This is called *mean centering*, and the rating is now estimated as follows (e.g. Ning et al., 2015):

$$\hat{y}_{ui} = \overline{y}_u + \frac{\sum_{v \in N_k(u)} w_{uv} (y_{vi} - \overline{y}_v)}{\sum_{v \in N_k(u)} w_{uv}} \tag{2.3}$$

In this equation, $\overline{y}_u$ denotes user u's arithmetic mean rating, and the weighted average over u's neighbors' deviations from their respective mean ratings is added to $\overline{y}_u$.

While mean centering mitigates the problem of different mean ratings for each user, there is still the issue of the spread of ratings, i.e. the standard deviation. Some users use the full range of possible rating values, i.e. 1 to 5 in this case, and therefore have a higher rating standard deviation, but other users mostly assign the same rating, e.g. 4, and only use 3 for movies they find very bad, and 5 for very

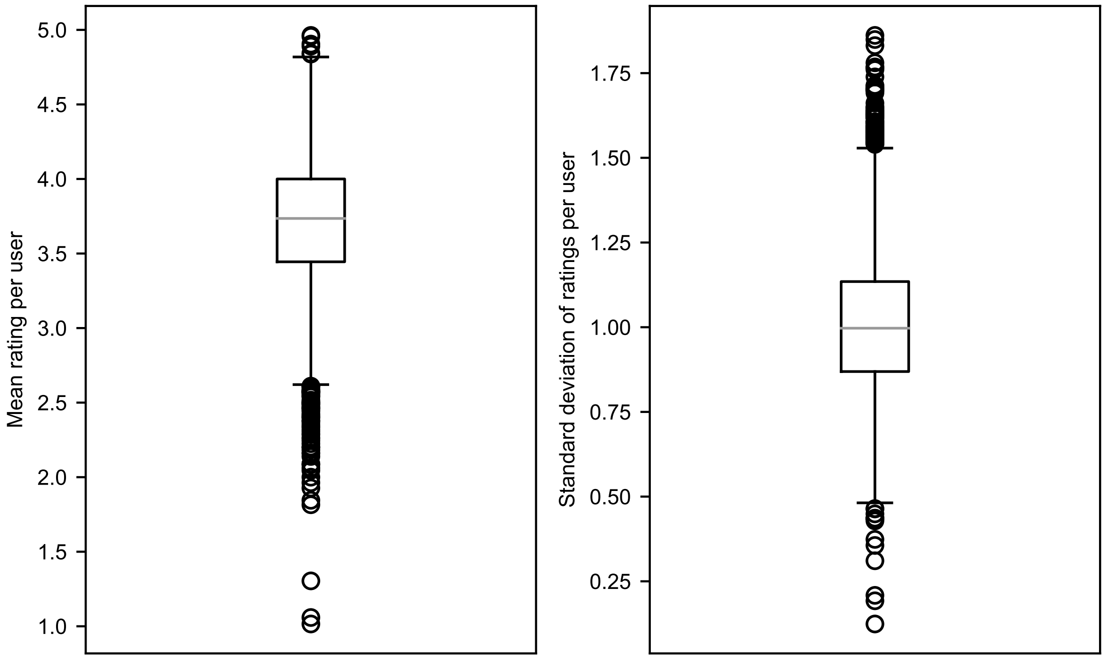


Figure 2.2.: Boxplot of the mean rating (left) and the standard deviation of ratings (right) per user in the MovieLens 1M data set.

good ones, therefore displaying a smaller spread. This is visible on the right side of Figure 2.2, which shows a boxplot of the standard deviations of each user's ratings in the MovieLens 1M data set. Again, it is apparent that there are very different types of users, with the standard deviations ranging from 0 to 2.

However, mean centering adds to a given user's mean rating the weighted average of that user's neighbors' *absolute* deviations from their respective mean ratings. This again introduces a bias, since an absolute deviation of, e.g., 0.5 from the mean rating can be relatively low for one user while it is a very significant deviation for another. Therefore, *z-score normalization* is often used to mitigate this issue. This approach (as well as mean centering) is also frequently used as a preprocessing step for variables in machine learning algorithms. The idea is to not only subtract a user's mean rating from a given rating, but additionally divide the rating by the user's rating standard deviation, thus normalizing the deviation over all users. Using z-score normalization, the rating is estimated as follows (e.g. Ning et al., 2015):

$$\hat{y}_{ui} = \overline{y}_u + \sigma_u \frac{\sum_{v \in N_k(u)} w_{uv} \frac{y_{vi} - \overline{y}_v}{\sigma_v}}{\sum_{v \in N_k(u)} w_{uv}} \tag{2.4}$$

In this formula, the weighted average over user u's neighbors' normalized ratings, i.e. *relative* deviations from their mean ratings, is multiplied with user u's rating standard deviation σ_u, and that term is again added to $\overline{y}_u$. While z-score normalization can have certain unwanted effects, such as a possible division by 0 for users who applied the same rating to all items, it has been shown to consistently improve predictive accuracy in case the rating matrix is not overly sparse (e.g. Howe and Forbes, 2008).

Table 2.1 summarizes the benefits and shortcomings of NCF (e.g. Ricci et al., 2015). Disadvantages with an asterisk (*) are partially mitigated by using item-based instead of user-based NCF.

Table 2.1.: Advantages and disadvantages of neighborhood-based collaborative filtering, based on Ricci et al. (2015). Disadvantages with an asterisk (*) are partially mitigated by using item-based instead of user-based NCF.

Pros	Cons
• No inputs required except rating matrix • *Serendipity*: can identify cross-genre niches	• *Cold-start*: no recommendations for new users or items • *Sparsity*: not enough common ratings between users* • *Scalability*: pairwise user similarities*

On the pro side, NCF only requires the rating matrix Y and no further inputs in order to estimate future ratings. Also, it is able to make serendipitous recommendations, since they are not based on the items' properties or content, but rather on similar users' tastes, which can be diverse.

On the other hand, NCF suffers from the cold-start problem. Both for new users and new items (with little to no ratings), no recommendations can be made using pure NCF. Very sparse rating matrices are a problem for NCF, since the probability of two users having rated enough common items to compute a meaningful similarity between their rating vectors decreases. Item-based NCF can mitigate this problem to a certain point: Since most real-world systems have far more users than items, the column vectors in Y are longer than the row vectors, and there is a higher chance of enough common ratings between items. This is also related to the point of scalability: If a system has millions of users, which is not uncommon, calculating pairwise user similarities becomes increasingly infeasible. Pairwise item similarities are usually more efficient to calculate.

The important shortcomings of NCF, especially user-based, make its sole usage in a productive environment difficult. Even the item-based variant still suffers from a number of issues. This motivates the usage of MCF, which is reviewed in the following section.

Model-Based Collaborative Filtering (MCF)

As described in the previous section, NCF is an intuitive and effective RS technique which, however, has a few major drawbacks. While item-based NCF can mitigate some of those, it also suffers from sparsity and scalability problems, especially when there are a lot of products.

For those reasons, Sarwar et al. (2000) introduced a novel family of RS algorithms, MCF, which are trained to learn a model that describes the user preferences. While in general, many different statistical or machine learning models can be used for MCF, the one introduced by the authors, and still heavily used today in different variants, builds on dimensionality reduction through matrix factorization. By decomposing the user-item matrix Y into lower-dimensional matrices, both the scalability and the sparsity issue are addressed. MCF pre-computes recommendations for all user-item combinations with unknown ratings. This is a key difference to NCF: While NCF calculates recommendations on-demand (and is therefore also called memory-based CF), MCF requires a — sometimes expensive — training stage, but afterwards, recommendations can be made in milliseconds (e.g. Koren et al., 2009).

In matrix factorization MCF, the user-item matrix Y is decomposed into a matrix of user factors and a matrix of item factors. Each user and each item are represented by a certain number of latent factors. The number of factors is a hyper-parameter of the algorithm which has to be set in advance. The factor matrices are learned e.g. by using an alternating least squares procedure. The prediction matrix $\hat{Y}$ is then derived by multiplying the two factor matrices.

Matrix factorization NCF has empirically been shown to be one of the most accurate and also computationally efficient RS algorithms. It has been observed that most of the time, a few latent factors suffice to reconstruct the rating matrix, thus reducing the dimensionality drastically. In addition, it is often possible to interpret the latent factors, which enables the RS operator to gain a better understanding of users' tastes (e.g. Koren and Bell, 2015).

Table 2.2 summarizes the advantages and disadvantages of MCF (e.g. Ricci et al., 2015). The first two benefits are identical to NCF. Additionally, as mentioned above, making recommendations to any given user is very fast after the model has been trained. This training phase, however, can be computationally expensive, depending on the data set and specific algorithm applied. At the same time, MCF tends to require larger data sets than NCF in order to converge. The cold-start problem, like with NCF, applies to both users and items. The sparsity and scalability issues which NCF suffers from are mitigated by using MCF, due to the applied dimensionality reduction.

Table 2.2.: Advantages and disadvantages of model-based collaborative filtering, based on Ricci et al. (2015).

Pros	Cons
• No inputs required except rating matrix	• Cold-start
• Serendipity	• Expensive training phase
• Recommendation is fast after training phase	• Requires large data set

While one of the advantages of CF algorithms is the fact that they only require the rating matrix Y as an input in order to generate suggestions, there is often additional useful metadata which cannot trivially be processed by a CF system. Therefore, we now review two RS paradigms which are able to take advantage of additional inputs, CB and DF.

Content-Based Filtering (CB)

The intuition of CB is to recommend to a given user those items most similar to the ones he or she purchased or rated positively in the past (e.g. Pazzani and Billsus, 2007). In that respect, it is related to item-based NCF. However, the important difference is that the pairwise similarities between items are not based on the items' rating vectors, i.e. the column vectors in Y, but rather based on the item metadata I.

As mentioned above, the raw metadata often contains mixed data types. E.g., in the MovieLens 1M data set, the item metadata consists of the title (text), year of release (integer), and one or more genres (categorical). In order to be able to calculate cosine similarities between the movies, i.e. the rows in I, all metadata must be transformed to a numerical format. This can be done in different ways, e.g. using natural language processing, one-hot encoding, and normalization. In addition, depending on the used similarity metric, it can be beneficial to normalize or standardize the metadata, e.g. again using the z-score, because otherwise the importance of variables with a broader value range might be overestimated.

Table 2.3 displays the benefits and shortcomings of CB (e.g. Ricci et al., 2015). It allows integrating additional input data which CF does not consider. This leads to the most important advantage, i.e. that the new-item cold-start problem is not an issue for CB. This is because the similarity calculation does not rely on ratings, but rather on feature vectors. Consequently, even new items which have not been rated yet can be recommended to users. Finally, CB is mostly computationally inexpensive, even more so than item-based NCF, since the metadata (row) vectors are usually much shorter than the rating (column) vectors.

The new-user cold-start remains an issue, since recommendations are made based on a user's past ratings. If a new user without any prior ratings registers in the system, CB is not able to calculate recommendations for that user. In addition, the problem of *over-specialization* arises: The very nature of CB, i.e. recommending items which are similar to liked items, leads to a lack of diversity in the recommendation list. E.g., in a movie RS, CB might only recommend movies of one specific genre to a user, and this effect reinforces itself over time. Apart from that, the feature vectors can only go so far in representing the content of an item. Therefore, two items which a human finds similar might not be recognized as such by the CB

Table 2.3.: Advantages and disadvantages of content-based filtering, based on Ricci et al. (2015).

Pros	Cons
• No cold-start for new items • Allows integration of item metadata • Computationally inexpensive	• New-user cold-start • *Over-specialization*: recommended items are too similar to each other • *Limited content analysis* • No implicit quality assessment

algorithm. This is called *limited content analysis*. Finally, other than CF, which only recommends items rated positively by a number of other users, CB does not have an implicit quality assessment. This can lead to scenarios where an item with only bad ratings is recommended to a user solely based on its objective properties. While this may be desired sometimes, mostly the "wisdom of the crowds" results in an accurate assessment of quality.

Due to its significant shortcomings, CB is seldom used as a standalone RS in production (e.g. Burke, 2002). However, it can be deployed as a valuable component in an HRS, especially because of its ability to recommend new items. HRSs are reviewed in Section 2.1.2.

Demographic Filtering (DF)

DF, like CB, exploits independent metadata in addition to Y. It is similar to user-based NCF, but the similarities between users are computed based on the user metadata U (e.g. age, gender, occupation). Users are recommended items which are popular in their demographic neighborhood (e.g. Pazzani, 1999).

Table 2.4 compares the pros and cons of DF (e.g. Ricci et al., 2015). It can also mitigate the cold-start problem. Other than CB, it is able to generate suggestions for new users who have not rated any items yet, based on their user metadata and their similarity to other users. However, DF is usually not very accurate as a stand-alone RS. Also, it has scalability issues when the number of users increases, like user-based

NCF (e.g. Ricci et al., 2015). Still, it can be an important component to an HRS.

Table 2.4.: Advantages and disadvantages of demographic filtering, based on Ricci et al. (2015).

Pros	Cons
• No cold-start for new users • Allows integration of user metadata • Serendipity	• New-item cold-start • Empirically not very accurate • Scalability • Privacy: Personal data required

In summary, there are different techniques and algorithms to estimate ratings and calculate recommendations, each with their benefits and shortcomings. HRSs, which are reviewed in the following section, are a simple and established means to mitigate those shortcomings and improve accuracy and robustness.

2.1.2. Weighted Hybrid Recommender Systems

In order for users to adopt RSs as a decision aid, the perceived personalization, i.e. the accuracy of recommendations, is key (Komiak and Benbasat, 2006). Higher quality RSs lead to better decision quality and increased repurchase intention (Zhang et al., 2011). The combination of multiple RSs is a proven means of increasing accuracy over the best individual algorithm. This is in line with findings from statistical forecast theory (e.g. Bates and Granger, 1969; Clemen, 1989; Makridakis and Hibon, 2000).

A combination of two or more RSs is called an HRS. HRSs combine multiple RSs in order to alleviate the aforementioned drawbacks of the individual algorithms as well as improve accuracy and robustness by reducing model variance. Each predictive method uses a certain part of the overall information available to make the prediction, and lacks other information with predictive value exploited by another method based on different data. E.g., CF only uses the user-item rating matrix as an input, while CB also exploits item metadata in order to compute similarities between items.

In addition to improved accuracy and robustness, HRSs have been shown to lead to higher decision quality, perceived usefulness, and satisfaction in comparison to pure collaborative filtering or content-based methods (Xiao and Benbasat, 2007). Different approaches to combine RSs have been proposed. Burke (2002) classifies them into seven types of HRSs:

- weighted
- switching
- mixed
- feature combination
- cascade
- feature augmentation
- meta-level

The focus of this thesis lies on WHRSs, where several individual RSs calculate predictions independently, and those predictions are then combined using an aggregation function, for instance a linear combination for a regression task (Claypool et al., 1999), or a majority vote for a classification problem (Pazzani, 1999).

With the user-item rating matrix Y as defined above,

$$\mathbf{y} \;=\; (y_{11}, \ldots, y_{1m}, \ldots, y_{n1}, \ldots, y_{nm})'$$

denotes the column vector derived by flattening Y by row. With a set of k individual RSs, $\hat{y}_{lui}$ denotes RS l's prediction of the rating of user u for item i, and

$$\hat{\mathbf{y}}_l = (\hat{y}_{l11}, \ldots, \hat{y}_{l1m}, \ldots, \hat{y}_{ln1}, \ldots, \hat{y}_{lnm})'$$

denotes the vector of RS l's estimations for the ratings in $\mathbf{y}$. A linear combination of the k estimation vectors using the weight vector $\mathbf{w} = (w_1, \ldots, w_k)' \in \mathbb{R}^k$, $\sum_{l=1}^{k} w_l = 1$ can then be calculated as

$$\hat{\mathbf{y}} = (\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_k)\,\mathbf{w}.$$

WHRSs are the most frequently applied hybridization technique in RS research (Çano and Morisio, 2017) and it has often been shown that using a weighted average of RSs' predictions leads to increased accuracy due to reduced model variance. In addition, the straightforward and intuitive way of adapting weights contributes to its popularity.

However, published work on the selection of combination weights is scarce. Mostly, uniform weights or manually selected weight vectors are used. Claypool et al. (1999) calculate weights per user or per item based on the "strength" of predictions, e.g. the number of ratings for a CF component. Pazzani (1999) combines the predictions of different RS algorithms based on their ranked recommendation lists, i.e. items which are higher in the list receive more points, and the points are summed up. Ghazanfar and Prugel-Bennett (2010) combine CB with a rule-based model to recommend e-learning materials. Similarly to Claypool et al. (1999), weights are set on a per-user base by the individual RS models based on their confidence in the respective prediction. De Campos et al. (2010) apply Bayesian networks for the combination of CB and CF. Jahrer et al. (2010) compare different supervised models like ridge regression, neural networks, or gradient boosted decision trees for learning weights.

In this thesis, we propose several analytical linear weighting schemes for HRSs which are based on statistical forecast combination. Therefore, the following section reviews literature from that area.

2.2. Robust Combination of Forecasts

This section introduces the foundations of statistical forecast combination which lay the groundwork for the methods developed in this thesis. Section 2.2.1 reviews the literature on combining predictions made by different prediction models, and Section 2.2.2 discusses the bias-variance trade-off and its connections to forecast combination.

2.2.1. Statistical Forecast Combination

In statistical forecasting, the combination of two or more prediction models in order to improve accuracy and robustness of forecasts has been subject to a large body of research. In their seminal paper, Bates and Granger (1969) introduce a weighting strategy which, for two combined models, can be shown to minimize the MSE in-sample, given the individual forecasts are unbiased, i.e. they do not consistently over- or underestimate the true values, and the performance of the individual forecasts is time-invariant. This weighting strategy is coined OW. The weight vector for k forecasts is denoted by $\mathbf{w} = (w_1, \dots, w_k)'$, and it must sum to 1, i.e. $\sum_{l=1}^{k} w_l = 1$. For $k = 2$ forecasts, we write $\mathbf{w} = (w_1, w_2)' =: (w, 1 - w)$, and focus on the optimization of $w_1 = w$, since $w_2 = 1 - w$ is determined by it. The OW is calculated as follows[3]:

$$w_{OW} = \frac{\sigma_2^2 - \rho\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2} \tag{2.5}$$

In Equation (2.5), σ_l is the standard deviation of past errors of forecast l, and ρ is the correlation between the past errors of the two forecasts. The better forecast, i.e. the one with the lower MSE, receives a higher weight. If the forecasts are biased, they must be corrected by subtracting the mean bias from each prediction.

OW can generally be calculated for k prediction models: Let $\mathbf{y}$ be the vector of actual outcomes and $\hat{\mathbf{y}}_l$ model l's predictions for the entries in $\mathbf{y}$. Assuming error vectors $\mathbf{e}_l = \mathbf{y} - \hat{\mathbf{y}}_l, l \in \{1, \dots, k\}$ of the individual models are multivariate normal with mean zero, OW can be learned that minimize the MSE over available ratings in $\mathbf{y}$. With Σ_E denoting the covariance matrix of the error matrix $E = (\mathbf{e}_1, \dots, \mathbf{e}_k)$, and $\vec{\mathbf{1}}$ as a k-dimensional column vector with all ones, the OW vector is given by (e.g. Timmermann, 2006)

$$\mathbf{w}_{OW} = \frac{\Sigma_E^{-1}\vec{\mathbf{1}}}{\vec{\mathbf{1}}'\Sigma_E^{-1}\vec{\mathbf{1}}}. \tag{2.6}$$

Note that Equation (2.6) minimizes the sum of squared deviations from zero (as of the unbiasedness assumption) subject to the constraint that the weights sum up to one, i.e. a weighted average.

[3] In Bates and Granger's original article, there is a sign error ($-$ instead of $+$) in the denominator.

2.2.2. The Bias-Variance Trade-Off in Forecast Combination

After Bates and Granger's initial article, much research was conducted on the combination of forecasts. Interestingly, OW and other learned weighting strategies have often been reported to be outperformed on unseen data by more robust weighting strategies such as giving equal weights to all forecasts, i.e. an SA (e.g. Clemen, 1989; Smith and Wallis, 2009). This observation has been coined "forecast combination puzzle" and has been subject to research for decades.

Smith and Wallis (2009) provide a theoretical explanation for the puzzle: For more sophisticated weighting schemes like OW, the weights must be estimated from past errors of the individual forecasts. While OW is guaranteed to yield minimal MSE in-sample, it does not necessarily deliver an estimate which is optimal out-of-sample, i.e. on unseen test data.

Empirical research in forecast combination typically uses rather small data sets such as time series of economic indicators with only a few dozens or hundreds of observations. The smaller the training sample on which the weights are learned, the higher the estimation uncertainty and the more learned weighting schemes like OW are adjusted to randomness in the training sample rather than to persistent structure in the data. Hence, OW can overfit the training data due to high model variance. Static weights such as SA, on the other hand, are not learned on training data and are therefore not subject to wrong fitting to randomness. Therefore, they do not overfit the training data; they have a sampling-variance-related error of 0.

On the other hand, SA does not exploit any structure from the training data, and therefore SA is subject to an underfitting error: It produces higher in-sample MSE than OW, which produces the lowest in-sample MSE of all linear weight combinations. The difference is, in statistical learning theory, called the model bias. Hence, for which of both combination models, OW versus SA, the aggregate of model bias and model variance is lower, produces the lower MSE out-of-sample. In the literature on statistical forecasting, this is coined the bias-variance trade-off (e.g. Smith and Wallis, 2009).

The forecast combination puzzle therefore results from the fact that model variance typically exceeds model bias, especially if there is a rather small number of observations, and therefore the SA is hard to beat in empirical settings. This is especially the case if the true, unknown out-of-sample optimal weight is close to the SA. However, if the true optimal weight is farther away from the SA, empirical results (e.g. Smith and Wallis, 2009) still show that the OW estimate is mostly outperformed by a strategy which is more sophisticated than SA but more robust than OW. This weighting scheme, which was also suggested by Bates and Granger (1969), ignores the error correlation between the individual forecasts (assumes $\rho = 0$, even if the true ρ is different) and only uses the inverse ratio of error variances. This strategy, which will be called "modified OW" or "OW ignoring error correlation" in the following, always yields a weight in between $w_{SA} = 0.5$ and w_{OW}, given $p > 0$, and is, for two forecasts, calculated as

$$w_{MOW} = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}. \tag{2.7}$$

The modified OW is more sophisticated than SA since it gives more weight to the better forecast, but is more robust than OW because OW can lead to extreme and instable values of w, especially if σ_1 and σ_2 are similar and ρ is close to 1, leading to the denominator of Equation (2.5) approaching 0. Another common practice to prevent overfitting in the selection of weights is the shrinkage of OW towards equal weights, i.e. a linear combination of OW and SA (e.g. Clemen, 1989; Blanc and Setzer, 2016).

In the following part, three studies are presented. In each of those chapters, a novel weighting method for WHRSs based on statistical forecast combination is proposed.

Part II.

Weighting Schemes for Hybrid Recommender Systems

Chapter 3.

