The Dissertation Committee for Meghana Jayant Deodhar
certifies that this is the approved version of the following dissertation:

Simultaneous Partitioning and Modeling: A Framework for Learning from Complex Data

Committee:

Joydeep Ghosh, Supervisor

Craig Chase

Inderjit Dhillon

Lizy John

Maytal Saar-Tsechansky

Simultaneous Partitioning and Modeling: A Framework for Learning from Complex Data

by

Meghana Jayant Deodhar, B.E.;M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2010

Acknowledgments

I am filled with nostalgia as I remember myself a shy and under confident (and rightfully so) new student in the lab in 2004. The journey from there to defending my dissertation has been a remarkable, enriching one, filled with learning a multitude of concepts and more importantly developing maturity of thought. This journey was made possible and also immensely enjoyable by my advisor, Dr. Joydeep Ghosh. Several times my friends have joked that my Ph.D. was a bit too smooth. In a way they are right; everything fell in place at the right time and there were no major times of distress, all thanks to the constant support of Dr. Ghosh. The best part about working with him was that he gave me the freedom to explore, but ensured that I was on the right track and gave an impetus to my ideas, which then grew into concrete problem formulations. Another thing I am very grateful to him for is that he is always available for his students; you can walk into his office at any time and he will make an immediate context switch from whatever he was doing. Something in particular I really appreciate is the importance he attached to learning and experiencing a variety of things rather than just getting the Ph.D. done. He had nothing but encouragement for the several related and unrelated courses I took, as well as for the hours I spent on art and marathon running.

Through my time at UT, I got the opportunity to work with several professors and students, from whom I had plenty to learn. Let me begin by thanking

Gunjan Gupta, a fellow lab member, mentor, co-author and of course a friend. I am indebted to him for all his help, right from designing my first resume, formulating research ideas with me, putting me in touch with the right people for internships and helping me with my full-time job search. Considering the fact that he himself was always very busy with multiple projects, his willingness to promptly take out time to help me and relentlessly encourage me is something I truly appreciate. Next, I would like to thank Srujana Merugu, Arindam Banerjee, Suju Rajan, Kunal Punera and Chase Krumpelman, alumni of our lab, for their patience in answering all my doubts, being ever willing to talk about my research and giving me concrete ideas, helping me get datasets, getting me set up with code, and most importantly motivating me and instilling confidence in myself. A big thank you to Hyuk Cho, whom it was wonderful to work with. In his quiet way he inspired a lot of my ideas and gave me the confidence to implement them.

I was fortunate to have worked with Maytal Saar-Tsechansky, Leigh McAlister and Andrea Godfrey from the McComb's Business school. Their systematic, organized approach is something I certainly hope to take with me to my work in industry. I would also like to thank them for the datasets they provided me with, which were invaluable in getting my research started. In the past couple of semesters I had the extremely enjoyable experience of working with Clinton Jones on developing a parallel implementation of the SCOAL algorithm. A task that seemed daunting at first was accomplished quickly thanks to his systems and programming expertise. A big thank you to Manish Kaytal for getting us started with Map-Reduce and taking out time to explain things to me. I would also like to thank Aayush Sharma,

whom I actively worked with in the past few months. I value all of our engaging discussions and brain-storming sessions.

A very important component of my Ph.D. was my interaction with people in industry. Firstly, I would like to thank Yasho Rao and Dr. Ghosh for giving me the opportunity to work for their startup company, Stadia Marketing, where I got to learn SAS and got my hands dirty with real data and problems. Each of my internships, at Kosmix, eBay, Amazon and Yahoo! Labs, was a very rewarding experience. In particular I would like to thank Vijay Narayanan from Yahoo! and Sansern Thongmee from Amazon for giving me a chance to apply the data mining concepts I knew in theory to very interesting industry applications. My sincere thanks to Sansern, Rajeev Bhatia and Joseph Sirosh, who were more than managers to me at Amazon. Their mentorship even after the internship, especially during my full-time job hunt, is something I absolutely value.

I would like to thank all my colleagues in IDEALab for making the lab atmosphere fun filled and stimulating. Special thanks to Sanmi Koyejo for leading the Gnofai discussion group, which I found very interesting and learnt a lot from. A big thank you to everyone for listening to my practice talks very patiently several times and giving me very good feedback each time. Our lab system administrators, Chase Krumpelman, Goo Jun and Clinton Jones did a great job in ensuring that all of the lab machines and servers were always in good shape. Not once did they mind our questions and emails demanding help with getting things to work. Finally, all kinds of paperwork and administrative issues were always easily taken care of by Amy Levin, Debi Prather, Melissa Campo and Melanie Gulick, whom I am indeed

grateful too.

I would like to thank all my friends in Austin, who made Austin feel very much like home. Special thanks to the Godboles for helping me set up and feel comfortable during my first few months here, the Kyles and Wads for created a family-like atmosphere for me, and Harish Ganapathy and several others for all their moral support during tough times. Last but not the least, I would like to thank my family for their unfailing support, encouragement and patience. Their happiness and pride in this achievement certainly makes me feel very special.

Simultaneous Partitioning and Modeling: A Framework for Learning from Complex Data

Publication No. _____

Meghana Jayant Deodhar, Ph.D.
The University of Texas at Austin, 2010

Supervisor: Joydeep Ghosh

While a single learned model is adequate for simple prediction problems, it may not be sufficient to represent heterogeneous populations that difficult classification or regression problems often involve. In such scenarios, practitioners often adopt a "divide and conquer" strategy that segments the data into relatively homogeneous groups and then builds a model for each group. This two-step procedure usually results in simpler, more interpretable and actionable models without any loss in accuracy. We consider prediction problems on bi-modal or dyadic data with covariates, e.g., predicting customer behavior across products, where the independent variables can be naturally partitioned along the modes. A pivoting operation can now result in the target variable showing up as entries in a "customer by product" data matrix. We present a model-based co-clustering framework that interleaves partitioning (clustering) along each mode and construction of prediction models to iteratively improve both cluster assignment and fit of the models.

This Simultaneous CO-clustering And Learning (SCOAL) framework generalizes co-clustering and collaborative filtering to model-based co-clustering, and is shown to be better than independently clustering the data first and then building models. Our framework applies to a wide range of bi-modal and multi-modal data, and can be easily specialized to address classification and regression problems in domains like recommender systems, fraud detection and marketing.

Further, we note that in several datasets not all the data is useful for the learning problem and ignoring outliers and non-informative values may lead to better models. We explore extensions of SCOAL to automatically identify and discard irrelevant data points and features while modeling, in order to improve prediction accuracy. Next, we leverage the multiple models provided by the SCOAL technique to address two prediction problems on dyadic data, (i) ranking predictions based on their reliability, and (ii) active learning. We also extend SCOAL to predictive modeling of multi-modal data, where one of the modes is implicitly ordered, e.g., time series data. Finally, we illustrate our implementation of a parallel version of SCOAL based on the Google Map-Reduce framework and developed on the open source Hadoop platform. We demonstrate the effectiveness of specific instances of the SCOAL framework on prediction problems through experimentation on real and synthetic data.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Divide and Conquer: A Methodology to Solve Difficult Problems

While it is common practice to develop a single learned model (e.g., a classification or regression model, or a single ensemble of multiple base-learners) to characterize a given dataset, many prediction problems involve a heterogeneous population where a single model may not be adequate. For many such difficult problems it is practically advantageous to adopt a "divide and conquer" strategy that partitions the population into multiple, relatively homogeneous segments and then develops separate models for each segment [8, 29, 82, 72]. For example, an e-tailor may attract different types of browsers, from casual shoppers to bulk purchasers, and one may want to model their purchasing inclinations separately. Similarly while forecasting electric load usage, it is advisable to build separate predictive models for weekdays, weekends and holidays. Advantages of such divide-and-conquer approaches include improved accuracy and reliability in general. Moreover, such approaches provide improved interpretability as well, since the component models are learnt on relatively homogenous data and are often far simpler [99].

Typically such partitioning is done *a priori* based on domain knowledge or a separate segmentation routine [8, 29]. The segmentation/clustering is hence

performed independent of the classification/regression model that is subsequently developed. The objective of partitioning is to find segments where the fit of the prediction models is good. However, *a priori* partitioning does not directly achieve this objective and may be sub-optimal. The solution we propose is to interleave partitioning and construction of prediction models. We are concerned with large-scale, heterogeneous data with a *dyadic structure*, i.e., where the independent variables can be naturally decomposed into three groups associated with two sets of elements and their combination. We propose a very general framework for simultaneous co-clustering (clustering along each mode) and modeling, which we show is a very effective technique for prediction from complex data.

1.2 Motivating Example: Recommender Systems

To concretize the above discussion, consider the problem of predicting customer purchase decisions, which can be used to recommend relevant products to customers. The dataset in this case can be represented as a matrix of customers by products, where the cell values are class labels representing whether a customer buys a certain product or not. This matrix will have missing values where the corresponding customer-product choice is unknown. Each customer is described by a set of attributes, for example, demographics and each product also has attributes, which could include the price, market share, quality, etc. Attribute information could also include features specific to a customer-product pair, e.g., whether the customer saw an advertisement for the product. The dataset and problem setting are illustrated in Figure 1.1.

Data with this structure is referred to as dyadic (bi-modal) data [4] and is now pervasive in a variety of real life applications. Dyadic data consists of measurements on dyads, which are pairs of elements from two different sets (modes); customers and products in this example. The measurements can be represented as the entries of a matrix, whose rows and columns are the two sets of elements. The independent variables, also referred to as covariates are associated with the entities along the two modes and their combination. In this thesis we are concerned only with Dyadic data with Covariates present (**DyaC** data). In the customer-product example, the covariates include the customer attributes, product attributes as well as attributes associated with customer-product pairs. Datasets consisting of web page-ad clicks, search query-web page relevance, user-movie ratings or customer-product preferences are other examples of DyaC data.

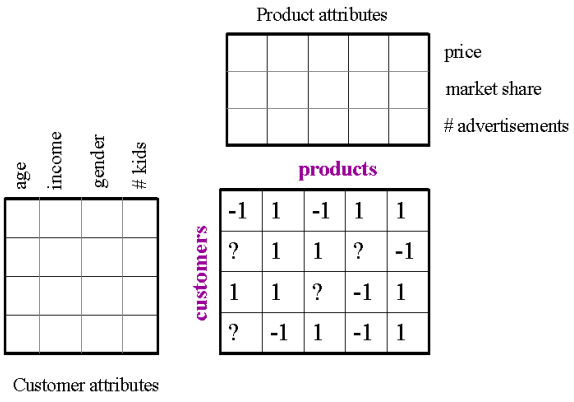


Figure 1.1: Problem setting for the recommender system application

The problem we focus on in the recommender system application is to predict the choices for the missing customer-product combinations, as well as behavior of new customers, or choices made for new products.

Collaborative filtering is a commonly used prediction technique for recommender systems, which will find a neighborhood of similar customers based on known choices and will predict the current customer purchase decision using the preferences of the neighborhood. However, collaborative filtering approaches for this problem will make use only of the matrix entries and ignore customer/product attributes [47, 38].

On the other extreme, a typical classification model will form a map between the feature vector for a given customer-product pair and the corresponding matrix entry, but will not consider nearby customers or products in this process. For the classifier, the dependent variable values are nothing but the matrix entries, and the independent variables are grouped into variables associated with the rows and variables associated with the columns. For a diverse population of customers and a wide range of different products, it is unlikely for all the customer-product preferences to be well explained by a single model. Hence, a single classification model may not be adequate to capture the heterogeneity in the data.

A co-clustering approach as described in Section 2.3 will simultaneously cluster the customers and products based on the matrix entries. It will then use the entries of the corresponding co-cluster to predict a missing value. If done properly, this gives better results than standard recommender systems [38], however this approach still ignores the customer and product attributes, i.e., the prediction is solely based on the value of the dependent variable in a suitably identified neighborhood.

Hence, traditional approaches to this problem do not exploit all the available information and data relationships. If there is sufficient heterogeneity in the data, as

described above it is difficult to learn a single model that can accurately represent all the customer-product choices. It is more natural for a model to closely represent the preferences of only a subset of the customers for a subset of the products. Our idea is to co-cluster (partition) the entire data matrix into blocks of customers and products such that each block can be well characterized by a single predictive model. By doing so, our approach exploits both neighborhood information as well as the available customer/product attributes. Moreover, our model based co-clustering-cum-learning algorithm interleaves clustering and construction of classification models to iteratively improve both cluster assignment and fit of the models.

1.2.1 Other Real Life Applications

Our simultaneous co-clustering and modeling approach is applicable to a wide range of problems in several domains involving DyaC data. A few real life applications of our approach are described below.

- Online ad CTR prediction. A key goal of search engines such as Yahoo! and Google is to maximize the click-through-rate (CTR) on online ads by serving users the ads they are most likely to click on. The massive scale of the ads targeting problem and its obvious business relevance has recently attracted attention from the data mining community [15, 3, 2].

  A typical data setup for the problem is as follows: Ads are partitioned into ad categories. Each category is a specific topic of interest, e.g., loans, travel, parenting and children. The categories are annotated by attributes such as descriptive keywords, historic CTR rates and volume. Users are also described

by features such as demographics, geographical location and metrics computed based on previous browsing behavior. For some (ad category, user) pairs, the target CTR value is known or easily estimated, and these form the training data. Given a user, the objective is then to select the categories to be served based on the highest predicted CTRs. This data is inherently dyadic, with users and ad categories representing two sets of entities. Also, such data is very large (typically several hundred million users per week and several thousand categories), very sparse and noisy, with little activity in some low traffic categories. Moreover, the data is very heterogeneous, with widely varying patterns of user behavior across different user and category groups.

Initial experiments on applying our proposed simultaneous co-clustering and regression framework to an internal Yahoo! dataset have shown promising results.

- Ecology. The analysis of population dynamics of different species and their interaction with the environment is a central problem in the field of ecology. A typical data setup for this problem consists of population count data for different species across different sites/locations. The objective then is to predict the unknown population counts for species of interest in certain locations. Typical datasets of this nature are highly heterogeneous with varying population patterns across different sites and species, and are also extremely sparse with counts available only for few species and locations. A sample dataset in this domain is the recently compiled and NSF funded eBird Reference Dataset [30] (avianknowledge.net, ebird.org), which contains over 27 million

observations of birds count for over 4000 species collected from more than 250,000 locations. Each location is annotated with over 50 covariates such as habitat, climatic conditions, elevation etc. Bird species are described by about 25 attributes such as preferred habitat, global population size, breeding information etc.

- Microarray data analysis. Gene expression data or microarray data is typically in the form of a matrix of genes versus experiments/conditions with the gene expression values as the matrix entries. Often, annotations for the genes and details of the experiments or conditions are available, e.g., regulatory network information, gene/condition metadata, co-location of names in medical abstracts, etc. Often, a large fraction of the matrix entries are missing and the data is very noisy, which makes learning in this domain challenging. Problem areas such as predicting the missing entries or identifying subsets of genes that are coregulated under subsets of experiments could benefit by application of our proposed approaches.

- Analysis of market survey data. Analyzing market survey data involves a matrix of consumer-product preferences with attribute information. The preferences could be real numbers in the form of ratings or purchase probabilities or class labels indicating whether the customer would purchase the product or not. The objective of the analysis is to obtain interpretable and actionable models of purchase behavior and also accurately predict unknown choices. Here, our simultaneous segmentation and modeling approach can be

used to segment the consumer-product choices into homogenous groups and understand the factors that influence purchase decisions within each group. Our approach can hence simultaneously perform "market segmentation" and "market structure", which are important problems in the marketing research community and are discussed further in Section 2.1.1.1.

- Fraud detection. This problem deals with automatically detecting fraudulent sales made by users in an online marketplace like eBay or Amazon. The data consists of a matrix of sellers and products, annotated with attributes like seller ratings based on transaction history, seller demographics, product price and description and transaction details. The additional challenges in this problem are that the fraud and non-fraud class are very imbalanced and often interesting patterns involve only a small fraction of the sellers and products.

1.3 Summary of Proposed Framework

Although most solutions to prediction problems have been based on learning a single model or sequential partitioning and modeling, some work has been done in simultaneous clustering and modeling [57, 88, 97]. This thesis will explore *simultaneous co-clustering and modeling* of dyadic (bi-modal) and multi-modal data with covariates, e.g. the dataset illustrated in Figure 1.1. We first propose Simultaneous CO-clustering And Learning (SCOAL), a framework for solving classification or regression problems on such datasets. This approach interleaves co-clustering and construction of prediction models to improve both, cluster assignment and fit of the models, which is better than independently clustering the data first and then build-

ing models. Our framework is very general and can be easily extended to several prediction models and loss functions. The learning (meta)-algorithm is based on a simple, iterative procedure that provably converges to a local minimum of a suitable classification/regression loss function. This naturally leads to a model selection procedure to select the appropriate number of row and column clusters. This procedure provides further improvement in the prediction accuracy of our approach. The SCOAL framework is described in Chapter 3. This chapter also includes a comparison of SCOAL with alternative predictive techniques on a variety of datasets. In addition, we highlight the practical utility of SCOAL through a case study of a real life marketing application.

The simultaneous co-clustering and learning algorithm is primarily designed to predict missing data values (choices) for an entity only when some other data values (previous choices) are known for that entity. We address the issue of making predictions for new entities by extending the co-clustering framework to an attribute based co-clustering and prediction algorithm. We explore this idea in Chapter 4 in the context of making recommendations to new customers/products in a recommender system setting [1].

Further, we realize that the datasets arising in several domains are large and noisy with coherent patterns occurring only across small regions of the data. In such situations it is likely that not all the data is useful for the learning problem. The data may include outliers or non-informative values and ignoring them while learning

[1] This work was done in collaboration with Andrea Godfrey, Maytal Saar-Tsechansky and Leigh McAlister.

models may lead to better models with higher prediction accuracy. For achieving this objective, we first need a partitioning approach that can automatically identify and discard irrelevant parts of the data to find multiple, coherent co-clusters. Drawing the conceptual idea from Robust Overlapping Co-Clustering (ROCC) [26], an unsupervised learning technique that detects dense co-clusters with a very general structure, i.e., arbitrarily positioned and possibly overlapping, we develop a robust predictive modeling technique for DyaC data (Robust SCOAL), described in Section 5.5. Robust SCOAL identifies and models only the most coherent regions of the data to give high predictive accuracy on the selected subset of target values.

Next, we leverage the multiple models provided by the SCOAL technique to address the following two prediction problems on DyaC data:

1. Ranking predictions to identify the most reliable/certain predictions (Chapter 5). This is particularly important in cases where due to finite resources or domain requirements, one wants to make decisions based only on the most reliable rather than on the entire set of predictions.

2. Developing novel active learning techniques (Chapter 6), where the learner intelligently selects the data points to be labeled such that the prediction model is improved the most. This contribution is of value in applications where acquiring labeled data is costly.

In Chapter 7 we extend SCOAL to prediction problems on dyadic/multimodal data with covariates, with additional ordering constraints along certain modes.

We focus on predictive modeling of time series data, where one of the modes represents a fixed temporal ordering. We demonstrate the effectiveness of our approach on a marketing application.

The SCOAL approach partitions the data matrix into a grid of blocks (co-clusters). While this structured partitioning technique allows for efficient and scalable algorithms, it may be too rigid in some situations. For instance, a dataset can have very high heterogeneity in some regions where a larger number of models are required, while other regions can be adequately represented at a coarser granularity. We address this by proposing Decoupled SCOAL, discussed in Chapter 8, which begins with a single co-cluster and recursively splits co-clusters if doing so improves prediction accuracy, resulting in a more general partitioning of the data matrix.

SCOAL is most suitable for modeling large-scale datasets, where there is large heterogeneity and sufficient data is available to learn multiple local models. A scalable implementation of SCOAL is hence of high practical value. There is massive scope for parallelization in the SCOAL algorithm that can be exploited by a distributed computing environment. In Chapter 9 we provide the details of our implementation of a parallel version of SCOAL based on the Google Map-Reduce framework [24], developed on the open source Hadoop (`hadoop.apache.org`) platform [2].

In several of the following sections, which describe the details of the si-

[2] This work was done in collaboration with Clinton Jones.

multaneous co-clustering and learning framework, we refer to the "row" mode as customers and "column" mode as products, based on our motivating example. Our approach is however not restricted to a customer-product matrix, and is applicable to any bi-modal/dyadic data with covariates. It can also be readily extended to multi-modal data, e.g., a 3-D tensor (data cube) with sets of variables associated with one or more of the axes.

1.4 Notation

Lower case letters represent scalars, e.g., a,z, lower case, bold face letters represent vectors, e.g., $\mathbf{b}, \mathbf{c}, \boldsymbol{\beta}$, upper case letters like Z, W represent matrices and calligraphic upper case letters like $\mathcal{Z}$ represent tensors. Bold face upper case letters like $\mathbf{T}$ represent sets. Individual elements of a matrix, e.g., Z are represented as z_{ij}, where i and j are the row and column indices respectively.

Chapter 2

Background and Related Work

This chapter discusses how different streams of prior work in predictive modeling of dyadic data relate to the contributions in this thesis. The related work associated more specifically with certain topics is included in the corresponding chapters.

2.1 Partitioning for Prediction from Complex Data

In this section, we discuss several "divide and conquer" approaches that have been used for solving complex classification and regression problems. This includes approaches where the partitioning is done *a priori* as well as simultaneous partitioning and modeling approaches.

2.1.1 Hard Partitioning of Input/Input-output Space

2.1.1.1 Sequential Partitioning and Modeling

There are several examples of the use of localized prediction models in load forecasting systems, where clustering is used to distinguish smaller, homogenous groups of data and a prediction model is then fitted for each cluster in a two-step sequential process [8, 29]. Clustering based prediction models have also been widely

used in economics [82, 72].

In the bioinformatics domain, clustering of genes is often used as a pre-processing step for the classification of experiments (samples) in microarray data analysis [71, 58]. A cluster is represented by the mean of the expression profiles across all its member genes, which acts as a dimensionality reduction step for the classification process. This helps to reduce gene redundancies and constructs parsimonious and more interpretable classification models.

Market segmentation, which partitions the customers into homogenous groups and market structure, which deals with identifying groups of "equivalent" products are two closely related problems in market research. Grover and Srinivasan use a cross classification matrix of the proportion of customers that switch from one brand to another [41]. Maximum likelihood is used to estimate latent class models that can be used to extract consumer segments. Competing products for each segment are determined by identifying the brands with large purchase probabilities in that segment. Similarly, Reutterer finds competing brands in different consumer segments by a one sided *a priori* clustering [90]. Self organizing maps are used to cluster individuals based on their vector of product choice probabilities. The products with high choice probabilities in the prototype of each customer cluster are said to be competing. Moe and Fader model customer and product segments simultaneously in their analysis of music CD sales [79]. Their model for the total sales consists of different consumer segments with different purchase parameters and different product clusters that have different contributions from each consumer segment. The sales for each product cluster are modeled as an additive combination

of contributions from different consumer segments.

2.1.1.2 Simultaneous Partitioning and Modeling

There have been some instances of simultaneous clustering and learning of localized prediction models in order to improve prediction accuracy. Sfetsos *et al.* [97] propose an iterative algorithm to cluster time series data such that each cluster consists of data points with similar linear models. This is followed by a heuristic to identify a single cluster to be used for future predictions. This clustering algorithm is one sided and linear model based, a special case of our "two-sided" co-clustering with associated generalized linear models (Section 3.2). A simultaneous clustering and classification algorithm is proposed by Zhang *et al.* [111], who use a voting based classifier ensemble to improve a clustering solution. The labels assigned by an initial clustering are used to train a set of diverse classifiers. Data points that lie on cluster boundaries are relabeled by combining the classifier predictions using a majority vote. This process is iterated to refine the clustering solution.

Another body of work that is related to input space partitioning and modeling is that comprised of decision tree based approaches. The well known CART system [12] approximates a non-linear function by local, piecewise constants. The M5′ approach proposed by Wang and Witten [106] builds on CART and the M5 method developed by Quinlan [86] to predict a continuous response variable value by inducing model trees. The basic idea is to build a tree using a splitting criterion that minimizes the variation in the response values going down each branch. A linear predictive model is learnt at each tree node using only the attributes in

the corresponding sub-tree. The tree is then pruned as long as the estimated error reduces. A smoothing procedure is used to compute the predicted value, in which the values output by linear models along the path from a leaf node to the root are suitably combined.

2.1.2 Soft Partitioning of Input/Input-output Space

2.1.2.1 Mixture-of-Experts Model

The mixture-of-experts framework [57, 88] is a supervised learning approach that divides the input training dataset into subsets corresponding to distinct subtasks. The prediction model is composed of a collection of different "experts", each of which specializes on different regions of the input space. Here, the partitioning of the input space and the training of the expert models is carried out simultaneously and a common objective function is optimized using gradient descent or maximum likelihood. The partitioning of the input space is soft, i.e., multiple experts are involved in varying amounts for producing any particular input-output map. A gating network determines the extent to which each expert contributes to a particular input-output map. Collections of experts along with their corresponding gating networks can be organized in a hierarchy to give rise to a hierarchical mixture of experts framework [56]. The soft partitioning of the input space makes the mixture-of-experts framework less interpretable and actionable as compared to our proposed approach. Our partitioning is based on co-clustering the data into a grid of rectangular blocks and is hence more structured. Moreover, our approach is able to smooth over the joint input-output space which is not achieved by a mixture-of-

experts model.

2.1.2.2 Fuzzy Clusterwise Regression

The idea of simultaneous clustering and regression was introduced in the marketing literature by Wedel and Steenkamp, who proposed a generalized fuzzy clusterwise regression technique to find both customer segments and market structure [107]. Each cluster includes fractional membership from all customers and products and is hence a fuzzy co-cluster. Each cluster has a regression model that predicts the preferences as a linear combination of the product attributes. The cluster memberships and models are estimated so as to reduce the total squared error between the actual preferences and the predicted preferences. However, the hard version of this method corresponds to diagonal co-clustering [76], where only a subset of products is associated with each customer group. In contrast, our simultaneous co-clustering and learning framework is concerned with partitional co-clustering, which covers the entire customer-product matrix.

2.1.2.3 Latent Class Modeling

The latent class modeling approach proposed by Hoffman and Puzicha [50] uses a soft partitioning technique to predict unknown choices of individuals based on their preference history. Their formulation uses only a data matrix of customer-product choices, without any attribute information. The proposed models include the aspect model and the two-sided clustering model. In the aspect model each customer-product combination is assumed to be generated from 1 of K latent classes

(clusters) and hence this model imposes very few structural constraints. In contrast, in the two-sided clustering model, each customer belongs to 1 of k customer clusters, each product belongs to 1 of l product clusters and an association parameter relates customer and product clusters. While the aspect model gives better prediction accuracy, the two-sided clustering model is better at structure discovery. These latent class models can be efficiently estimated using the EM algorithm.

2.1.3 Output Space Partitioning for Multi-class Problems

Rather than solving multi-class classification problems directly, it is often beneficial to decompose them into simpler 2-class problems and combine the outputs of the 2-class problems to solve the original problem. The "1 class vs. all others" approach followed by voting, the pairwise classification approach [44] and the error correcting output codes (ECOC) approach [28] are commonly used techniques to decompose a multi-class problem. However, none of these approaches exploit relationships between classes or take advantage of the fact that certain classes are more similar than others. For example, in case of the hyperspectral dataset acquired from the Bolivar peninsula [46], the 12 classes representing different land cover types form a hierarchy. This hierarchical structure can be exploited to obtain a more natural and meaningful decomposition of the output space.

Kumar et al. propose a learning technique that given a k-class problem automatically generates a binary hierarchy of $k-1$ classifiers, each solving a 2-class problem and having its own feature space [67]. At the root node, the set of k classes is partitioned into 2 meta classes, which are further partitioned recursively

until each meta-class is reduced to one of the k classes. A deterministic annealing technique is used for the feature extraction and class partitioning at each node of the hierarchy. This binary hierarchical classification technique groups classes by natural affinities and hence results in simple decision boundaries. The ability to use node specific feature extraction/selection techniques and classification models makes this approach very flexible. Besides improving classification accuracy, this approach also yields an interpretable ensemble of classifiers and reveals inherent class similarities.

2.2 Techniques for Predictive Modeling of Dyadic Data

While dyadic data, discussed in Section 1.2 arises in several domains such as online advertising, movie/book/news recommendation and market-basket analysis, there has been significant work on predictive modeling of such data particularly in the context of recommender systems. Traditional approaches in recommender systems include, (i) collaborative filtering, which utilizes similarity information among users to recommend items and, (ii) content based filtering, where the predictions are based on descriptive attributes of the items, e.g., annotations associated with books/blogs. Hence, although the underlying data is DyaC, it is not completely utilized; collaborative filtering uses only the matrix of response values while content based filtering uses only the covariates. Basilico and Hofmann [7] proposed a unified approach that integrates past ratings (response values) and user/item covariates. Their main contribution is the design of a joint kernel over user-item pairs that captures the similarity of past ratings as well as covariate information.

2.2.1 Latent Factor Models

Recently, latent factor models have become popular and successful approaches for recommender systems. Several authors have applied variants of these models to the Netflix problem [94, 65, 66]. For instance, Probabilistic Matrix Factorization (PMF) [94] is based on the idea that user-movie ratings can be represented as a product of user specific and movie specific latent feature vectors. Different ways of regularizing the latent factors results in techniques with different properties and generalization capabilities. While these matrix factorization techniques scale well, and are able to address the sparsity and imbalance of the Netflix data and improve accuracy, none of them consider user/movie covariate information.

Agarwal et. al propose a latent factor model approach [4] to the problem of predicting dyadic response variables (e.g. ratings in a customer-product matrix), when covariate information (e.g. customer and product attribute information) is available (DyaC data). The Probabilistic Discrete Latent Factor (PDLF) model simultaneously incorporates the effect of the covariates as well as any local structure that may be present in the data. The data matrix is partitioned into a grid of co-clusters, each one representing a local region. The mean of the response variable is modeled as a sum of a function of the covariates (representing global structure) and a co-cluster specific constant (representing local structure). The co-cluster specific constant can also be thought of as part of the noise model, teased out of the global model residues. Scalable, generalized EM based algorithms are formulated to estimate the parameters of hard or soft versions of the proposed model. The PDLF model is very robust and resistant to outliers and is even applicable in situ-

ations where the training data is limited. Experimental results on the MovieLens dataset and a user click count dataset illustrate the benefits of this approach over traditional prediction models. This approach is complementary to the simultaneous co-clustering and learning approach described in Section 3, which constructs an independent prediction model in each co-cluster.

Matrix factorization approaches for predicting missing entries in a matrix, e.g., movie ratings in a user-movie matrix [94, 65, 66], have recently been extended to use side-information, i.e., covariates associated with the two modes. Agarwal and Chen [2] further generalize the PMF [94] approach to regression based latent factor models (RLFM), where covariate information is incorporated in the priors over the latent factors. RLFM provides a unified framework to smoothly handle both cold-start and warm-start scenarios. It can hence effectively make predictions for new "customers" and "products". RLFM is a two stage hierarchical model with regression based priors used to regularize the latent factors. The Probabilistic Discrete Latent Factor model, described above is a special case of RLFM, where the latent factors are discrete and denote cluster memberships. A scalable Monte Carlo EM approach is used for fitting the regression based latent factor models. However, the performance of the algorithm depends on suitable tuning of the MCMC steps. Another recent approach that comes under the umbrella of matrix factorization with "side-information" is Spatio-Temporal Kalman Filtering [73]. This approach provides a joint model that simultaneously incorporates both spatial and temporal structure in user-item ratings to accurately predict future ratings. The spatial component regularizes the factors through a Markov random field prior that

takes into account user/item correlations based on covariate information. The temporal component ensures that the factors suitably adapt to changes that occur in time. Incorporating covariate information also helps in handling the cold-start scenario. An inference algorithm based on mean-field approximation allows scalability to large datasets.

Note that the proposed SCOAL approach can also be considered as a latent factor modeling technique, where the latent factors are the cluster memberships and regularization is achieved through a common model for all entries in a co-cluster.

2.3 Co-clustering

Co-clustering, also known as biclustering, is a technique that simultaneously clusters data along multiple axes, e.g., in the case of a recommender system application, it simultaneously clusters the users and the items [38]. Co-clustering hence exploits the duality between the data points and the features to improve on one-sided clustering. Co-clustering has been used in several diverse data mining applications like clustering microarray data [16, 17], which involves simultaneous clustering of genes and experiments, text mining [27], where documents and words are clustered and marketing applications [107], where customers and products are clustered.

Bregman Co-clustering is a generalized framework for partitional co-clustering [76] proposed by Banerjee et al. [5], which includes several previously developed co-clustering algorithms like information theoretic co-clustering [27] and minimum sum squared co-clustering [17] as special cases. Bregman co-clustering is very

efficient and scalable and works with any distance measure that is a Bregman divergence. It can also be thought of as a matrix approximation technique that approximates the data matrix Z by a matrix $\hat{Z}$, which is constructed such that it preserves a certain set of sufficient statistics within each co-cluster. Banerjee et al. identify 6 possible sets of summary statistics, of increasing complexity, that one might be interested in preserving in the reconstructed matrix $\hat{Z}$. These 6 sets of statistics lead to 6 different approximation schemes referred to as co-clustering bases. For example, basis 2 approximates all the values within a co-cluster by the mean of the values and hence preserves the co-cluster means.

2.3.1 Co-clustering for Prediction

Most clustering or co-clustering approaches cannot handle missing data and assume a full data matrix. However the Bregman co-clustering [5] formulation readily handles missing data and can also be used for missing value prediction. It has been shown to perform significantly better than traditional collaborative filtering techniques in a recommender system setting [38], where the data is a matrix of user-movie ratings. The known ratings are used to simultaneously cluster users and movies and compute summary statistics for the co-clusters, which are then used to predict unknown ratings, using a suitable instance of the Bregman co-clustering algorithm [5].

Chapter 3

A Framework for Simultaneous Co-clustering and Learning from Complex Data

3.1 Introduction

Simultaneous Co-clustering and Learning (SCOAL) is an effective framework for predictive modeling of large-scale, heterogeneous, dyadic data. The key idea behind SCOAL is to simultaneously partition the two sets of entities that comprise the dyadic dataset ("customers" and "products") into clusters such that their cross product results in dividing the data matrix into a grid of blocks (co-clusters). Concurrently, a predictive model is learnt in each co-cluster, which relates the independent variables to the response variable in the co-cluster. The aim is to obtain a partitioning such that each co-cluster can be well characterized by a single predictive model. Specifically, SCOAL aims at finding a co-cluster assignment and corresponding set of co-cluster models that minimize a global objective function, namely the prediction error summed over all the known entries in the data matrix. A simple iterative algorithm that steadily decreases the objective function can be applied to guarantee convergence to a local minimum. This simultaneous approach is better than independently clustering the data first and then building prediction models.

The SCOAL approach forms a very versatile framework for prediction problems in general. Depending on the application, SCOAL can construct local models that are generalized linear models (GLMs) or even other predictive models like neural networks (MLP, RBF, etc.). In order to present a concrete description of SCOAL, we focus on two special cases (i) classification using logistic regression, which is discussed in Section 3.2 and (ii) linear regression models, discussed in Section 3.3. Further, in Section 3.4 we show how these 2 instances are special cases of a GLM based soft generative model. We then illustrate how the SCOAL framework can be used with a number of regularization techniques that provide robustness and improve accuracy (Section 3.6). In Section 3.7, we develop a simple but effective model selection approach to determine an appropriate number of local models to cover the problem space. To highlight the effectiveness of our approach, we consider diverse applications involving the following classification and regression problems (i) predicting course choices made by graduate students in Section 3.8.2 (ii) predicting the number of items of a given product purchased by a customer, using a formidable, real dataset in Section 3.9.1 and (iii) predicting unknown user-movie ratings using the MovieLens dataset in Section 3.9.2.

3.2 Simultaneous Co-clustering and Classification

3.2.1 Problem Definition

We first describe the problem formulation for the classification setting. Let m be the total number of customers and n the total number of products. The data can be represented as an $m \times n$ matrix Z of customers and products, with cells z_{ij}

representing the corresponding class labels, e.g., whether customer i buys product j or not. Throughout the following discussion we assume that we are dealing with a 2 class problem and $z_{ij} \in \{-1, +1\}$, however the algorithm can easily be generalized to deal with multiclass settings. The problem formulation and solution for regression models is given in Section 3.3.

Every customer and product pair is described by a vector of covariates $\mathbf{x_{ij}}$. The covariate vector is typically composed of the customer attributes, $\mathbf{c_i}$, the product attributes, $\mathbf{p_j}$ as well as annotations associated with the customer-product pair, $\mathbf{a_{ij}}$. Note that up to two of these three vectors ($\mathbf{c_i}$,$\mathbf{p_j}$,$\mathbf{a_{ij}}$) could be empty. It is assumed that each class label z_{ij} is primarily determined by the attributes of the corresponding customer-product pair via a logistic regression model [1]. Thus the log odds is modeled as a linear combination of the customer and product attributes given by

$$ln\frac{P(z_{ij}=1|\mathbf{x_{ij}})}{1-P(z_{ij}=1|\mathbf{x_{ij}})} = f(\mathbf{x_{ij}}),$$

where $\mathbf{x_{ij}}^T = [1, \mathbf{c_i}^T, \mathbf{p_j}^T, \mathbf{a_{ij}}^T]$ is a vector consisting of the customer-product covariates, and $f(\mathbf{x_{ij}}) = \boldsymbol{\beta}^T\mathbf{x_{ij}}$ is a linear model with parameters $\boldsymbol{\beta}^T = [\beta_0, \boldsymbol{\beta_c}^T, \boldsymbol{\beta_p}^T, \boldsymbol{\beta_a}^T]$. The similarity of the cell values is now defined based on the similarity of their underlying logistic regression models. The aim is to simultaneously cluster the rows (customers) and columns (products) into a grid of k row clusters and l column clusters [2], such that the class labels within each co-cluster are predicted by a single,

[1] The focus of this work is on simultaneous co-clustering and classification, rather than obtaining the best possible classifier, therefore we have chosen a standard, fairly flexible classifier rather than experiment with a multitude of classifier options.

[2] This form of co-clustering is often called partitional co-clustering [76]

common classification model. The co-cluster assignments along with the classification models for the co-clusters can be used to predict the class labels for missing customer-product combinations.

Formally, let ρ be a mapping from the m rows to the k row clusters and γ be a mapping from the n columns to the l column clusters. A weight w_{ij} is associated with each cell z_{ij}. The weights of the known (training) matrix cell values are set to 1. The missing cell values, that are to be predicted are given a weight of 0. In general, the weight is not restricted to 0 or 1 and can take other values. This formulation allows the prediction framework to deal with data uncertainties, where less certain values can be given comparatively lower but non-negative weights. We now want to find a co-clustering defined by (ρ, γ) and associated set of classification models $\{\boldsymbol{\beta^{gh}}\}$ that minimize the following objective function

$$\sum_{g=1}^{k}\sum_{h=1}^{l}\sum_{u:\rho(u)=g}\sum_{v:\gamma(v)=h} w_{uv} ln(1 + exp(-z_{uv}\boldsymbol{\beta^{gh}}^T\mathbf{x_{uv}})), \tag{3.1}$$

where z_{uv} is the original value (class label) in row u, column v of the matrix, with associated weight w_{uv}. Here $\boldsymbol{\beta^{gh}}$ denotes the vector of coefficients of the model associated with the co-cluster that the cell value z_{uv} is assigned to. Since the weights for the missing z_{uv} values are set to 0, the objective function essentially ignores them and is simply the log loss summed only over all the known elements of matrix Z. Minimizing this objective function is equivalent to maximizing the log-likelihood of the data.

3.2.2 Algorithm for Logistic Regression Based Classification

A co-clustering (ρ, γ), that reduces the cost function (3.1) can be obtained by a simple iterative algorithm. Since the objective function is the log loss summed over all the elements of the matrix, it can be expressed as a sum of row or column losses. If row u is assigned to row cluster g (i.e., $\rho(u) = g$), the row error is

$$E_u(g) = \sum_{h=1}^{l} \sum_{v:\gamma(v)=h} w_{uv} ln(1 + exp(-z_{uv} \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}})).$$

Since any missing values in the row u will have a weight 0, the error $E_u(g)$ is effectively computed only over the known values in row u. For a given column clustering and model parameter sets $\{\boldsymbol{\beta^{gh}}\}$, the best choice of the row cluster assignment for row u is the g that minimizes this error, i.e.,

$$\rho^{new}(u) = argmin_g E_u(g).$$

Each row is hence assigned to the row cluster that minimizes the row error. A similar approach is used to (re)-assign columns to column clusters. Such row and column cluster updates hence decrease the objective function and improve the clustering solution. Note that updating column cluster assignments could cause the best row assignments to change and vice versa. Thus optionally, the row and column cluster reassignment steps can be repeated several times and in arbitrary order until both row and column cluster memberships converge.

Given the current row and column cluster assignments, the co-cluster models need to be updated, i.e., the co-efficient vector $\boldsymbol{\beta}$ has to be updated for each co-cluster. To update the model for a row cluster g of size r and column cluster

h of size c, train a logistic regression model with the $r \times c$ values within the co-cluster, weighted by their corresponding weight values. The missing values present in the co-cluster have weights of 0 and are essentially ignored. The logistic regression model is hence trained using only the known training samples $(\mathbf{x_{uv}}, z_{uv})$. In the more general case of arbitrary valued weights, this step involves updating a weighted logistic regression model [68] rather than a simple logistic regression model. The output will be an updated vector $\boldsymbol{\beta^{gh}}$ of coefficients that minimizes the model log loss given by

$$L = \sum_{u=1}^{r} \sum_{v=1}^{c} w_{uv} ln(1 + exp(-z_{uv} \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}})).$$

The model update step is hence guaranteed to decrease the objective function (3.1).

The resulting algorithm is a simple iterative algorithm described in Figure 3.1. Step 1 minimizes the objective function due to the property of logistic regression, steps 2(a) and 2(b) directly minimize the objective function. The objective function hence decreases at every iteration. Since this function is bounded from below by zero, the algorithm is guaranteed to converge to a local minimum.

Predicting missing class labels: After the co-cluster assignments and the co-clusterwise classification models are obtained by the algorithm, the missing class labels can be predicted easily. Let z_{uv} be a missing cell value that has been assigned to row cluster g and column cluster h. $\mathbf{x_{uv}}$ is the vector of attributes of row u and column v and $\boldsymbol{\beta^{gh}}$ represents the model parameters of the logistic regression model of the assigned co-cluster. The logistic regression model is used to obtain

the probability of z_{uv} of belonging to the positive class as follows

$$P(z_{uv} = 1) = \frac{1}{1 + e^{-\boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}}}}$$

A suitable threshold t is used to convert the probabilities into class labels, i.e., $z_{uv} = 1$ if $P(z_{uv} = 1) > t, z_{uv} = -1$ otherwise.