Applying Optimal Weight Combination in Hybrid Recommender Systems

In this chapter, we propose a method for learning weighting schemes in WHRSs that is based on the ideas of statistical forecast combination as introduced in Section 2.2. To determine the weighting of individual RSs, we learn OW from the covariance matrix of available error data of individual RSs that minimize the error of an HRS comprised of respectively two RSs. We test the method on the well-known MovieLens 1M data set, and, contrary to the forecast combination puzzle, as introduced in Section 2.2.2, the out-of-sample results show that the learned weights consistently outperform the individually best RS as well as an SA combination.

The chapter is organized as follows: Section 3.1 motivates the contribution of a novel weighting scheme for WHRSs, based on existing approaches and their shortcomings. Section 3.2 introduces the proposed weighting approach as well as its assumptions. Section 3.3 describes the experimental design for evaluating the weighting method on the MovieLens 1M data set. Section 3.4 reports and discusses the empirical results, and Section 3.5 concludes.

3.1. Motivation and Related Work

As described in Chapter 2.1.2, not a lot of research has been published on the selection of combination weights in WHRSs. The most similar study to this one is by

Jahrer et al. (2010), who compare different weighting strategies like ridge regression, neural networks, or gradient boosted decision trees. Their ridge regression combination is similar to OW, but some differences exist: They solve a ridge regression with a regularization parameter λ in order to avoid overfitting. This approach can lead to weight vectors which do not sum to 1, especially in the case of systematically biased predictors, which are not checked for or preprocessed. Our OW approach, which is described in detail in the following, instead checks the predictors for mean biases, corrects them if necessary, and then solves a constrained least squares problem with the condition that the resulting weight vector must sum to 1. Furthermore, instead of learning the weight vector on the whole training set using cross-validation, they train the individual methods on the training set and learn the combination weights on a holdout set which is rather small in relation to the training set and might not properly represent the whole data set. This way, the resulting weight vector might overfit to the holdout set.

We show how OW can be derived from a large user-item rating matrix and demonstrate that, in contrast to the usual findings in the realm of the forecast combination puzzle, OW are beneficial to combine individual RSs given the large sets of training data which are mostly available in the RS context. We propose a simple and computationally efficient combination strategy for HRSs that can be expected to outperform other weighting schemes given sufficient training sample size. In addition, we analyze the bias-variance trade-off along the dimension s_t, i.e. how the performance of OW develops in relation to SA and the more robust modified OW when the number of available training observations decreases.

3.2. Methodology

This section explains the method of estimating OW for two RS algorithms, based on the errors made by those algorithms.

As introduced earlier, Y denotes the user-item rating matrix with s ratings, made by n users, represented by the rows of the matrix, for m items, represented by the columns. Note that in general, $s << nm$, i.e. each user on average has only rated a small fraction of available items, and vice versa. Therefore, most of the entries in the matrix are unknown. As for the known ratings, user u's rating for item i is denoted by $y_{ui} \in \{1, \dots, 5\}$.

A five-fold cross-validation is run on the available ratings in Y. Each rating is randomly assigned to one of five folds. In each of the five iterations, Y is subdivided into two matrices: The training set Y_t, which contains four of the five folds, i.e. $s_t = \frac{4s}{5}$, and the holdout set Y_h, containing the remaining fold, i.e. $s_h = \frac{s}{5}$. This is done in such a way that each fold constitutes the holdout set exactly once.

In each iteration of the cross-validation, the two individual RSs are fitted on the training matrix Y_t. The holdout matrix Y_h is flattened into a column vector $\mathbf{y}_h \in \{1, \dots, 5\}^{s_h}$ which contains all known ratings in Y_h, indexed by user and item. This is necessary for the next step: Both RS methods now make predictions for the ratings in $\mathbf{y}_h$. The prediction vectors are denoted by $\hat{\mathbf{y}}_{hl} \in \mathbb{R}^{s_h}, l \in \{1, 2\}$.

The prediction vectors and the holdout vector are then compared, which yields an error vector $\mathbf{e}_{hl} = \mathbf{y}_h - \hat{\mathbf{y}}_{hl}, l \in \{1, 2\}$ for each of the two RSs. Finally, the covariance matrix Σ_h of the two error vectors is calculated.

After finishing all five iterations, the estimate $\hat{\Sigma}_E$ of the out-of-sample error covariance matrix is calculated as the arithmetic mean of the five matrices derived in the cross-validation. This approach is taken because, as described above, OW is only guaranteed to be the optimal combination strategy in-sample, i.e. on the training set. Using cross-validation to compute an estimate of the out-of-sample error covariance matrix allows to derive a more robust weighting strategy which does not suffer as much from overfitting as in-sample OW.

The OW estimate $\hat{\mathbf{w}}$ can then be computed by inserting $\hat{\Sigma}_E$ into Equation (2.6). Since we combine respectively $k = 2$ RSs in this chapter, we introduce w as $\mathbf{w} = (w, 1-w)'$. For the remainder of this chapter, we only focus on the estimation of w_{OW}, as it contains all information of the weight vector due to the constraint that the vector must sum to 1.

Finally, both individual RS algorithms are re-fitted on all available ratings in Y in order to take advantage of the additional ratings which were used to estimate out-of-sample errors in the previous step and maximize the RSs' accuracy. The trained RSs as well as the OW estimate $\hat{w}_{OW}$ are the output of this method and can be used to make predictions for new, unknown ratings. This is described in more detail in the following.

3.3. Experimental Design

We now describe the design of the experiments we conduct to analyze the out-of-sample accuracy using the OW estimation method as described in Section 3.2 compared to SA and the individual models, respectively. Section 3.3.1 characterizes the MovieLens 1M data set used in the experiments. Section 3.3.2 describes the individual RS methods which are applied and combined. Finally, Section 3.3.3 details the procedure of evaluating the OW estimation method.

3.3.1. Data Set

For the experiments, the MovieLens 1M data set (Harper and Konstan, 2015) was used. It consists of approximately one million ratings of $3,706$ movies made by $6,040$ users. This amounts to a sparsity, i.e. ratio of missing ratings in the user-item matrix Y, of around 96%. The ratings are based on a five-star Likert scale, i.e., they have integer values from 1 (worst) to 5 (best). In addition to the ratings, the data set contains metadata for users and items: Each user is described by his or her gender, age (seven intervals), occupation, and US ZIP code; each movie is represented by its title, genre(s), and year of release. We denote the user (item) metadata as a matrix U (I), where each row vector represents a user (item).

Figure 3.1 displays the distribution of ratings in the data set. The most frequent rating value is 4, followed by 3 and 5. The arithmetic mean rating lies at 3.58, and the standard deviation at 1.1171. Hence, a naive model which simply predicts the mean rating would lead to an RMSE of 1.1171, which can be regarded as an upper bound for more sophisticated prediction methods.

Figure 3.2 shows a histogram of the number of ratings per user, using equal-width bins with a width of 50. The bin between 0 and 50 movies rated contains the largest group of users amongst all bins, and the higher the number of ratings, the lower the number of users with that number of ratings. Still, there are some users with a very high number of ratings, some over 1000. The user with the highest number of ratings in the data set rated $2,314$ movies. The minimum number is 20, which is due to the fact that users with less than 20 movies rated were removed from the data set (Harper and Konstan, 2015). On average, each user rated 166 movies. However, the median is substantially lower at 96 and the standard deviation is very high at 193.

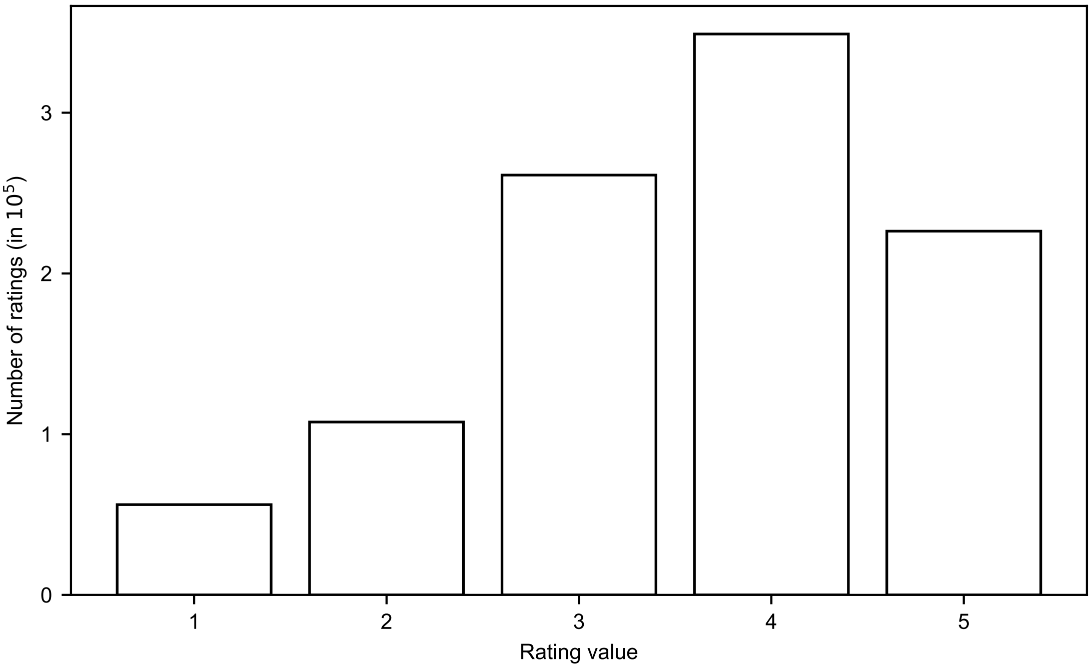


Figure 3.1.: Distribution of ratings in the MovieLens 1M data set.

Due to this skewedness, the arithmetic mean is not very meaningful here.

Figure 3.3 contains the same plot, but this time the ratings per movie are displayed instead of per user. The plot looks similar to the user plot, with most movies by far having been rated 50 times or less. In contrast to the users, movies with less than 20 ratings were not removed from the data set. In fact, 114 movies, and thus the largest group, were only rated once.

3.3.2. Individual Recommender Systems

In order to evaluate the proposed weighting method, we use the four RS algorithms which were introduced in Section 2.1.1, namely NCF, MCF, CB, and DF. They are chosen in such a way that each method either uses different input data or makes different assumptions about the underlying preference model. While NCF and MCF both use only the user-item rating matrix Y as an input, they work in different ways: NCF derives neighborhoods of users or items based on the rating similarities and calculates rating predictions as a weighted average of neighbors' ratings. MCF, on the other hand, uses matrix factorization to reduce dimensionality and extract

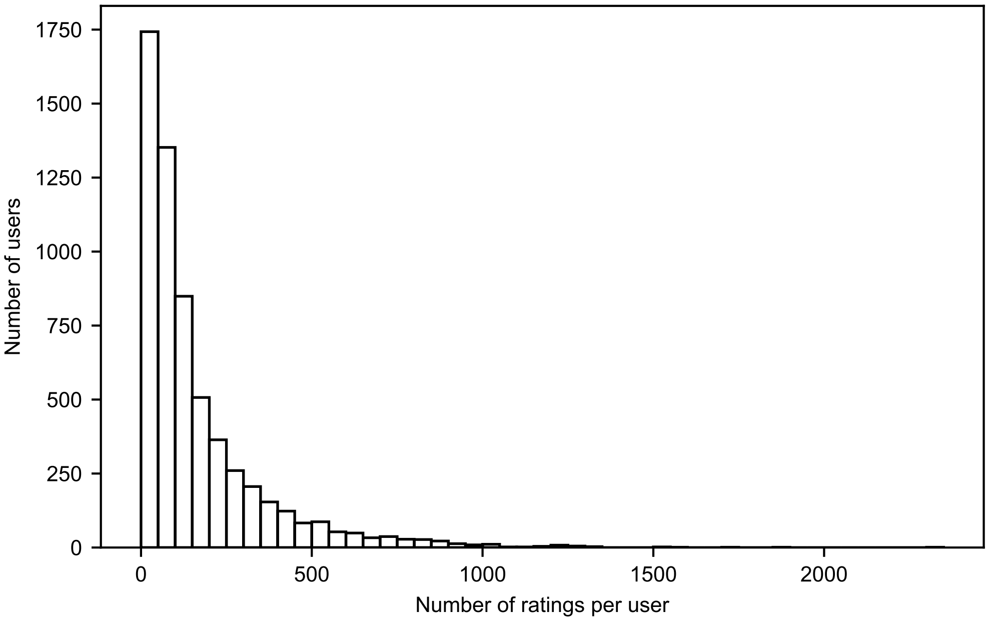


Figure 3.2.: Histogram of the number of ratings per user in the MovieLens 1M data set.

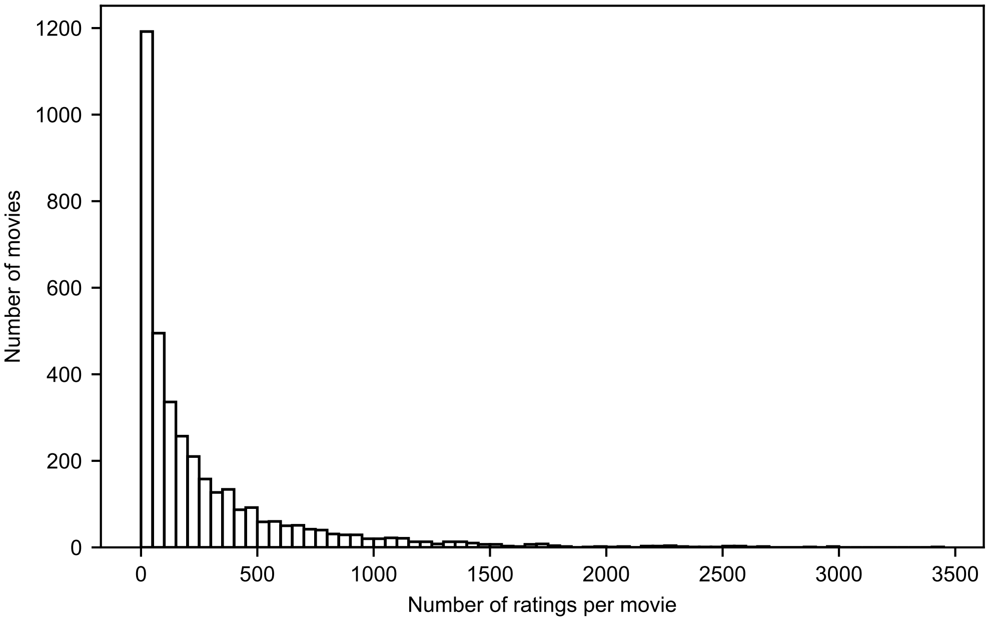


Figure 3.3.: Histogram of the number of ratings per movie in the MovieLens 1M data set.

relevant information out of the sparse user-item matrix. The remaining two methods utilize further input data for their predictions: CB calculates similarities between items based on their metadata I, while DF does so analogously for users based on their metadata U. For a more detailed description of all four algorithms, refer to Section 2.1.1. The reason why RSs with different input data or different analytical methods were chosen for this experiment is that the higher the diversity of predictions, i.e. the lower the pairwise error correlation, the more accuracy an HRS can gain over the individual components.

As mentioned above, in this chapter, all HRSs are comprised of two individual RSs. Since four algorithms were used, this amounts to six pairwise combinations which are evaluated against their individual components as well as an SA combination.

3.3.3. Evaluation and Benchmarks

This section explains the process how the proposed OW estimation method is evaluated. In a first step, it is checked whether the assumptions under which OW lead to an in-sample minimal MSE, namely that the individual RSs are unbiased, are met. In case these assumptions are violated, we preprocess and transform the data aimed at meeting the presumptions.

Second, the data set is split into a training set, containing 75% of ratings, and a test set which contains the remaining 25%. Please note that the training set as referred to here constitutes the whole input to the OW method as described in Section 3.2 and is therefore denoted by Y. The cross-validation procedure further splits Y into five folds for estimating OW. The test set, denoted by Y_p, is not fed into any training algorithm. Instead, it is solely used for evaluating the OW estimation method on unseen data and comparing it to the benchmarks.

In order to obtain robust and reliable results and to verify that the performance comparison between methods is not based on randomness, the experiment is run ten times, splitting the MovieLens data set randomly into training and test set in each run. The evaluation metric used in this experiment is the RMSE, a widely used quality criterion in RS research. The RMSE of each RS or HRS is averaged over the ten runs, and the standard deviation is also reported. The following prediction methods are fitted on the training set and their performances are compared on the test set:

- **Individual RSs**: MCF, NCF, CB, DF
- **SA**: all pairwise combinations of individual RSs using equal weights $\hat{w}_{SA} = 0.5$
- **Modified OW estimate**: all pairwise combinations using the more robust weighting strategy from Equation (2.7)
- **OW estimate**: all pairwise combinations using the weighting method proposed in this chapter

The forecast combination puzzle, which states that simpler and more robust weighting strategies such as SA are often more accurate in practice than learned weighting schemes such as OW, has often been shown on rather small forecasting data sets. The fewer training observations are available, the higher the model variance and therefore the potential to learn non-optimal weights. In order to analyze this effect in an RS setting, we also conduct the experiments with two subsets of the data, where we use random samples consisting of 50% and 10% of the available ratings, respectively.

3.4. Results and Discussion

This section reports and discusses the empirical results from the experimental evaluation of the proposed OW estimation method. First, the assumptions and requirements of OW are checked in Section 3.4.1. Second, the results of the comparison on the MovieLens 1M data set are reported and discussed in detail in Section 3.4.2. Finally, the experiment is run with only 50% and 10% of the ratings, respectively, and those results are described in Section 3.4.3.

3.4.1. Data Preparation

As mentioned in Section 2.2.1, for the OW model to derive the (at least in-sample) best weight vector, errors per RS should have mean zero and the errors are expected to be multivariate normal. Table 3.1 displays the mean errors of the individual RS methods as well as the standard error of the mean[4]. Figure 3.4 shows the histogram of errors per RS.

Table 3.1.: Mean errors of the individual recommender systems, together with the standard error of the mean. The mean errors deviate slightly from zero.

	Mean error	Standard error of mean
Model-based CF	0.000027	0.000671
Neighborhood-based CF	-0.015527	0.000680
Demographic filtering	0.013650	0.000889
Content-based filtering	-0.009644	0.000847

Although the distributions exhibit mean errors close to zero, they deviate from zero and a Jarque-Bera test with a p-value approaching zero reveals that the null hypothesis of a multivariate normal distribution is not met. Therefore, the four individual RSs are combined using OW learned on a training set (a) without any transformation and (b) with a multivariate Box-Cox transformation in which the data is corrected to conform to the requirements. The predictions are evaluated in terms of RMSE on the independent test set Y_p. The comparison is run ten times with random training-test splits.

Table 3.2 displays the mean, median, standard deviation, minimum, and maximum values of the RMSEs for both treatments. Correcting the data for a multivariate normal distribution does not lead to significant improvement in the accuracy of the HRS. That is because the deviation from a normal distribution can be considered too marginal to disturb the weight estimation. For this reason, it is chosen not to include any transformation steps in further experiments using these four RS algorithms on the MovieLens 1M data set.

[4] Note that the assumption checks described in this subsection were conducted during the study described in Chapter 4, i.e. later than the study described in this chapter. Therefore, in this subsection only, the OW combination is shown for $k = 4$ RSs at once, instead of the pairwise combinations applied in the rest of this chapter. Also, $s_p = 200,000$ observations are used as a test set for the assumption checks, instead of $250,000$ as in the rest of this chapter. However, the assumption checks are already reported in this chapter, since the same data set and the same four RS methods were used in both studies.

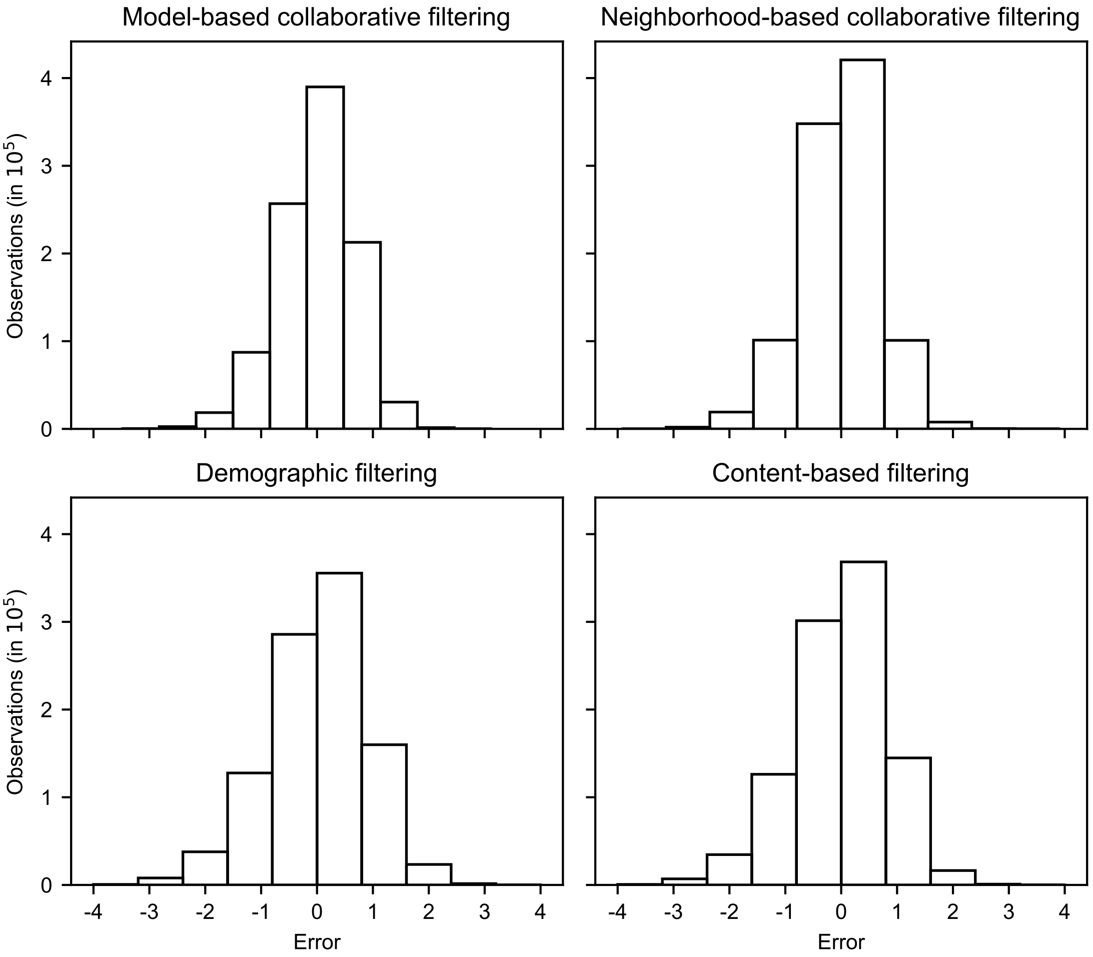


Figure 3.4.: Histograms of the individual recommender systems' errors.

Table 3.2.: Comparison of the root mean squared error of a hybrid recommender system using an optimal weight estimate with and without a Box-Cox transformation. The transformation does not lead to a smaller error.