Algorithm: SCOAL
Input: $Z_{m \times n}$, $W_{m \times n}$, covariates
Output: Co-clustering (ρ, γ) and co-cluster models $\boldsymbol{\beta}$s
1. Begin with a random co-clustering (ρ, γ)
2. Repeat
    Step 1
3. Update co-cluster models
4. for $g = 1$ to k do
5. for $h = 1$ to l do
6. Train a logistic regression model with training samples $(\mathbf{x_{uv}}, z_{uv})$ in
7. co-cluster (g, h), with weights w_{uv}, to obtain updated $\boldsymbol{\beta^{gh}}$.
8. end for
9. end for

    Step 2(a)
10. Update ρ - assign each row to the row cluster that minimizes the row error
11. for $u = 1$ to m do
12. $\rho(u) = argmin_g \sum_{h=1}^{l} \sum_{v:\gamma(v)=h} w_{uv} ln(1 + exp(-z_{uv} \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}}))$
13. end for

    Step 2(b)
14. Update γ - assign each col. to the col. cluster that minimizes the col. error
15. for $v = 1$ to n do
16. $\gamma(v) = argmin_h \sum_{g=1}^{k} \sum_{u:\rho(u)=g} w_{uv} ln(1 + exp(-z_{uv} \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}}))$
17. end for
18. Optional: repeat steps 2(a) and 2(b) until convergence
Until convergence
19. Return (ρ, γ) and $\boldsymbol{\beta}$s

Figure 3.1: Pseudo-code for simultaneous co-clustering and classification

3.3 Simultaneous Co-clustering and Regression

In the regression setting, Z is an $m \times n$ matrix of "customers" and "products", with cells representing the corresponding customer-product preference values, ratings or choice probabilities. Here we assume the generative model to be a linear model, where the preference value $z_{ij} \in \mathbb{R}$ is modeled as a linear combination of the corresponding covariates. The preference value is estimated as $\hat{z}_{ij} = \beta_0 + \boldsymbol{\beta_c}^T \mathbf{c_i} + \boldsymbol{\beta_p}^T \mathbf{p_j} + \boldsymbol{\beta_a}^T \mathbf{a_{ij}}$. Similar to the problem definition in Section 3.2.1, the aim is to simultaneously cluster the customers and products into a grid of k row clusters and l column clusters, such that preference values within each co-cluster have similar linear models and can be represented by a single common model. We want to find a co-clustering defined by (ρ, γ) and the associated $k \times l$ regression models that minimize the following objective function

$$\sum_{g=1}^{k} \sum_{h=1}^{l} \sum_{u:\rho(u)=g} \sum_{v:\gamma(v)=h} w_{uv}(z_{uv} - \hat{z}_{uv})^2, \tag{3.2}$$

where

$$\hat{z}_{uv} = \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}}. \tag{3.3}$$

The only difference here as compared to the classification case is the loss function, which is now squared loss rather than log loss. A co-clustering (ρ, γ), that minimizes the objective function can be obtained by an algorithm similar to the one described in Section 3.2. The cluster reassignment steps assign each row or column to the row or column cluster that minimizes the row or column error.

The model for row cluster g of size r and column cluster h of size c is

updated by finding the $\boldsymbol{\beta}$ that minimizes

$$\sum_{u=1}^{r}\sum_{v=1}^{c} w_{uv}(z_{uv} - \boldsymbol{\beta}^{\boldsymbol{gh}^T}\mathbf{x_{uv}})^2.$$

In case of 0/1 weights, this is equivalent to ignoring missing values and updating the $\boldsymbol{\beta}$ for each co-cluster by least squares regression using only the non-missing (training) values within the co-cluster. In case of a general set of weights, the $\boldsymbol{\beta}$ is a solution to a weighted least squares problem. Close to convergence, since the co-cluster memberships change only by a few matrix entries in each iteration, it is more efficient to incrementally update the regression models rather than retraining from scratch. This can be achieved by updating the QR factorization of the coefficient matrix to reflect the addition and deletion of data points [39]. Least squares regression finds a solution for $\boldsymbol{\beta}$ that minimizes the sum of the squared errors between the original values and the predicted values. The model update step is hence guaranteed to decrease the objective function.

After the algorithm converges, the co-cluster assignments and co-clusterwise regression models can be used to predict unknown customer-product preference values. A missing value z_{uv} is predicted as $\hat{z}_{uv} = \boldsymbol{\beta}^{\boldsymbol{gh}^T}\mathbf{x_{uv}}$.

3.4 Generative Model for Soft SCOAL

Logistic and linear regression models with which we developed the SCOAL algorithm in Sections 3.2 and 3.3 are special cases of a large class of models known as generalized linear models (GLMs) [77]. We begin by stating the relevant properties of GLMs. If the response variable y follows a GLM, then a function g of the

mean response is modeled as an unknown linear function $\boldsymbol{\beta}^T\mathbf{x}$ of the independent variables $\mathbf{x}$. g is referred to as the link function. For example, in linear least-squares regression the link function is the identity, while in logistic regression, g is the logit function ($g(y) = log(\frac{y}{1-y})$). Another important characteristic of GLMs is that the distribution of the response variable conditioned on $\mathbf{x}$ belongs to a member of the exponential family. A single-parameter exponential family [6] is a set of distributions whose probability density function can be expressed as

$$f(x;\theta) = h(x)\exp(\theta t(x) - \psi(\theta)),$$

where θ is known as the natural parameter, $\psi(\theta)$ is the cumulant generating function and $h(x)$ and $t(x)$ are functions of x. The Gaussian, Poisson and Bernoulli distributions are examples of exponential families. Under linear least-squares regression and logistic regression models, the response variable belongs to the Gaussian and Bernoulli exponential families respectively.

The formulation of the SCOAL meta-algorithm with GLMs for co-cluster models is based on an intuitive generative model, consisting of a mixture of $k \times l$ exponential family distributions, corresponding to the $k \times l$ co-clusters. Each matrix entry z_{ij}, given the covariates $\mathbf{x_{ij}}$ is assumed to be generated from this mixture model as follows

$$P(z_{ij}|\mathbf{x_{ij}}) = \sum_{I=1}^{k}\sum_{J=1}^{l}\alpha_{IJ}f_{\psi}(z_{ij};\boldsymbol{\beta}^{\boldsymbol{I},\boldsymbol{J}^T}\mathbf{x_{ij}}), \quad [i]_1^m, [j]_1^n, \tag{3.4}$$

where α_{IJ} denotes the mixture component priors and f_ψ is an exponential family distribution with cumulant $\psi(.)$. The natural parameter $\theta_{i,j,I,J}$ of the exponential

family is the linear function $\boldsymbol{\beta}^{\boldsymbol{I},\boldsymbol{J}^T}\mathbf{x_{ij}}$, where $\boldsymbol{\beta}^{\boldsymbol{I},\boldsymbol{J}}$ denotes the regression coefficients of component (I, J). Response values z_{ij} hence belong fractionally to the components (I, J), with the mean response for each component modeled as a linear function of the corresponding covariates. Note that the co-cluster assignments here are *soft*, in contrast with the hard assignments in the formulation in Section 3.2, where each row/column belonged to exactly 1 row/column cluster.

By assuming that the matrix elements are generated i.i.d with weights w_{ij}, the incomplete data log-likelihood is given by

$$L(\boldsymbol{\beta}, \boldsymbol{\alpha}|Z) = \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} logP(z_{ij}), \tag{3.5}$$

where $\boldsymbol{\beta}, \boldsymbol{\alpha}$ collectively denote all the model coefficients and priors. It is however not possible to directly maximize this data log-likelihood and we hence associate latent variables $\rho(i)$ and $\gamma(j)$, denoting row and column cluster membership, with each element z_{ij}. $\rho(i)$ and $\gamma(j)$ take values from 1 to k and 1 to l respectively. We first construct the free energy function [80] as a sum of the expected complete data log-likelihood and the entropy of the latent variables with respect to a distribution $\tilde{p}(\rho(i), \gamma(j))$, i.e.,

$$F(\tilde{p}, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \sum_{ij} w_{ij} E_{\tilde{p}_{ij}}[logP(z_{ij}, \rho(i), \gamma(j))] + \sum_{ij} w_{ij} H(\tilde{p}_{ij}), \text{ where} \tag{3.6}$$

$$E_{\tilde{p}_{ij}}[logP(z_{ij}, \rho(i), \gamma(j))] = \sum_{IJ} \tilde{p}_{ij}(I, J) log(\alpha_{IJ} f_{\psi}(z_{ij}|\theta_{i,j,I,J})),$$

$$H(\tilde{p}_{ij}) = -\sum_{IJ} \tilde{p}_{ij}(I,J) log(\tilde{p}_{ij}(I,J)).$$

It can be shown that maximizing $F(\tilde{p}, \boldsymbol{\beta}, \boldsymbol{\alpha})$ with respect to $\tilde{p}$, $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ is equivalent to maximizing the data log-likelihood given by Equation 3.5 [80]. $F(\tilde{p}, \boldsymbol{\beta}, \boldsymbol{\alpha})$ can be maximized by the classical EM procedure, which alternates between maximizing F w.r.t. $\tilde{p}$ for fixed $\boldsymbol{\beta}, \boldsymbol{\alpha}$ (E-step) and maximizing F w.r.t. $\boldsymbol{\beta}, \boldsymbol{\alpha}$ for fixed $\tilde{p}$ (M-step) and is guaranteed to converge to a local maximum of F.

However, the standard EM algorithm used to estimate the model is not scalable and can be very slow in practice for large m and n. To address this issue, one can impose structural constraints on the generative mixture model. SCOAL does so by restricting $\tilde{p}_{ij}$ to the form $\tilde{p}_{ij}(\rho(i), \gamma(j)) = \tilde{p}_i(\rho(i))\tilde{p}_j(\gamma(j))$, i.e., $\rho(i)$ and $\gamma(j)$ are independent a-posteriori. Hence, we assume that every row i of the data matrix Z belongs to row cluster $I = 1$ to k with probability $\tilde{p}_i(I)$ and similarly every column j belongs to column cluster $J = 1$ to l with probability $\tilde{p}_j(J)$. By decoupling the row and column cluster assignments, SCOAL estimates the model very efficiently by an EM based algorithm. The steps of the algorithm are illustrated in Figure 3.2. The exact form of the regression coefficients update step can be derived based on the assumed exponential family distribution. For the special case of a Gaussian distribution, the coefficients are computed by a weighted least squares regression. Each step monotonically increases the free energy function, finally converging to a locally optimal solution. Note that this algorithm is a generalization of the instances of the *hard* SCOAL algorithm described in Sections 3.2 and 3.3, for soft cluster assignments. Specifically, SCOAL with linear least-squares

regression models corresponds to a mixture of Gaussians in Eq. 3.4, which results in the squared error cost function in Eq. 3.2. This model is also related to the soft PDLF model [4], where the mean of the z_{ij}s in each co-cluster is modeled as a sum $\boldsymbol{\beta}^T \mathbf{x_{ij}} + \delta_{I,J}$, where $\boldsymbol{\beta}$ is a global regression model and $\delta_{I,J}$ is a co-cluster specific offset.

Algorithm: Soft SCOAL
Input: $Z_{m\times n}$, $W_{m\times n}$, covariates, k, l, exponential family with cumulant ψ
Output: Regression coefficients $\boldsymbol{\beta}^{I,J}$, priors $\boldsymbol{\alpha}$ and co-cluster assignments $\tilde{p}$

Begin with arbitrarily initialized assignments $\tilde{p}$
Repeat

E-step
Update Row Cluster Assignments:

$\tilde{p}_i(I) = c_i(\prod_{j,J} (\alpha_{IJ} f_\psi(z_{ij}; \boldsymbol{\beta^{I,J}}^T \mathbf{x_{ij}}))^{w_{ij}\tilde{p}_j(J)})^{\frac{1}{w_i}}, \quad \forall [i]_1^m, [I]_1^k,$
where c_i is a normalizing factor s.t. $\sum_{I=1}^k \tilde{p}_i(I) = 1$ and $w_i = \sum_j w_{ij}$.

Update Column Cluster Assignments:

$\tilde{p}_j(J) = c_j(\prod_{i,I} (\alpha_{IJ} f_\psi(z_{ij}; \boldsymbol{\beta^{I,J}}^T \mathbf{x_{ij}}))^{w_{ij}\tilde{p}_i(I)})^{\frac{1}{w_j}}, \quad \forall [j]_1^n, [J]_1^l,$
where c_j is a normalizing factor s.t. $\sum_{J=1}^l \tilde{p}_j(J) = 1$ and $w_j = \sum_i w_{ij}$.

M-step
Update Priors:
$\alpha_{IJ} = \frac{\sum_{ij} w_{ij}\tilde{p}_i(I)\tilde{p}_j(J)}{\sum_{ij} w_{ij}}, \quad \forall [I]_1^k, [J]_1^l,$

Update Regression Coefficients:
$\forall [I]_1^k, [J]_1^l,$
$\boldsymbol{\beta}^{I,J} = argmax_{\boldsymbol{\beta}} \sum_{ij} w_{ij} \sum_{IJ} \tilde{p}_i(I)\tilde{p}_j(J)(z_{ij}\boldsymbol{\beta^{I,J}}^T \mathbf{x_{ij}} - \psi(\boldsymbol{\beta^{I,J}}^T \mathbf{x_{ij}})).$

Until convergence
Return $(\tilde{p}, \boldsymbol{\beta}, \boldsymbol{\alpha})$

Figure 3.2: EM algorithm for estimation of the soft SCOAL model

3.5 Extensions

Extension to Other Regression and Classification Models. The algorithm presented in Sections 3.2 and 3.3 is not restricted to linear or logistic regression models, but actually suggests a meta-algorithm that can be extended to other predictive models by modifying (3.2) and/or (3.3). For example, in the regression setting, the data can be modeled by a collection of neural network models (MLP, RBF, etc.) or regression models with the L_1 norm regularization (Lasso [45]). For classification problems one can use decision trees or Naive Bayes classifiers. The model update and the cluster reassignment steps will parallel those in Figure 3.1 to now reduce the loss function corresponding to the assumed predictive model.

Extension to Tensor Data. In the above analysis we assumed that the data was dyadic or bi-modal. In this section we discuss its extension to the more general tensor setting. Let $\mathcal{Z}$ represent a tensor dataset with N modes. Let $\mathcal{X}$ denote the set of covariates associated with the elements of $\mathcal{Z}$. If there is heterogeneity along all the N modes, the aim is to assign entities along each of the N modes to clusters, such that all the response variables within each co-cluster can be represented by a common prediction model. Note that each co-cluster will now be an N-mode sub-tensor.

The problem formulation and objective function for the N-mode case are a natural extension of the 2-mode case. In case of a real valued $\mathcal{Z}$, assuming linear regression models, the objective function is the squared error summed over all the elements of $\mathcal{Z}$. Since the objective function is still additive over the individual tensor elements, the iterative algorithm in Figure 3.1 can be readily extended to

achieve a locally optimal solution. The co-cluster model update step now uses all the data entries within the sub-tensor corresponding to each co-cluster as training examples for the prediction models. The cluster update steps cycle through the modes, treating them like rows/columns in the bi-modal case. As in the case of bi-modal data, the algorithm is guaranteed to converge to a locally optimal solution.

3.6 Regularization

The SCOAL framework involves fitting $k \times l$ independent models to the data, one in each co-cluster. If p is the number of covariates, the total number of parameters to be learnt in case of linear models is $k \times l \times (1+p)$. The overall model will have too many parameters for large values of k and l and may overfit in cases where training data is limited. In this section we explore a number of regularization approaches that perform smoothing/shrinkage across the parameters to alleviate the overfitting problem.

3.6.1 Shrinkage Methods

One possible alternative to obtaining a generalizable set of $k \times l$ models is to update the model loss function to include a regularization term. Such approaches shrink the coefficients of each co-cluster model by imposing a penalty on their size. For instance, in a regression setting, assuming linear models, shrinkage can be achieved by ridge regression or lasso. Such a modification does not however change the SCOAL loss function (3.2) and the meta-algorithm for simultaneous co-clustering and regression can still be used to achieve a local minimum of the loss

function. The only change is that the model update step will now involve learning ridge regression or lasso models rather than least squares models.

3.6.1.1 Ridge Regression

Ridge regression adds a penalty term equal to the sum of the squared model coefficients to the total squared residue to be minimized [45]. The model coefficients hence minimize the following loss function,

$$\boldsymbol{\beta}^{\text{ridge}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\{\sum_{i=1}^{N}(y_i - \boldsymbol{\beta}^T\mathbf{x_i})^2 + \lambda\sum_{j=1}^{p}\boldsymbol{\beta}_j^2\},$$

where y_i and $\mathbf{x_i}$ are the response variable and the vector of independent variables respectively. $\lambda \geq 0$ is a complexity parameter that controls the amount of regularization. The effect of increasing the value of λ is to reduce the magnitude of the regression coefficients.

The ridge regression solution is given by:

$$\boldsymbol{\beta}^{\text{ridge}} = (X^TX + \lambda I)^{-1}X^T\mathbf{y}\ ,$$

where X is the design matrix and $\mathbf{y}$ is the vector of responses. Another advantage of using ridge regression is that the additional of λ to the diagonal of X^TX makes it non-singluar, which is useful if X^TX is not full rank.

3.6.1.2 Lasso

Like ridge regression, lasso adds a penalty term to the sum squared residue [45]. The difference is that the penalty term in this case is the L_1 norm penalty, as com-

pared to the L_2 norm ridge penalty. The model coefficients hence minimize the following,

$$\boldsymbol{\beta}^{\text{lasso}} = \underset{\boldsymbol{\beta}}{\text{argmin}}\{\sum_{i=1}^{N}(y_i - \boldsymbol{\beta}^T\mathbf{x_i})^2 + \lambda\sum_{j=1}^{p}|\boldsymbol{\beta}_j|\}.$$

As the complexity parameter $\lambda \geq 0$ is increased, the model coefficients are driven to zero, leading to parsimonious models. This formulation makes the coefficients non-linear in the y_is, which can be computed by a quadratic programming approach [3]. Another important advantage of lasso is that it improves model interpretability by identifying the subset of the predictors that are the most influential.

In general, regularization can be achieved by Bridge regression [36], which defines a family of regression loss function penalized with the p-norm penalty. Note that ridge regression and lasso are special cases of bridge regression with the L_2 and L_1 norm penalties respectively.

3.6.2 Reduced Parameter Approach

The simultaneous co-clustering and prediction approach with $k \times l$ independent models and a single prediction model for all the data are two extremes of the modeling spectrum in terms of the number of model parameters to tune. We now propose an intermediate approach, which constructs $k \times l$ models but with smoothing or regularization achieved by sharing parameters across certain sets of models. Here we assume that the covariates $\mathbf{x_{ij}}$ include only customer attributes $\mathbf{c_i}$

[3]The following matlab code http://www.cs.ubc.ca/~schmidtm/Software/lasso.html was used for experimentation.

and product attributes $\mathbf{p_j}$. The co-cluster models are constructed in such a way that the customer coefficients of all models for the same row cluster and the product coefficients of all models for the same column cluster are constrained to be identical. This reduced setting has $(1 + |C|) \times k + (1 + |P|) \times l$ parameters, where $|C|$ and $|P|$ are the number of customer and product attributes respectively, which will be considerably lower than the total number of parameters for an independent set of $k \times l$ models. We now describe how the model parameters can be obtained for the regression problem.

If z_{uv} is the original value in row u, column v of the matrix Z that is assigned to row cluster g and column cluster h, the predicted value $\hat{z}_{uv}$ is now given by

$$\hat{z}_{uv} = \beta^g_{c0} + \boldsymbol{\beta}^{\boldsymbol{g}T}_{\boldsymbol{c}} \mathbf{c_u} + \beta^h_{p0} + \boldsymbol{\beta}^{\boldsymbol{h}T}_{\boldsymbol{p}} \mathbf{p_v},$$

where β^g_{c0} and β^h_{p0} are the customer and product intercepts and $\boldsymbol{\beta}^{\boldsymbol{g}}_{\boldsymbol{c}}$ and $\boldsymbol{\beta}^{\boldsymbol{h}}_{\boldsymbol{p}}$ are the customer and product coefficient vectors for the row cluster g and column cluster h respectively.

We still want to find a co-clustering defined by (ρ, γ) and associated regression models that minimize the objective function (3.2). The row and column cluster assignment steps remain the same. The update models step now involves solving a constrained optimization problem. Instead of updating $k \times l$ linear models independently, this step now updates the k row cluster models, such that the product coefficients are fixed and the customer coefficients are updated, and then the l column cluster models, in which the customer coefficients are fixed and the product coefficients are updated.

To update the model for row cluster g with r rows and n columns solve

$$min\ \mathbf{w}^T((\mathbf{y} - X_c[\beta^g_{c0}, \boldsymbol{\beta}^{\boldsymbol{g}T}_{\boldsymbol{c}}]^T) \otimes (\mathbf{y} - X_c[\beta^g_{c0}, \boldsymbol{\beta}^{\boldsymbol{g}T}_{\boldsymbol{c}}]^T)),$$

where $\otimes$ represents elementwise multiplication. X_c is an $(r*n)\times(1+|C|)$ matrix of the customer attributes, with the first column set to 1 for the intercept. The response variable $\mathbf{y}$ is a vector with $r * n$ elements, given by

$$\mathbf{y} = \mathbf{z} - X_p[\beta^h_{p0}, \boldsymbol{\beta}^{\boldsymbol{h}T}_{\boldsymbol{p}}]^T,$$

where $\mathbf{z}$ is a vector of all the $r * n$ preference values in the row cluster g with associated weights $\mathbf{w}$, $[\beta^h_{p0}, \boldsymbol{\beta}^{\boldsymbol{h}T}_{\boldsymbol{p}}]$ is a vector of the product coefficients of the corresponding column clusters and X_p is a matrix of size $(r*n)\times(1+|P|)$ representing the product attributes corresponding to the preference values. The column cluster models are updated similarly. This update ensures that all the customer coefficients of models within the same row cluster and the product coefficients of models within the same column cluster are updated simultaneously and are identical.

3.7 Model Selection

The SCOAL meta-algorithm described in Section 3.2 requires the number of row clusters k and the number of column clusters l as an input. If the input k and l values are larger than the "true" k and l, then trying to learn the large number of parameters would tend to cause overfitting and lead to poor generalization on test data. In this section we deal with the possibility of overfitting by choosing the k and l values in a systematic way.