	Box-Cox transformation	No transformation
mean	0.8685	0.8685
median	0.8686	0.8684
std	0.0012	0.0013
min	0.8667	0.8660
max	0.8703	0.8710

3.4.2. Experimental Results

Table 3.3 displays the aggregated experimental outcomes with the complete MovieLens 1M data set. The first column denotes the weighting scheme used, i.e., whether the individual RS methods (rows 1-4), SA over mutual combinations of the methods (rows 5-11), the modified OW scheme ignoring error correlation (rows 12-18), or the OW combination scheme proposed in this chapter (rows 19-25) has been applied. The second column displays the individual methods used or combined. For better comparison, there is one additional row ("Total") for SA, modified OW, and OW, respectively, which reports the arithmetic mean over all pairs. The third column reports the out-of-sample RMSE of the respective RS method or weighting scheme. As described above, the RMSE is averaged over ten experimental runs. The respective standard deviation of the RMSE over those ten runs is reported in parentheses.

In order to make the results more accessible, in the fourth column, we additionally display the percentage difference in RMSE of the OW estimation and the modified OW estimation to the respective SA combination of the same two RS methods. For instance, a value of -5% means that the learned weights resulted in an RMSE 5% lower than the one obtained when using SA, i.e. a negative value represents a better performance in comparison to the SA combination.

Finally, the two outer-right columns display the estimated weight $\hat{w}$ (i.e., the first element of the weight vector, which is the weight given to the first mentioned RS algorithm, while the other one is assigned a weight of $1-\hat{w}$), and the estimated error correlation $\hat{\rho}$ of the two combined methods. Analogously to the third column, all reported values are the averages over ten experimental runs with random training-test split, and the standard deviation of the respective value is given in parentheses.

We now discuss the experimental results on the full MovieLens 1M data set, beginning with the individual RS methods, i.e. rows 1-4 in Table 3.3. The table clearly shows that MCF using matrix factorization performs the best out of the four algorithms, with an RMSE of 0.8780. The second-best individual RS is NCF with an RMSE of 0.8964, around 2% worse than MCF. This difference is to be expected since, as described in Section 3.3.2, matrix factorization techniques are known to be among the most accurate RS algorithms, especially in the case of a sparse user-item matrix Y. As described in Section 3.3.1, the sparsity of the MovieLens 1M

Table 3.3.: Aggregated predictive results of the individual recommender systems and their pairwise combination using the optimal weight (OW) estimate, the modified OW estimate, and a simple average (SA) on the MovieLens 1M data set.

Weighting	Method(s)	RMSE (std.)	ΔSA	$\hat{w}$ (std.)	$\hat{\rho}$ (std.)
Individual recommender systems	Content-based filtering (CB)	1.0224 (0.0009)			
	Demographic filtering (DF)	0.9768 (0.0011)			
	Neighborhood-based CF (NCF)	0.8964 (0.0012)			
	Model-based CF (MCF)	0.8780 (0.0016)			
Simple average ($\hat{w} = 0.5$)	DF + CB	0.9488 (0.0007)			
	NCF + CB	0.9279 (0.0010)			
	NCF + DF	0.9155 (0.0012)			
	MCF + CB	0.9103 (0.0012)			
	MCF + DF	0.8962 (0.0011)			
	MCF + NCF	0.8738 (0.0014)			
	Total	0.9121 (0.0011)			
Modified optimal weight estimate	DF + CB	0.9478 (0.0007)	-0.10%	0.52 (0.0002)	
	NCF + CB	0.9202 (0.0010)	-0.83%	0.56 (0.0002)	
	NCF + DF	0.9123 (0.0012)	-0.35%	0.54 (0.0001)	
	MCF + CB	0.9007 (0.0012)	-1.05%	0.57 (0.0002)	
	MCF + DF	0.8917 (0.0011)	-0.49%	0.55 (0.0003)	
	MCF + NCF	0.8737 (0.0014)	-0.01%	0.51 (0.0002)	
	Total	0.9078 (0.0011)	-0.47%		
Optimal weight estimate	DF + CB	0.9460 (0.0007)	-0.29%	0.61 (0.0011)	0.80 (0.0003)
	NCF + CB	0.8964 (0.0010)	-3.40%	0.96 (0.0009)	0.87 (0.0002)
	NCF + DF	0.8966 (0.0011)	-2.06%	0.95 (0.0013)	0.91 (0.0002)
	MCF + CB	0.8770 (0.0014)	-3.66%	0.92 (0.0008)	0.84 (0.0006)
	MCF + DF	0.8761 (0.0012)	-2.24%	0.86 (0.0021)	0.88 (0.0003)
	MCF + NCF	0.8725 (0.0015)	-0.15%	0.60 (0.0044)	0.95 (0.0002)
	Total	0.8941 (0.0012)	-1.97%		

data set amounts to 96%, which explains the higher accuracy of the model-based over the neighborhood-based CF approach — or vice versa, the difficulties of the neighborhood-based model to compute robust pairwise similarities based on very few common ratings between two users, respectively.

The DF and CB models have significantly higher RMSEs than the two CF models, with 0.9768 and 1.0224, respectively. This is probably due to the lack of higher-quality metadata in the MovieLens 1M data set: As described in Section 3.3.1, the user metadata consists of gender, age group, occupation, and US ZIP code. The item (movie) metadata is even less meaningful, containing the movie title, genre(s), and year of release. The extent of metadata is probably the explanation that CB performs worse than DF, while both perform significantly worse than the CF approaches.

However, the higher errors of CB and DF in comparison to the two CF approaches do not imply that those algorithms should be dropped entirely. They can still contribute valuable information into an HRS by utilizing input data which the CF approaches do not, i.e. the user and item metadata. In general, a combination of prediction algorithms, RS or not, yields a higher improvement, the lower the error correlation between the individual approaches is. Additional independent input data is one way to lower this error correlation. The rightmost column in Table 3.3 shows that the error correlations of combinations containing CB or DF are significantly lower than the combination of the two CF approaches. The combination of MCF and NCF has the highest estimated error correlation of all pairwise combinations with $\hat{\rho}_{MCF,NCF} = 0.95$. On the other hand, the combination of DF and CB has the lowest estimated error correlation of all pairwise combinations with $\hat{\rho}_{DF,CB} = 0.80$. The other four combinations, all with one CF model and one of CB or DF, respectively, have error correlations which lie somewhere between those two extremes. For those reasons, it is still beneficial to include CB and DF in the combinations, despite their inferior individual accuracy in comparison to the two CF methods.

Rows 5-11 in Table 3.3 display the results obtained using an SA combination for each pair of RS methods. Since two algorithms are combined, respectively, the weight is always $\hat{w}_{SA} = 0.5$. The numbers show that an SA combination of the two best individual RSs, i.e. MCF and NCF, leads to an RMSE of 0.8738 which is lower than any of the RMSEs with the individual RSs. This amounts to an improvement of 0.5% over pure MCF. The fact that combining two strong prediction algorithms using SA leads to an improvement over both individual methods is in line with the literature on forecast combination.

Another insight is the strong improvement when combining the two relatively weak individual methods CB and DF. The RMSE of this combination is 0.9488, an improvement of 2.9% over the stronger individual RS which is DF. This relatively high improvement can be explained by the low error correlation, in comparison to all other pairs, of those two.

For the other four pairs, i.e. the combinations with one CF method and either DF or CB, respectively, the RMSE is higher than that of the more accurate individual algorithm (which is the respective CF method). In this case, the combination using equal weights leads to a performance deterioration in comparison to the better method. This result is to be expected, since (a) the error correlations between the respective two RSs are all quite high, and (b) assigning equal weights to two algorithms with such a disparity in performance results in the fact that the weaker of the two has too much of an influence on the final prediction.

For this reason, learned weighting strategies such as OW assign higher weights to prediction methods with lower errors. As mentioned earlier, OW has often been reported to overfit the training data in the literature on forecast combination. There, a more robust weighting strategy is recommended which assigns a higher weight to the more accurate forecast, yet does not derive as “extreme” weights as OW. This weighting strategy, which we call modified OW here, ignores the error correlation between the two forecasts and just calculates the weight as the ratio of MSEs, as displayed in Equation (2.7).

Rows 12-18 in Table 3.3 contain the results using this modified OW weighting. The weights which were estimated using this strategy are reported in the fifth column. They are all in the interval between 0.5 and 0.6, confirming that while higher weights are assigned to the better methods, they are still fairly close to the SA weight of 0.5. The fourth column shows that the RMSEs of all pairwise combinations are improved using the modified OW weighting as opposed to an SA combination. On average, the HRSs using this weighting strategy have a 0.47% lower RMSE than the ones using SA. The improvement is smaller for some combinations and larger for others. Intuitively, the higher the disparity in performance between the individual methods, the further away $\hat{w}_{MOW}$, the weight estimated using the modified OW approach, is from $\hat{w}_{SA} = 0.5$, and the higher the performance gain using the more sophisticated weighting strategy. The highest improvement when using modified OW instead of SA is achieved for the combination of MCF and CB, i.e. the best and worst individual RS, respectively. On the other hand, the performance for the combination of the two CF methods is virtually unchanged. This is also due to the fact that in this case, the weight $\hat{w}_{MOW} = 0.51$ is almost the same as for SA.

As with the SA weighting, the combination is only better than the best individual method for one of the six pairs, which is the pair of CF algorithms. For the other five pairings, the HRS leads to a higher RMSE than the better of the two RSs alone. This does not come as a surprise since, as mentioned above, the calculated weights using the modified OW estimate are close to an SA weighting, which is sub-optimal for two algorithms with a high disparity in performance.

Finally, rows 19-25 in Table 3.3 report the results when using the OW estimate including error correlation, where the covariance matrix is estimated using five-fold cross-validation on the training set, i.e. the approach proposed in this chapter. The results clearly show that the OW estimate including error correlation leads to a significantly lower RMSE than both the SA and the modified OW. As displayed in the last row of the table, the OW estimate results in an RMSE on average 1.97% lower than an SA combination.

As with the modified OW strategy, the estimated weight $\hat{w}_{OW}$ is reported in the fifth column. By definition, it is further away from the SA (0.5) than the modified OW for all pairings. However, this difference is much higher for some combinations than for others. For the respective combination of the two weakest (DF and CB) and the two strongest (MCF and NCF) individual algorithms, the weights estimated using OW are rather close to equal weights (0.61 and 0.60, respectively). For the two CF methods, this can be explained by their similarly good individual accuracies. For the pairing of DF and CB, the weight $\hat{w}_{DF,CB} = 0.61$ results, in addition to their similarly low individual accuracies, from their relatively low error correlation of $\hat{\rho}_{DF,CB} = 0.80$. Although one learner is significantly weaker than the other, the diversity it brings into the combination adds to the HRS's accuracy. Due to $\hat{w}_{OW}$ being relatively close to equal weights, the performance improvement of OW over SA is rather small for those two combinations (0.29% and 0.15%, respectively).

For the remaining four pairs, again each consisting of one CF algorithm and one of CB or DF, the weight $\hat{w}_{OW}$ calculated using the OW estimate is very high and closer to 1 (pure CF) than to 0.5 (equal weights). Therefore, the improvement over SA is also much greater for those combinations, ranging from 2.06% to 3.66%. Interestingly, $\hat{w}_{OW}$ is higher when combining NCF with CB (0.96) or DF (0.95) than when combining MCF with those two, respectively (0.92 and 0.86), even though MCF is more accurate individually than NCF. This difference is might be explained by the fact that NCF uses a more similar approach to CB and DF than MCF and therefore also has higher error correlations to these two methods. This also leads to another effect: The combination of NCF with CB or DF does not lead to a smaller RMSE than using pure NCF; the RMSE is virtually unchanged. For MCF, on the other hand, even the combination with an individually significantly weaker algorithm like CB or DF leads to a slight improvement over the pure CF.

Finally, the overall lowest RMSE of 0.8725 is obtained with an OW combination of MCF and NCF. However, the improvement over the individually most accurate algorithm (0.6%) is not much higher than when using SA. This is also caused by the fact that both CF methods have a very high error correlation in 0.95, as mentioned above.

At a first glance, the results contradict empirical findings in forecast combination research, indicating that SA is hard to beat out-of-sample, and if so, then rather by the modified OW which ignores ρ than by OW. Looking at the table in more detail, we observe that OW is the dominant combination strategy for each pair of RSs. However, for some combinations, the difference to SA is small and the difference to the individual methods is comparably large, while for other combinations, the difference to SA is large and the difference to the individual methods is smaller.

The forecast combination puzzle states that typically, the estimated OW are too unstable to improve the RMSE over SA out-of-sample. Specifically, the puzzle states that when OW is close to SA, SA will outperform OW, and when OW is further away from SA, the modified OW is to be preferred. Our results contradict this advice: OW is the best weighting strategy for all cases, and the further $\hat{w}_{OW}$ is away from SA, the larger the performance improvement by using OW over SA.

Figure 3.5 provides more insight on the findings displayed in Table 3.3. Each subplot represents a combination of one pair of prediction methods. For each combination, the out-of-sample RMSE is plotted as a function of the chosen weighting parameter $\hat{w}$; hence, the curves show which RMSE would have been observed with a particular $\hat{w}$, again averaged over ten runs. The vertical lines mark the values of $\hat{w}$ which result from using the estimated OW (dashed line), the modified OW (dotted line), and the true out-of-sample optimal value of w_{OW} (dash-dotted line), which is unknown upfront. SA ($\hat{w} = 0.5$) is marked on the horizontal axis.

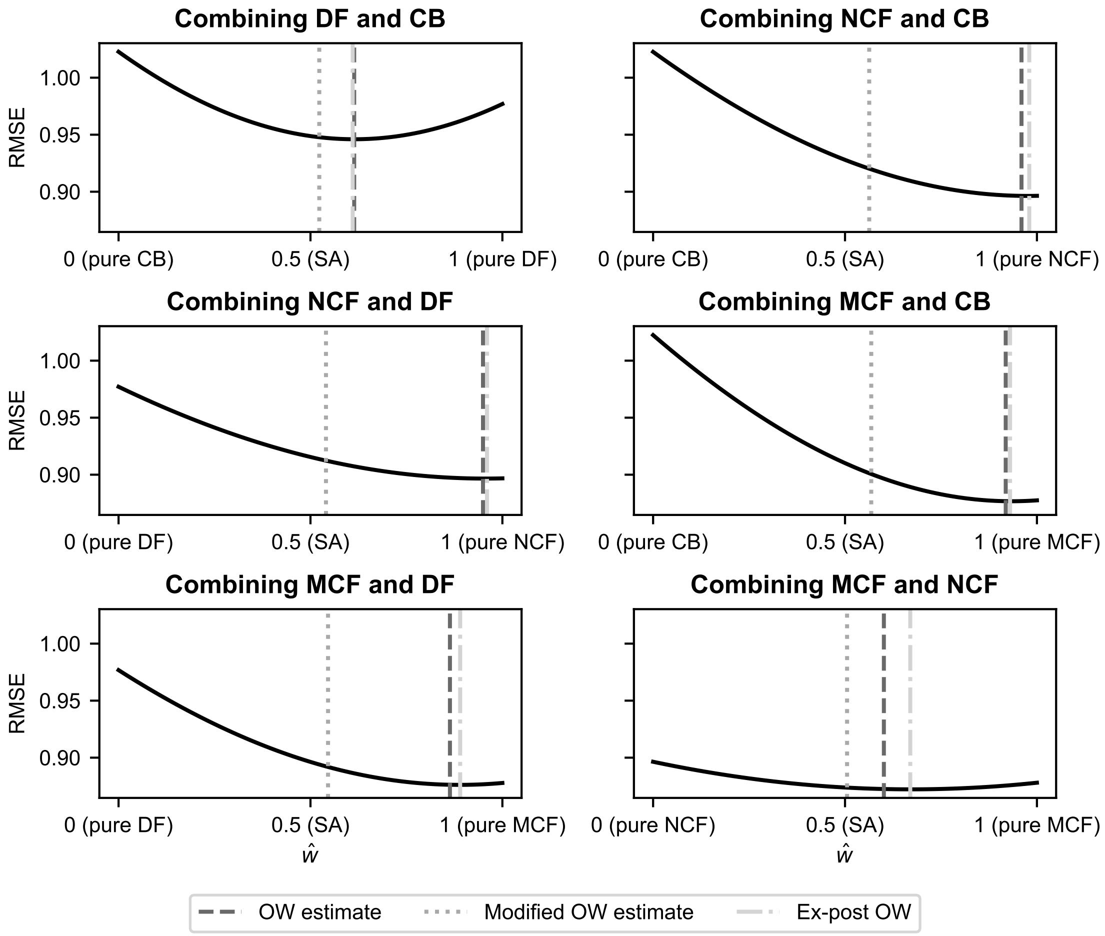


Figure 3.5.: Comparison of optimal weight (OW) estimate, simple average (SA), individual recommender systems, and modified OW ignoring error correlation for pairwise combinations of recommender systems on the MovieLens 1M data set.

In each subplot, a characteristic U-shape can be observed, which shows that an HRS comprised of the two methods yields a smaller RMSE than each of the individual methods. The graphs show that the OW estimate is close to the out-of-sample optimal weight. This is especially the case for those combinations where the error correlation $\hat{\rho}$ is relatively low ($\hat{\rho}_{DF,CB} = 0.8$, $\hat{\rho}_{MCF,CB} = 0.84$). For the combinations with higher correlation, especially $\hat{\rho}_{MCF,NCF} = 0.95$, the OW estimate is farther off the real out-of-sample optimal weights. This is to be expected, since the higher the error correlation, the more volatile the OW estimate.

3.4.3. Results with Fewer Observations

As aforementioned, the same experiments were also conducted on random samples of 50% as well as 10% of the available data. This was done in order to analyze whether the OW estimate is more likely to overfit the training data or estimate instable or biased weights when there are fewer training observations to learn from.

The results for 50% of the data are reported in Table 3.4. The structure of the table is the same as in Table 3.3. We also observe similar results to the ones with the full data set; in particular, SA improves over the best individual method, OW estimates ignoring error correlation improve over SA, while the lowest RMSEs are achieved with the OW combinations, as before. However, the error difference is smaller than on the full data set. The OW estimate is similarly accurate as with the full data set, but the true, out-of-sample optimal weight is closer to SA for all pairings except the combination of CB and DF.

This can be seen in Figure 3.6, again depicting the RMSE as a function of $\hat{w}$, but this time for the 50% sample. The graphs show relationships similar to the ones observed on the full data set presented in Figure 3.5, but the actual, out-of-sample best weight is closer to the middle and thus closer to $\hat{w}_{SA} = 0.5$ for almost all subplots. Therefore, the difference in performance is smaller than on the full data set.

Finally, the results for 10% of the data are reported in Table 3.5. The table shows that even when using only $\frac{1}{10}$ of the available data, the OW estimate is still the combination strategy that results in the lowest RMSE and outperforms SA as well as the modified OW. However, as expected, the error difference is smaller than for 50% and 100% of the data, since for the decreased number of observations the model variance and thus the level of overfitting of the OW estimates is stronger. The OW estimate is only the best weighting strategy for five of the six pairs of RSs, while for the combination of NCF and DF, SA outperforms the estimated OW. For this combination, the OW estimate is also outperformed by the more robust OW estimate

Table 3.4.: Results analogous to Table 3.3 for 50% of the training data. Principle relationships are similar to the full data set, but the differences are smaller.

		RMSE (std.)	ΔSA	$\hat{w}$ (std.)	$\hat{\rho}$ (std.)
Weighting	Method(s)				
Individual recommender systems	Content-based filtering (CB)	1.0279 (0.0020)			
	Demographic filtering (DF)	0.9801 (0.0012)			
	Neighborhood-based CF (NCF)	0.9125 (0.0013)			
	Model-based CF (MCF)	0.9181 (0.0018)			
Simple average ($\hat{w} = 0.5$)	DF + CB	0.9518 (0.0015)			
	NCF + CB	0.9384 (0.0013)			
	NCF + DF	0.9273 (0.0011)			
	MCF + CB	0.9388 (0.0021)			
	MCF + DF	0.9239 (0.0012)			
	MCF + NCF	0.9062 (0.0015)			
	Total	0.9311 (0.0015)			
Modified optimal weight estimate	DF + CB	0.9507 (0.0015)	-0.12%	0.52 (0.0002)	
	NCF + CB	0.9318 (0.0013)	-0.71%	0.56 (0.0002)	
	NCF + DF	0.9250 (0.0011)	-0.24%	0.53 (0.0002)	
	MCF + CB	0.9331 (0.0021)	-0.62%	0.55 (0.0003)	
	MCF + DF	0.9222 (0.0012)	-0.19%	0.53 (0.0004)	
	MCF + NCF	0.9062 (0.0015)	-0.00%	0.50 (0.0003)	
	Total	0.9281 (0.0015)	-0.32%		
Optimal weight estimate	DF + CB	0.9488 (0.0016)	-0.32%	0.62 (0.0012)	0.79 (0.0004)
	NCF + CB	0.9121 (0.0013)	-2.81%	0.91 (0.0011)	0.86 (0.0004)
	NCF + DF	0.9119 (0.0012)	-1.66%	0.94 (0.0014)	0.93 (0.0002)
	MCF + CB	0.9158 (0.0022)	-2.45%	0.90 (0.0017)	0.87 (0.0004)
	MCF + DF	0.9140 (0.0012)	-1.07%	0.79 (0.0030)	0.90 (0.0005)
	MCF + NCF	0.9062 (0.0015)	-0.01%	0.36 (0.0074)	0.96 (0.0002)
	Total	0.9181 (0.0015)	-1.39%		

Table 3.5.: Results analogous to Table 3.3 for 10% of the ratings. OW still dominates SA and the modified OW overall, but the gap is smaller due to the higher model variance.