Since the SCOAL algorithm directly tries to minimize a prediction loss function, it can be extended to use a cross validation procedure to select k and l that give the lowest validation set error. This is done by splitting the training dataset to obtain a validation set, on which the parameters k and l are tuned. We propose an efficient, top-down "bisecting" greedy algorithm that begins with $k = 1$ and $l = 1$ and then iteratively tries to increase the number of row and column clusters using a model selection procedure as long as the validation set error reduces. The detailed steps of the complete algorithm referred to as M-SCOAL are given below. The pseudo-code for the model selection procedure, which increases the number of models in each step by suitably splitting a row/column cluster, is illustrated in Figure 3.3. (ρ, γ) denote the current row and column cluster assignments and $\{\boldsymbol{\beta}\}$ represents the set of co-cluster models. $\mathcal{V}$ denotes the matrix entries z_{uv} that form the validation set and valError($\{\boldsymbol{\beta}\}$,ρ,γ) is the validation set error for the current co-cluster assignments and models.

1. Run SCOAL with $k = 1$ and $l = 1$ and initialize (ρ, γ) and $\{\boldsymbol{\beta}\}$.

2. Execute the model selection procedure (algorithm ModelSel described in Figure 3.3) to attempt to increase the number of co-cluster models. The detailed steps of algorithm ModelSel are as follows:

   (a) **Attempt to split a row cluster.** Let $RC(g)$ denote the average error of row cluster g on the validation set $\mathcal{V}$. For SCOAL with linear regression models

$$RC(g) = \frac{\sum_{z_{uv} \in \mathcal{V} \& \rho(u) = g} w_{uv} (z_{uv} - \boldsymbol{\beta}^{\boldsymbol{gh}^T} \mathbf{x_{uv}})^2}{\sum_{z_{uv} \in \mathcal{V} \& \rho(u) = g} w_{uv}},$$

where $h = \gamma(v)$. The row cluster $\tilde{g}$ with maximum average validation set error is selected as a candidate for splitting. The rationale for splitting the row cluster with the highest prediction error is that the current set of models in row cluster $\tilde{g}$ do not adequately capture the heterogeneity of this row cluster. Thus, appropriately splitting $\tilde{g}$ into two row clusters to induce a larger number of co-cluster models may produce models that characterize the input space better. This hypothesis is then evaluated computationally by determining the effectiveness of the split. Let $R(i)$ denote the average validation set error of row i belonging to row cluster $\tilde{g}$, i.e., $\rho(i) = \tilde{g}$. For SCOAL with linear models, $R(i)$ is computed as follows:

$$R(i) = \frac{\sum_{z_{iv} \in \mathcal{V}} w_{iv} (z_{iv} - \boldsymbol{\beta}^{\tilde{\boldsymbol{g}}\boldsymbol{h}^T} \mathbf{x_{iv}})^2}{\sum_{z_{iv} \in \mathcal{V}} w_{iv}},$$

where $h = \gamma(v)$. The row cluster $\tilde{g}$ is split into two by assigning half the rows with the largest row errors $R(i)$, represented by set $\mathcal{S}_R$, to a new row cluster, and ρ is updated accordingly. Rows with the largest errors contain the response variable values that are most inaccurately predicted by the models in row cluster $\tilde{g}$. The next step is to determine whether assigning the rows in $\mathcal{S}_R$ to a different row cluster results in inducing a more accurate set of models in the new co-clusters. The revised co-cluster assignments, with k increased by 1, are used to initialize a new run of the SCOAL algorithm and the validation set error of the new set of models is computed. If the error is lower than the error before

splitting the row cluster, the split is accepted; otherwise, the previous co-clustering and set of models are retained.

(b) **Attempt to split a column cluster.** The column cluster $\tilde{h}$ with maximum average error on the validation set, denoted by $CC(h)$ for column cluster h, is selected for splitting. Analogous to the row cluster case, the column cluster $\tilde{h}$ is split into two based on column errors $C(j)$, for j such that $\gamma(j) = \tilde{h}$, and γ is updated accordingly. Note that the average column cluster and column errors ($CC(h)$,$C(j)$) are computed in the same manner as the average row cluster/row errors. The updated co-clustering, with l increased by 1, is used to initialize a new run of SCOAL. As in step 1, the split is accepted if it reduces the validation error.

(c) **Application of the best split.** If both the row and column cluster splits decrease the validation set error, the one that decreases the error the most is selected and the current co-clustering (ρ, γ) and models $\{\boldsymbol{\beta}\}$ are updated accordingly.

(d) Note that if neither a row or column cluster split reduces the validation set error, the co-cluster assignments (ρ, γ) and the set of co-cluster models $\{\boldsymbol{\beta}\}$ remain unchanged.

3. If k and l remain unchanged after executing the ModelSel algorithm, terminate and use the current (ρ, γ) to initialize a final run of SCOAL; otherwise loop back to Step 2.

Algorithm: ModelSel
Input: $\{\boldsymbol{\beta}\}$, (ρ,γ)

1. V = valError($\{\boldsymbol{\beta}\}$,ρ,γ)
2. Try to split a row cluster
3. $\quad \tilde{g} = \arg_g \max\ RC(g)$
4. $\quad \mathcal{S}_R$ = $r/2$ rows i with highest $R(i)$, where $\rho(i) = \tilde{g}$ and r = # rows in $\tilde{g}$
5. $\quad \tilde{\rho} = \rho, \quad \tilde{\rho}(u) = k+1, \quad \forall u \in \mathcal{S}_R$
6. $\quad \{\boldsymbol{\beta_1}\}$, ($\rho_1$,$\gamma_1$) = output of SCOAL initialized with ($\tilde{\rho}$,γ)
7. $\quad V_1$ = valError($\{\boldsymbol{\beta_1}\}$,ρ_1,γ_1)
8. $\quad$ if $(V_1 < V) \quad \{\boldsymbol{\beta_{row}}\} = \{\boldsymbol{\beta_1}\}$, $\rho_{row} = \rho_1$, $\gamma_{row} = \gamma_1$
9. $\quad$ else $\quad \{\boldsymbol{\beta_{row}}\} = \{\boldsymbol{\beta}\}$, $\rho_{row} = \rho$, $\gamma_{row} = \gamma$

10. Try to split a column cluster
11. $\quad \tilde{h} = \arg_h \max\ CC(h)$
12. $\quad \mathcal{S}_C$ = $c/2$ cols j with highest $C(j)$, where $\gamma(j) = \tilde{h}$ and c = # cols in $\tilde{h}$
13. $\quad \tilde{\gamma} = \gamma, \quad \tilde{\gamma}(v) = l+1, \quad \forall v \in \mathcal{S}_C$
14. $\quad \{\boldsymbol{\beta_2}\}$, ($\rho_2$,$\gamma_2$) = output of SCOAL initialized with (ρ,$\tilde{\gamma}$)
15. $\quad V_2$ = valError($\{\boldsymbol{\beta_2}\}$,ρ_2,γ_2)
16. $\quad$ if $(V_2 < V) \quad \{\boldsymbol{\beta_{col}}\} = \{\boldsymbol{\beta_2}\}$, $\rho_{col} = \rho_2$, $\gamma_{col} = \gamma_2$
17. $\quad$ else $\quad \{\boldsymbol{\beta_{col}}\} = \{\boldsymbol{\beta}\}$, $\rho_{col} = \rho$, $\gamma_{col} = \gamma$

18. Apply the best split
19. $\quad$ if $(V_1 < V_2) \quad \{\boldsymbol{\beta_{new}}\} = \{\boldsymbol{\beta_{row}}\}$, $\rho_{new} = \rho_{row}$, $\gamma_{new} = \gamma_{row}$
20. $\quad$ else $\quad \{\boldsymbol{\beta_{new}}\} = \{\boldsymbol{\beta_{col}}\}$, $\rho_{new} = \rho_{col}$, $\gamma_{new} = \gamma_{col}$

21. return $\{\boldsymbol{\beta_{new}}\}$, $(\rho_{new},\gamma_{new})$

Figure 3.3: Pseudo-code for the model selection procedure used by M-SCOAL.

A key advantage of the model selection procedure is that it initializes each run of SCOAL with the previous co-clustering, which is likely to alleviate the local minima problem caused by poor initialization and lead to a better final solution. Although the M-SCOAL algorithm involves several runs of SCOAL, the initialization from the previous run causes each one to converge much faster as compared to random initialization. Hence M-SCOAL is not much slower than SCOAL. We found this to be well-supported by empirical evidence.

3.8 Experimental Evaluation of Classification Results

3.8.1 Synthetic Datasets

The algorithm described in Section 3.2 was first evaluated on a number of synthetic datasets. We used synthetic data for experimentation as an initial sanity check before working with real data. These experiments also indicate the amount of improvement localized classification models provide when the model assumptions match the generative model for the data. Another motivation for using synthetic data was to evaluate the SCOAL model selection procedure (M-SCOAL) in a controlled setting, where the true k and l values are known. Each of the four synthetic datasets (Table 3.1) was created by generating $k \times l$ blocks of data, each corresponding to a true cluster of class labels generated by a logistic regression model. Each block has a different logistic regression generative model, with its coefficient vector ($\boldsymbol{\beta}$) set randomly. The blocks are then appended together in the form of a grid to form a data matrix, whose rows and columns are then randomly shuffled. To assign a class label to a cell z_{ij} within each block, we begin by taking a linear combination of randomly generated customer and product attributes with the coefficient vector $y_{ij} = \boldsymbol{\beta}^{\boldsymbol{T}}\mathbf{x_{ij}}$. We then add random Gaussian noise with variance σ^2 to all the y values. We obtain the probability of a cell belonging to the positive class as $P(z_{ij} = 1) = \frac{1}{1+e^{-y_{ij}}}$. A threshold of 0.5 is used to convert the probabilities into class labels.

Table 3.1 describes the synthetic datasets that were used for experimentation. Dataset 1 and 2 are very similar, dataset 2 has more noise and hence a weaker relationship with the underlying generative model as compared to dataset 1. Dataset

3 and 4 are larger, also with a substantial amount of noise. The datasets along with details of their generative models can be accessed at http://www.ece.utexas.edu/~deodhar/modelCCData.

	m,n	$\|C\|$,$\|P\|$	k,l	Noise σ^2
Dataset 1	100, 80	3,4	3,2	5
Dataset 2	100, 80	3,4	3,2	15
Dataset 3	500, 300	3,4	4,3	5
Dataset 4	900, 500	5,5	8,6	5

Table 3.1: Synthetic datasets

3.8.1.1 Results on Synthetic Data

We evaluate the ability of SCOAL to classify unknown matrix values in the synthetic datasets. The data is split as 90% training and 10% test (missing) and the technique in Section 3.2 is used to predict the class label of the missing values, given the true k and l values as input. M-SCOAL, on the other hand selects the most suitable number of row and column clusters. The predicted labels are compared to the true labels and the classification quality is evaluated using precision, recall, F-measure and classification error. We compare the SCOAL approaches to the partitional co-clustering algorithm, Bregman co-clustering [5], which uses only the matrix Z without any attribute information. The Bregman co-clustering algorithm is very flexible and can work with several distance measures and co-cluster definitions. The special case of Bregman co-clustering that we compare with uses squared Euclidean distance as the distance measure and tries to find uniform co-clusters that minimize the distance of the data points within the co-cluster to the

co-cluster mean [4], since this case best matches the data generation process.

In order to apply Bregman co-clustering (CC) to this problem, in matrix Z, we encode positive class labels by the value 1 and negative by 0. The co-clustering algorithm approximates the cell values within each co-cluster by the co-cluster mean μ_{gh}. If a missing cell z_{ij} is assigned to row cluster g and column cluster h, with co-cluster mean μ_{gh}, we assign a class label to z_{ij} using the rule $z_{ij} = 1$ if $\mu_{gh} >$ threshold , $z_{ij} = -1$ otherwise. If the threshold is selected to be 0.5 this rule can be interpreted as assigning a missing cell the majority class label within its co-cluster.

In order to validate our claim that simultaneous co-clustering and modeling does better than a sequential, two-step approach of first partitioning the data and then learning models, we compare with Co-cluster Models, which partitions the data *a priori* by applying Bregman co-clustering to matrix Z, and then learns logistic regression models in each co-cluster. We also compare our approach with a single logistic regression classification model (Global Model), which is SCOAL with $k = 1$ and $l = 1$.

Table 3.2 displays the precision, recall, F-measure and classification error for SCOAL, M-SCOAL, co-clustering (CC), Co-cluster models and a single logistic regression model (Global Model). The results are averaged over 5 random 90-10% splits of the data. The values in parentheses are the standard errors. The threshold is set to 0.5 for all these experiments since the same threshold was used to obtain

[4]This corresponds to scheme 2 of the Bregman co-clustering algorithm [5] with squared Euclidean distance

class labels from probability values while generating the synthetic data. One can observe that on all the synthetic datasets SCOAL and M-SCOAL do significantly better than CC, Co-cluster models and Global Model in terms of both the F-measure and the classification error. The performance of M-SCOAL is comparable to that of SCOAL with the true k and l. In some cases, M-SCOAL actually does better than SCOAL, which is explained by the initialization procedure of M-SCOAL that reduces the susceptibility of SCOAL to poor local minima. Table 3.3 shows the k and l values selected by M-SCOAL on the synthetic datasets, over 5 trials, highlighting that the selected values are consistently very close to the true k and l values.

On these datasets we also evaluate the ability of SCOAL to reconstruct the original data matrix. We find that on all the datasets this approach is consistently able to recover a close approximation of the original data matrix. Additionally, the cluster assignments made by the algorithm closely match the true underlying cluster labels.

3.8.2 Recommender System Application

This classification problem deals with predicting the course choices made by masters students at a large Midwestern University. The objective is to use the information of previous known course choices of students to predict unknown choices. These predictions can be used to recommend the right courses for students to take in the future. The data includes a matrix of 326 students vs. 32 courses with class labels = 1 if the student took the course and -1 otherwise. Each student has attributes including the student's career aspiration and undergraduate degree.

Algorithm	Precision	Recall	F-measure	Classification Error
Dataset 1				
Global Model	0.894 (0.004)	0.952 (0.004)	0.922 (0.004)	0.131 (0.006)
CC	0.881 (0.007)	0.948 (0.003)	0.913 (0.003)	0.149 (0.006)
Co-cluster Models	0.91 (0.006)	0.95 (0.005)	0.929 (0.003)	0.117 (0.005)
SCOAL	**0.967 (0.003)**	**0.979 (0.002)**	**0.973 (0.002)**	**0.044 (0.002)**
M-SCOAL	0.961 (0.008)	0.969 (0.005)	0.965 (0.006)	0.058 (0.009)
Dataset 2				
Global Model	0.883 (0.008)	0.927 (0.009)	0.904 (0.007)	0.151 (0.01)
CC	**0.910 (0.002)**	0.880 (0.004)	0.895 (0.003)	0.159 (0.004)
Co-cluster Models	0.883 (0.003)	0.929 (0.007)	0.906 (0.004)	0.149 (0.005)
SCOAL	0.908 (0.006)	**0.937 (0.005)**	**0.922 (0.005)**	**0.124 (0.007)**
M-SCOAL	0.906 (0.007)	0.934 (0.004)	0.920 (0.005)	0.127 (0.007)
Dataset 3				
Global Model	0.926 (0.001)	0.972 (0)	0.948 (0)	0.092 (0.001)
CC	0.935 (0.004)	0.956 (0.004)	0.945 (0.001)	0.096 (0.001)
Co-cluster Models	0.928 (0.002)	0.969 (0.001)	0.948 (0.001)	0.091 (0.002)
SCOAL	0.968 (0.002)	0.979 (0)	0.974 (0.001)	0.046 (0)
M-SCOAL	**0.974 (0.001)**	**0.981 (0.001)**	**0.978 (0.001)**	**0.038 (0.001)**
Dataset 4				
Global Model	0.928 (0)	0.971 (0)	0.949 (0)	0.09 (0)
CC	0.931 (0.003)	0.968 (0.002)	0.949 (0.001)	0.09 (0.001)
Co-cluster Models	0.936 (0)	0.969 (0)	0.952 (0)	0.083 (0)
SCOAL	**0.979 (0)**	**0.984 (0)**	**0.981 (0)**	**0.032 (0)**
M-SCOAL	0.979 (0)	0.984 (0)	0.981 (0)	0.032 (0.005)

Table 3.2: Comparison of classification performance on 4 synthetic datasets.

The course attributes include the department offering the course, the course evaluation score and a binary variable indicating whether the course is quantitative. The dataset is skewed with unequal priors of the positive and negative classes. Around 25% of the student course choices are positive and the rest negative.

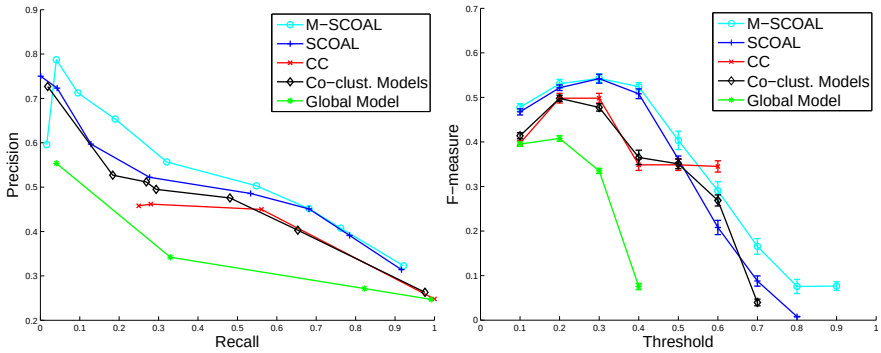
Run	Data 1 $k=3, l=2$	Data 2 $k=3, l=2$	Data 3 $k=4, l=3$	Data 4 $k=8, l=6$
1	3,2	2,3	4,3	8,6
2	2,4	3,2	4,3	8,6
3	3,2	3,2	4,3	8,6
4	3,2	3,2	4,3	8,7
5	2,2	3,2	4,3	8,6

Table 3.3: k and l values selected by M-SCOAL on the synthetic datasets.

3.8.2.1 Classifying Missing Cell Values

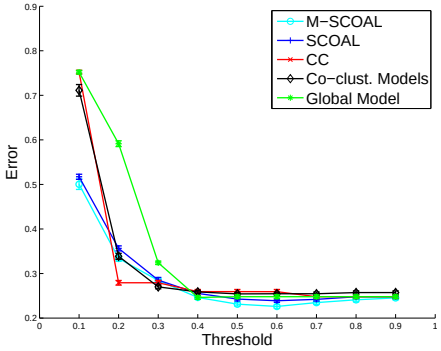
In order to test the classification capability of SCOAL on this problem, the data is split as 90% training and 10% test and the class labels in the test set, i.e., the unknown student-course choices, are predicted using the co-cluster models. Results are obtained by averaging over 10 random 90-10% data splits. We compare the SCOAL (with logistic regression models) results with CC, Co-cluster Models and Global Model. The number of row and column clusters for SCOAL, CC and Co-cluster Models are set to 2. We additionally compare with the SCOAL model selection procedure described in Section 3.7 (M-SCOAL). For assigning class labels to the missing matrix entries we use a threshold that is varied from 0.1 to 0.9 to get a range of precision-recall tradeoffs.

Figures 3.4(a), 3.4(b), 3.4(c) display the precision-recall curves, the F-measure and the classification error of the algorithms at different values of the threshold. Beyond a certain threshold CC, Co-cluster Models and Global Model classify all the data points as belonging to the negative class, causing the F-measure to be undefined. Such points are excluded from the Precision-Recall curve and the F-measure plot. Both M-SCOAL and SCOAL do significantly better than CC, Co-



(a) Precision-Recall curve

(b) F-measure



(c) Classification error

Figure 3.4: Evaluation of classification of missing student-course choices in the recommender system application. Note that CC has previously been shown to perform better than classic collaborative filtering approaches [38].

cluster Models and Global Model in terms of precision and recall as can be seen in the Precision-Recall curves and hence their F-measure is consistently better than the other approaches over different values of the threshold. The classification error of the SCOAL approaches is also lower than the other approaches. At threshold value 0.2 however, CC has a lower error as it has a much smaller number of false positives.

3.9 Experimental Evaluation of Regression Results

3.9.1 Real Marketing Dataset

We applied SCOAL to a challenging marketing application. Given purchase information for a set of customers and products along with customer and product attributes, we simultaneously clustered customers and products and used the co-clustering solution to predict unknown customer-product purchase information. The obtained co-clusters also provide information about customer segments in the market and equivalent product groups, achieving simultaneous market segmentation and structure, an important problem in marketing research [41]. The dataset that we used is the publicly available ERIM dataset [5] consisting of household panel data collected by A.C. Nielsen, which is well known in the marketing community and has been used by several researchers [61, 62, 95]. This dataset has purchase information for six product categories over a period of 3 years from 1985-1988 for households in Sioux Falls, South Dakota. The dataset includes household demographics and product characteristics.

The data preprocessing steps we took are similar to the data selection procedure used by Seetharam et al. [95]. We have 6 product categories (ketchup, tuna, sugar, tissue, margarine and peanut butter) with a total of 121 products. Brands with very low market share in each product category are omitted. We select households that made at least 2 purchases in each product category, resulting in a set of 1714 households. We select 6 household attributes - income, number of residents, male

[5] URL: http://www.gsb.uchicago.edu/kilts/research/db/erim/

head employed, female head employed, total visits and total expense and 3 product attributes - market share, price, number of times the product was advertised. The data can be represented by a data matrix of households and brands where the cell values are the number of units of a brand purchased by a household, aggregated over the time the household was tracked. The number of units purchased can be used as an indicator of brand preference.

3.9.1.1 Dataset Properties

The data matrix is extremely sparse, with 74.86% of the values being 0. The distribution of the number of units purchased is very skewed. 99.12% of the values are below 20, while the remaining values are very large and range upto around 200. These few, large values that form the tail in the histogram of the matrix entries, can be considered as outliers with respect to the rest of the values.

3.9.1.2 Standardization of the Data

The dimensions (items) in this application are products from 6 different product categories. The product attributes such as price and extent of advertising could vary widely from one category to another. When we construct a linear model for a co-cluster we weigh the attributes of all the products in the co-cluster by the same set of coefficients. However, the products in the co-cluster could be from different categories with very different ranges of attribute values. We hence need to standardize the product attributes to make them comparable across categories. We transform each product attribute value a to $a^{'} = \frac{a-\mu_c}{\sigma_c}$, where μ_c and σ_c are the mean

and standard deviation within the corresponding product category c. This problem does not arise in case of the customer attributes since they are relatively comparable. The matrix cell values, which are the number of units purchased could also be very different across categories and have to be standardized. The cell values z_{ij} within each sub-matrix of all the products belonging to a specific product category c are transformed to $z'_{ij} = \frac{z_{ij} - \mu_{z_c}}{\sigma_{z_c}}$ where μ_{z_c} and σ_{z_c} are the mean and standard deviation of the all the values in the sub-matrix. Since the standardization of the data is a linear transformation, the dataset properties described in section 3.9.1.1 continue to hold.

3.9.1.3 Data Reconstruction

Table 3.4 displays the mean squared error of the approximated data matrix obtained by the different algorithms on the entire standardized dataset, averaged over 10 runs. Global Model is a single linear regression model, while CC is Bregman Co-clustering [5], which does not use the attribute information. Co-cluster Models is a two step modeling procedure of first co-clustering and then learning linear regression models in each co-cluster. "Row Clustering" is SCOAL with the number of column clusters set to 1 and "Column Clustering" is SCOAL with the number of row clusters set to 1. Reduced Model is the SCOAL approach with the reduced set of parameters as described in Section 3.6.2. Note that SCOAL obtains the best reconstruction of the original matrix as compared to the other approaches in terms of MSE (mean squared error). Table 3.4 also shows the average R^2 of the linear models constructed within each co-cluster. The R^2 values are actually

quite low, indicating that a strong linear relationship does not really exist in the data, which is to the disadvantage of the simultaneous co-clustering and regression algorithm.

Algorithm	MSE	Avg. R^2
Global Model (k=1,l=1)	0.930 (0)	0.0696
CC (k=10, l=4)	0.842 (0.003)	-
Co-cluster Models (k=10, l=4)	0.835 (0.002)	0.0122
Row Clustering (k=10, l=1)	0.887 (0)	0.0896
Column Clustering (k=1, l=4)	0.88 (0.001)	0.0714
SCOAL (k=10, l=4)	**0.794 (0.005)**	0.1308
Reduced Model (k=10, l=4)	0.876 (0)	0.0426

Table 3.4: Reconstruction error on entire ERIM dataset.

3.9.1.4 Predicting Unknown Data Values

The prediction error of the different approaches on this problem is computed by averaging over 10 random 90-10% training and test data splits. Here we additionally compare with a two-step sequential approach (Cluster Models) that first clusters the customers based on their attributes and then fits regression models in each customer cluster. Table 3.5 shows the training error (Training Err.), the test set error (Test Err.) on the standardized data and the test set error on the original, unstandardized dataset obtained by back transforming the standardized data (Test Err. Original). A more comprehensive comparison of the SCOAL approaches with alternative modeling techniques is presented in Section 3.9.3.

As mentioned in Section 3.9.1.1 the data is skewed with a few very large outliers. In the presence of outliers the clusterwise models overfit the training data and do not generalize well to the test data. SCOAL is the most susceptible to

Algorithm	Training Err.	Test Err.	Test Err. Original	Avg. R^2
Global Model (k=1,l=1)	0.931 (0.003)	0.928 (0.023)	16.776 (0.584)	0.067
Cluster Models (k=4)	0.927 (0.003)	0.917 (0.026)	16.854 (0.51)	0.067
CC (k=4,l=4)	0.853 (0.003)	0.905 (0.022)	15.501 (0.511)	-
SCOAL (k=4,l=4)	0.823 (0.002)	0.905 (0.021)	15.688 (0.519)	0.113
Reduced Model (k=4,l=4)	0.878 (0.002)	0.887 (0.023)	15.339 (0.553)	0.056
M-SCOAL	0.874 (0.004)	**0.873 (0.019)**	**15.032 (0.416)**	0.071

Table 3.5: Comparison of prediction error on ERIM dataset.

overfitting since it is the most complex and involves the most number of parameters. SCOAL does better than Global Model and Cluster Models but slightly worse than CC. Reduced Model does better than SCOAL in this scenario since it has fewer parameters and a simpler overall model, which generalizes better. Reduced Model does slightly better than CC as well, indicating that using the attribute information in the prediction process helps. However, this improvement is small, which can be explained by the fact that the data does not show a very strong linear relation.

Linear "least squares" regression is very sensitive to outliers and a few outliers could skew the model results. Hence on this dataset, for a fair comparison of linear model based techniques with respect to prediction of missing values, we need some way of dealing with outliers. It is unreasonable for a model based on linear regression to capture both small as well as extremely large values simultaneously and a more suitable approach would be to separate out these two very different sets of values and model them independently. A threshold of 20 number of units purchased was used to separate the bulk of the matrix entries (99.12%) from the tail of high values. This can easily be handled by the co-clustering algorithm by setting the weight of the outlier points to 0. We now focus on the prediction problem and

Algorithm	Training Err.	Test Err.	Test Err. Original	Avg. R^2
Global Model (k=1,l=1)	0.913 (0.001)	0.920 (0.01)	4.24 (0.06)	0.093
Cluster Models (k=4)	0.91 (0.001)	0.917 (0.01)	4.228 (0.059)	0.084
CC (k=4,l=4)	0.833 (0.001)	0.890 (0.009)	4.002 (0.056)	-
SCOAL (k=4,l=4)	0.804 (0.001)	0.883 (0.007)	3.965 (0.044)	0.143
Reduced Model (k=4,l=4)	0.849 (0.001)	0.872 (0.009)	3.893 (0.052)	0.08
M-SCOAL	0.848 (0.004)	**0.856 (0.007)**	**3.832 (0.035)**	0.081

Table 3.6: Prediction error on ERIM dataset (low valued entries).

the model for the bulk of the entries, the results of which are illustrated in Table 3.6.

Results on the low valued matrix entries:

Table 3.6 shows the training mean squared error and the test set error for the different approaches on this problem. One can see that M-SCOAL outperforms all the other approaches. The number of customer and product clusters identified by M-SCOAL range from $1-2$ and $4-6$ respectively. All the SCOAL approaches do significantly better than Global Model and Cluster Models on the test set. SCOAL and Reduced Model also do slightly better than CC. Avg. R^2 is the average R^2 of the regression models. One can observe that while the average R^2 for SCOAL is the best among the different approaches, it is still relatively small. Moreover, Reduced Models and M-SCOAL have a lower R^2 than Global Model, but also a lower test MSE. The improvement in the prediction error is hence primarily due to representing the response values in each co-cluster by a different mean as compared to one global mean. Learning multiple linear models does not help as much due to the weak inherent linear relationship in the data.

Our complete model for the prediction problem consists of the model constructed for the bulk of the matrix entries as described above and a linear model for

the outliers. A classifier is trained to appropriately select one of the two models to predict each unknown matrix value. An alternative way of dealing with outliers is to reduce the influence of the extremely high values on the constructed models, allowing them to generalize better. This is achieved by giving high valued matrix entries a very small fixed weight, enabling the linear models to focus on the bulk of the values and rather than fit a few high values. Through experiments conducted for both these cases [25], we observe that here as well, the SCOAL based approaches do better than the other techniques.

3.9.1.5 Interpretability and Actionability

In this section, we delve into this marketing application in further detail and demonstrate the ability of SCOAL to additionally provide interpretable and actionable results which are extremely valuable in real life marketing settings. We focus on the specific problem of simultaneous market segmentation and structure [41]. Market segmentation involves dividing the market into groups of consumers who are homogenous in terms of their choice of products. Market structure deals with clustering products into groups such that the products within each group are equivalent in some respect, and can be used to find groups of competing products in a given market. While both these problems are typically addressed separately by most researchers, Grover and Srinivasan were among the first to realize that these two problems are complementary and it would be more suitable to consider them simultaneously [41]. Other approaches to simultaneous market segmentation and structure include the work by Reutterer [90] and Moe and Fader [79]. More details

on these approaches are discussed in Section 2.1.1.1. The SCOAL framework is promising for this problem since it simultaneously clusters customers and products and builds explanatory models for each co-cluster, which indicate the factors that influence the preference for a group of products.

Market Segmentation and Structure with SCOAL. For this application, the aim is to identify customer segments and groups of "equivalent" products. The details of the SCOAL result with 4 customer and 4 product clusters (a total of 16 co-clusters) are illustrated here. Table 3.7 displays the average number of units purchased in each of the 16 co-clusters. Table 3.8 shows the linear regression coefficients for 3 interesting co-clusters (Cust. Seg. 3-Prod. Seg. 3, Cust. Seg. 4-Prod. Seg. 4, Cust. Seg. 1-Prod. Seg. 2). This table also compares the co-cluster coefficients with the coefficients of a global model. The values in parentheses are the p-values from the significance tests of the coefficients. One can observe that the coefficients of the sample co-clusters presented here are significantly different from those of the global model, validating our intuition that a single model is inadequate for capturing the heterogeneity in the population. By comparing the regression coefficients across all the co-cluster models [6], we observed that they differ quite a bit across the co-clusters, indicating that different factors are important for different customer and product subsets.

Product cluster 3 is a small cluster consisting of the cheapest, most popular and most advertised products. As expected the average number of units purchased

[6]The entire set of coefficients along with details of the customer and product clusters identified is available at http://users.ece.utexas.edu/~deodhar/ERIM

	Prod. Seg. 1	Prod. Seg. 2	Prod. Seg. 3	Prod. Seg. 4
Cust. Seg. 1	1.379	1.378	7.891	0.409
Cust. Seg. 2	0.884	0.650	2.503	0.317
Cust. Seg. 3	1.061	1.197	4.934	1.454
Cust. Seg. 4	1.400	1.480	4.741	0.483

Table 3.7: Mean # of units purchased in each co-cluster.

is highest for this product cluster across all customer segments (Table 3.7). All the co-clusters that include product cluster 3, e.g. Cust. Seg. 3, Prod. Seg. 3 as observed in Table 3.8 have models with very high negative coefficients for price and high positive coefficients for market share and advertising (statistically significant with low p-values), correctly indicating that these are the most influential factors for product preference in these co-clusters. On the other extreme, product cluster 4 consists of products with below average pricing and very low market share and advertising. Overall, across all customer clusters, the average number of units purchased is among the lowest for this product segment. In all the co-cluster models corresponding to this product cluster e.g. Cust. Seg. 4, Prod. Seg. 4, the coefficient for market share is the highest, indicating that this is a major factor influencing the purchase of these products, which is intuitive given the really low market share of the products in this segment. Co-cluster Cust. Seg. 1, Prod. Seg. 2 (Table 3.8) consists of the customer group with the highest average income and slightly expensive, not well advertised products with average market share. It is interesting that the coefficient for price is positive, indicating that these high income customers, who might not consider price to be an important factor in product choice, don't mind paying more for the products. What drives the sale of the products in this co-cluster is probably their market share and advertising and the large number of

shopping visits and amount spent, which end up being very explanatory predictors in the corresponding regression model as seen in Table 3.8.

Another observation we make based on the co-cluster model coefficients is that on the whole, product attributes are more indicative of customer choice as compared to the customer attributes. Among customer attributes, the total amount spent is a dominating explanatory variable in almost all the regression models. The rest of the customer attributes are not as predictive, probably because they do not vary as much across customer segments and the segment averages for these attributes are close to the global averages. The statistically insignificant predictors for each model can be ignored to get sparse, simpler models.

Attribute	Global	Cust. Seg. 3, Prod. Seg. 3	Cust. Seg. 4, Prod. Seg. 4	Cust. Seg. 1, Prod. Seg. 2
intercept	0.000 (1.000)	-0.423 (0.001)	-0.136 (0.000)	0.151 (0.000)
income	-0.021 (0.000)	-0.089 (0.313)	-0.025 (0.000)	-0.019 (0.424)
# members	0.027 (0.000)	0.028 (0.736)	0.039 (0.000)	-0.042 (0.064)
male head emp.	0.002 (0.424)	-0.060 (0.418)	-0.001 (0.870)	0.052 (0.040)
female head emp.	0.001 (0.617)	-0.071 (0.305)	-0.003 (0.469)	0.016 (0.464)
# shopping visits	0.024 (0.000)	-0.106 (0.054)	0.010 (0.056)	0.112 (0.000)
total amount spent	0.097 (0.000)	0.478 (0.000)	0.029 (0.000)	0.094 (0.000)
price	-0.020 (0.000)	-0.746 (0.000)	-0.022 (0.000)	0.424 (0.000)
market share	0.174 (0.000)	0.425 (0.000)	0.089 (0.000)	0.162 (0.000)
# times advertised	0.096 (0.000)	0.481 (0.000)	0.037 (0.000)	0.041 (0.066)

Table 3.8: Coefficients of the global model and of 3 co-cluster models.

The results on the ERIM dataset demonstrate that the SCOAL approach not only improves prediction accuracy over existing approaches but also produces easily interpretable and actionable models in a segmentation setting. The inferences made from the analysis of the learned models can be potentially very valuable in driving marketing decisions, e.g., advertising strategy and designing targeted dis-

counts and deals.

3.9.2 MovieLens Dataset

Another application on which we evaluated the SCOAL framework is that of predicting user-movie ratings in a recommender system setting. We applied SCOAL to the MovieLens dataset, which consists of 100,000 ratings (1-5) from 943 users on 1682 movies made available by the GroupLens Research Project at the University of Minnesota [7]. We used a set of 23 attributes including user demographics like age, gender, employment status and movie date and genre. The dataset is extremely sparse, with the proportion of known user-movie ratings as small as 6.3 % of the size of the data matrix.

Table 3.9 displays the training and test set mean squared error of different approaches on this problem, computed by averaging over 10 random 80-20% training and test data splits. As in the case of the ERIM dataset, the SCOAL based approaches do well, with Reduced Model doing significantly better than the competing techniques. Reduced Model and M-SCOAL improve upon SCOAL as well, indicating the importance of reducing the number of parameters to be tuned. The k and l selected by M-SCOAL range from $4-5$ and $3-5$ respectively. The average R^2 of the SCOAL approaches is significantly larger than that of the Global Model. Hence, the improvement in the prediction error of SCOAL is due to both, a collection of better fitting linear models as well as a co-cluster specific mean of

[7] The data can be downloaded from http://www.grouplens.org/system/files/ml-data.tar__0.gz

Algorithm	Training Err.	Test Err.	Avg. R^2
Global Model (k=1,l=1)	1.192 (0)	1.192 (0.004)	0.059
Cluster Models (k=4)	1.185 (0.001)	1.189 (0.003)	0.065
CC (k=4,l=4)	0.881 (0.001)	0.942 (0.004)	-
Reduced Model (k=4,l=4)	0.887 (0.002)	**0.924 (0.004)**	0.16
SCOAL (k=4,l=4)	0.876 (0.001)	0.943 (0.004)	0.257
M-SCOAL	0.883 (0.002)	0.937 (0.003)	0.256

Table 3.9: Comparison of prediction error on the MovieLens dataset.

the response variables, which is in contrast to the situation on the ERIM dataset (Table 3.6).

Figure 3.5 provides interesting insights into the nature of the structure in the data learnt by a single global model versus the collection of local models constructed by SCOAL. Both, the global model and SCOAL ($k = 4, l = 4$) are initially trained on a small subset (10%) of the training data. The number of training points input to the two predictive techniques is then progressively increased along the x-axis in 80 batches each of 1000 randomly selected points. SCOAL uses the previously learnt co-clustering to initialize each new run with an increased training dataset size. Mean squared error on a held out test set is plotted on the y-axis. One can notice that the global model captures the linear structure in the data fairly quickly and attains a reasonable accuracy on the test set with relatively few training examples. In contrast, the performance of SCOAL with a very small amount of training data is not as good, since the available data is not sufficient to fit the 16 local models. However, as more training data is provided, the global model saturates in performance, while SCOAL continues to fit the data better by capturing more complex local structures.

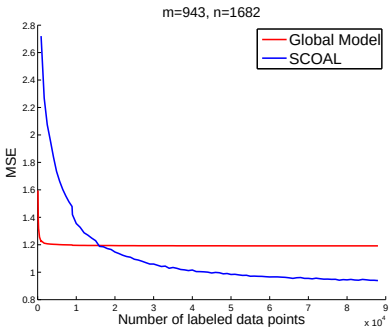


Figure 3.5: Error on a held out test set of models learnt at increasing amounts of training data on the MovieLens dataset.

3.9.3 Comparison with Alternate Approaches

In this section we present a comparative study of SCOAL evaluated against a number of alternate predictive modeling approaches, described below.

1. Two-way cluster models. The customers are clustered into k row clusters based on the customer attributes using a 1-sided clustering algorithm like k-means. The products are independently clustered into l column clusters based on the product attributes. The 1-sided row and column cluster assignments induce a grid of $k \times l$ co-clusters on the data matrix. A linear regression model that relates the independent variables to the target values is then learnt in each co-cluster.

2. Co-cluster models. The partitional Bregman co-clustering algorithm [5] is applied to the matrix of data values to obtain $k \times l$ co-clusters. Linear regression models are then trained in each co-cluster. This approach is included to indicate how a simultaneous co-clustering and learning strategy compares

with a two-step sequential co-clustering and learning procedure.

3. M5′. This is the model tree algorithm proposed by Wang and Witten [106], discussed in Section 2.1.1.2. This is again a sequential approach, in which the input space is first partitioned using a decision tree, followed by the construction of predictive models for each partition.

4. Neural network. A multilayer perceptron (MLP) model is evaluated on the datasets to give some idea of how a single non-linear model would perform in contrast to modeling the data using a collection of linear models.

Table 3.10 compares and contrasts the test set mean squared error of the different approaches on the ERIM and MovieLens datasets. The SCOAL and reduced parameter approaches with ridge regression and M-SCOAL are included in this table. Two-way cluster models and co-cluster models use $k = 4, l = 4$. The parameter in M5′ that specifies the minimum number of instances required to split a node is set to 100. The MLP has 2 hidden layers, each with 10 hidden units and has a learning rate of 0.3. The SCOAL approaches perform consistently well, giving better results than even M5′ and the MLP model. Two-way cluster models and co-cluster models have higher error than SCOAL, validating the benefits of a simultaneous partitioning and modeling strategy as compared to *a priori* segmentation and modeling. Non-linear models like the MLP possibly need more extensive tuning to give better performance.

Algorithm	ERIM Dataset	ERIM Dataset (low valued entries)	MovieLens Dataset
SCOAL	15.807 (0.49)	3.965 (0.044)	0.943 (0.004)
Reduced Model	**14.021 (0.443)**	3.902 (0.04)	**0.93 (0.004)**
M-SCOAL	15.032 (0.416)	**3.832 (0.035)**	0.935 (0.005)
Two-way cluster models	17.062 (1.038)	4.091 (0.0267)	1.168 (0.005)
Co-cluster models	16.049 (0.588)	3.967 (0.034)	0.931 (0.003)
M5′	15.457 (0.816)	3.877 (0.037)	1.116 (0.006)
MLP	17.169 (0.938)	4.395 (0.193)	1.462 (0.127)

Table 3.10: Mean squared error of different modeling techniques on the ERIM and MovieLens datasets, averaged over 10 90-10 % train-test data splits.

3.9.4 Evaluation of Regularized Models

We now evaluate the effects of the regularization methods discussed in Section 3.6.1 on the prediction error of SCOAL. Table 3.11 compares the mean squared error of SCOAL without regularization with SCOAL based on ridge regression and lasso models on the ERIM and the MovieLens datasets. Each value in the table is an average obtained over 10 random 90-10 % train-test splits of the data. The value of the regularization parameter λ in each case is selected by cross-validation. One can see that on the ERIM dataset, which is known to have a very skewed distribution of response values and the presence of outliers, ridge regression and lasso boost the performance of SCOAL.

3.9.5 Non-Linear Co-cluster Models

As described in Section 3.5, the SCOAL algorithm is not restricted to logistic or linear regression models but can use other predictive models in each co-cluster by suitably modifying the objective function (3.2) and/or (3.3). In this section, we

Regularization	Global Model	SCOAL	Reduced Model
ERIM Dataset			
No regularization	16.776 (0.584)	15.688 (0.519)	15.339 (0.553)
Ridge	17.013 (0.528)	15.807 (0.49)	**14.021 (0.443)**
Lasso	16.776 (0.584)	15.051 (0.315)	15.154 (0.475)
ERIM Dataset (low values entries)			
No regularization	4.24 (0.06)	3.965 (0.044)	3.893 (0.052)
Ridge	4.24 (0.06)	3.949 (0.033)	3.902 (0.04)
Lasso	4.24 (0.06)	3.957 (0.031)	**3.866 (0.044)**
MovieLens Dataset			
No regularization	1.192 (0.004)	0.943 (0.004)	**0.924 (0.004)**
Ridge	1.192 (0.004)	0.946 (0.004)	0.93 (0.004)
Lasso	1.192 (0.004)	0.938 (0.006)	0.94 (0.018)

Table 3.11: Comparison of regularized linear regression models on the ERIM and MovieLens datasets.

evaluate simultaneous co-clustering and regression with a non-linear model (Multi-Layer Perceptron) learnt in each co-cluster. Table 3.12 compares the test MSE of a single MLP modeling all the data (Global) with the SCOAL and M-SCOAL approaches, averaged over 10 random 90-10 % train-test data splits. The MLP used is a feed-forward backpropagation network, with the hidden layer using the log-sigmoid transfer function and the output layer using the linear transfer function. The MLP model complexity for the global model and the SCOAL models is tuned through cross-validation. The learning rate for all models is set to 0.05. SCOAL is run with 4 row and 4 column clusters respectively, while M-SCOAL determines the most suitable number of row and column clusters. The trend observed here is similar to that in the comparisons with linear models illustrated in Sections 3.9.1 and 3.9.2. M-SCOAL consistently outperforms the global model, while SCOAL seems to overfit slightly on the ERIM dataset due to the presence of outliers. Com-

Algorithm	ERIM Dataset	ERIM Dataset (low valued entries)	MovieLens Dataset
Global	15.406 (0.67)	4.0183 (0.0533)	1.138 (0.004)
SCOAL	17.575 (0.433)	4.076 (0.037)	0.967 (0.002)
M-SCOAL	**15.204 (0.434)**	**3.917 (0.033)**	**0.953 (0.005)**

Table 3.12: Mean squared error of SCOAL techniques, using MLPs as predictive models in each co-cluster vs. a single global MLP, on the ERIM and MovieLens datasets.

paring these results with the ones in Table 3.10, one can observe that on all datasets, SCOAL with linear models does better than SCOAL with MLP models. This is probably because a collection of simple linear models is sufficient to capture the heterogeneity and structure in these datasets. Multiple non-linear models, with a larger number of parameters require more tuning and may be an overkill.