Weighting	Method(s)	RMSE (std.)	ΔSA	$\hat{w}$ (std.)	$\hat{\rho}$ (std.)
Individual recommender systems	Content-based filtering (CB)	1.0649 (0.0040)			
	Demographic filtering (DF)	1.0062 (0.0041)			
	Neighborhood-based CF (NCF)	0.9869 (0.0033)			
	Model-based CF (MCF)	0.9534 (0.0031)			
Simple average ($\hat{w} = 0.5$)	DF + CB	0.9696 (0.0036)			
	NCF + CB	0.9736 (0.0044)			
	NCF + DF	0.9727 (0.0042)			
	MCF + CB	0.9810 (0.0038)			
	MCF + DF	0.9583 (0.0039)			
	MCF + NCF	0.9517 (0.0028)			
	Total	0.9678 (0.0038)			
Modified optimal weight estimate	DF + CB	0.9680 (0.0036)	-0.16%	0.53 (0.0008)	
	NCF + CB	0.9717 (0.0044)	-0.19%	0.53 (0.0008)	
	NCF + DF	0.9728 (0.0042)	0.01%	0.50 (0.0007)	
	MCF + CB	0.9751 (0.0038)	-0.60%	0.56 (0.0007)	
	MCF + DF	0.9570 (0.0038)	-0.14%	0.53 (0.0006)	
	MCF + NCF	0.9508 (0.0028)	-0.10%	0.53 (0.0006)	
	Total	0.9659 (0.0038)	-0.20%		
Optimal weight estimate	DF + CB	0.9661 (0.0038)	-0.36%	0.62 (0.0031)	0.75 (0.0009)
	NCF + CB	0.9675 (0.0045)	-0.62%	0.61 (0.0039)	0.78 (0.0011)
	NCF + DF	0.9733 (0.0042)	0.06%	0.47 (0.0064)	0.88 (0.0013)
	MCF + CB	0.9544 (0.0033)	-2.71%	0.97 (0.0059)	0.89 (0.0007)
	MCF + DF	0.9511 (0.0038)	-0.75%	0.78 (0.0063)	0.91 (0.0005)
	MCF + NCF	0.9482 (0.0028)	-0.36%	0.80 (0.0081)	0.90 (0.0010)
	Total	0.9601 (0.0038)	-0.80%		

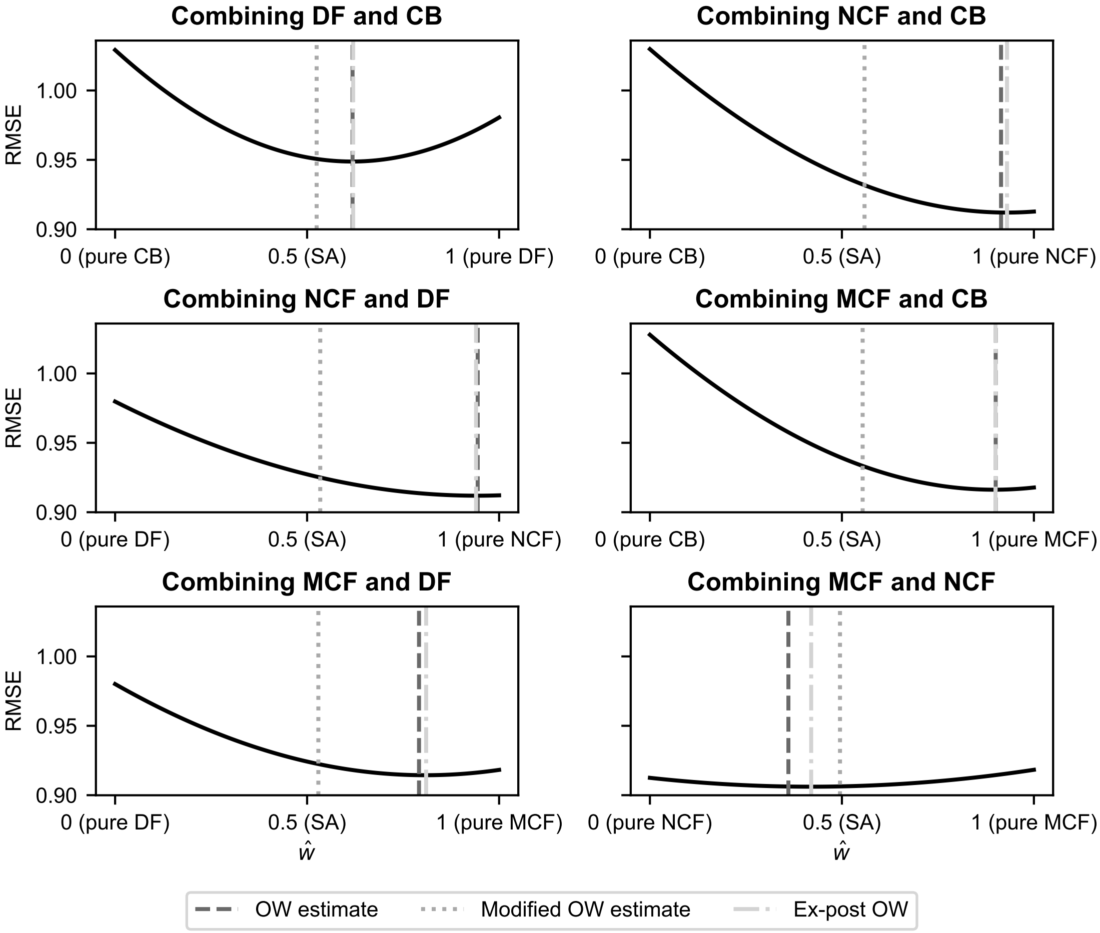


Figure 3.6.: Display analogous to Figure 3.5 for 50% of the training data. Results are similar to the full data set, but the ex-post out-of-sample best weights are closer to the simple average (SA).

ignoring error correlation.

The increasing instability of the OW estimates can also be seen in Figure 3.7. We observe that the distance between the true out-of-sample optimal weight and the OW estimate increased as compared to the full data set as well as the 50% sample. For instance, for the combination of NCF and CB, the OW estimate is off by 0.07 $(0.68-0.61)$ on the 10% data set, whereas for the whole data set, it is only off by 0.02 $(0.98-0.96)$. Similarly, for the combination of NCF and DF, the OW estimate is off by 0.12 $(0.59-0.47)$ on the 10% sample, in comparison to only 0.01 $(0.96-0.95)$ on the full data set. This is also the one combination where the OW estimate is outperformed on the 10% sample by both SA and OW ignoring error correlation, as

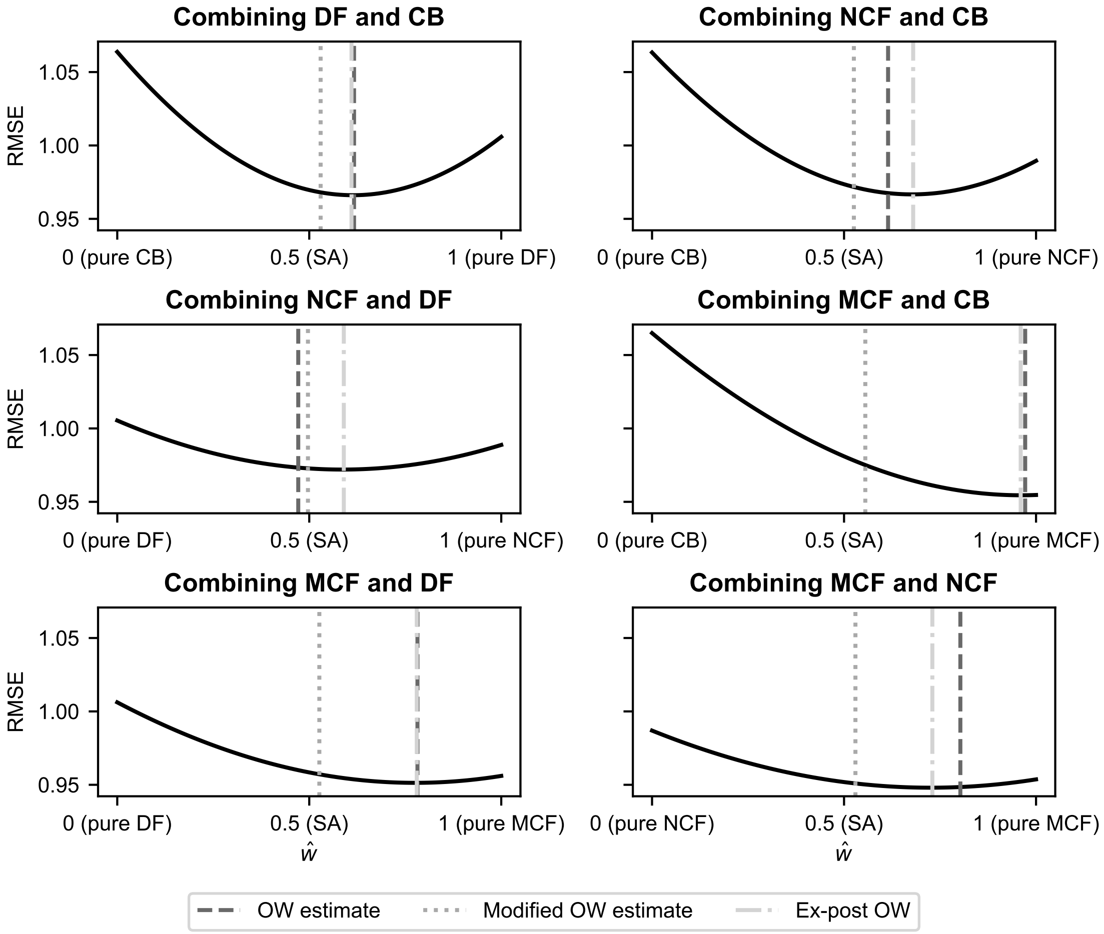


Figure 3.7.: Display analogous to Figure 3.5 for 10% of the ratings. The smaller number of observations increases estimation uncertainty for the optimal weight (OW) estimate.

described above.

3.5. Conclusion

Our experimental findings show that OW estimates can be learned in the realm of RSs that significantly outperform the individually best RS as well as the SA of RS algorithms. While this finding contradicts the forecast combination puzzle, stating that most likely SA will outperform learned weights, this outcome is explained by the fact that (a) the presumptions of OW optimality on training data are approximately met, and (b) that a comparably large set of training observations is available, which is typically not available in business forecast scenarios usually considered in the forecasting literature.

As large sets of observations are often available for RS, e.g. in the area of media platforms or e-commerce stores, OW are likely to be a beneficial choice for WHRS in many practical settings. This result is of importance due to the key role of RSs in today's digital world, and as OW estimates can be computed efficiently. Applying OW is a viable strategy in WHRSs, where, surprisingly, today there is very little research on strategies for choosing the weight.

It is worthwhile to mention that, considering decreasing benefits of using OW with decreasing training sample size, OW will not be the optimal choice for sample sizes below a certain threshold, e.g. for small enterprises or platforms with few error observations, where one would then face the forecast combination puzzle. Our results indicate that we started to experience the puzzle when reducing the data set size to 10% of available ratings. In such cases, the model variance with OW increases and regularization is required, for instance by shrinking OW towards SA by a degree that can be learned using cross-validation. Shrinkage has been shown in the realm of business forecasting to be a beneficial strategy in case the number of observations decreases or the number of forecasts increases (e.g. Blanc and Setzer, 2016).

In the following chapter, we extend the weighting approach presented in this chapter in two ways: First, we propose a weight projection method which mitigates the bias introduced by the fact that weights change with more observations, and second, we combine all four RS algorithms instead of pairwise combinations. Both of those extensions lead to a further increase in accuracy.

Chapter 4.

Projection of Optimal Weights in Hybrid Recommender Systems

In Chapter 3, we proposed a weight estimation approach for WHRSs with $k = 2$ individual RS algorithms based on statistical forecast combination. While already effective, it still has a few shortcomings. In this chapter, we extend the method into a weight projection approach. In addition to being able to combine more than two RS algorithms, this novel technique measures the error covariance and OW for increasing numbers of training observations and extrapolates them to the full size of the available data set. Therefore, it leads to an increase in accuracy over the approach from Chapter 3.

This chapter is organized as follows. Section 4.1 motivates why the weighting scheme from Chapter 3 should be extended, and how. Section 4.2 introduces the procedure to project the development of the OW, as well as error covariances used to derive OW, over different sizes of training observations, and shows its theoretical improvement potential as a function of different properties of the data set and the individual RSs. In Section 4.3, we describe the experimental design and benchmarks for evaluating the proposed method. Experimental results are presented in Section 4.4 and discussed in Section 4.5. Finally, in Section 4.6, we conclude and discuss the impact and limitations of this work.

4.1. Motivation and Related Work

As described in Section 2.1, accurate RSs are vital in order for customers to accept them as a decision aid and to take their recommendations. WHRSs are an established means to improve RSs' accuracy and robustness over the individual algorithms. There are, albeit few, methods for estimating optimal combination weights in HRSs. However, they are only guaranteed to be optimal in-sample, i.e. on the training set. Hence, to learn weights in practice, usually a subset of available data is used to train individual RSs, while portions of the data are held out to derive (out-of-sample) error structures for the individual RSs, determine weighting schemes, evaluate the out-of-sample performance with a weighting scheme, and estimate and tune further parameters such as regularization factors. In order to predict new and unknown ratings, the individual models are then trained on the entire available data set to maximize their individual performance. In Chapter 3, we proposed a method to estimate optimal combination weights for $k = 2$ RS algorithms, respectively, and found that it leads to accuracy improvements over the best individual RS as well as the SA and a modified OW estimate. That method, too, uses a holdout set in the form of cross-validation to estimate out-of-sample errors and thus OW.

Typically, an RS's accuracy increases with the number of training observations and is therefore underestimated when using data subsets. However, the strength of decrease (or the slope of the MSE curve over the training set size) is not generally the same for all RS algorithms. Some methods rapidly improve with more data, while others flatten out as they have already converged and do not profit as much from the additional data. Since OW directly depends on the ratio and correlations of errors of the individual methods, which may change with more data, weights learned on subsets of available data can be biased.

This problem is apparent with sampling techniques such as a simple holdout of ratings but also with re-sampling techniques such as bootstrapping procedures or cross-validation. Unfortunately, re-sampling that uses almost the whole training data available, such as leave-one-out cross-validation, are not recommended since those have a high variance in estimating out-of-sample errors. There is a bias–variance trade-off in the size of the holdout set: A larger holdout set leads to a bias since the individual models are trained on fewer observations and their errors are overestimated, and a smaller holdout leads to higher variance in estimating the errors (James et al., 2013).

To mitigate this problem, we propose a novel weight projection approach in this chapter. The approach measures and stores the out-of-sample errors of the individual RSs as well as the OW vector on the holdout set for increasing numbers of training observations and then extrapolates the error development to the full size of the available data set using a supervised learning model. This projection can be done in two ways: Either the error covariance of the individual RSs is projected and the OW estimate is calculated from it (referred to as *covariance projection*), or the development of OW itself is projected (referred to as *direct projection*). The aim is to estimate weights that would be learned with the full set of available data, while holding a part of the training data out for evaluation and further parameter tuning. As a result, the bias in estimating combination weights introduced by fitting the components of the HRS on a subset of the available data is mitigated, given certain assumptions, and the accuracy of the HRS is increased. Portions of data can still be used for evaluation and tuning, while individual RSs can finally be fitted on all observations without data leakage.

A simulation study is presented that illustrates benefits with this approach depending on the predictability of covariances and weights. Experimental results when applying the approach to the MovieLens 1M data set are then provided, exhibiting decreasing error levels with the proposed projection approach compared to the individual approaches, an equal weights combination, and OW estimated without projection.

4.2. Methodology

In this section, we introduce the methodology to project OW from a subset of rating data to the full set of available data. Section 4.2.1 lists the assumptions and requirements of the proposed approach. Section 4.2.2 describes the projection method in detail. Finally, Section 4.2.3 discusses asymptotic potentials with the method assuming zero projection error, i.e. perfect extrapolation of the out-of-sample OW, together with its sensitivities using a simulation study.

4.2.1. Model Assumptions

The method proposed in this chapter assumes a few requirements to be (roughly) met. We list them here and check whether they are met later in Section 4.4.1.

Since the projection method builds on the OW estimation approach proposed in Chapter 3, the requirements for OW must be met. In contrast to Chapter 3, we combine $k = 4$ RSs into one HRS instead of applying pairwise combinations of $k = 2$ methods, respectively. The assumption, however, is the same: Assuming error vectors $\mathbf{e}_l = \mathbf{y} - \hat{\mathbf{y}}_l, l \in \{1, \ldots, k\}$ of the RSs are multivariate normal with mean zero, OW minimizes the MSE over available ratings in $\mathbf{y}$.

Additionally, for the projection approach to provide benefits, the RMSE must change (presumably decrease) with increasing numbers of training observations, and the development must, at least to some extent, exhibit predictable patterns. This is because the idea of the projection approach is to extrapolate the development of error variance and error correlation, which is described in more detail in the following section.

4.2.2. Projection Methodology

Figure 4.1 describes in pseudocode the method of projecting OW learned on smaller training samples to the full available training data set. We will explain the algorithm in detail in the following. Note that the method outlined in the following paragraphs describes the direct projection. The small adjustments which have to be made for the covariance projection are explained afterwards.

Input to the algorithm is a set of s ratings in rating data $\mathbf{y}$, k individual RS models, and a supervised learning model f, such as a regression model or a time series model. The following hyper-parameters have to be set:

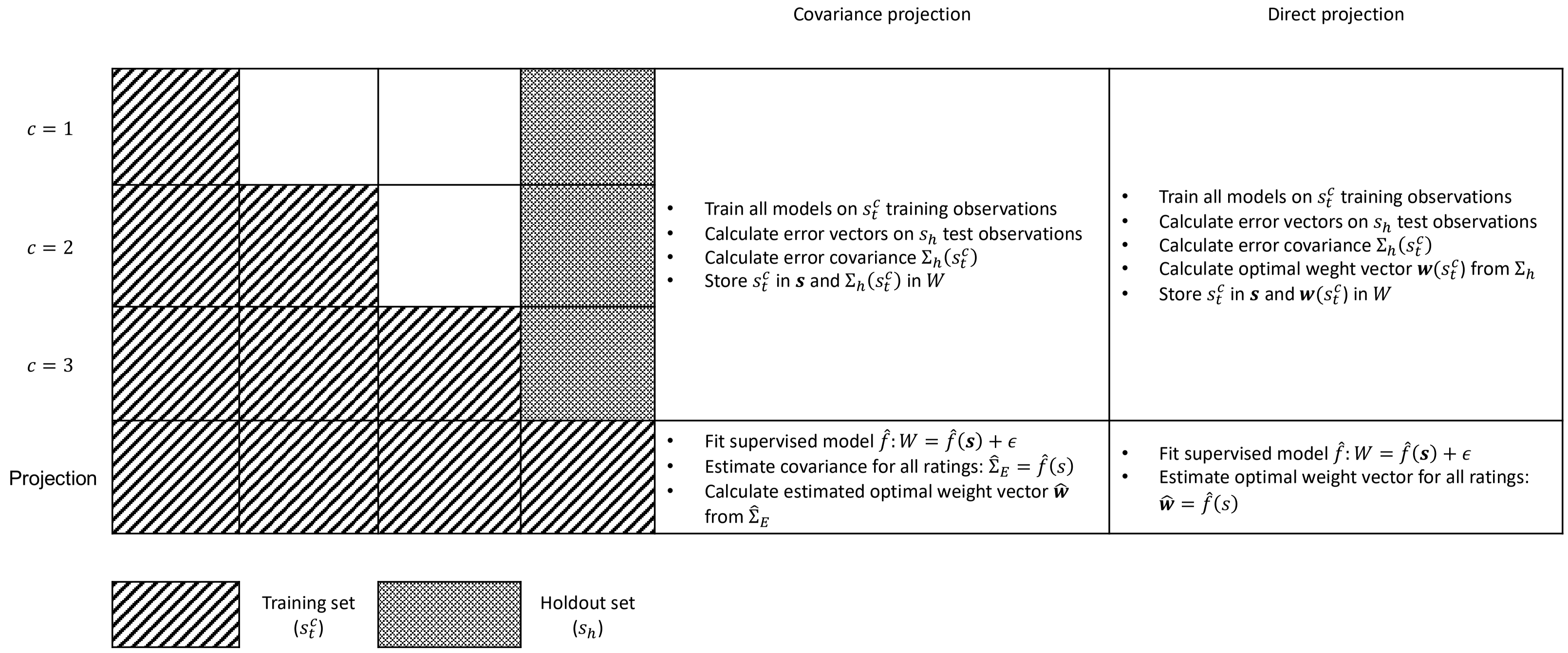
- the cardinality s_h of the holdout set $\mathbf{y}_h$
- the initial training sample size s_t^0
- the number of observations o in the projection series (number of iterations of the algorithm)

```
Input: Rating data $\mathbf{y}$ with $s$ entries; $k$ RS models; supervised learning model $f$
Parameters: Holdout set size $s_h$, initial training sample size $s_t^0$, number of training
            sample sizes (i.e. algorithm iterations) $o$
Output: Projected OW estimate $\hat{\mathbf{w}}$
Initialize ordered sets $\mathbf{s} \leftarrow \emptyset, W \leftarrow \emptyset$;
Define maximum training set size $s_t^{max} := s - s_h$;
Define step size $s_t^{new} := (s_t^{max} - s_t^0)/(o-1)$;
Randomly draw $s_h$ ratings from $\mathbf{y}$ into holdout set $\mathbf{y}_h$;
Randomly draw $s_t^0$ ratings from $\mathbf{y} \setminus \mathbf{y}_h$ into training set $\mathbf{y}_t$;
// take $o$ measurements and store them in $\mathbf{s}$ and $W$
for $c \leftarrow 1$ to $o$ do
    Training set size $s_t^c = |\mathbf{y}_t|$;
    Train all $k$ models on $\mathbf{y}_t$;
    Get predictions by all models $\hat{\mathbf{y}}_{hl}(s_t^c), l \in \{1, \dots, k\}$ for ratings in $\mathbf{y}_h$;
    Calculate error vectors $\mathbf{e}_{hl}(s_t^c) = \mathbf{y}_h - \hat{\mathbf{y}}_{hl}(s_t^c), l \in \{1, \dots, k\}$;
    Calculate error covariance matrix $\Sigma_h(s_t^c)$ from error vectors;
    Calculate out-of-sample OW $\mathbf{w}(s_t^c) = (\Sigma_h(s_t^c)^{-1}\vec{\mathbf{1}})/(\vec{\mathbf{1}}'\Sigma_h(s_t^c)^{-1}\vec{\mathbf{1}})$;
    Store training set size: $\mathbf{s} \leftarrow \mathbf{s} \cup \{s_t^c\}$;
    Store OW: $W \leftarrow W \cup \{\mathbf{w}(s_t^c)\}$;
    if $c < o$ then
        Randomly draw $s_t^{new}$ ratings from $\mathbf{y} \setminus (\mathbf{y}_h \cup \mathbf{y}_t)$ into $\mathbf{y}_t^{new}$;
        Append new ratings to training set: $\mathbf{y}_t \leftarrow \mathbf{y}_t \cup \mathbf{y}_t^{new}$;
    end
end
// fit the prediction model and project to $s$ observations
Fit prediction model: $W = \hat{f}(\mathbf{s}) + \epsilon$;
Predict OW for $s$ ratings: $\hat{\mathbf{w}} = \hat{f}(s)$;
return $\hat{\mathbf{w}}$
```

Figure 4.1.: Pseudocode for projection of optimal weights.

In each iteration c, $s_t^{new} = (s_t^{max} - s_t^0)/(o-1)$ ratings are randomly drawn and appended to the training set $\mathbf{y}_t$, causing the training set to increase in equidistant steps per iteration. Each individual RS is then fitted to the respective training sample with s_t^c observations, and all RSs predict ratings in the holdout set of size s_h, yielding error vectors $\mathbf{e}_{hl}(s_t^c), l \in \{1, \dots, k\}$, where then the error covariance $\Sigma_h(s_t^c)$ is estimated and out-of-sample OW $\mathbf{w}(s_t^c)$ are derived on the holdout set.