3.10 Summary

Based on the results in Sections 3.8 and 3.9 we observe that the simultaneous co-clustering and prediction approach holds promise for both classification and regression problems, at least for the datasets examined so far. One should be able to further improve the results by using alternative prediction models within each co-cluster that more closely conform to the data characteristics. This is particularly true for the ERIM dataset which is known to have significant outliers, but we still used linear least squares regression as it is widely adopted and understood. Note that overfitting of the models to outliers is addressed to some extent by the regularization techniques. Further improvements can be achieved by using a more robust error function rather than squared error [52]. Moreover, non-linear models would

help because the limitations of linear models for this dataset is quite evident by the low R^2 values obtained by several researchers who previously applied such models to ERIM. We would like to point out that the SCOAL framework shows the most value on large datasets that are heterogeneous enough to require a collection of localized models for adequate representation and where enough data is available to tune the parameters of each of the models.

The results in Section 3.9.3 illustrate that SCOAL does very well in terms of prediction accuracy as compared to a variety of competing techniques. Apart from accuracy, SCOAL additionally provides the following features that distinguish it from related predictive modeling techniques.

1. SCOAL captures the inherent structure in the data by a collection of simple (possibly linear) local models. The models indicate the degree to which predictors influence the response in different regions of the input space and are hence easily actionable. This ease of interpretability of the models is valuable to applications in domains such as marketing.

2. The SCOAL algorithm is simple and computationally efficient. For instance, in case of least squares linear regression models, each iteration is linear in the size of the data.

3. The independence of the operations involved in the SCOAL meta-algorithm allows it to be easily parallelized and adapted to multi-core architectures. This enables the SCOAL framework to scale to very large real life datasets.

4. The generic SCOAL framework can be suitably tailored to the application by the right choice of predictive models, cost function and regularization mechanism.

5. The fit of the predictive models in their local regions provides an idea of the goodness of representation of different regions of the input space. This intuition can be used to device a mechanism that estimates the reliability or certainty of predicted response values.

Chapter 4

Attribute Based Co-clustering for Recommender System Application

4.1 Introduction

The simultaneous co-clustering and learning approach discussed in Chapter 3 assumed that for each customer and product, some previous purchase choices were known. In this section we address the problem of making recommendations in the absence of previous choice history, i.e., recommending products to a new customer or predicting target customers for a new product. Collaborative filtering techniques as well as co-clustering applied in a recommender system setting [38] cannot easily deal with new customers and simply recommend the most popular products to new customers. We propose an attribute based co-clustering algorithm, which extends Bregman block average co-clustering, a special case of the Bregman co-clustering algorithm with basis 2 [5], to make use of customer and product attributes in addition to the matrix of customer product choices/ratings. Attribute based co-clustering uses the notion of similarity of new customers/products to existing ones based on attribute information in order to make recommendations.

4.2 Problem Definition

Let $Z_{m \times n}$ be a matrix of customers and products as described in Section 3.2.1, with cells representing the corresponding customer-product preference values i.e., ratings or binary choices. Each customer-product combination is represented by a vector $\mathbf{x_{uv}} = [z_{uv}, \mathbf{C_u}^T, \mathbf{P_v}^T]^T$ consisting of the actual matrix entry z_{uv} and the attributes of the corresponding customer ($\mathbf{C_u}$) and product ($\mathbf{P_v}$). w_{uv} is the weight associated with each customer-product combination. Let $d_{\phi 1}$, $d_{\phi 2}$ and $d_{\phi 3}$ be suitable Bregman divergences for computing the distances between the matrix entries and the customer and product attribute vectors respectively. Then, the distance between two customer-product vectors $\mathbf{x_{ab}}$ and $\mathbf{x_{cd}}$ is defined as the weighted combination of the Bregman divergences between the cell values and the customer and product attributes as follows:

$$d(\mathbf{x_{ab}}, \mathbf{x_{cd}}) = d_{\phi 1}(z_{ab}, z_{cd}) + w_1 d_{\phi 2}(\mathbf{C_a}, \mathbf{C_c}) + w_2 d_{\phi 3}(\mathbf{P_b}, \mathbf{P_d}).$$

If squared Euclidean distance is the selected Bregman divergence,

$$d(\mathbf{x_{ab}}, \mathbf{x_{cd}}) = (z_{ab} - z_{cd})^2 + w_1(\mathbf{C_a} - \mathbf{C_c})^T(\mathbf{C_a} - \mathbf{C_c}) + w_2(\mathbf{P_b} - \mathbf{P_d})^T(\mathbf{P_b} - \mathbf{P_d}).$$

The aim is to simultaneously cluster the m customers and n products into a grid of k row clusters and l column clusters such that the data reconstruction preserves the mean preference value and the mean customer and product attributes within each co-cluster. Note that the parent Bregman block average co-clustering algorithm preserves the co-cluster means in the reconstructed data matrix $\hat{Z}$ by approximating all the values in a co-cluster by the mean value [5]. Let ρ be a mapping

from the m rows to the k row clusters and γ be a mapping from the n columns to the l column clusters. We want to find a co-clustering (ρ, γ), that minimizes the following objective function

$$\sum_{g=1}^{k}\sum_{h=1}^{l}\sum_{u:\rho(u)=g}\sum_{v:\gamma(v)=h} w_{uv}d(\mathbf{x_{uv}}, \hat{\mathbf{x}}_{\mathbf{uv}}), \tag{4.1}$$

where $\mathbf{x_{uv}}$ is the vector corresponding to customer u and product v. $\hat{\mathbf{x}}_{\mathbf{uv}}$, the reconstructed vector is the mean vector for the co-cluster (g, h) given by

$$\hat{\mathbf{x}}_{\mathbf{uv}} = [\mu_{gh}, \boldsymbol{C\mu_{gh}}^T, \boldsymbol{P\mu_{gh}}^T]^T.$$

μ_{gh} is the mean of the customer-product preference values in co-cluster (g, h). $\boldsymbol{C\mu_{gh}}$ and $\boldsymbol{P\mu_{gh}}$ are the mean vectors of the customer and product attributes respectively within co-cluster (g, h). Each co-cluster hence has a customer prototype, a product prototype and a mean preference value.

Attribute based co-clustering hence aims at finding co-clusters of customers and products with similar preference values, customer attributes and product attributes. In the ideal case, all the customers and products in a co-cluster will have the same attributes and a constant preference value. Any deviations from this are modeled as the effects of noise. The generative noise model assumed by this approach is a mixture of $k \times l$ exponential family distributions, corresponding to the $k \times l$ co-clusters. Minimizing the objective function (4.1) is equivalent to maximizing the log likelihood of the underlying generative model. In the special case of the Bregman divergence being squared Euclidean distance, the generative noise model is a Gaussian mixture model consisting of $k \times l$ components. Each component

is a multivariate Gaussian with a diagonal covariance matrix, where the diagonal elements are $1, 1/w1, 1/w1...., 1/w2, 1/w2....$

4.3 Attribute Based Co-clustering Algorithm

A simple iterative algorithm can be used to find a co-clustering (ρ, γ), that minimizes the objective function (4.1). The objective function is the distance (error) between the original and predicted vectors summed over all the customer-product pairs in the matrix. It can hence be expressed as a sum of row/column errors. If row (customer) u is assigned to row cluster g i.e. $(\rho(u) = g)$, the row error is

$$E_u(g) = \sum_{h=1}^{l} \sum_{v:\gamma(v)=h} w_{uv} d(\mathbf{x_{uv}}, \hat{\mathbf{x}}_{\mathbf{uv}}).$$

The best choice of the row cluster assignment for row u is the g that has the smallest error i.e. $\rho^{new}(u) = argmin_g E_u(g)$. A similar approach is used to assign columns to column clusters. Assigning each row/column to the row/column cluster with the smallest error reduces the objective function. Given the current row and column cluster assignments (ρ, γ), the data reconstruction has to be updated by computing the required co-cluster statistics, which involves solving the Minimum Bregman Information (MBI) problem for basis 2 [5]. For squared Euclidean distance as the Bregman divergence, the MBI solution is simply the mean of all the customer-product vectors $\mathbf{x_{uv}}$ in the co-cluster. The MBI solution for the model update step is guaranteed to decrease the objective function. Since the objective function is bounded from below by zero the algorithm is guaranteed to converge

to a locally optimal solution. The pseudo-code for the resulting algorithm is given in Figure 4.1. The weights $w1$ and $w2$ are determined by cross-validation. Note that Bregman block average co-clustering is a special case of the attribute based co-clustering algorithm with $w1 = w2 = 0$.

Algorithm
Input: $Z_{m\times n}$, $W_{m\times n}$, $C = [\mathbf{C_1}..\mathbf{C_m}]$, $P = [\mathbf{P_1}..\mathbf{P_n}]$
Output: Co-clustering (ρ, γ) and co-cluster prototypes
1. Begin with a random co-clustering (ρ, γ)
2. **Repeat**
Step 1
3. Update co-cluster models
4. for $g = 1$ to k do
5. for $h = 1$ to l do
6. Compute the mean vector $[\mu_{gh}, \boldsymbol{C\mu_{gh}}^T, \boldsymbol{P\mu_{gh}}^T]^T$
7. of all the customer-product pairs in co-cluster (g, h)
8. end for
9. end for

Step 2(a)
10. Update ρ - assign each row to the row cluster that minimizes the row error
11. for $u = 1$ to m do
12. $\rho(u) = argmin_g \sum_{h=1}^{l} \sum_{v:\gamma(v)=h} w_{uv} d(\mathbf{x_{uv}}, \hat{\mathbf{x}}_{\mathbf{uv}})$
13. end for

Step 2(b)
14. Update γ - assign each column to the column cluster that minimizes the column error
15. for $v = 1$ to n do
16. $\gamma(v) = argmin_h \sum_{g=1}^{k} \sum_{u:\rho(u)=g} w_{uv} d(\mathbf{x_{uv}}, \hat{\mathbf{x}}_{\mathbf{uv}})$
17. end for
Until convergence
18. return (ρ, γ) and the co-cluster prototypes

Figure 4.1: Pseudo-code for the attribute based co-clustering algorithm

Making recommendations for new customers/products.

The co-clustering (ρ, γ) along with the co-cluster statistics, namely the pro-

totype customer and product attributes and the mean preference value can be used to make recommendations for new customers or products. To recommend products to a new customer c with attributes $\mathbf{C_{new}}$, the customer is first assigned to the row cluster g that has the closest customer prototype $\boldsymbol{C\mu_g}$ to $\mathbf{C_{new}}$ i.e.

$$\rho(c) = \text{argmin}_g d_\phi(\mathbf{C_{new}}, \boldsymbol{C\mu_g}).$$

From among the l co-clusters in the assigned row cluster g, the co-cluster (g, h) with the highest mean preference value μ_{gh} consists of the products that customers very similar to the new customer prefer the most. The set of products belonging to co-cluster (g, h) are hence recommended to the new customer. A similar approach is used to predict the customers who would be most likely to purchase a new product. The product is first assigned to the closest column cluster h. From among the co-clusters in the column cluster h, the one with the highest mean preference value is selected and the new product is recommended to the customers in the selected co-cluster.

4.4 Experimental Evaluation

We compare the attribute based co-clustering approach with a traditional collaborative filtering recommender system and a latent class segmentation approach, commonly used in the marketing community. Specifically, we use the product and customer segmentation approaches [40] as representatives of a latent class segmentation approach. The product segmentation approach, used to recommend products to a customer uses latent class modeling to identify product segments such

that the products in a segment attract similar customer types. The customer segmentation approach predicts target customers for a product and is based on identifying customer segments that prefer similar product types. Additionally, the comparison includes a baseline, a single logistic regression model with the customer-product choices as the target variable and the corresponding attributes as the independent variables.

The dataset used to compare the recommendations made by the different algorithms consists of elective course choices made by masters students in the MBA program at a large Midwestern University [40]. The 32 included courses are labeled by the department offering the course: (1) Accounting, (2) Finance, (3) Management of Information Systems, (4) Management, and (5) Marketing. The dataset includes 326 students, labeled by the student's career aspiration: : (1) Investment Banker, (2) Corporate Finance, (3) Technology Manager, (4) General Manager, (5) Brand Manager, (6) Consultant, and (7) Other. The student-course choice data is represented as a binary matrix of students vs. courses.

We evaluate the algorithms based on two tests conducted on the dataset described above. The first test investigates the effectiveness of each algorithm at recommending courses for a new student to take, given the student's career aspiration. The second test compares the algorithms on the basis of their accuracy in predicting the set of students a new course should be targeted to, given the department offering the course. For both tests, the leave-one-out evaluation technique is used, in which given a dataset of N observations, 1 observation is withheld at a time, the model is estimated with the remaining $N - 1$ observations and a recommendation is then

made for the withheld observation. The evaluation metrics are averaged over the N withheld observations. Since collaborative filtering requires knowledge of previous choices to make recommendations, a randomly selected subset of 4 courses taken by a withheld student and 10 students who selected a withheld course are given as inputs to the collaborative filtering approach for tests 1 and 2 respectively.

Algorithm	Precision	Recall	F-measure
Product Segmentation	0.27 (0.01)	0.50 (0.02)	0.33 (0.01)
Attr. Based Co-clustering	0.37 (0.01)	0.42 (0.02)	0.37 (0.01)
Collaborative Filtering	**0.48 (0.02)**	0.38 (0.02)	**0.39 (0.01)**
Logistic Regression	0.19 (0.01)	**0.78 (0.02)**	0.29 (0.01)

Table 4.1: Comparison of recommendation approaches for Test 1 (recommending a set of courses for a new student)

Algorithm	Precision	Recall	F-measure
Customer Segmentation	**0.37 (0.04)**	**0.60 (0.05)**	**0.42 (0.04)**
Attr. Based Co-clustering	0.35 (0.04)	0.56 (0.04)	0.38 (0.03)
Collaborative Filtering	0.31 (0.03)	0.52 (0.03)	0.35 (0.02)
Logistic Regression	0.30 (0.05)	0.57 (0.08)	0.24 (0.04)

Table 4.2: Comparison of recommendation approaches for Test 2 (predicting a set of students for a new course to target)

The precision, recall and F-measure of the recommendations made by the different approaches for tests 1 and 2 are illustrated in Tables 4.1 and 4.2. The performance of collaborative filtering and the latent class modeling approaches is heavily influenced by the size and dimensionality of the dataset [40]. Collaborative filtering gives the best F-measure on test 1, but when the data dimensionality increases in test 2, its performance degrades. The product and customer segmentation approaches rely on the informativeness of the customer and product attributes

respectively and give poor performance in the absence of informative attributes. Hence, while the customer segmentation approach performs well on test 2, due to informative course attributes with an average NMI of 0.23 with the target variable, the product segmentation approach does not do very well due to the less informative student attributes (NMI = 0.02 with target variable). In contrast, attribute based co-clustering is more stable and gives consistent performance on the two tests, with an F-measure in between that of the collaborative filtering and latent class modeling approaches. Since co-clustering simultaneously clusters along both axes and performs simultaneous dimensionality reduction, it is not very sensitive to data size and dimensionality. Moreover, the weights of the customer and product attributes w_1 and w_2 are automatically tuned based on the utility of the attributes in the prediction process. We observed that for both tests the weights selected by cross validation had values > 0, indicating that including attributes in the co-clustering process does better as compared to ignoring them as Bregman co-clustering does. The low F-measures of the logistic regression model on both tests highlight that a single model is not adequate to capture the data heterogeneity and "divide and conquer" approaches that make use of neighborhood information give better results on such problems.

Chapter 5

Mining for the Most Certain Predictions from Dyadic Data

5.1 Introduction

In certain prediction problems, one is interested in only the most accurate or useful predictions rather than in the performance over the entire test/scoring data. Such scenarios typically arise in applications where acting on each prediction requires a certain quantity of a limited resource and/or the cost incurred depends on the inaccuracy of the prediction. Perhaps the most well known example comes from direct marketing where one is interested in identifying a subset of the target population most likely to respond to a solicitation (positive class) [100]. Since making irrelevant offers is not only unprofitable but also harmful to customer experience, the choice of the target sub-population is critical. In this context, the performance of a model is evaluated based on the response rate in the top x quintiles via a gain (lift) chart, rather than on the overall classification accuracy. Determining the most relevant documents/pages for a search query has a similar flavor. Note that these are both classification examples in which there is a rare positive class, and one tries to identify a small subset of the records that most likely belong to this class.

A qualitatively different consideration is encountered in certain regression

settings. Consider a stock market player in options, who wants to predict closing stock prices. The sign of price change is not critical as he can use both "puts" and "calls", and he can bet on only a limited number of stocks because of a limited budget. In this situation, the accuracy of prediction (difference from the actual closing value) for a limited number of stocks is key to this person's ability to maximize his expected returns. A similar situation is faced in predicting other futures (energy usage, currency rates, etc.) for arbitraging strategies. Note that there has been little study of selection/ranking of regression results based on accuracy in the data mining literature. This chapter focuses on the problem of ranking predictions of numeric quantities by their reliability, so that one is able to identify a subset of predictions that are the most accurate. Though the emphasis is on regression settings, we will also show how the methodology can be applied to classification problems as well.

The problem of estimating the uncertainty associated with the prediction of a regression model has been examined in the statistics and neural networks literature [48, 9, 54]. The typical approach is to learn a linear/non-linear map from the independent variables to the response variable and then estimate the uncertainty of each prediction as a function of the uncertainty in the regression coefficients. In this chapter we show that a radically different and more effective approach can be taken for DyaC data (1.2).

The problem we address in this chapter is to predict the missing entries in the customer-product matrix Z, along with an estimate of the goodness/certainty of the predictions, which is to be used for ranking. We begin by formulating a simple but effective ranking strategy using a single model learned on the data. Motivated

by the observation that in a realistic setting the dataset is typically too large and heterogeneous to be adequately represented by a single global model, this chapter proposes ranking strategies based on the Simultaneous CO-clustering And Learning (SCOAL) framework. Two novel ranking schemes are formulated, (i) Row-Column Ranking, and (ii) Block Ranking, and the use of multiple localized models learnt on dyadic data to obtain a better as well as more actionable ranking is demonstrated.

Another contribution of this chapter is Robust SCOAL, a novel predictive modeling technique that selectively models only the most coherent parts of the data. This approach is based on the key idea behind the Robust Overlapping Co-Clustering (ROCC) algorithm [26], which involves mining dense, coherent co-clusters from large, noisy datasets, by pruning away irrelevant regions of the data. Note that ROCC is only focused on clustering; no prediction models are formed. A different viewpoint of the Robust SCOAL technique is as an approach for automatic outlier detection and removal that is carried out simultaneously with the modeling process. Robust SCOAL identifies and retains only those parts of the data where accurate predictive models can be learnt. The predictions within the selected data subset can be subsequently ranked. Finally, we introduce a new certainty lift measure that evaluates the improvement in error for a selected subset of predictions.

In contrast to traditional techniques for error bar estimation [48, 9, 54], the approaches presented here can take advantage of the structure of dyadic data rather than treat it as a flat collection of points. Moreover, while most existing techniques are based on learning a single model using the available training data, we learn a collection of localized models that provide a better representation of the input space

and a finer resolution of prediction uncertainty. Comparisons of our ranking procedure with ranking based on standard methods of prediction error estimation on real life datasets highlight the benefits of our approach. Additionally, our ranking strategies with a collection of localized models are readily applicable to both regression and classification problems.

5.2 Related Work

Determining the standard error of the predictions of a linear regression model is well established, text-book material [55]. This is not as direct for non-linear models, and there has been a considerable amount of work in the neural network community on determining the uncertainty of network predictions [81, 54]. There are several references in the literature on the application of resampling based techniques for prediction error estimation. Heskes [48] uses the variance of the outputs of an ensemble of neural networks, trained on bootstrap replicates of the training dataset, for computing confidence intervals on the output for new data points. This technique is shown to do better than [81] on synthetic data. Tibshirani [105] conducts an empirical comparison of the delta estimator, the sandwich estimator and bootstrap methods to estimate the standard error for values predicted by a multi layer perceptron and concludes that the bootstrap methods provide the most accurate estimates. High computational effort however is one of the main drawbacks of resampling approaches. While resampling techniques construct multiple global models, each of which represents the entire input space, our approach is based on a collection of local models (Section 5.4). To the best of our knowledge, no one

has used an ensemble of localized models for estimating the accuracy of individual predictions.

Bayesian techniques provide another way of estimating prediction uncertainty based on the variance of the predictive distribution computed as a function of the input [10, 9]. We compare our proposed techniques with a Bayesian linear regression approach [10] in Section 5.9.

An example of a recent non-Bayesian approach is the work by Cawley et al. [14] where they assume heteroscedasticity as in [9] and use an iterative procedure based on kernel ridge regression to simultaneously model the mean and variance of the target distribution as functions of the input. A novel approach by Shrestha and Solomatine [101] involves clustering the residuals into homogeneous groups using fuzzy c-means and then constructing the prediction limits for each cluster, based on the distribution of the errors within the cluster. The prediction limits for each training point are then estimated according to its membership in each cluster. A linear/non-linear mapping learnt from each input point to its prediction intervals is used to make estimates for test points. While the learning of the model and clustering the model residues are sequential steps in this work, our approach, discussed in Section 5.4, is based on performing these two tasks simultaneously.

Another related body of work, specific to classification problems, is that on abstaining/cautious classifiers, where the classifier refrains from making a decision in cases where its prediction is uncertain or unreliable [18, 32]. The challenge here is to suitably trade off the reduction of the number of misclassified instances (errors) and the number of unclassified instances (abstentions). Recently, techniques that

use ROC analysis to efficiently build an abstaining binary classifier that is optimal according to three different, practically relevant performance criteria have also been proposed [85]. We develop a strategy to rank predictions based on their reliability (Section 5.6), using which, the most reliable n predictions can be selected, given n. Cautious classifiers are complementary since they can provide a suitable value of n based on the error rate/abstention tradeoff.

5.3 Ranking with a Single Model

A simple approach to the above problem is to train a single "global" model that learns a map from the covariates to the response variable. For ease of exposition, consider a linear model $y_k = \boldsymbol{\beta}^T \mathbf{x_k} + u_k$, where $(\mathbf{x_k}, y_k)$ is the k^{th} data point. u_k is the unobserved error term, assumed to be a random variable with $E(u_k|\mathbf{x_k}) = 0$.

In the dyadic data notation, the response variable $z_{ij} \in \mathbb{R}$ is estimated by a linear combination of the corresponding covariates: $\hat{z}_{ij} = \boldsymbol{\beta}^T \mathbf{x_{ij}}$, where $\mathbf{x_{ij}}^T = [1, \mathbf{r_i}^T, \mathbf{c_j}^T, \mathbf{a_{ij}}^T]$ is a vector consisting of the covariates augmented with $x_0 = 1$ for the intercept, and the least squares estimates of the model parameters are $\boldsymbol{\beta}^T = [\beta_0, \boldsymbol{\beta_x}^T]$. Let $\mathbf{T}$ represent the set of training data instances, of size s and let p be the number of independent variables.

Since $\boldsymbol{\beta}$ is estimated with imprecision, the predicted value $\hat{z}_{ij}$ is also subject to error. For the above multivariate linear regression model, the error variance is estimated by the following standard approach [55], which we refer to as Global Var

$$E_{ij}^{GlobalVar} = var(z_{ij} - \hat{z}_{ij}) \tag{5.1}$$
$$= \sigma^2[1 + \mathbf{x_{ij}}^T (Y^T Y)^{-1} \mathbf{x_{ij}}].$$

Y is the design matrix, with rows of the form $\mathbf{x_{uv}}^T$, where $(u, v) \in \mathbf{T}$, i.e., $w_{uv} = 1$ for a 0-1 weight matrix W. σ^2 can be estimated by the mean squared error, given by

$$\hat{\sigma^2} = \frac{\sum_{(u,v)\in\mathbf{T}} (z_{uv} - \hat{z}_{uv})^2}{s - p}.$$

The square root of (5.1) gives the standard error of the prediction $\hat{z}_{ij}$. In the experiments in Section 5.9, we use ridge regression with error variance estimated as

$$E_{ij}^{GlobalVar} = \sigma^2[1 + \mathbf{x_{ij}}^T (Y^T Y + \lambda I)^{-1} (Y^T Y)(Y^T Y + \lambda I)^{-1} \mathbf{x_{ij}}], \tag{5.2}$$

where $\lambda \geq 0$ controls the amount of regularization [34].

Other alternatives to estimate prediction uncertainty, as discussed in Section 5.2, are to employ a resampling based technique [48] or a Bayesian technique [10], where the uncertainty is estimated by the variance of the predictive distribution. However, all these procedures implicitly flatten the data matrix and hence fail to exploit the inherent dyadic structure of the data.

For dyadic data fitted with a single (linear or non-linear) model, we propose a new approach, referred to as Global Rank, to rank predictions by their estimated accuracy.

1. The average error for row i is its training mean squared error, computed as

$$E_i = \frac{\sum_{j=1}^{n} w_{ij}(z_{ij} - \hat{z}_{ij})^2}{\sum_{j=1}^{n} w_{ij}}.$$

2. Similarly, the average error for column j is given by

$$E_j = \frac{\sum_{i=1}^{m} w_{ij}(z_{ij} - \hat{z}_{ij})^2}{\sum_{i=1}^{m} w_{ij}}.$$

3. Assuming independence of row and column errors, the error for a missing entry z_{ij} is given by

$$E_{ij}^{GlobalRank} = E_i + E_j. \tag{5.3}$$

4. A ranking of the predicted entries is obtained based on their computed errors.

While a single learned model is adequate for simple prediction problems, it may not be sufficient to represent the heterogeneous population that very large classification or regression problems often involve. For instance, the problem of predicting customer preferences for products typically involves a diverse population of customers and a wide range of different products. It is unlikely that all the customer-product preferences can be well explained by a single model. Since it is more natural for a model to closely represent the preferences of only a subset of the customers for a subset of the products, the data might be better modeled by a collection of suitably trained localized models, e.g., obtained by using the SCOAL approach.

5.4 Ranking with a Collection of Local Models

By generating multiple localized models, SCOAL provides novel ways to rank predictions by their expected quality. In this section we propose two such approaches: (i) Row-Col Ranking, (ii) Block Ranking.

Row-Col Ranking. The Row-Col Ranking technique is based on an estimation of the prediction errors as given by Equation (5.3). The average error for row i, where $\rho(i) = g$ is given by

$$E_i = \frac{\sum_{h=1}^{l} \sum_{j:\gamma(j)=h} w_{ij}(z_{ij} - \boldsymbol{\beta^{gh}}^T \mathbf{x_{ij}})^2}{\sum_{j=1}^{n} w_{ij}}.$$

The average error for column j is computed similarly. As before, the error for a missing entry z_{ij} is the sum of the corresponding average row and column errors. Intuitively, since the collection of local models characterize the data better than a single model, the resulting ranking would be expected to be better as well.

Block Ranking. The intuition here is that for a heterogeneous dataset the fit of the co-cluster models, quantified by the training reconstruction error (MSE), differs across co-clusters. Models in noisy and sparse regions of the input space will have poorer fit, while other models may capture more coherent relationships between the covariates and the response variables. Hence, the first step is to rank the predictions by the mean squared error of the co-clusters they are assigned to. The mean error for co-cluster (g, h) is computed as

$$\frac{1}{\sum_{i:\rho(i)=g} \sum_{j:\gamma(j)=h} w_{ij}} \sum_{i:\rho(i)=g} \sum_{j:\gamma(j)=h} w_{ij}(z_{ij} - \boldsymbol{\beta^{gh}}^T \mathbf{x_{ij}})^2.$$

Ranking by co-cluster membership helps in identifying regions of the input space that are not adequately represented by the learnt models and gives the predictions in these regions a lower rank. Further, a second level of ranking can be done within each co-cluster (g, h), in which the predictions are ranked by estimating their prediction errors as a sum of the average co-cluster row and co-cluster column errors, i.e., the error for a predicted value in row i, column j is $E_i + E_j$, where,

$$E_i = \frac{\sum_{k:\gamma(k)=h} w_{ik}(z_{ik} - \hat{z}_{ik})^2}{\sum_{k:\gamma(k)=h} w_{ik}},$$

$$E_j = \frac{\sum_{k:\rho(k)=g} w_{kj}(z_{kj} - \hat{z}_{kj})^2}{\sum_{k:\rho(k)=g} w_{kj}}.$$

SCOAL and the associated Row-Col and Block Ranking procedures are readily extensible to other predictive models as described in Section 3.5. In our experiments described in Section 5.9 we use ridge regression, which reduces the possibility of overfitting while learning a larger number of model parameters ($k \times l$ $\boldsymbol{\beta}$ vectors) as compared to a single linear regression model.

5.5 Modeling a Selected Data Subset

Robust SCOAL is a robust predictive modeling technique that identifies and models only the most coherent regions of the data to give high predictive accuracy on the selected subset of target values. This idea is motivated by our work in co-clustering, Robust Overlapping Co-clustering (ROCC) [26], a novel approach for discovering dense, arbitrarily positioned co-clusters in large, possibly high dimensional datasets. Note that [26] is only focused on clustering; no prediction models

are formed. ROCC is robust in the presence of noisy and irrelevant objects as well as features, which it automatically detects and prunes during the clustering process. ROCC is also robust to the noise model of the data and can be tailored to use the most suitable distance measure for the data, i.e., any Bregman divergence. The final objective of ROCC is achieved in two steps. In the first step, the Bregman Co-clustering algorithm [5] is adapted to automatically prune away non-informative data points and perform feature selection by eliminating non-discriminative features and hence cluster only the relevant part of the dataset. This step finds co-clusters arranged in a grid structure, but only a predetermined number of rows and columns are assigned to the co-clusters. An agglomeration step then appropriately merges similar co-clusters to discover dense, arbitrarily positioned and even overlapping co-clusters. ROCC is very well suited to several real life applications such as microarray data clustering and market basket analysis.

Robust SCOAL aims at simultaneously co-clustering DyaC data and learning predictive models in each co-cluster, with the added constraint that only a predetermined number of rows and columns are assigned to row and column clusters. More concretely, our aim here is to simultaneously select and cluster $s_r \leq m$ rows and $s_c \leq n$ columns of Z, into a grid of k row clusters and l column clusters, such that the response values in each co-cluster are predicted by a common regression model. The exact value of s_r and s_c is not critical in the setting of this paper, as this induces the first stage of selection only, and a second stage is needed in any case so long as s_r and s_c are not too small. Let $\mathbf{K}$ and $\mathbf{L}$ denote the sets consisting of the s_r clustered rows and the s_c clustered columns respectively. Let ρ be a mapping from

the s_r rows $\in \mathbf{K}$ to the k row clusters and γ be a mapping from the s_c columns $\in \mathbf{L}$ to the l column clusters. We want to find a co-clustering defined by (ρ, γ), sets $\mathbf{K}$ and $\mathbf{L}$ and the associated set of regression models $\{\boldsymbol{\beta}\text{s}\}$ for specified s_r and s_c that minimize the following objective function:

$$\frac{1}{\sum_{u\in\mathbf{K},v\in\mathbf{L}} w_{uv}} \sum_{g=1}^{k}\sum_{h=1}^{l} \sum_{u\in\mathbf{K}:\rho(u)=g} \sum_{v\in\mathbf{L}:\gamma(v)=h} w_{uv}(z_{uv}-\hat{z}_{uv})^2, \tag{5.4}$$

where $\hat{z}_{uv} = \boldsymbol{\beta}^{\boldsymbol{gh}^T}\mathbf{x_{uv}}$.

Note that the objective function (5.4) (i) involves the squared error summed only over $s_r \times s_c$ elements of Z, and that (ii) it is based on the mean error rather than the total error. In the presence of missing entries in the data matrix, the denominator in (5.4) will differ for different choices of sets $\mathbf{K}$ and $\mathbf{L}$ and hence needs to be considered in the cost function. This point is especially important for sparse data, where a substantial number of matrix entries are missing. In this scenario, if the total error was optimized rather than the mean error, the solution would be a very sparse subset of the data with highly overfit models, since missing entries have $w_{uv} = 0$ and do not contribute to the error. In the most extreme case, the solution would be an empty $s_r \times s_c$ block of the matrix Z with zero error.

5.5.1 Robust SCOAL Algorithm

As in the case of SCOAL, a co-clustering (ρ, γ), that decreases the cost function can be obtained by an iterative algorithm that alternately updates co-cluster models and row/column cluster assignments. The algorithm begins by initializing

(ρ, γ) either randomly or by a suitable clustering procedure followed by a selection of s_r rows and s_c columns. Given the initial cluster assignments (ρ, γ) of the selected s_r rows and s_c columns, the co-cluster models are constructed, i.e., the coefficient vector $\boldsymbol{\beta}$ is learnt for each co-cluster. In case of 0/1 weights, this is done by least squares regression using only the non-missing (training) values within the co-cluster. In case of a general set of weights, the $\boldsymbol{\beta}$ is a solution to a weighted least squares problem.

For a fixed column cluster assignment and a set of co-cluster models, the row cluster update step involves assigning rows to row clusters and selecting s_r of m rows to participate in the current clustering. If row u is assigned to row cluster g, i.e., $\rho(u) = g$, the row error is the error summed over the appropriate s_c elements in the row given by

$$E_u(g) = \sum_{h=1}^{l} \sum_{v \in \mathbf{L}: \gamma(v)=h} w_{uv}(z_{uv} - \hat{z}_{uv})^2.$$

For a fixed $\mathbf{L}$,γ and set of $\boldsymbol{\beta}$s, the best choice of the row cluster assignment for row u is the g that minimizes the row error, i.e., $\rho^{new}(u) = \arg_g \min\, E_u(g)$. Each of the m rows is hence assigned to the row cluster that minimizes the row error.

Since the objective function (5.4) includes the sum of weights term in the denominator, it cannot be decomposed over the rows. A naive greedy solution of selecting the s_r rows with lowest mean row error is hence not optimal for the row selection problem. Instead, we select the s_r rows that minimize

$$\frac{\sum_{u \in \mathbf{K}} E_u(\rho^{new}(u))}{\sum_{u \in \mathbf{K}} \sum_{v \in \mathbf{L}} w_{uv}}.$$

An optimal selection of s_r rows can now be obtained by using a standard dynamic programming approach akin to the solution of the knapsack problem in [63].

A similar approach is used to update the column clusters. Note that the rows/columns that are not included in the current s_r/s_c rows/columns assigned to co-clusters are still retained since they could be included in the co-clusters in future iterations. The algorithm alternates between the row/column cluster update and model update steps until convergence. The overall algorithm is described in Figure 5.1. Step 1 reduces the objective function because of the optimality of the pseudo-inverse solution to linear regression, steps 2 and 3 are greedy updates and directly reduce the objective function. The objective function hence decreases in every iteration. Since this function is bounded from below by zero, the algorithm is guaranteed to converge to a local minimum.

If the data is extremely sparse then, for given s_r and s_c, the row/column selection step could involve empty rows or columns, which have zero weight and hence zero cost. Although selecting such rows/columns will be optimal from the point of view of the cost function, it will not result in a well generalizable set of models for making accurate predictions on test data. We hence restrict the dynamic programming steps (2b and 3b) to select from the set of non-empty rows/columns. This avoids the modeled part of the data matrix from including rows and columns with all entries missing.

While steps 1, 2a and 3a of the Robust SCOAL algorithm are linear in the size of the data, the complexity of steps 2b and 3b is O(ms_r) and O(ns_c), i.e., quadratic in the worst case. If the number of missing entries in the data matrix is

Algorithm: Robust SCOAL
Input: $Z_{m\times n}$, $W_{m\times n}$, covariates, s_r, s_c,k,l
Output: (ρ, γ), co-cluster models $\boldsymbol{\beta}$s
1. Begin with an initial co-clustering (ρ, γ)
2. Repeat
3. **Step 1: Update co-cluster models**
4. $\forall\, [g]_1^k$,$[h]_1^l$,
5. Train a linear regression model with
6. all the training samples $(\mathbf{x_{uv}}, z_{uv})$ in
7. co-cluster (g, h), with weights w_{uv},
8. to obtain an updated $\boldsymbol{\beta^{gh}}$.

9. **Step 2: Update** ρ
10. **2a.** $\forall\, [u]_1^m$,
11. $\rho(u) = \arg_g \min \sum_{h=1}^{l} \sum_{v\in L:\gamma(v)=h}$
12. $w_{uv}(z_{uv} - \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}})^2$
13. **2b.** **K** = the set of s_r rows selected
14. by dynamic programming.

15. **Step 3: Update** γ
16. **3a.**$\forall\, [v]_1^n$,
17. $\gamma(v) = \arg_h \min \sum_{g=1}^{k} \sum_{u\in K:\rho(u)=g}$
18. $w_{uv}(z_{uv} - \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}})^2$
19. **3b.** **L** = the set of s_c columns selected
20. by dynamic programming.
21. Until Convergence
22. Return (ρ, γ) and $\boldsymbol{\beta}$'s

Figure 5.1: Pseudo-code for Robust SCOAL

comparatively small and the entries are missing at random, then the $\sum_{u\in\mathbf{K},v\in\mathbf{L}} w_{uv}$ term in the objective function (5.4) does not vary too much over different $s_r \times s_c$ blocks of the matrix and can be ignored. An efficient heuristic then is to replace the dynamic programming steps 2b and 3b by a greedy selection of the s_r rows and s_c columns with the lowest row and column errors respectively. This greedy selection can be performed in linear time by the efficient order statistics based al-

gorithm proposed by Blum et al. [11]. We verified empirically on the ERIM dataset (Section 5.9) that there is no significant difference between the greedy strategy and dynamic programming in terms of prediction accuracy.

Similar to the Row-Col Ranking procedure described in Section 5.4, the Robust SCOAL predictions can be ranked by an estimation of the prediction error, which is the sum of the average row and column errors. Note that the average row/column error is computed over only the rows and columns that belong to sets $\mathbf{K}$ and $\mathbf{L}$ respectively.

Analogous to the discussion in Section 3.5, Robust SCOAL can also be generalized to other predictive models and/or loss functions. For instance, the experiments in Section 5.9 use Robust SCOAL with ridge regression.

5.6 Extension: Ranking for Classification Problems

The focus so far has been on assessing the reliability of the output of a regression model. In classification problems as well, it may be of interest to select a specified number of the most accurate or reliable predictions. A motivating example is the ecological problem of predicting the absence/presence of a set of species in a set of locations. Extremely reliable predictions for a few location-species pairs lead to more fruitful ground research rather than the burdensome task of investigating a larger set of predictions with lower overall accuracy. In this section, we show that if a classification problem involves data with a dyadic structure as described in Section 5.1, we can adapt the proposed ranking technique to obtain a ranking of the classifier predictions by their reliability. Analogous to the regression setting,

note that for a 2-class problem we treat the 2 classes symmetrically and aim to find the most accurate predictions, irrespective of the predicted class label. Our perception of prediction "accuracy" in this case is simply the probability of correct classification for $\mathbf{x_{ij}}$, and so aposteriori class probabilities very close to 0 or 1 are ranked higher as compared to probabilities close to the prior. Hence, if the data is modeled by a single, "global" probabilistic classifier, $\max_i(P(\text{class}_i|x_k))$ can be used to rank predicted values.

For DyaC data, the response variable is represented as the entries of a matrix Z as before, with covariates associated with the corresponding modes. The entries in matrix Z are now class labels, e.g., $+1, -1$ for 2-class problems. We use the SCOAL (meta)-algorithm to simultaneously partition the matrix into co-clusters and learn classification models in each co-cluster. One can use logistic regression classifiers, in which case the SCOAL objective function will be the total log loss. We then get a ranking of the missing class labels based on the class membership probabilities estimated by the co-cluster models. We illustrate in Section 5.9.3 that the ranking provided by this approach is better than that obtained by a global model. Experimental evaluation shows that the ranking provided by this approach is better than that obtained by a global model. The Robust SCOAL algorithm described in Section 5.5.1 can also be easily extended to classification models by suitably changing the objective function (5.4) to the loss function of the classifier.

5.7 Contrasting Different Ranking Techniques

The ranking techniques discussed so far in Sections 5.3, 5.4 and 5.5 result in 3 conceptually different approaches for selecting the subset of the missing entries to be predicted. Both Global Rank and Row-Col Ranking select individual entries in the data matrix. In contrast, for Block Ranking the unit of selection is a co-cluster rather than individual matrix entries, since the first level of ranking is based on co-cluster membership. Finally, Robust SCOAL selects the most accurately predictable parts of the data matrix by throwing away rows and columns with large errors. Its unit of selection is hence individual rows and columns. The selection technique that is the most relevant to a certain problem depends on the application.

In applications involving extremely large datasets, Robust SCOAL identifies a smaller subset of the data that can be focused on and analyzed in further detail. For example, in a direct marketing application, it identifies a group of customers and products for which predictions can be made with very high confidence. This result is possibly more actionable from a marketing viewpoint than selecting individual customer-product combinations in an unstructured manner from the entire customer-product base. Robust SCOAL is also well suited to datasets known to have substantial outliers since it performs simultaneous outlier detection and pruning during the modeling process.

Selection at the co-cluster level is the most appropriate when the subset of the predictions on which decisions/actions are taken is required to satisfy certain criteria imposed by the application. For example, a certain e-commerce application might constrain the predictions to be localized to a certain geographical region or to

users in a certain age or income group. A retail business may want to identify customers whose purchase decisions are heavily influenced by the price, and provide special discounts to them on products they are most likely to purchase. Ranking by co-clusters is also very interpretable since the co-cluster models can provide insights on the degree to which different factors influence customer purchases in various customer-product subgroups.

An interesting point is that the proposed ranking approaches can be combined to make selections at multiple resolutions. Robust SCOAL can first select a set of rows and columns, which can be followed by a ranking of the co-clusters, with a finer ranking of individual entries within each co-cluster.

5.8 Certainty Lift

Having discussed a number of approaches to select the most certain predictions, we now formulate a "certainty lift" measure that evaluates the improvement in the error of a selected subset of model predictions as compared to the error of a baseline model on the entire test set. With squared error as the loss function, the certainty lift is defined as

$$\text{Certainty Lift}(model, \mathbf{V}) = \frac{\frac{1}{s}\sum_{i\in\mathbf{U}}(y_i - \bar{y}_{\mathbf{T}})^2}{\frac{1}{\acute{s}}\sum_{i\in\mathbf{V}}(y_i - y^{model})^2}, \tag{5.5}$$

where $\mathbf{U}$ represents the test set, $\mathbf{V} \subseteq \mathbf{U}$, and $s = |\mathbf{U}|$ and $\acute{s} = |\mathbf{V}|$. The baseline model is simply the mean of the response values in the training dataset, given by $\bar{y}_{\mathbf{T}}$ and y^{model} is the value output by a predictive model learnt on $\mathbf{T}$. For

example, for a given model, a certainty lift of 5 on a subset $\mathbf{V}$ of $\mathbf{U}$, implies that the mean squared error on $\mathbf{V}$ is 5 times lower than the baseline error on $\mathbf{U}$.