This procedure is repeated for increasing values of s_t up to a defined maximum value $s_t^{max} = s - s_h$, where for each s_t considered the obtained OW are captured. With the records containing s_t (stored in $\mathbf{s} \in \mathbb{N}^o$) and OW (stored in $W \in \mathbb{R}^{o \times k}$),



Covariance projection, $c = 1, 2, 3$:

- Train all models on s_t^c training observations
- Calculate error vectors on s_h test observations
- Calculate error covariance $\Sigma_h(s_t^c)$
- Store s_t^c in $\boldsymbol{s}$ and $\Sigma_h(s_t^c)$ in W

Direct projection, $c = 1, 2, 3$:

- Train all models on s_t^c training observations
- Calculate error vectors on s_h test observations
- Calculate error covariance $\Sigma_h(s_t^c)$
- Calculate optimal weght vector $\boldsymbol{w}(s_t^c)$ from Σ_h
- Store s_t^c in $\boldsymbol{s}$ and $\boldsymbol{w}(s_t^c)$ in W

Covariance projection, Projection:

- Fit supervised model $\hat{f}: W = \hat{f}(\boldsymbol{s}) + \epsilon$
- Estimate covariance for all ratings: $\hat{\Sigma}_E = \hat{f}(s)$
- Calculate estimated optimal weight vector $\hat{\boldsymbol{w}}$ from $\hat{\Sigma}_E$

Direct projection, Projection:

- Fit supervised model $\hat{f}: W = \hat{f}(\boldsymbol{s}) + \epsilon$
- Estimate optimal weight vector for all ratings: $\hat{\boldsymbol{w}} = \hat{f}(s)$

Figure 4.2.: Example of the projection approach. In this case, 25% of the available data is used as a holdout set and initial training set, respectively ($s_h = s_t^0 = \frac{s}{4}$). $o = 3$ iterations are run, and finally OW are projected to all s observations.

developments of OW over s_t are fed into a supervised model extrapolating the development of $\mathbf{w}$ when using the s samples as training data. Finally, individual RSs are re-fitted on all s observations and their estimates are combined using the projected weight estimate $\hat{\mathbf{w}}$ to predict unseen ratings.

For the covariance projection, a few adjustments are made: Instead of the OW, the error covariance $\Sigma_h(s_t^c)$ is flattened and stored in each step (line 12 in the pseudocode). The supervised model then delivers the projected error covariance $\hat{\Sigma}_E(s)$, which is used to calculate the projected OW (line 21 in the pseudocode).

Figure 4.2 illustrates the methodology for $s_h = \frac{s}{4}$, $s_t^0 = \frac{s}{4}$, and $o = 3$. Both the derivation of OW from the projected covariance and the direct projection of OW are displayed. The training set size is increased step-wise, while the holdout set of size s_h is kept constant for the calculation of the error covariance. For each of the three values of s_t, error covariance matrices and OW are determined on the holdout sample. The development of the covariance (or OW) is then projected to the whole set of s observations, and the OW estimated by projection is applied to the RS predictions for new test data, where the RSs are fitted on all s data points.

As the advantages with the proposed projection clearly depend on the change in OW over training sample sizes as well as the predictability of these changes, we will now discuss how much improvement the projection approach can yield over the conventional method of learning OW, and the sensitivity of potential improvements.

4.2.3. Improvement Potential

For demonstration, we consider the simple case of two RSs ($k = 2$) in this subsection only. We assume bi-variate normal distributed errors with mean zero to meet the assumptions of the OW model. For brevity, we write $\mathbf{w} = (w, 1 - w)'$ and focus on the optimization of w, since the second element of the weight vector is determined by the first one and the restriction that the vector must sum to 1. We also assume that the error correlation ρ between the two models is constant over all s_t, and that the development of the RMSE over s_t is extrapolated perfectly to s for both models (i.e. zero forecasting error).

Since both models are unbiased, their error standard deviation $\sigma_l, l \in \{1, 2\}$ is equivalent to their RMSE, decreasing with s_t. Let $\alpha(s_t) := \sigma_1(s_t)/\sigma_2(s_t)$ be the ratio of the two models' error standard deviations when trained on s_t observations. Inserting α into Equation (2.6), the OW is then given by

$$w_{OW}(s_t) = \frac{1 - \rho\alpha(s_t)}{1 + \alpha(s_t)^2 - 2\rho\alpha(s_t)}. \tag{4.1}$$

Now, with s_t as the cardinality of the training subset used for fitting the RSs, $s_h > 0$ as the size of the holdout set used for calculating the error covariance matrix and deriving OW, and $s = s_t + s_h$, we define $\beta(s_t, s) := \alpha(s)/\alpha(s_t)$. For brevity, we will use $\alpha := \alpha(s_t)$ and $\beta := \beta(s_t, s)$. The change in the OW from s_t to s training observations is given by

$$\Delta w = w_{OW}(s) - w_{OW}(s_t) = \frac{1 - \rho\alpha\beta}{1 + \alpha^2\beta^2 - 2\rho\alpha\beta} - \frac{1 - \rho\alpha}{1 + \alpha^2 - 2\rho\alpha}. \tag{4.2}$$

The absolute value $|\Delta w|$ of this change is the maximum improvement in estimating the OW when using the weight projection method rather than just applying $w_{OW}(s_t)$ to s. Figure 4.3 illustrates the improvement potential in estimating the OW as a function of β for three different values of α (0.5, 0.8, and 1) and three different values of ρ (0, 0.5, and 0.9). The plot illustrates sensitivities of the OW change on combinations of α, β and ρ values.

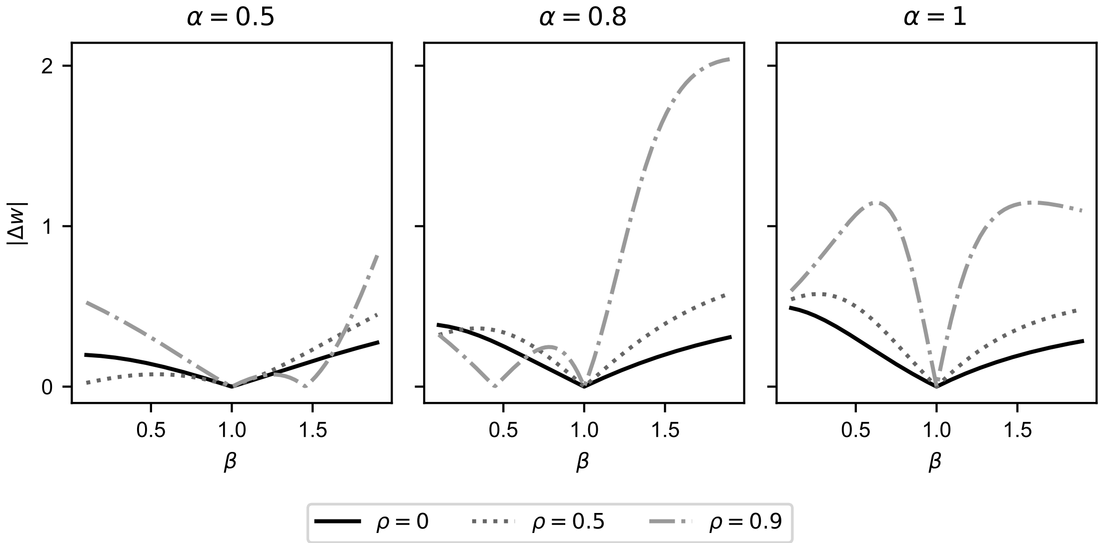


Figure 4.3.: Sensitivity of the optimal weight to changes (β) in the root mean squared errors of the individual recommender systems, depending on the initial ratio of RMSEs (α) and the error correlation ρ.

The figure shows that the (absolute) change of w_{OW} between s_t and s training observations increases with ρ. In addition, for values of α closer to 1 (i.e. similar initial RMSEs of the two RSs), the change in the OW is more sensitive than when the initial RMSEs are very different[5].

What can also be seen in the figure is that, depending on the initial ratio of error standard deviations (RMSEs), the change in the ratio of RMSEs, and the error correlation, the absolute difference in w_{OW} from the training subset to the whole training set can be very high (up to a value of 2). For instance, a value of $\Delta w = 0.5$ could mean that the out-of-sample OW for the whole training set is $\mathbf{w}_{OW}(s) = (0.8, 0.2)$, but the OW estimate learned on the training subset is $\hat{\mathbf{w}}(s) = \mathbf{w}_{OW}(s_t) = (0.3, 0.7)$. Consequently, this weight estimate can be quite far off the out-of-sample OW, motivating a projection of the weights in such settings.

In order to analyze the practical relevance of the projection approach, we now test sensitivities of the RMSE of the HRS regarding the change in OW. Figure 4.4 displays the increase in RMSE depending on the error in estimating w_{OW} for the same three values of ρ as above and $(\sigma_1, \sigma_2) = (0.9, 0.95)$. Values of the RMSE are shown for a test set containing $s_p = 200,000$ observations.

The lower the error correlation between the individual methods, the more the RMSE increases with the absolute error in estimating w. This again means that the lower ρ, the higher the potential for the weight projection approach to improve the RMSE of the combination over learning weights on the smaller training set.

As an example, we assume that $\Delta w = -0.25$, i.e. the real $w_{OW}(s)$ is 0.25 lower than $w_{OW}(s_t)$ and therefore the conventional method of weight estimation $\hat{w}(s) = w_{OW}(s_t)$ has an absolute error of 0.25 in estimating $w_{OW}(s)$. For $\rho = 0$, a perfect projection (i.e. an error of 0 in estimating OW on the full data set) leads to an RMSE improvement of 0.093, more than 10% of the better individual RS's RMSE. For $\rho = 0.5$, the improvement is 0.046, still quite high. However, for $\rho = 0.9$, the RMSE improvement is only 0.019.

The reason that the RMSE improvement potential using a projection approach over a conventional weight estimation procedure decreases with an increasing ρ is that a higher error correlation means more similar predictions by the RSs. The more similar the predictions, the lower the impact of a non-optimal weighting scheme on

[5] $\alpha = 2$ is equivalent to $\alpha = 0.5$, but with the individual model variances interchanged.

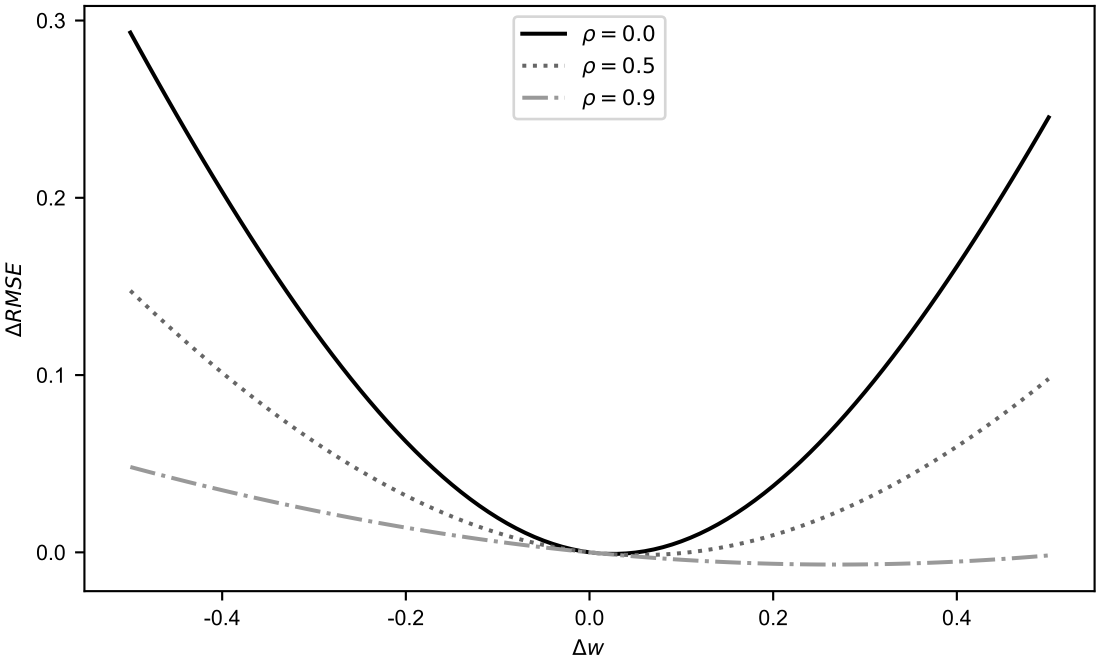


Figure 4.4.: Improvement of the root mean squared error (RMSE) of a hybrid recommender system with perfect extrapolation of the optimal weight (OW), depending on the error correlation ρ and the deviation Δw of OW between the training set and the entire data set.

the RMSE.

In summary, depending on the difference between OW estimated on a training subset and the unknown OW on all available data, and given that the projection of OW to the complete data set is accurate, the weight projection approach can yield accuracy improvements over the conventional method of learning OW without projecting. Clearly, the magnitude of improvement depends on different parameters of the data set as well as the RS methods, such as their error correlation and their relative performance improvement when given new data. Furthermore, the model assumes multivariate normal errors with mean zero, which might be violated in RS tasks in practice.

The following section describes the experimental design to evaluate the proposed projection method on a real-world data set and compare it to different benchmarks.

4.3. Experimental Design

This section describes the experiments and benchmarks used to evaluate the projection approach introduced in this chapter. Like in Chapter 3, all methods were tested on the MovieLens 1M data set (Harper and Konstan, 2015). For details about the data, refer to Section 3.3.1.

In a first step, we check whether the model assumptions listed in Section 4.2.1 are met. In case these assumptions are violated, we preprocess and transform the data aimed at meeting the presumptions.

Second, the data set is split into a training and a test set. In order to evaluate the HRSs as well as the individual RS models, we keep an independent test set $\mathbf{y}_p$ with s_p ratings which are not used for training. 20% of the available ratings are used as $\mathbf{y}_p$ for evaluating and comparing the different weighting strategies out-of-sample, which leaves 80% of the ratings for $\mathbf{y}$. The holdout set $\mathbf{y}_h$, used to record the development of the error covariance and OW over different training set sizes, also contains 20% of all ratings. Using 20% of the ratings for deriving OW (s_h) and compare the weighting strategies (s_p), respectively, ensures that the out-of-sample error estimates are robust and do not exhibit a high variance.

This leaves a maximum of $s_t^{max} = s - s_h \approx 600,000$ observations for fitting the RS models to derive OW. The number of levels of s_t is set to $o = 10$, i.e. ten iterations with increasing s_t are conducted. This value proved high enough to capture the complexity of the error development in preliminary experiments. s_t is initialized as $s_t^0 \approx 150,000$ and increased in steps of $s_t^{new} = (s_t^{max} - s_t^0) \ / \ (o - 1) \approx 50,000$ up to $s_t^{max} \approx 600,000$, where the c-th value of s_t is denoted by s_t^c.

The following prediction methods are fitted on the training set and their performances are compared on the test set:

- **Individual RSs**: We use the four RS approaches described in Section 2.1, namely MCF, NCF, DF, and CB. They are chosen in such a way that each method either uses different input data or makes different assumptions about the underlying preference model: While MCF and NCF both use only the user-item rating matrix Y as an input, they apply different statistical models. DF and CB utilize further input data for their predictions, namely the user and item metadata, respectively.

- **SA**: An HRS assigning equal weights to all individual RSs. In this case, with $k = 4$, each RS is assigned a weight of 0.25.

- **Naive forecast**: The OW estimation method introduced in Chapter 3 is used as a benchmark. The difference to that study is that here we use a simple holdout instead of cross-validation for better comparability to the projection approaches. It will also be referred to as "naive forecast" in the following since it assumes that OW with s training observations equals OW with s_t^{max} observations, similar to a naive time series forecast which predicts the current value as the value for the next point in time.

- **Covariance and direct projection**: Those are the two variants of the projection method proposed in this chapter (see Section 4.2 for details). In the experiment, Holt's exponential smoothing (Holt, 2004) is used to project error covariances and OW at $s_t^c \leq s_t^{max}, c \in \{1, \ldots, o\}$ to $s \approx 800,000$. Exponential smoothing is chosen because of its simplicity and its ability to extrapolate linear as well as nonlinear developments.

All weighting approaches are additionally compared to the ex-post out-of-sample OW, derived by fitting all RS methods to the full $\mathbf{y}$ (80% of ratings), calculating the error covariance on the test set $\mathbf{y}_p$, and inserting it into Equation (2.6). To obtain reliable results, the procedure is repeated ten times with random allocation of ratings to $\mathbf{y}_t$, $\mathbf{y}_h$, and $\mathbf{y}_p$.

The following section reports the experimental results of comparing projected weights to the benchmarks.

4.4. Empirical Evaluation

This section reports the results of the experimental evaluation. In Section 4.4.1, the requirements of OW and the projection method are checked. Section 4.4.2 presents experimental outcomes.

4.4.1. Data Preparation

As mentioned above, the proposed projection method requires OW assumptions to be fulfilled. In Section 3.4.1, we checked whether the individual RSs' errors were multivariate random with mean 0. The test results showed that those assumptions were violated, however a mutivariate Box-Cox transformation as an intermediate step before calculating OW did not lead to an increase in accuracy. Since the same individual RSs and the same data set are used in this chapter, no additional transformations are performed.

In addition, the projection approach requires the development of the RMSE (error standard deviation) of the individual RSs over s_t to display a predictable pattern, i.e. not fluctuate randomly. Figure 4.5 displays that development with $s_p \approx 200,000$ and increasing s_t, again averaged over ten runs with random training-test splits. For DF, CB, and NCF, the development curves are very similar, steeper at the beginning and flattening towards the end. The curve for MCF differs in that it decreases almost in a linear fashion, indicating that it can still gain significant performance benefits from being fed more training data. The curves exhibit structure that should be partly predictable.

4.4.2. Experimental Results

Table 4.1 displays overall comparative results with the projection approach. The first column reports which individual or hybrid RS was used. The first four rows contain the individual RSs. The fifth row contains results for an SA combination. The sixth row displays results for the conventional OW estimation method as introduced in Chapter 3. Finally, the seventh and eighth row of the table show results for the two variants of the projection method introduced in this chapter, the covariance projection and the direct projection. The mean RMSE as well as the standard deviation over ten runs is reported in the second column. The third column reports, for all methods, the relative difference to the ex-post OW, i.e. the best possible RMSE using a linear weight vector.

Please note that the CB and DF algorithms used in this study are further developed versions of the ones used in Chapter 3. Among other changes, they use debiasing techniques. That is why their individual RMSEs in this table are signif-

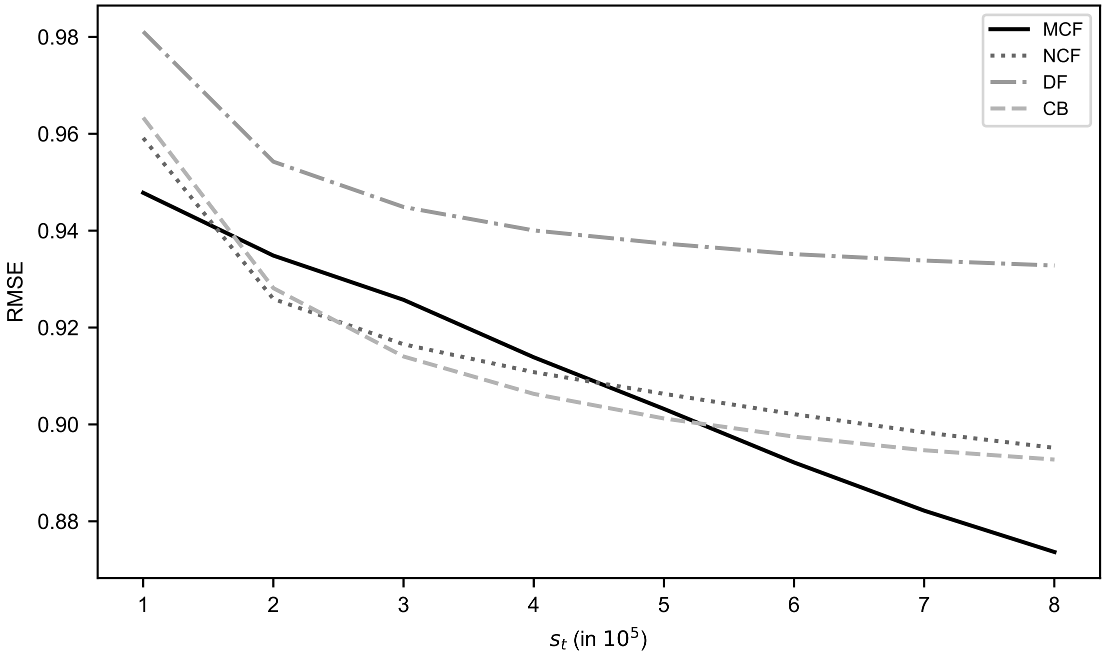


Figure 4.5.: Development of the root mean squared error (RMSE) of the individual recommender systems depending on the number of training observations s_t (average over ten runs with random allocation of $s_t^{max} \approx 800,000$ observations for training and $s_p \approx 200,000$ for evaluation).

Table 4.1.: Comparison between root mean squared errors (RMSE) of weighted hybrid recommender systems (RS) using the weighting strategies simple average, naive forecast (weights learned on a training subset), projection of the error covariance, and direct projection of optimal weights. For reference, the RMSEs of the individual RSs are also included.

	RMSE (std.)	Δ(best RMSE)
Model-based CF	0.8736 (0.0012)	1.126%
Neighborhood-based CF	0.8951 (0.0012)	3.615%
Demographic filtering	0.9331 (0.0010)	8.014%
Content-based filtering	0.8925 (0.0009)	3.310%
Simple average	0.8765 (0.0010)	1.453%
Naive forecast	0.8646 (0.0010)	0.087%
Covariance projection	0.8639 (0.0010)	0.005%
Direct projection	0.8640 (0.0010)	0.015%

icantly lower than in Table 3.3. For the other two methods, MCF and NCF, their performances are similar to Table 3.3. For both, the RMSE is slightly lower, which can be explained by the fact that in this study, 80% instead of 75% of the ratings were used as training data. Interestingly, the error of MCF decreases slightly stronger than that of NCF, which confirms the insight from Figure 4.5.

Looking at the HRSs, the table shows that a learned weighting strategy clearly outperformed an SA in this instance, also confirming results from Chapter 3. The worst-performing learned weighting strategy, which is the naive forecast (0.8646), had an RMSE which was 1.3% lower than the one when using an SA combination (0.8765). Since the difference between the best (MCF with 0.8736) and the worst individual RS (DF with 0.9331) was relatively large here, an SA combination was even slightly outperformed by using pure MCF. Possibly, an SA combination of the three best RSs, excluding DF, might have led to a lower RMSE than MCF alone.

The table also reveals that both projection strategies outperformed the naive forecast. Interestingly, calculating the OW estimate from the projected error covariance led to a slightly better result than direct projections of the OW. Both projection weighting strategies came very close to the best possible RMSE using a linear weight vector, with an RMSE only 0.005% and 0.015% higher, respectively. This is illustrated in Figure 4.6, comparing the difference to the best possible RMSE for the naive forecast and the two projection methods. The direct projection outperformed the naive forecast by a factor of about 6 in terms of the difference to the best RMSE, while for the projection of the error covariance the factor is about 17.

In order to understand why the projection approach leads to an improved performance over the conventional OW estimate, we now analyze how well the projection approach was able to extrapolate the different parameters, i.e. error standard deviations, error correlations, and the elements of the OW vector. Table 4.2 reports the MAE in estimating the error covariance and OW of the two projection approaches and compares them to the naive forecast, i.e. just using the parameters learned on the training set for the entire data set.

The first column of the table displays for which parameter the results are. They are divided into three blocks: the error standard deviation σ_l for each RS l, the error correlation ρ_{lq} for each pair (l, q) of RSs, and the weight w_l assigned to each RS l. E.g., if the MAE for w_{MCF} is 0, this means that the component of the OW

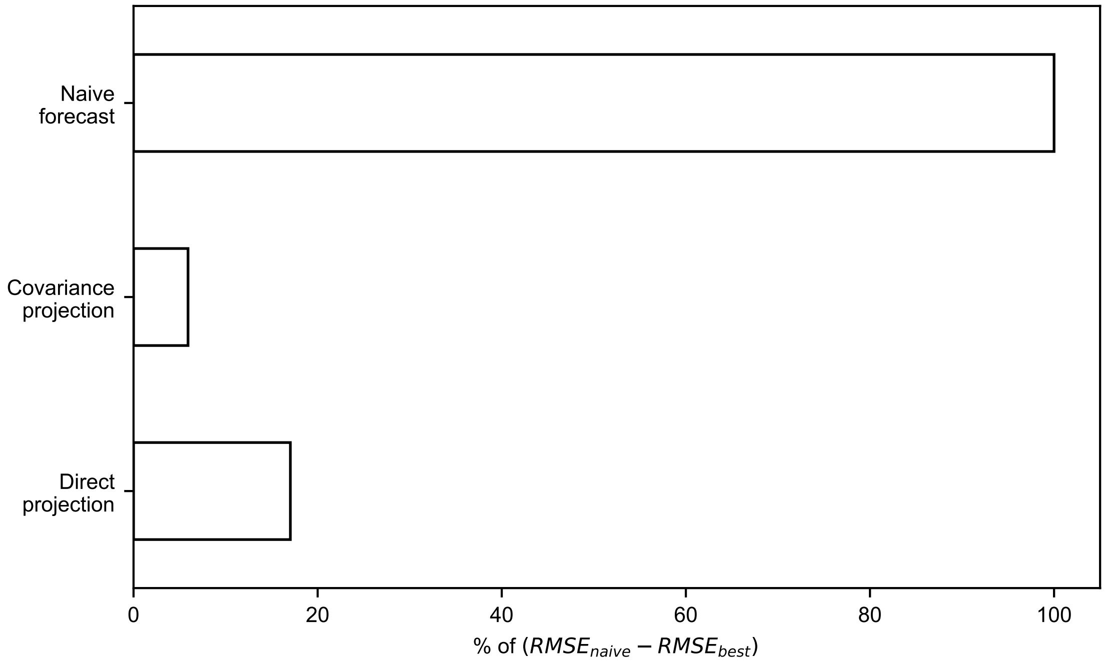


Figure 4.6.: Comparison of the weight estimation approaches in terms of their difference in root mean squared error (RMSE) to the ex-post optimal weight (OW). The values are displayed as a percentage of the difference between the RMSE of the naive forecast ($RMSE_{naive}$) and the ex-post OW ($RMSE_{best}$).

vector for MCF was estimated perfectly. In addition, there is one "Total" row for each of those blocks, reporting the arithmetic mean over all entries in the respective block, in order to ease the comparison between the two projection methods and the benchmark.

The second column contains the results when applying the naive forecast, i.e. just using the values measured at s_t^{max} training observations for the whole data set. This is equivalent to the absolute change of the respective parameter from s_t^{max} to s observations. The third and fourth columns display the results with the covariance projection and the direct projection, respectively. Note that in the direct projection column, there are no values for the parameters σ and ρ. That is because this variant of the projection method directly extrapolates the development of OW without considering the error covariance.

As described above, the experiment was run ten times with random allocations of training, holdout, and test set, in order to derive robust results and enhance

Table 4.2.: Comparison between mean absolute error (MAE) of the naive forecast, the covariance projection, and the direct projection in estimating the error standard deviations σ (where applicable), the error correlations ρ (where applicable), and the optimal weights w assigned to each individual recommender system. The experiment was run ten times with random training-holdout-test splits, and all values are averages over those runs, while the standard deviations are displayed in parentheses.

	Naive forecast	Covariance projection	Direct projection
σ_{MCF}	0.0187 (0.0017)	0.0026 (0.0025)	
σ_{NCF}	0.0068 (0.0018)	0.0017 (0.0011)	
σ_{DF}	0.0021 (0.0010)	0.0017 (0.0012)	
σ_{CB}	0.0049 (0.0013)	0.0020 (0.0012)	
Total	0.0081 (0.0015)	0.0020 (0.0015)	
$\rho_{MCF,NCF}$	0.0089 (0.0004)	0.0024 (0.0008)	
$\rho_{MCF,DF}$	0.0128 (0.0005)	0.0027 (0.0013)	
$\rho_{MCF,CB}$	0.0083 (0.0004)	0.0017 (0.0009)	
$\rho_{NCF,DF}$	0.0017 (0.0003)	0.0003 (0.0001)	
$\rho_{NCF,CB}$	0.0008 (0.0003)	0.0006 (0.0003)	
$\rho_{DF,CB}$	0.0016 (0.0005)	0.0005 (0.0003)	
Total	0.0057 (0.0004)	0.0014 (0.0006)	
w_{MCF}	0.1139 (0.0074)	0.0191 (0.0147)	0.0393 (0.0258)
w_{NCF}	0.0290 (0.0118)	0.0239 (0.0173)	0.0359 (0.0237)
w_{DF}	0.0240 (0.0078)	0.0110 (0.0062)	0.0076 (0.0055)
w_{CB}	0.0609 (0.0118)	0.0113 (0.0084)	0.0129 (0.0077)
Total	0.0569 (0.0097)	0.0163 (0.0116)	0.0239 (0.0157)

comparability. The values reported in the table are therefore averages over those ten runs, with the respective standard deviation in parentheses.

The table shows that the projection approach leads to higher accuracy than the naive forecast for all parameters. For the assigned weights w_l, the mean MAE of the direct projection (0.0239) is less than 50% that of the naive approach (0.0569). Calculating OW via the extrapolated error covariance performs even better. Here, the mean MAE (0.0163) is about 3.5 times smaller compared to the naive forecast. Extrapolating the error standard deviation σ results in an MAE (0.0020) which is, on average, around four times smaller compared to the usage of a naive forecast (0.0081). For the error correlation ρ, the extrapolation (0.0014) also outperforms the naive forecast (0.0057) by a factor of around 4.

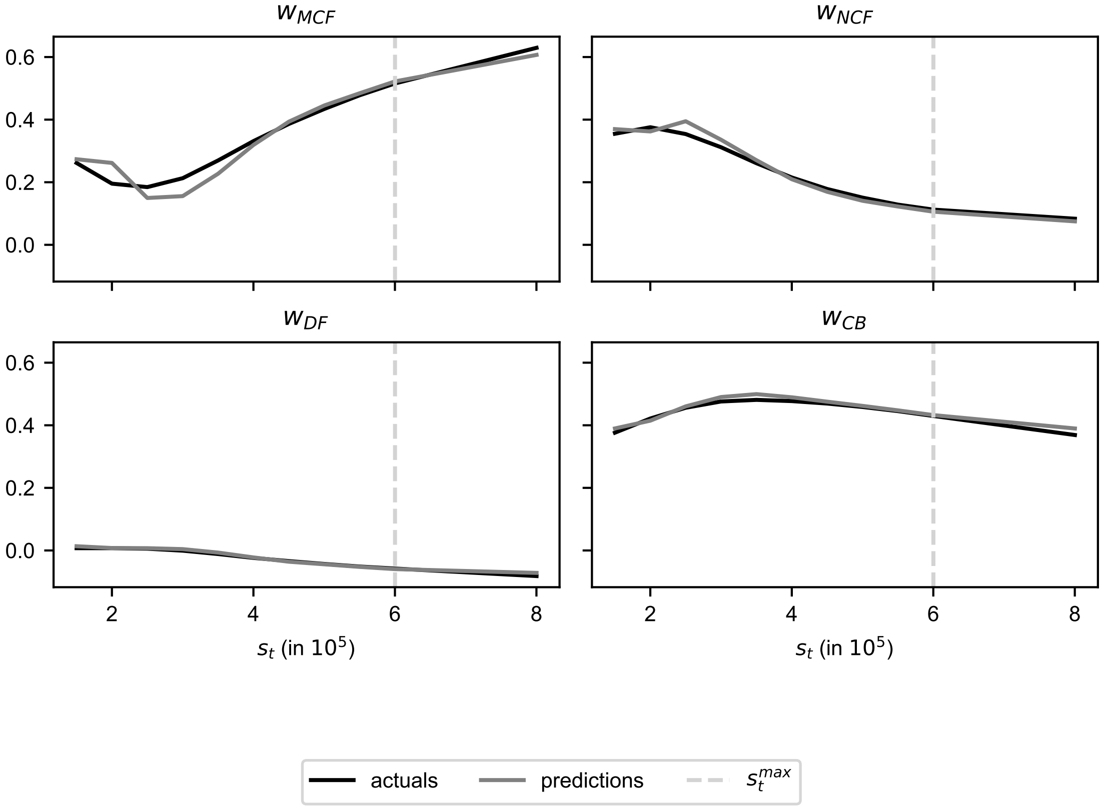


Figure 4.7.: Results for the projection of the optimal weights (OW) estimated for each recommender system. The black curves represent actual OW, the gray curves show the predictions. The vertical dashed line displays s_t^{max}, i.e. predictions beyond this point are out-of-sample.

Figure 4.7 shows the performance of the direct OW extrapolation, where w_{MCF} is, for instance, the weight assigned to MCF. The black curves display the actual values collected in the stepwise approach, while the gray curves show the predictions using the exponential smoothing model. A vertical dashed line is plotted at s_t^{max}, i.e. all predictions beyond this point are the out-of-sample extrapolations.

Figure 4.8 displays the extrapolation performance for the error standard deviation σ, or the RMSE development for the four individual RSs. Again, the black lines represent the actual values, while the gray curves show the predictions, and the vertical dashed line is set at s_t^{max}.

Finally, Figure 4.9 presents the performance for extrapolating the error correlation ρ of the pairwise combinations of RSs. The lines are equivalent to Figure 4.7 and 4.8.

Those three figures indicate that the exponential smoothing model was quite ac-

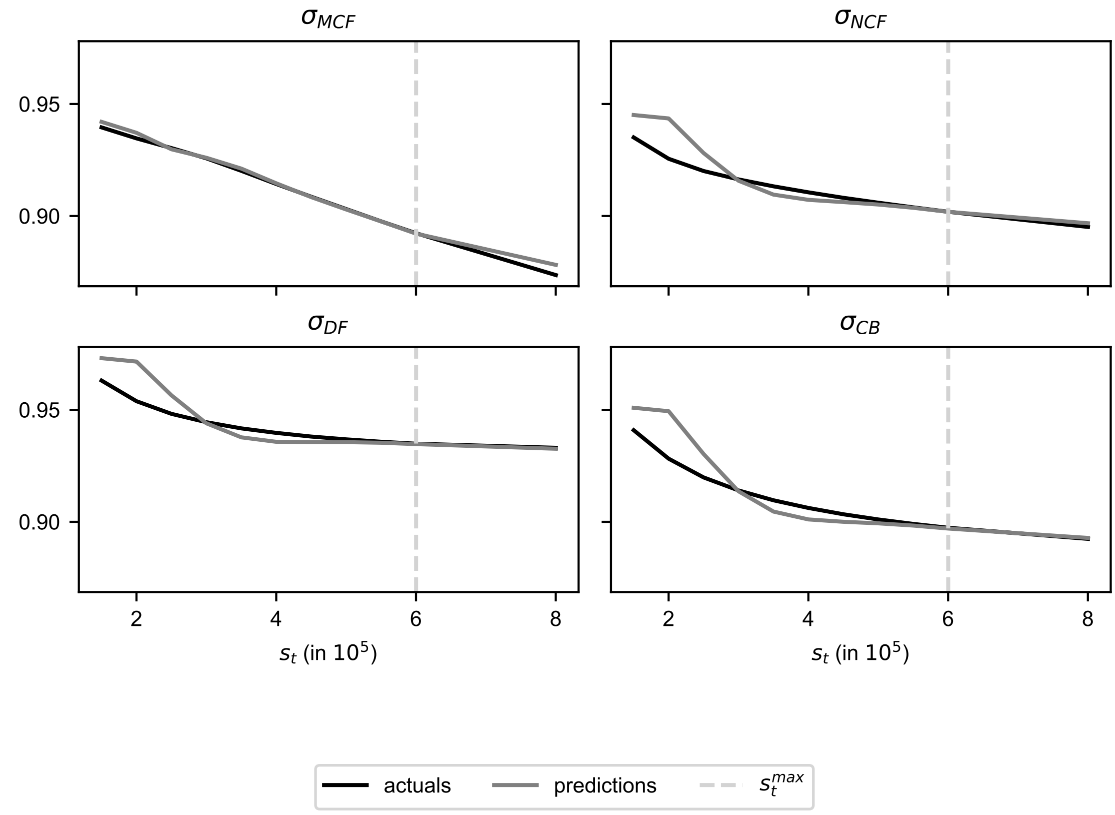


Figure 4.8.: Results for the projection of the error standard deviation of each recommender system. The lines are analogous to Figure 4.7.

curate in extrapolating both OW and error covariance (comprised of error standard deviations and error correlations). This is shown by the fact that the gray lines (predictions) are close to the black lines (actuals) beyond the light-gray dashed line, i.e. out-of-sample. Since the parameters were predicted accurately, both the direct projection and the covariance projection led to a smaller RMSE than the naive forecast.

In the following section, we discuss and interpret the results obtained in the experimental evaluation.

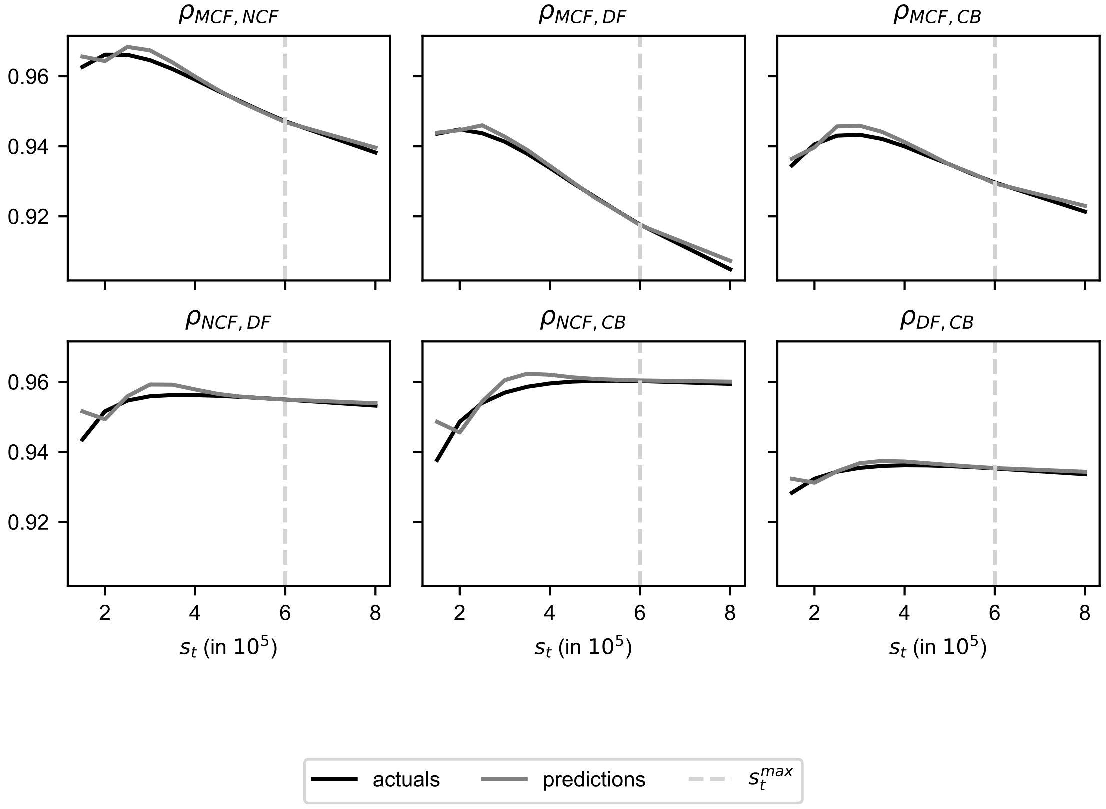


Figure 4.9.: Results for the projection of the error correlation of each pairwise combination of recommender systems. The lines are analogous to Figure 4.7.

4.5. Discussion

Results on the MovieLens 1M data set show that using the naive approach of estimating OW on the training set and then applying it to the whole data set improves accuracy. On average, the RMSE using the naive forecast only exceeds the lowest possible RMSE with linear models, i.e. the RMSE with out-of-sample OW, by 7.84×10^{-4}. This is because OW learned with $600,000$ observations are already rather stable and change only slightly with additional $200,000$ observations.

However, results show that projected weights consistently improve the accuracy of the naive approach. On average, the direct weight projection leads to a decrease in the difference to the best possible RMSE with a linear combination approach by 83% (from 7.8×10^{-4} to 1.3×10^{-5}). The projection of the pairwise error covariances led to an even higher decrease in RMSE of 94% (from 7.8×10^{-4} to 4.4×10^{-5}). The improvement is consistent over all individual runs, making the projection of weights a promising approach, given sufficient predictability of error patterns with increasing training sample size.

This is generally in line with findings in Chapter 3, where we applied pairwise combinations of the same four RS methods to the same data set. In this chapter, we used combinations of $k = 4$ methods instead of only combining two algorithms at a time. Still, applying OW led to a smaller RMSE compared to using equal weights, even though there are more degrees of freedom — and therefore increased instability — when estimating a vector $\mathbf{w}_{OW}$ of length 4 instead of 2. This shows that given a sufficiently large training data set, OW is a feasible strategy for combining RS methods as this improved predictive accuracy over individual methods or pairwise combinations significantly.

In our experiment, the error correlations between the individual RSs were strong (see Figure 4.9), and while the RMSEs decreased for all RSs over s_t, the change of their ratio(s) was moderate (see Figure 4.8). In Section 4.2.3, we used a simulation study to show the magnitudes of potential improvements with the projection over the naive approach as a function of those values (error correlations and changes in RMSE ratios). Here, due to the properties of the data set and the RSs used, the margin of improvement over the conventional OW approach was rather small. This can be seen in Table 4.1, which shows that the naive forecast led to an RMSE that was 0.087% higher than the best possible RMSE when using a linear weight vector. However, projection further reduced the relative error and led to RMSE values close to the out-of-sample OW, which is apparent in Figure 4.6.

The novel projection method shows very promising results, and, with individual RSs exhibiting (a) lower error correlations, and (b) less similar RMSE development, larger improvements over OW learned on the training set are expected.

4.6. Conclusion

In this chapter, we introduced a novel approach to learning optimal combination weights in HRSs by calculating the error covariance matrix and, based on that, the OW for different numbers of training observations and then projecting this series of covariance matrices or weights to the full size of available data.

We applied the projection approach to the MovieLens 1M data set by combining standard RS algorithms, where it consistently led to the lowest RMSE out-of-sample compared to individual RSs, a simple averaging of RSs, or OW estimates not using a weight projection (i.e., a naive projection).

As a limiting factor, the window of improvement between the naive forecast and the ex-post OW was very small in this study, i.e. the naive forecast was already very close to the best possible linear weight vector. Still, the projection approach was able to cut that window by 81% and 94%, respectively. We expect the empirical improvement in accuracy of the projection method in comparison to the naive forecast to be significantly higher for other RS algorithms with smaller error correlations, and possibly for other data sets and use cases.

In the following chapter, we present a study in which the weight estimation method introduced in Chapter 3 is adapted to be used for the weighted combination of classification algorithms.

Chapter 5.

Hybrid Recommender Systems for Next Purchase Prediction Based on Optimal Combination Weights

As described earlier, RSs play a key role in e-commerce by pre-selecting presumably interesting products for customers. Hybrid RSs using a weighted average of individual RSs' predictions have been widely adopted for improving accuracy and robustness over individual RSs.

The literature on how to select combination weights in WHRSs is very limited. While for regression tasks, approaches to estimate accurate weighting schemes based on individual RSs' out-of-sample errors exist, there is scant literature in classification settings. Providing products or product categories of interest to a current user is key to content and affiliate marketing, generating leads and developing existing customers in terms of up- and cross-selling endeavors.

In this chapter, we adapt the weighting method proposed in Chapter 3 to a classification problem, specifically in the area of purchase predictions. We use the Brier score, the equivalent of the MSE for classification tasks, in order to generate error vectors. Based on those error vectors, the error covariance matrix of all individuall classifiers is calculated and fed into the OW estimation. Other than in Chapter 3, more than two algorithms are combined at once in this study.

The approach is evaluated on a labeled real-world data set from a large European telecommunications provider, where it is used to predict purchasing probabilities for three categories of mobile devices. The task is to predict the conditional probability, given that a certain customer is going to buy a mobile device, in which category it will be. Thus, the problem can be regarded as a top-1 recommendation task. Experimental results show that the proposed classifier weighting method leads to significant improvements, both in the Brier score and the accuracy score, compared to both the individual models as well as an SA combination.

This chapter is organized as follows. Section 5.1 motivates the importance of a weighting scheme for WHRSs in purchase prediction or other classification tasks. Section 5.2 describes the proposed classifier weighting method. Section 5.3 outlines the experimental design to evaluate the proposed method on a real-world data set, and Section 5.4 reports the empirical results from those experiments. Section 5.5 discusses the benefits and shortcomings of the proposed approach, and Section 5.6 concludes.

5.1. Motivation and Related Work

Little research has been published on the selection of combination weights in WHRSs. As described above, there exist some weighting strategies for regression scenarios, mainly rating prediction, but for classification tasks, binary or multi-class, we are not aware of analytical methods for selecting weight vectors in WHRSs. However, those kinds of problems appear very often in e-commerce, where a company wants to estimate, for a given user, purchasing probabilities of different products or product categories in order to show personalized advertisements or select suitable customers for marketing campaigns.

In Chapter 3, we proposed a method of estimating OW for combining multiple RSs in a rating prediction scenario. That approach derives in-sample OW that minimize the MSE of the HRS on the training data given certain assumptions. In this chapter, we propose an analytical weighting procedure to increase the accuracy and robustness of a multi-class classifier ensemble over the best individual classifier as well as SA without requiring expensive brute-force search or additional input data. The proposed approach transfers the weighting method from regression to a multi-category classification problem, where the goal is to predict the next purchase of a given customer based on the customer profile. For that, we use the Brier score, which quantifies the mean squared deviation of the estimated purchase probability from the true outcome.

The Brier score, introduced by Brier (1950), can be viewed as the classification version of the MSE. It measures the accuracy of probabilistic predictions. For m possible outcomes (classes) and s observations (users)[6], it is calculated as

$$BS = \frac{1}{sm} \sum_{u=1}^{s} \sum_{i=1}^{m} (y_{ui} - \hat{y}_{ui})^2. \tag{5.1}$$

y_{ui} represents the actual outcome for observation u and class i, which is either 0 or 1, and $\hat{y}_{ui}$ represents the estimated probability with $0 \leq \hat{y}_{ui} \leq 1$ and $\sum_{i=1}^{m} \hat{y}_{ui} = 1,\ u \in \{1, \ldots, s\}$. The Brier score is used in this work to estimate weights for combining probability scores of multiple classifiers. We combine classifying RSs based on the covariance structures of the individual models' probability scores such that the Brier score is minimized in the same fashion as the MSE is minimized in regression settings. With k classification models and m possible class labels, we calculate error vectors $\mathbf{e}_l = \mathbf{y} - \hat{\mathbf{y}}_l, l \in \{1, \ldots, k\}$, where

$$\mathbf{y} = (y_{11}, \ldots, y_{1m}, \ldots, y_{s1}, \ldots, y_{sm})'$$

is the vector of actual outcomes and $\hat{\mathbf{y}}_l$ is the vector of probability estimations for $\mathbf{y}$ by classifier l. Calculating OW from those error vectors can be shown to minimize the Brier score in-sample, analogous to the MSE in a regression setting. Next purchase (class) predictions on unseen data are then derived as the class with the highest probability score.

Our weighting approach is evaluated on a labeled real-world e-commerce data set, which contains purchases of three different categories of mobile devices. The problem is formulated as a top-1 recommendation model, i.e. the question is: Given that a customer is going to buy a device, which category is the most likely one? We combine seven classification models which act as demographic filtering RSs (e.g. Pazzani, 1999), learning purchasing probabilities from customer attributes.

[6] Note that in this chapter, each user is represented by one observation, i.e. one customer profile. Therefore, the number of users n and the number of observations s are identical.

5.2. Methodology

This section introduces the approach to estimate OW for combining predictions of classification algorithms. Section 5.2.1 considers the assumptions and requirements of the weighting method. Section 5.2.2 describes the estimation of optimal combination weights in detail.

5.2.1. Model Assumptions

The classifier weighting is based on statistical forecast combination, as introduced in Section 2.2.1. In regression settings, Equation (2.6) ensures a minimal in-sample MSE given that the individual models' errors follow a multivariate normal distribution with mean 0, i.e. the models are unbiased.

As mentioned above, our adapted classifier weighting scheme relies on the Brier score (Brier, 1950) as the classification equivalent of the MSE. For each observation and each class label, we calculate the deviation between the predicted probability of the observation pertaining to the class and the true outcome. For each observation, the predicted probabilities sum to 1, and the true outcome is 1 for one class label and 0 for all the other labels. Since the errors are flattened, yielding error vectors of length sm, the deviations between prediction and ground truth sum exactly to 0 for each observation. Consequently, the mean deviation for each flattened error vector is also 0. Regarding the multivariate normality of the error vectors, respective analyses of the data set are provided in Section 5.4.1.

Another assumption of the classifier weighting method is that the minimization of the Brier score of a classifier ensemble results in an accuracy gain over all individual classifiers as well as an equal weights combination. We expect the Brier score to be an appropriate metric due to its interpretation as the MSE in probability estimation.

5.2.2. Classifier Weighting Method

Input to the method is a labeled classification data set with s observations and k classification models. The output is $\hat{\mathbf{w}}$, the estimate of the out-of-sample OW with $\hat{\mathbf{w}} \in \mathbb{R}^k$ and $\sum_{l=1}^{k} \hat{w}_l = 1$. The number of classes in the data set is denoted by m. A portion of the input data is held out, resulting in two subsets, the training set with

s_t observations and the holdout set with s_h observations. The split is performed stratified, i.e. the class distributions in the training and holdout set are practically equal.

All classifiers are fitted on the training set. Each classifier $l \in \{1, \ldots, k\}$ then makes probability predictions $\hat{Y}_{hl} \in [0, 1]^{s_h \times m}$ on the holdout set. These predictions are flattened into a prediction vector $\hat{\mathbf{y}}_{hl}$ of length $s_h m$, which contains predicted probabilities for each instance $u \in \{1, \ldots, s_h\}$ and for each class $i \in \{1, \ldots, m\}$. $\mathbf{y}_h = (y_{11}, \ldots, y_{s_h m})'$ denotes the vector of true outcomes in the holdout set. For each instance, $\mathbf{y}_h$ contains 1 for the actual class label of the instance, and 0 for all other class labels. Figure 5.1 shows an example of two prediction vectors and the true outcome. For simplicity, we omit the h subscript in this and the following figures.

		$\hat{\mathbf{y}}_1$	$\hat{\mathbf{y}}_2$	$\mathbf{y}$
Instance	Class			
1	1	0.3343	0.2531	0
	2	0.1396	0.3511	0
	3	0.5261	0.3958	1
2	1	0.0192	0.0982	0
	2	0.4492	0.4895	1
	3	0.5316	0.4123	0
3	1	0.2614	0.1163	0
	2	0.4296	0.4690	1
	3	0.3091	0.4148	0

Figure 5.1.: Example for prediction vectors and actual outcomes with $s = 3$ observations, $m = 3$ classes, and $k = 2$ classification models.

For each classifier l, the vector $\hat{\mathbf{y}}_{hl}$ of predicted probabilities is then compared to the vector $\mathbf{y}_h$ of actual outcomes. The error vector for classifier l is calculated as $\mathbf{e}_{hl} = \mathbf{y}_h - \hat{\mathbf{y}}_{hl}$. For each of the k classifiers, this error vector is computed, yielding an error matrix $E_h = (\mathbf{e}_{h1}, \ldots, \mathbf{e}_{hk})$. Calculating OW from those error vectors can be shown to minimize the Brier score in-sample, analogous to the MSE in a regression setting. Figure 5.2 displays the error matrix for the predictions from Figure 5.1.

With Σ_h as the variance-covariance matrix of E_h, the OW estimate $\hat{\mathbf{w}}$ can be computed using Equation (2.6). The variance-covariance matrix of the error matrix from Figure 5.2 is given by

$\mathbf{e}_1$	$\mathbf{e}_2$
-0.3343	-0.2531
-0.1396	-0.3511
0.4739	0.6042
-0.0192	-0.0982
0.5508	0.5105
-0.5316	-0.4123
-0.2614	-0.1163
0.5704	0.5310
-0.3091	-0.4148

Figure 5.2.: Example for error matrix based on the predictions in Figure 5.1.

$$\Sigma_h = \begin{pmatrix} 0.1789 & 0.1730 \\ 0.1730 & 0.1825 \end{pmatrix}.$$

The weight vector estimated in this example is $\hat{\mathbf{w}} = (0.6163, 0.3837)'$. Finally, training and holdout sets are concatenated again, and all k individual classifiers are re-fitted on all s observations. This is to ensure that all models can process as many observations as possible in the training phase. The classifiers' probability predictions on new, unseen data are subsequently combined using the weight vector $\hat{\mathbf{w}}$ estimated on the training set.

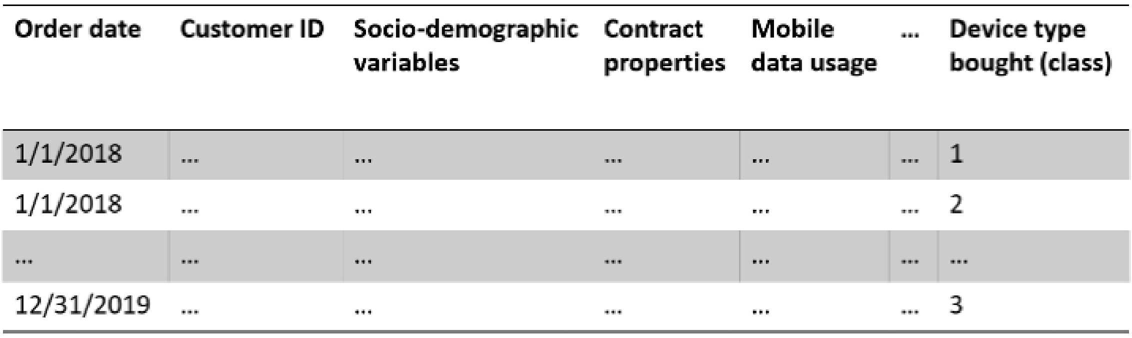
Many classification algorithms yield class membership scores that can be used to rank observations based on their likelihood to pertain to a certain class. However, those scores in general cannot be interpreted as proper probability estimates, since they are not well-calibrated, i.e. predicted class membership scores do not match ex-post probabilities (Niculescu-Mizil and Caruana, 2005). While this is not an issue for class predictions of single classifiers, in classifier ensembles it is important to have reliable probability estimates. Therefore, we compare the OW estimation with and without calibration. For the calibration setting, we use isotonic regression as introduced by Zadrozny and Elkan (2002).

5.3. Experimental Design

We now describe the experiments conducted to evaluate the proposed weighting approach. Section 5.3.1 describes the use case and data set used for evaluation. Section 5.3.2 introduces the individual classifier methods used for the WHRS. In Section 5.3.3, details about the experiments and evaluation criteria are given.

5.3.1. Use Case and Data Set

For the evaluation of the proposed classifier weighting scheme, we use a proprietary real-world data set from a large European telecommunications provider. Figure 5.3 displays the schema of the data set. It contains several hundred thousand purchases of mobile devices by customers. All purchases occurred in the years 2018 and 2019. In the figure, the last column represents the target variable, the first two columns are metadata for identification, and the columns in between are predictors.



Order date	Customer ID	Socio-demographic variables	Contract properties	Mobile data usage	...	Device type bought (class)
1/1/2018	...	...	...	...	...	1
1/1/2018	...	...	...	...	...	2
...	...	...	...	...	...	...
12/31/2019	...	...	...	...	...	3

Figure 5.3.: Schematic display of the real-world data set used for evaluation. The proprietary data set stems from a large European telecommunications provider.

The mobile devices are divided into three categories. The goal is to predict, for each of the $m = 3$ categories, the conditional probability that the given customer will select the respective category, given a purchase. The category with the highest estimated probability is then recommended. The most frequent of the three class labels occurs in 48% of cases in the data set. Thus, a simple classifier which always predicts that label would already achieve an accuracy score of 48%, which can serve as a lowest bound for more sophisticated models.

The data set contains more than 40 predictor variables, consisting of customer properties such as sociodemographics, characteristics of the customer's contract, and aggregated behavioral information such as mobile data usage. The data types of the predictors are mixed, comprising binary, integer, real-valued as well as categorical variables. All values of the predictors were measured immediately before the respective purchase, representing a snapshot of the respective customer and contract in order to recognize purchasing patterns.

5.3.2. Individual Classifiers

This section describes the individual models used to test the weighting method. Since the model's inputs are vectors representing customers via their respective properties, the method used here can be classified as demographic filtering (e.g. Pazzani, 1999), although the predictors do not only contain demographic information. In total, $k = 7$ classifying algorithms were combined, which are briefly outlined here. We used the implementations in the Python package `scikit-learn` (Pedregosa et al., 2011) for the individual classifiers.

- **Logistic regression**: The logistic regression model assumes a linear relationship between the predictor variables and the log-odds of the positive outcome of a binary dependent variable (e.g. Hastie et al., 2009). The model can be extended for non-binary classification either fitting a one-vs-all model for each class label or minimizing the multinomial logistic loss. The latter is used here.

- ***k*-nearest neighbors classifier**[7]: A k-nearest neighbors classifier predicts, for a given instance to classify, the class which most occurs in the k training points with the smallest distance to that instance (Altman, 1992). For probabilistic predictions, the class distribution of those k instances is predicted. The distance metric used here is the Euclidean distance, and the number of neighbors considered is set to $k = 5$.

- **Multi-layer perceptron**: A multi-layer perceptron is a frequently-used form of neural networks, consisting of an input layer with p nodes (the number of features), an output layer with m nodes (the number of classes), and one or more hidden layers (e.g. Hastie et al., 2009). The nodes of the hidden

[7] Note that in this bullet point only, k represents the number of neighbors. In the rest of this chapter and thesis, k is used to denote the number of individual models combined in the WHRS.

layer use a nonlinear activation function, in our case the rectified linear unit $f(x) = \max\{0, x\}$. The weights between nodes are initialized randomly and then sequentially updated using the backpropagation algorithm, which computes the gradient of the loss function with respect to each weight. The weight optimization is done using the efficient stochastic gradient descent method *Adam* (Kingma and Ba, 2014), and the maximum number of iterations is set to 1000.

- **Decision tree**: The decision tree algorithm (Breiman et al., 1984) learns simple "if-else" style decision rules by recursively splitting the data set with respect to a certain variable and value in order to create subsets which are more pure in terms of class distribution. We use a maximum depth of 5 in order not to overfit the training set.

- **Random forest**: The random forest algorithm (Breiman, 2001) fits an ensemble of decision trees on the training data. By randomly selecting bootstrap samples of data and randomly selecting a subset of variables available for splitting at each node, the trees in the ensemble are partially independent, reducing the model variance and thus alleviating a single decision tree's tendency to overfit the training data. For predicting probabilities on new data, the average of predicted probabilities of all trees in the ensemble is calculated. We choose a number of 100 trees with a maximum depth of 5 for the forest.

- **AdaBoost**: AdaBoost (Freund and Schapire, 1997) is an ensemble method which fits simple base learners sequentially, where in each iteration, weights for previously misclassified instances are increased such that the next base learner is forced to focus on more difficult cases. For prediction, the outputs of all base learners are aggregated. We use 50 decision trees with a depth of 1, also known as "decision stumps".

- **Gradient boosting**: Gradient boosting (Friedman, 2001) sequentially builds an additive model. In each iteration, m (the number of classes) regression trees are fitted on the negative gradient of the loss function, which for probabilistic outputs is the deviance. We choose a value of 100 iterations.

5.3.3. Evaluation and Benchmarks

As mentioned in Section 5.3.1, the prediction task in this business case is to recommend one of $m = 3$ possible categories of mobile devices to each customer in the test set. Therefore, an ensemble of three-class classifiers is used. In order to evaluate the classifier weighting approach we propose, the following methods are compared:

- **Individual classifiers**: For each of the seven models described in Section 5.3.2, the individual performance on the test set is calculated.
- **SA**: An equal weights average of the predictions of all seven classifiers on the test set is used as a benchmark for the hybrid approach.
- **OW estimate**: This is the approach proposed in this chapter (see Section 5.2.2 for details).
- **Out-of-sample OW**: The linear weight vector with ex-post minimal Brier score, calculated on the test set, serves as an upper performance bound for any linear weight vector. The goal of the OW estimate is to come as close as possible to this performance.

For each of the mentioned methods, there are two treatments: First, the probabilistic predictions are taken as-is. Second, the predicted probabilities are calibrated using isotonic regression before making predictions or combining the probabilistic predictions. The un-calibrated and calibrated treatments are compared.

In order to evaluate the weighting approach and compare it to other strategies, 10% of the data set is used as a test set. Another 10% of the data set is used as a holdout set to calculate out-of-sample errors of the individual classifiers in order to estimate OW, as described in Section 5.2.2. This leaves 80% of the data set as a training set.

We use two evaluation metrics in the experiment: the accuracy score and the Brier score which is also used for weight estimation. Those two metrics are chosen because the proposed weighting approach aims at increasing the accuracy of a classifier ensemble over all individual components as well as an SA combination by minimizing the Brier score. As mentioned in Section 5.2.1, we expect the minimization of the Brier score to result in a significant accuracy gain.

For reasons of robustness, the experiment is repeated ten times with random training-holdout-test allocations. Accuracy and Brier scores are averaged over those runs, and their standard deviations are reported.

5.4. Empirical Evaluation

This section contains the experimental results of comparing the proposed classifier weighting method to the aforementioned benchmarks. Section 5.4.1 describes the data preparation, i.e. checking the model requirements. Section 5.4.2 reports the results.

5.4.1. Data Preparation

As mentioned above, the weighting method requires unbiased individual estimators (mean errors of 0) and multivariate normal error vectors. We described in Section 5.2.1 that the mean errors are 0 when using the flattened deviation between predicted probabilities and true outcomes as error vectors. Now, we inspect the distribution of deviations.

Figure 5.4 displays, for each individual classifier, a histogram of out-of-sample errors, using ten equal-width bins. The vertical axes are not labeled since the number of observations in the test set would give away the number of observations in the entire data set (see Section 5.3.1). The figure shows the histograms when feeding error vectors into the weight estimation as-is, i.e. without calibration. It is clear to see that the errors are not normally distributed. Some of the classifiers partially exhibit a bell-shaped distribution, however all with a gap around 0. Others, especially AdaBoost, are nowhere near bell-shaped.

However, applying the mentioned classifier calibration technique using isotonic regression changes the error distribution. Figure 5.5 shows the same plot, but this time after isotonic calibration of each classifier. While there is still a gap near 0, all distributions now exhibit a bell-shaped form. They are still not normally distributed, but the calibration helps to better approach the assumption.

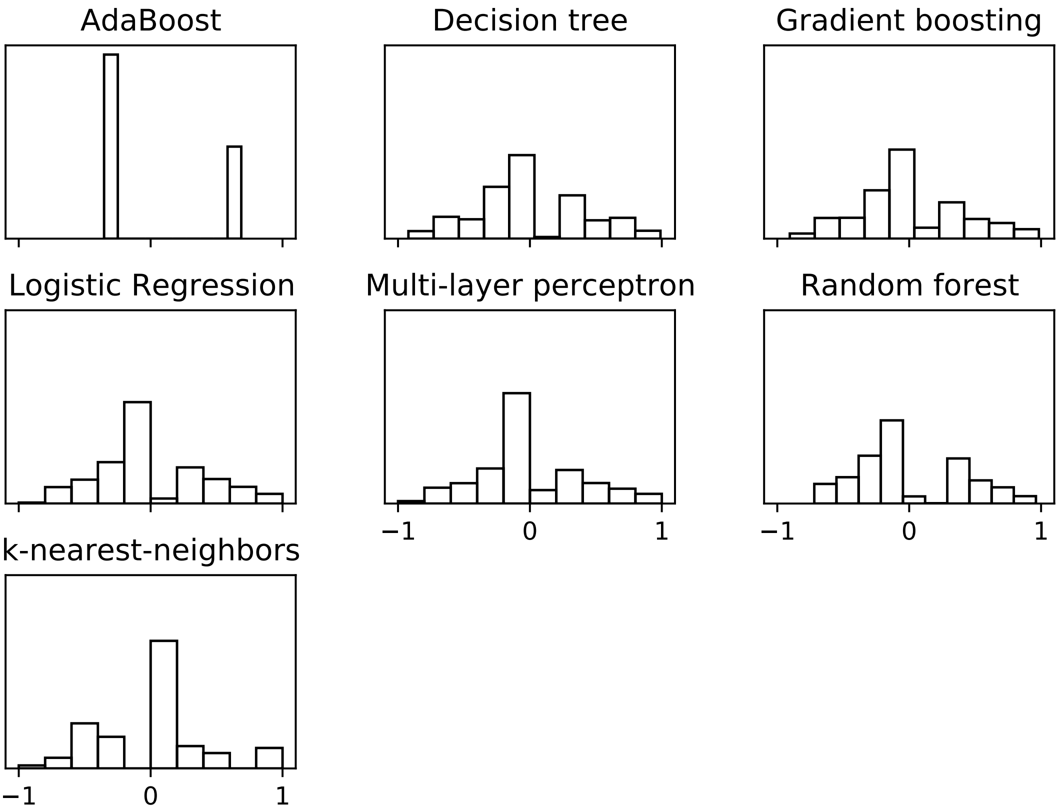


Figure 5.4.: Histograms of all individual classifiers' out-of-sample errors using ten equal-width bins. The vertical axes are not labeled in order not to disclose the confidential number of observations in the proprietary data set.

5.4.2. Results

Table 5.1 displays the results of comparing the WHRS using the OW estimation introduced in this chapter, a WHRS using SA combination, and the individual classifiers. Both for the accuracy and the Brier score, the mean and standard deviation over ten runs are reported. For better comparability, the percentage differences between the OW estimation and the other methods is also reported for both metrics (columns "Diff."). As mentioned in Section 5.3.3, in addition to the individual classifiers, the SA combination and the OW estimate, the results using out-of-sample OW are also reported (last row) as an upper bound for the performance of a linear weighting vector.

The table shows that the combination using OW estimation clearly outperforms all individual methods as well as the SA combination. The best-performing individual

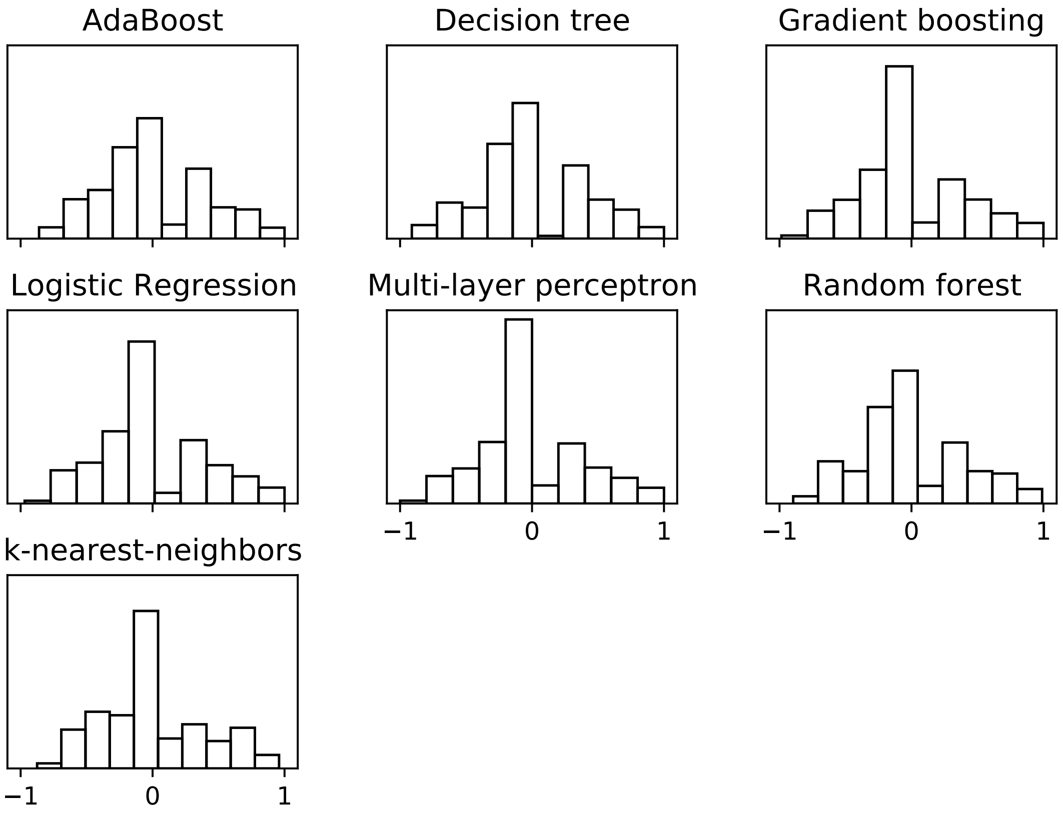


Figure 5.5.: Histograms of all individual classifiers' out-of-sample errors using ten equal-width bins after applying isotonic calibration. The vertical axes are not labeled in order not to disclose the confidential number of observations in the proprietary data set.

classifier, which is the neural network with an accuracy of 66.8%, has a 2.4% lower accuracy and a 4.99% higher Brier score than the HRS using the OW estimate. An SA combination using equal weights slightly outperforms the best individual method, but leads to a 2.2% lower accuracy and a 6.25% higher Brier score than estimated OW.

The results also show that the estimated OW vector is very close in both accuracy (0.07% lower) and Brier score (0.07% higher) to the ex-post, out-of-sample OW vector. This indicates that the weighting approach proposed in this study can yield weight vectors that are close to the best possible linear weighting HRS.

Table 5.2 displays the results when all classifiers' probability estimates are calibrated using isotonic regression before estimating OW. After calibration, all classi-

Table 5.1.: Comparison of performance between a hybrid recommender system using optimal weight (OW) estimation, a simple average (SA) combination, and all individual classifiers.

Method	Accuracy (std.)	Diff.	Brier score (std.)	Diff.
Logistic regression	0.6440 (0.0012)	+6.14%	0.1595 (0.0004)	-10.70%
k-nearest-neighbors	0.6280 (0.0016)	+8.84%	0.1677 (0.0004)	-15.09%
Multi-layer perceptron	0.6675 (0.0015)	+2.40%	0.1499 (0.0005)	-4.99%
Decision tree	0.6463 (0.0013)	+5.76%	0.1600 (0.0003)	-10.98%
Random forest	0.6430 (0.0014)	+6.30%	0.1649 (0.0004)	-13.63%
AdaBoost	0.6503 (0.0012)	+5.10%	0.2187 (0.0000)	-34.88%
Gradient boosting	0.6590 (0.0013)	+3.71%	0.1529 (0.0003)	-6.88%
Simple average	0.6688 (0.0012)	+2.20%	0.1519 (0.0002)	-6.25%
Optimal weight estimate	0.6835 (0.0013)		0.1424 (0.0003)	
Ex-post optimal weights	0.6840 (0.0013)	-0.07%	0.1423 (0.0003)	+0.07%

fiers' Brier scores are in a range between 0.15 and 0.16. Their individual accuracy scores are not significantly affected, with one exception: The calibrated nearest-neighbors classifier has an accuracy of 64.8%, as compared to 62.8% for the non-calibrated version. This indicates that the calibration changed the order of the class ranking for some instances, leading to a higher number of correct class predictions.

As for the WHRSs, the SA combination of calibrated classifiers has a slightly better (smaller) Brier score and worse (smaller) accuracy than the non-calibrated SA combination. Now, the best individual classifier, which is again the neural network, slightly outperforms the SA combination in terms of accuracy. For the estimated OW, the Brier score is virtually unchanged, while the accuracy slightly increases when using calibration. This is probably caused by the nearest-neighbors classifier's gain in accuracy. The performance increase of the OW estimation over the best classifier (2.68%) as well as the SA (3.07%) is even higher than in the no-calibration treatment.

Table 5.2.: Results analogous to Table 5.1 after applying isotonic calibration to all individual classifiers.

Method	Accuracy (std.)	Diff.	Brier score (std.)	Diff.
Logistic regression	0.6440 (0.0014)	+6.40%	0.1593 (0.0004)	-10.83%
k-nearest-neighbors	0.6484 (0.0017)	+5.69%	0.1561 (0.0003)	-9.01%
Multi-layer perceptron	0.6673 (0.0015)	+2.68%	0.1500 (0.0005)	-5.29%
Decision tree	0.6460 (0.0013)	+6.07%	0.1600 (0.0003)	-11.21%
Random forest	0.6461 (0.0018)	+6.05%	0.1593 (0.0003)	-10.80%
AdaBoost	0.6487 (0.0011)	+5.63%	0.1579 (0.0003)	-10.03%
Gradient boosting	0.6590 (0.0012)	+3.97%	0.1528 (0.0003)	-7.03%
Simple average	0.6648 (0.0015)	+3.07%	0.1500 (0.0002)	-5.29%
Optimal weight estimate	0.6852 (0.0013)		0.1421 (0.0003)	
Ex-post optimal weights	0.6861 (0.0013)	-0.12%	0.1419 (0.0003)	+0.12%

5.5. Discussion

In this section, we discuss and interpret the results obtained from the experimental evaluation in the previous section.

First, as apparent in Table 5.1, the technique of estimating OW using a subset of the available training data and then applying the learned weighting to the full data set clearly outperforms the SA combination as well as all individual methods. Although seven classification models were combined, meaning a weight vector of length $k = 7$ had to be estimated, the estimated OW comes very close to the ex-post OW, both in terms of accuracy and Brier score. This is probably due to the rather large data set used in the experiments, leading to robust weight estimates. Large e-commerce vendors usually have large data sets available, making the proposed approach a feasible and effective means to boost predictive accuracy.

In the use case of this study, the proposed classifier weighting approach was able to increase the accuracy over the best individual classifier as well as SA by more than 2%. The impact of such an improvement depends on the business case at hand. For the project partner that provided the data set, this improvement is significant. Due to the high number of customers, being able to predict the right purchase in 2% more of the cases can lead to a considerable profit enhancement. Other large corporations, especially in e-commerce, could benefit in a similar way. On the other hand, for smaller companies with fewer customers as well as smaller data sets, other factors are more important, such as the model interpretability.

The theoretical model requirements were not entirely fulfilled in the use case, since the classifier error vectors were not normally distributed. Real-world use cases often differ substantially from theoretical requirements, which is why many approaches do not work well under those circumstances. However, the proposed approach was still able to reach an accuracy and Brier score very close to the ex-post best possible, and to improve performance over individual classifiers and an SA combination. This shows that the classifier weighting is well suited for practitioners, even if the data is messy, as it often is in practice.

The classifier weighting method can be integrated into existing machine learning pipelines rather easily. Due to its analytical nature, it does not require expensive computations, and the weights are readjusted automatically without regular human intervention. Therefore, the cost-benefit ratio calculates favorably. The potential of gaining significant performance was shown in this study, and because of the added robustness, there is minimal risk of losing accuracy given sufficient data.

Second, as can be seen when comparing Table 5.1 to Table 5.2, although calibrating the probability estimates of the individual classifiers using isotonic regression led to decreasing Brier scores of the individual models (especially AdaBoost), it did not lead to significant differences in the accuracy of the OW combination. This is probably because almost all classifiers already had mostly well-calibrated scores.

Finally, while in general, a minimal Brier score does not automatically lead to the highest accuracy, in our case, the methods with the lowest Brier score (neural network for the individual methods and OW estimate for the HRSs) did have the highest accuracies as well. Especially the OW estimate, which aims at minimizing the Brier score of an HRS, outperformed all other models by more than 2% in terms of accuracy. This indicates that selecting combination weights based on the Brier score is a good strategy for creating accurate HRSs and therefore confirms our last assumption from Section 5.2.1.

5.6. Conclusion

In this chapter, we presented an approach to estimate optimal combination weights for WHRSs in a classification context, e.g. for purchase prediction. The weighting method fits all individual classifiers on a subset of the available training data and calculates out-of-sample errors of probabilistic predictions on the rest of the training data. Using the variance-covariance matrix of those out-of-sample errors, a weight vector is calculated which is optimal on the holdout set. Then, all classifiers are re-fitted on the entire available data set, and the calculated weight vector is used for combining predictions on new, unseen data.

Results on a real-world e-commerce data set show that this approach significantly outperforms both an SA combination, assigning equal weights to all components, as well as all individual classifiers. This is an encouraging finding, indicating that OW estimated using the Brier score is an adequate and simple method for increasing accuracy and robustness of classifiers.

This study contributes to research and practice. First, a novel and accurate analytical weighting scheme for classifiers is proposed. It contributes to the literature on WHRSs as well as classifier ensembles in general. For practitioners, especially companies with many customers and large data sets as well as different classifying models in use, the method provides a computationally efficient means of increasing accuracy and robustness, and thus revenue and profit, without requiring great effort to set up or maintain.

Commonly, e-commerce companies already test and compare different algorithms with the goal of maximizing predictive accuracy. Depending on the size of a company and its number of customers, an accuracy increase as small as 1% can lead to a significantly higher profit. The proposed approach offers a simple and efficient means to combine their existing methods and thus achieve higher levels of performance and profit.

As described in Section 1.1, both the importance of accurate RSs and the benefits of HRSs have been demonstrated in IS research. The weighting method proposed in this chapter therefore provides a relevant contribution both to the existing body of research and to practitioners, mainly large companies with substantial data available and many customers.

As a limiting factor, we did not engage in extensive hyper-parameter tuning for the individual classification algorithms, since the goal of this study was to demonstrate the improvement of a WHRS using an OW estimate over all individual models as well as an SA combination. In addition, we did not perform any feature selection or engineering. Performing both of those tasks might have improved the accuracy of the HRS even further.

The following and final part summarizes the contributions of this thesis, addresses the research questions, and proposes directions for future research.

Part III.

Finale

Chapter 6.

Conclusion

In today's world shaped by information overload and a large number of options in many areas of life, decision support systems and information filtering are becoming ever more important. RSs provide a means of automatically pre-selecting a set of interesting items in a personalized manner for each user. The accuracy of RSs increases the perceived personalization and the likelihood of users accepting the RS as a decision aid. In addition, each RS algorithm has its unique benefits and shortcomings and is, in certain situations, unable to generate recommendations.

The combination of multiple RSs into a WHRS is a means of mitigating those shortcomings as well as increasing accuracy and robustness of recommendations. While the advantages of WHRSs have been shown in the literature, little prior research has dealt with the estimation of OW for maximizing the WHRS's accuracy. In this thesis, multiple approaches were proposed to advance the research on weight selection in WHRSs. In this final chapter, the contributions of the thesis are summarized, and interesting directions for future research, possible extensions and additional application opportunities of the proposed methods are discussed.

6.1. Summary of Contributions

In Chapter 3, a weighting scheme for minimizing the RMSE of a WHRS with $k = 2$ individual RSs was developed. It was shown to outperform, on the MovieLens 1M data set, all individual RS algorithms as well as an SA combination and a more robust weighting scheme which uses the inverse ratio of error variances of the two algorithms as the weight estimate.

Those results and insights allow us to address the first research question of this thesis, which is subdivided into three parts. We begin with *RQ 1a: Given the assumptions of OW, does a weighting scheme for a WHRS with two individual RSs based on OW estimation lead to lower errors than each individual RS as well as an SA combination on an RS benchmark data set?*

While the assumptions of OW, i.e. multivariate normal errors with mean 0, were not entirely met in the study of Chapter 3, the requirements were still approximately fulfilled such that a transformation of the individual RSs' errors did not make a difference in the accuracy of the WHRS. Regarding the accuracy of the proposed weighting scheme, the question can be answered positively: The OW estimation method significantly outperformed, for all six pairs of RSs respectively, the two individual RSs as well as an SA combination. MovieLens 1M, a well-established benchmark data set in RS research, was used for the empirical evaluation.

The second part of the research question is *RQ 1b: Can the weighting scheme based on OW estimation lead to lower errors than the more robust modified OW estimate ignoring error correlations on the benchmark data set?*

This question can also be answered positively. While the modified OW estimate ignoring error correlation led to lower errors than an SA combination, it was still outperformed significantly by the OW estimate which considers the error correlation.

Finally, we address *RQ 1c: Does the forecast combination puzzle, i.e. OW being outperformed by more robust weighting schemes, occur in the combination of RSs on the benchmark data set, and if so, for which number of ratings?*

The forecast combination puzzle did show when repeating the experimental evaluation of the proposed weighting scheme on smaller subsets of the MovieLens 1M data set containing 50% and 10% of the ratings, respectively. While overall, OW was still the most accurate method, the differences in RMSE to SA and modified OW decreased with the number of observations in the data set. For the subset with 10%, OW was on average the best strategy, but was for one pair of RSs outperformed by the more robust modified OW estimate as well as an SA combination.

In Chapter 4, the weighting scheme from Chapter 3 was extended in two ways, namely (a) to be able to combine more than two RS algorithms at once, and (b) to mitigate the bias which stems from estimating OW based on the individual RSs fitted on a subset of available ratings, but then using the learned weights with the individual RSs fitted on all available ratings. The MovieLens 1M data set was again used for evaluation, and it was shown that (a) the combination of $k = 4$ individual RSs increases the boost in accuracy over the individual algorithms, and that (b) the weight projection method led to smaller errors compared to the weighting approach without projection.

Based on those results, we address the second research question of this thesis, which is again subdivided into three parts. We start with *RQ 2a: How does the accuracy of an OW combination on the benchmark data set develop in relation to the individual RSs when the number of RSs in the WHRS is increased from 2 to 4?*

While the estimation of an OW vector of length $k = 4$ instead of 2 increases the degrees of freedom and therefore also the margin of error, the extended weighting approach was able to estimate very accurate weights, resulting in an even greater increase of accuracy than in Chapter 3 over the individual RS algorithms. While in Chapter 3, an OW combination of the two best RSs decreased the RMSE by 0.6% compared to the best individual algorithm, in Chapter 4, an OW combination of all four RSs decreased the RMSE by more than 1% in comparison to the best individual method.

The second part of the second research question is *RQ 2b: Assuming perfect extrapolation, what is the performance improvement potential of a weight projection approach over a conventional OW estimate?*

This question can be answered with insights from Section 4.2.3. There, we analyzed the improvement potential of the weight projection approach depending on different characteristics of the data set and the individual RSs. We found out that the improvement potential is larger when the error correlations between the individual RSs is lower, when their initial ratio of RMSEs is more similar, and when the development curve of their RMSEs over the number of training observations is more heterogeneous. Given favorable values of those parameters, the projection approach can decrease the RMSE of a combination of $k = 2$ RSs by more than 10% of the better individual RS, compared to the conventional estimation of OW.

This leads to *RQ 2c: Does the projection of error covariances or OW lead to a lower error than the conventional OW estimate on the benchmark data set?*

In the empirical evaluation on the MovieLens 1M data set, the circumstances were unfavorable due to high error correlations and similar RMSE developments. Still, the projection method consistently led to lower errors than the conventional method of estimating OW, introduced in Chapter 3. The direct projection outperformed the naive forecast by a factor of about 6 in terms of the difference to the best RMSE, while for the projection of the error covariance, the factor was about 17.

Finally, in Chapter 5, the weighting method from Chapter 3 was adapted to be used in a classification setting for predicting a given customer's next purchase. The weighting scheme was developed to calculate weighted averages of the classifiers' estimated probabilities of an instance pertaining to each class. The Brier score was used as the classification equivalent of the MSE, and it was shown on a real-world data set from a large European telecommunications provider that the classifier weighting approach led to a significantly higher accuracy score than all individual classifiers as well as an SA combination.

Based on the results and insights from that study, we now address the third and final research question of this thesis, which is subdivided into three parts as well. We start with *RQ 3a: Can the estimation of OW for the combination of regressors into a WHRS be adapted to be used for the combination of classifiers?*

This question can be answered positively. The OW estimation approach for WHRSs introduced in Chapter 3 is adaptable to classification scenarios, as shown in this chapter. Combining the probability predictions of multiple classifiers using the Brier score as the equivalent to the MSE allows to learn weights which outperform all individual classifiers as well as an SA combination significantly.

The second part of the third research question is *RQ 3b: Does the minimization of the Brier score of a WHRS comprised of classifying algorithms lead to an increase of the accuracy score over all individual classifiers as well as an SA combination?*

The answer to this question is also positive. While in general, the classifier with the lowest Brier score is not guaranteed to have the highest accuracy score, the Brier score proved to be an adequate proxy for the accuracy score in the experimental evaluation of the proposed classifier weighting on the real-world telecommunications data set: The method with the lowest Brier score had the highest accuracy score in all cases. This holds true both for the individual classifiers and the WHRSs. The minimization of the Brier score for estimating OW led to a significant increase in the accuracy score.

Finally, we address the third and last part of the third research question, *RQ 3c:*

Does the calibration of the classifiers' probability estimates lead to an improvement in estimating OW in a WHRS?

Comparing Table 5.1 to Table 5.2, it can be seen that the intermediate step of calibrating all classifiers' probability scores before combining them into a WHRS does not lead to a significant improvement, although the accuracy score of the calibrated OW WHRS is slightly higher than that of the un-calibrated OW WHRS. Due to the small difference, the disadvantages of the additional estimation step, which introduces more degrees of freedom, outweigh the advantages.

6.2. Future Research

In this final section of the thesis, directions for future research are proposed. One interesting alley for further work is the more detailed analysis of the bias-variance trade-off in the weighted combination of RSs. In this thesis, it was shown that given a sufficient number of ratings or purchases as an input, weights can be learned which are very close to the out-of-sample best possible linear weighting vector. This was the case for the combination of two, but also for four individual RSs, and even for seven classifiers. The more models are combined, the higher the degrees of freedom and thus the higher the model variance of OW in comparison to more robust weighting schemes like SA or the modified OW estimate. However, it was also shown that when the number of training observations decreases, the forecast combination puzzle is visible as the advantage of OW over other weighting methods declines. Future research could analyze the relationship between the size of the training set, the number of RSs combined, and the relative performances of OW, SA, modified OW, and OW using direct or covariance projection. In situations with less training data or more combined models, the shrinkage of OW toward SA might further improve accuracy and robustness. Linear shrinkage methods, i.e., linear combinations of OW and SA weights, as well as non-linear approaches to shrink OW towards SA might be explored.

Another direction for future work is the application of the proposed weighting schemes in areas other than RS. The approach introduced in Chapter 3 was adapted from the forecast combination literature. However, we are not aware of a weight projection method such as the one proposed in Chapter 4, neither in WHRS research nor in the forecast combination or machine learning literature. While the technique's benefits and potentials were shown in a rating prediction context in this thesis, there is no reason why it should not be able to increase the accuracy of an ensemble of statistical or machine learning algorithms for other regression tasks.

Similarly, the classifier weighting scheme introduced in Chapter 5 can be used for any classification task in order to maximize the accuracy of an ensemble of classifiers by optimizing the weights with which the predicted class probabilities are combined.

Apart from that, an interesting direction for future studies is to test the approach using other algorithms, e.g. implicit feedback CF methods (which would require other data sets), and study whether it is also able to improve an ensemble of RSs in terms of other metrics, such as ranking metrics which are often relevant in a top-N recommendation setting.

Regarding the projection approach proposed in Chapter 4, future work might investigate the effect of this method when applied to different individual RS algorithms. We expect further improvements with the projection weighting method if RSs have lower error correlations. In addition, further improvements can be expected in case the developments of the RMSEs over the number of training observations are more heterogeneous among the components of the WHRS. This could also be achieved by using a different data set containing additional independent input data.

The two proposed projection methods also have a bias-variance trade-off: The covariance projection has a lower bias, but a higher variance, due to the higher degrees of freedom when extrapolating the error covariance rather than just the weight vector. Our supposition is that the direct projection approach might be better suited in situations with smaller data sets or a higher number of individual RSs due to its higher robustness. Even a combination of the two approaches, e.g. via linear shrinkage, is worth exploring.

Finally, the idea of measuring optimal hyper-parameter values for increasing numbers of training observations and projecting those to the full size of the available data set is not restricted to weight estimation in linear combination, but can also be investigated for the selection of other parameters such as regularization or shrinkage factors.

Bibliography

Adomavicius, G. and Tuzhilin, A. (2005). Toward 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.

Aggarwal, C. C. (2016). *Recommender Systems*. Springer International Publishing, Cham.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Bates, J. M. and Granger, C. W. (1969). The combination of forecasts. *Journal of the Operational Research Society*, 20(4):451–468.

Blanc, S. M. and Setzer, T. (2016). When to choose the simple average in forecast combination. *Journal of Business Research*, 69(10):3951–3962.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. CRC press.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370.

Çano, E. and Morisio, M. (2017). Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, 21(6):1487–1524.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M. (1999). Combing content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*, Berkeley, CA.

Clemen, R. T. (1989). Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5(4):559–583.

De Campos, L. M., Fernández-Luna, J. M., Huete, J. F., and Rueda-Morales, M. A. (2010). Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785–799.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Ghazanfar, M. and Prugel-Bennett, A. (2010). An improved switching hybrid recommender system using Naive Bayes classifier and collaborative filtering. In *Proceedings of the International Multiconference of Engineers and Computer Scientists*, Hong Kong.

Gigone, D. and Hastie, R. (1997). Proper analysis of the accuracy of group judgments. *Psychological Bulletin*, 121(1):149–167.

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

Harper, F. M. and Konstan, J. A. (2015). The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):1–19.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Science & Business Media.

Haubner, N. and Setzer, T. (2020). Applying optimal weight combination in hybrid recommender systems. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*.

Haubner, N. and Setzer, T. (2021). Hybrid recommender systems for next purchase prediction based on optimal combination weights. In *Proceedings of the 16th International Conference on Wirtschaftsinformatik*.

Hill, G. W. (1982). Group versus individual performance: Are N+1 heads better than one? *Psychological Bulletin*, 91(3):517.

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10.

Howe, A. E. and Forbes, R. D. (2008). Re-considering neighborhood-based collaborative filtering parameters in the context of new data. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 1481–1482.

Jahrer, M., Töscher, A., and Legenstein, R. (2010). Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 693–702. ACM.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning*, volume 112. Springer.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Komiak, S. Y. and Benbasat, I. (2006). The effects of personalization and familiarity on trust and adoption of recommendation agents. *MIS Quarterly*, 30(4):941–960.

Koren, Y. and Bell, R. (2015). Advances in collaborative filtering. In Ricci, F., Rokach, L., and Shapira, B., editors, *Recommender Systems Handbook*, pages 77–118. Springer US, Boston, MA.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Makridakis, S. and Hibon, M. (2000). The M3-Competition: Results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476.

Malgonde, O., Zhang, H., Padmanabhan, B., and Limayem, M. (2020). Taming the complexity in search matching: Two-sided recommender systems on digital platforms. *MIS Quarterly*, 44(1a):48–84.

Mannes, A. E., Soll, J. B., and Larrick, R. P. (2014). The wisdom of select crowds. *Journal of Personality and Social Psychology*, 107(2):276–299.

Niculescu-Mizil, A. and Caruana, R. (2005). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 625–632.

Ning, X., Desrosiers, C., and Karypis, G. (2015). A comprehensive survey of neighborhood-based recommendation methods. In Ricci, F., Rokach, L., and Shapira, B., editors, *Recommender Systems Handbook*, pages 37–76. Springer US, Boston, MA.

Oestreicher-Singer, G. and Sundararajan, A. (2012). Recommendation networks and the long tail of electronic commerce. *MIS Quarterly*, 36(1):65–83.

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5):393–408.

Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 325–341. Springer Berlin Heidelberg, Berlin, Heidelberg.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In *Pro-*

ceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, pages 175–186.

Ricci, F., Rokach, L., and Shapira, B. (2015). Recommender systems: Introduction and challenges. In Ricci, F., Rokach, L., and Shapira, B., editors, *Recommender Systems Handbook*, pages 1–34. Springer US, Boston, MA.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Application of dimensionality reduction in recommender system - a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295.

Smith, J. and Wallis, K. F. (2009). A simple explanation of the forecast combination puzzle. *Oxford Bulletin of Economics and Statistics*, 71(3):331–355.

Timmermann, A. (2006). Forecast combinations. In Elliott, G., Granger, C. W. J., and Timmermann, A., editors, *Handbook of Economic Forecasting*, volume 1, pages 135–196. Elsevier.

Xiao, B. and Benbasat, I. (2007). E-commerce product recommendation agents: Use, characteristics, and impact. *MIS Quarterly*, 31(1):137–209.

Xu, J., Benbasat, I., and Cenfetelli, R. T. (2014). The nature and consequences of trade-off transparency in the context of recommendation agents. *MIS Quarterly*, 38(2):379–406.

Zadrozny, B. and Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 694–699.

Zhang, T. C., Agarwal, R., and Lucas Jr, H. C. (2011). The value of IT-enabled retailer learning: Personalized product recommendations and customer store loyalty in electronic markets. *MIS Quarterly*, 35(4):859–881.