5.9 Experimental Evaluation

In this section we evaluate the proposed ranking procedures on real life datasets from two different regression applications. Section 5.9.1 presents results on the MovieLens data, while Section 5.9.2 discusses results on the ERIM Marketing dataset.

5.9.1 MovieLens Dataset

We evaluate our ranking techniques on the MovieLens dataset described in Section 3.9.2. We assume ridge regression models to predict the movie ratings and evaluate the quality of the predictions using mean squared error. We compare with three traditional approaches for computing the uncertainty of predictions of a linear regression model. The first is Global Var, which computes the standard error of a predicted value using Eq. (5.2) [1]. The second approach, Global Resample, trains a set of ridge regression models on bootstrap samples of the training dataset and computes the variance of the values predicted by the ensemble of models. A ranking of the data points in the test set is then obtained by sorting them in ascending order of their estimated standard error/variance values. We also consider Bayesian linear regression (Bayesian LinReg), with a zero-mean, isotropic Gaussian prior [10].

[1] The value of the regularization parameter (λ), selected by cross validation, came out to be 0 for a global ridge regression model. Hence, the standard error is equivalent to that given by Eq. (5.1).

The precision parameter of the Gaussian prior is selected by cross validation, while the variance of the additive Gaussian noise is set to the MSE obtained with a linear least-squares model. The ranking is based on the estimated variance of the predictive distribution for each test point. Figure 5.2 compares the Global Var, Global Resample (with 25 bootstrap samples) and Bayesian LinReg techniques with Global Rank, which uses the dyadic data structure to provide the ranking (Eq. (5.3)), the SCOAL based Row-Col Ranking and Block Ranking and Robust SCOAL. $k = 4$ and $l = 4$ for the SCOAL techniques. Robust SCOAL is run to select $s_r = 890$ rows and $s_c = 1550$ columns, followed by Row-Col Ranking to rank order the selected predictions. The SCOAL techniques use ridge regression models with $\lambda = 0.78$, selected by cross validation. The x-axis specifies the fraction of the points in the test set that are predicted, while the y-axis gives the cumulative mean squared error on the predicted set of points. At the last point on the x-axis all the entries in the test set are predicted and included in the MSE. The plotted values are averaged over 10 random $80 - 20$ % splits of the data as the training and test set.

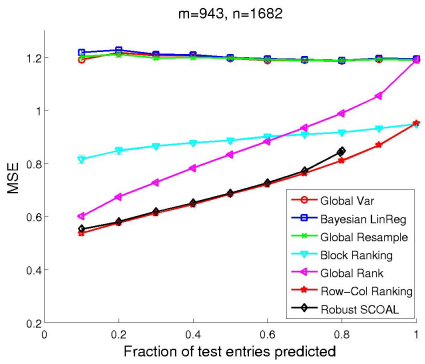


Figure 5.2: Comparison of ranking techniques on the MovieLens dataset.

The plot shows a significant improvement in the ranking obtained by techniques that use the dyadic data structure as compared to Global Var, Global Resample and Bayesian LinReg (almost superimposed in Figure 5.2, with MSE around 1.2). Since a collection of co-cluster models characterizes the data better than a single global model, the MSE of SCOAL on the entire training dataset is significantly lower than the MSE of a single global model, illustrated by the last point on the x-axis. Row-Col Ranking results in a useful ranking of the predicted values, with 10% of the test entries very accurately predicted with a MSE of 0.52. Robust SCOAL performs as good as Row-Col ranking, while Block Ranking, which ranks predicted values by co-cluster MSE, results in a flatter plot, possibly due to its more constrained ranking mechanism.

Figure 5.3 illustrates the certainty lift, computed according to Eq. (5.5), for different prediction techniques. The SCOAL based approaches as well as Global Rank provide substantial improvement over the baseline and also do much better than Global Var.

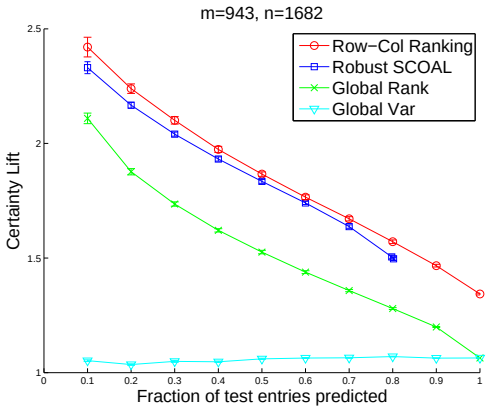


Figure 5.3: Certainty Lift of different prediction approaches on the MovieLens dataset.

5.9.2 ERIM Marketing Dataset

In this section we use the publicly available ERIM dataset described in Section 3.9.1. The results below are obtained on the entire dataset, including outliers. Figure 5.4 compares various ranking techniques on the ERIM dataset. The SCOAL techniques are run with $k = 4$ and $l = 4$ and Robust SCOAL is run with $s_r = 1700$, $s_c = 150$. The ridge regression parameter λ for the SCOAL techniques is set to 0.98 and is selected by cross validation. The λ selected for a global ridge regression model is 0.84. Note that the MSE is computed on the original data, by back transforming the standardized values. The trend of the results is similar to that in Figure 5.2. Global Rank and the SCOAL approaches (Row-Col and Block Ranking) that use the dyadic data structure do significantly better than the resampling, the Bayesian linear regression and the standard error estimation approaches. Although the average prediction error of Row-Col Ranking and Global Rank on the entire test set is almost the same, one can observe that over different fractions of the test entries predicted, Row-Col Ranking is slightly better than Global Rank. This is possibly due to a better estimation of prediction errors caused by an improvement in the overall fit of the co-cluster models as compared to a single global model. Figure 5.5 compares the certainty lift for different prediction techniques. Observe that Row-Col Ranking gives an improvement of almost 25 times on 10% of the test set.

Linear regression based on the squared error cost function is not very robust to outliers and a few outliers could completely skew the model results. While the use of ridge regression models mitigates the effect of outliers to some extent, there

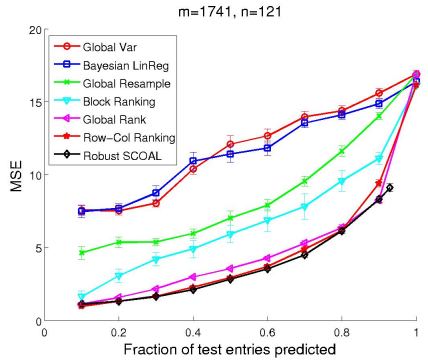


Figure 5.4: Comparison of ranking techniques on the ERIM dataset.

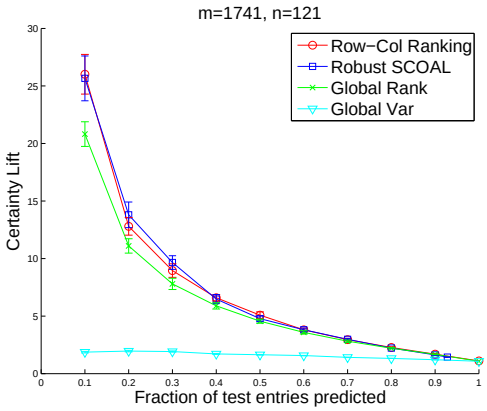


Figure 5.5: Certainty Lift of different prediction approaches on the ERIM dataset.

is still some sensitivity to outliers, resulting in a large MSE on the entire test set. We now evaluate the ability of Robust SCOAL to automatically identify and prune away outliers, where an outlier is defined as a response value (# of units purchased) > 20. Table 5.1 displays the percentage of the total number of outliers detected and discarded by Robust SCOAL for different sizes of the data modeled, specified by the s_r and s_c values. k and l are both set to 4 for Robust SCOAL. The results are averaged over 10 random $80 - 20$ % train-test splits of the data. The last row of the table is for the case where $s_r = m$ and $s_c = n$, i.e., no rows and columns are

discarded. One can observe that the % of outliers discarded is consistently much larger relative to the % of data discarded. The ratio of the % of outliers discarded to the % of data discarded (last column) is hence high, indicating that Robust SCOAL is able to selectively prune away outliers. It is interesting that discarding even a few rows and columns results in throwing away a large fraction of the outliers. This results in a dramatic reduction in the prediction error on the selected subset of test points, which is still a substantial fraction of the missing values, e.g., the MSE on 85.9 % of the test points is 8.225 when $s_r = 1573$ and $s_c = 115$, as compared to a MSE of 15.787 on the entire test set.

s_r	s_c	% outliers discarded	% data discarded	Ratio
900	90	98.414	61.549	1.599
1068	96	96.128	51.330	1.873
1236	102	91.359	40.154	2.275
1405	109	83.889	27.303	3.073
1573	115	70.770	14.130	4.949
1741	121	0	0	-

Table 5.1: Outlier pruning by Robust SCOAL for varying s_r and s_c values on the ERIM dataset.

5.9.3 MBA Student Course Dataset

In this section, we evaluate the proposed ranking techniques on the classification problem that arises on the MBA student course recommendation dataset (described in Section 3.8.2). We use logistic regression models for prediction. Figure 5.6 compares the ranking of the predicted values obtained by a collection of local models, learnt using SCOAL with that obtained by a single global model. For a missing entry z_{ij} in the student-course matrix Z, both approaches

estimate $P(z_{ij} = 1|\mathbf{x_{ij}})$. The missing entries are ranked in descending order of $\max(P(z_{ij} = 1|\mathbf{x_{ij}}), P(z_{ij} = -1|\mathbf{x_{ij}}))$. The y-axis in Figure 5.6 is the log loss averaged over the predicted entries. SCOAL is run with $k = 2, l = 2$. As in the regression case, since the local models represent the data better as compared to a single global model, they result in a better ranking.

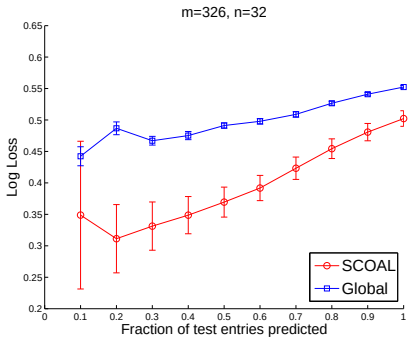


Figure 5.6: Comparison of ranking using SCOAL vs. a single classification model on the MBA dataset.

5.10 Concluding Remarks

The task of selecting the top n predictions based on their certainty has not been adequately explored, although it is very naturally applicable in a wide range of practical scenarios. In this chapter we propose novel approaches for mining the most certain predictions in regression and classification problems involving large, heterogeneous data with a DyaC structure. Our approaches are based on the key idea of modeling the data by a collection of multiple, localized models learnt by exploiting the dyadic structure. As illustrated in Section 5.9, the proposed ranking techniques give superior results on a variety of real life datasets as compared to classical uncertainty estimation approaches based on global models. Of the pro-

posed ranking methods, Robust SCOAL and Block Ranking are more useful than Row-Col Ranking from a practical viewpoint since they enable selection of structured subspaces of the data, which is more actionable. Another advantage of our multiple models based approach is that the selection of "certain" predictions can be done at varying resolutions, providing application specific flexibility as well as interpretability.

The idea of "certainty lift" can be useful in several applications. In addition, it can be used as a basis for developing active learning techniques that are radically different from traditional uncertainty sampling approaches. Another aspect worth exploring is the use of robust error functions, which suitably weigh errors from different entries in the data matrix differently. This will provide a "soft" selection of the subset of the data to be modeled, which may be more flexible than pruning entire matrix rows/columns as done by Robust SCOAL.

Chapter 6

Active Learning with Multiple Localized Models for Dyadic Data

6.1 Introduction

In several business domains, obtaining labeled training data for prediction tasks is costly. For instance, for effective prediction of consumers' ratings in large scale recommender systems, it is essential to have many customers rate a large number of products. However, consumers often do not provide their preferences without proper incentives. Given a budget to reward consumers for their feedback, it would be beneficial to have a policy to suggest which customers' ratings and for what products would be most cost-effective to acquire so as to improve the prediction of consumer ratings the most. Similarly, in the problem of predicting click-through rates for a given advertisement served with a given web page, the numerical response variable values are often acquired through costly online experimentation [64]. Because of the opportunity costs incurred when serving a suboptimal ad for a given page in order to acquire the response rate, it is desirable to identify what ad-page click-through rates would be most cost-effective to acquire in order to improve the model's prediction accuracy the most. This general challenge can be mapped to the problem of pool-based *active learning* [20] of regression models, wherein a learner aims to intelligently acquire the numerical dependent variable value of particularly

informative instances from a pool of prospective acquisitions, so as to improve the generalization accuracy of the prediction model the most.

In this chapter we focus on pool based active learning for regression problems, such as predicting customer-product preferences. Most research on active learning considers the induction of classification models and these policies are not directly applicable to actively learning regression models. There is some prior theoretical work on active learning in a regression setting [31, 109]. However, there has been little study of pool based active regression approaches that are applicable in certain important practical scenarios. Specifically scenarios such as learning customer-product preferences or web page-ad click-through rates entail that the active learner evaluate the benefits of an available set of plausible prospective acquisitions. In contrast, existing approaches for active regression [31, 109] *derive* the instance features whose labels would be optimal to acquire. Moreover, existing approaches assume that the predictive model induced is a single (typically linear) model.

Recall that SCOAL provides us with a way to represent the input data by a collection of localized models. This chapter develops and empirically evaluates novel active learning policies that leverage the multiple local models learnt on the data. While the policies are developed using SCOAL as the local modeling technique, they are in general applicable to other prediction approaches that induce multiple local models on input data. The BlockRank acquisition strategy we propose aims to acquire examples in regions of the input space that are not well represented by the current set of local models. The benefit of acquiring a new labeled example

is hence computed based on the fit of the local model that represents that example. The intuition is that by selecting such examples, weaker models can be improved quickly, which is more beneficial than acquiring additional training examples from regions of the input space where the current local models perform well. We demonstrate how the prediction accuracy of SCOAL can be further improved by acquiring the labels of instances selected using BlockRank.

Next, we extend BlockRank to a hybrid approach, where the prediction variance for each individual prospective acquisition in a given local region is estimated to help differentiate between the benefits of different prospective acquisitions within that region. We show that this can improve the selection of informative instances. Another contribution of this work is a new acquisition framework that caters to the relationship between model complexity and the increasing availability of training data. Capturing more complex patterns in the data via an increasing number of local models is likely to yield higher prediction accuracy as more training data is acquired; in contrast, a single (hence less complex) model would tend to yield a higher accuracy than multiple models when the dataset is sparse (i.e., when many customers' ratings have not yet been acquired). We develop Hierarchical SCOAL, a hierarchical active learning approach that initially induces a single global model and *adaptively* increases the number of local models as more training data is acquired.

We empirically demonstrate the advantages of the approaches we propose on real-world datasets, such as the MovieLens recommender system data, and the ERIM customer purchase data [62, 95]. While this chapter primarily considers DyaC data, the extension of our proposed policies to tensor (multi-modal) data

with covariates is straightforward.

6.2 Related Work

Early work in the area of active learning for regression includes the optimal experimental design literature [60, 31] and label acquisition policies for improving regression models [21, 37, 59]. A key distinguishing element of these population based works is that instances to be obtained are derived to minimize a closed-form expression of the model's generalization error. Thus, work in optimal experimental design assumes that the input data distribution is known, that acquisitions can be derived rather than selected from a pool and that there exists a closed form term capturing the model's generalization error. For example, a well established result is a least squares based active learning approach which derives input points such that the variance term in the closed-form error formulation is minimized [31]. Similarly, Wiens and Sugiyama [109, 102] have developed active learning methods based on weighted least squares that are robust to the misspecification of models. Recently Sugiyama et al. [104] proposed an ensemble active learning technique that aims to simultaneously address the problems of model selection (selection of the number and type of basis functions of a linear regression model) and the selection of input points. In contrast to these policies, in this chapter we consider common business settings where useful acquisitions cannot be simply derived and acquired, but are rather selected from a pool of plausible acquisitions. Moreover, the methods discussed above require a closed-form representation of the generalization error. However, in most practical settings we consider, the input data distribution is not

known and it is often not feasible to derive a closed-form expression of the model's generalization error.

Although most work on active learning for regression is population based, as described above, there are some notable exceptions of pool based active learning techniques. The query by committee algorithm [96], well known for classification problems, is extended along several directions for application to the regression setting [89, 87]. Sugiyama et al. [103] have recently extended two population-based active learning criterion, ALICE [102] and Full-expectation Variance-only active learning for WLS (FV_W) [109], to pool based scenarios. Since ALICE is more accurate than FV_W, but less efficient due to lack of a closed form solution, the authors propose a heuristic to combine the two to develop a practical active learning procedure. However, none of these policies are tailored to local model induction techniques, which are particularly beneficial for modeling heterogeneous data, often encountered in large-scale business applications.

Most work on active learning addresses the acquisition of instances to improve classification models [75, 20, 35, 1, 91, 93]. In particular, many approaches consider a single probabilistic classification model and propose different measures to estimate the benefit of prospective acquisitions from a pool of unlabeled instances. Perhaps the most well known and generic policy is uncertainty sampling [69], originally proposed to minimize the amount of human labeling required to train a binary text classifier to a given accuracy. Like many other approaches, uncertainty sampling employs the posterior class probability estimation of the model induced from the current training data to evaluate the benefit from prospective ac-

quisitions. In particular, uncertainty sampling prefers instances for which the current model's posterior class probability estimation is close to 0.5. Roy and McCallum [91] proposed an approach that is particularly applicable to improving a Naive Bayes model, which estimates the entropy of the model's estimated posterior class distribution for each prospective acquisition. Perhaps the most effective of the classification active learning approaches is Query by Bagging, which was proposed by Abe and Mamitsuka [1] for binary classification models and was inspired by the theoretical Query by Committee algorithm [96]. The Query by Bagging policy prefers prospective acquisitions at which the class memberships predicted by an ensemble of models are the most evenly split. Note that all of these active learning policies aim specifically to improve classification models and do not apply directly to regression problems.

To the best of our knowledge, no prior active learning policy has taken advantage of multiple localized models, where each represents a different region of the input space. The most relevant piece of work is a technique for population based active learning in the regression problem domain [21], based on selecting input points that minimize the expected variance of the learner. The authors show that by using a mixture of Gaussians model the expected variance can be computed exactly in closed form, which gives rise to an efficient active learning algorithm. Hence, in this context, a collection of local models is useful from the point of view of efficiency of the active learning algorithm. Most existing methods however, assume that a global model (or an ensemble of models) is induced from the entire input space. Prior work on active learning also does not consider designing approaches

specifically to perform well on large-scale, heterogeneous DyaC datasets.

Recently, there have been a few papers that explore active learning techniques in a recommendation context [51, 92, 53]. For instance, Huang [51] proposes a simple, neighborhood based strategy to selectively acquire ratings from customers to improve product recommendations. He addresses the "new product problem", where the aim is to identify which customers to query to improve the accuracy of the predicted customer ratings for a newly introduced product. While his formulation is limited to a single new product, our work considers multiple products simultaneously and exploits the relation between them. Note also that in contrast to our work, all these proposed approaches use only the available customer-product ratings and no covariate information.

6.3 Problem Definition

Let $\mathcal{L}$ be the set of initially labeled data points and $\mathcal{U}$ the pool of unlabeled points. It is assumed that the target values for points in $\mathcal{U}$ can be acquired at a cost. In the DyaC data setting, set $\mathcal{L}$ comprises of the customer-product pairs for which the corresponding response values z_{uv} in matrix Z are known. Customer-product combinations with missing (unknown) entries in Z form the unlabeled set $\mathcal{U}$. The active learning objective is to acquire the response values for as few customer-product pairs in $\mathcal{U}$ as possible to achieve a desired accuracy.

6.4 Active Learning with a Global Model

Given an initial labeled training set $\mathcal{L}$ and an unlabeled set of prospective acquisitions $\mathcal{U}$, the baseline active learning algorithm for a single global model iteratively acquires examples for labeling as follows: In each iteration, every instance $z_{uv} \in \mathcal{U}$ is assigned a score given by the variance across the predictions for z_{uv} produced by multiple models, where each model is induced from a different bootstrap sample of the current training set $\mathcal{L}$. A subset of (b) instances with the highest scores is selected for labeling. The iterations continue until a suitable stopping criterion is met, for instance, a desired model accuracy has been reached. This active learning policy, which we refer to as Estimation Variance-based Active Learning (EVAL) is presented in Figure 6.1. This policy is essentially an extension of the query by committee algorithm [96] to a regression setting. The survey paper by Burbidg et al. [87] includes several variants of this idea. In particular, the EVAL algorithm is very similar to the one proposed by RayChaudhari et al. [89] to minimize data collection.

Note that the above method is generic and the model learnt on the training data can be any regression model, e.g., an MLP. However, due to the re-sampling step required for each acquisition, the approach is computationally intensive and hence not scalable to very large datasets and complex models. In Section 6.6, we will employ this approach to learn a global ridge regression model and compare its performance to our proposed local models based active learning policies.

Algorithm: EVAL
Input: $Z_{m\times n}$, $W_{m\times n}$, covariates, sets $\mathcal{L}, \mathcal{U}$

1. Train a global model M with parameters $\boldsymbol{\beta}$ on training set $\mathcal{L}$
2. Repeat
3. Create N bootstrap replicates of $\mathcal{L}$ by sampling with replacement
4. Train N models, one on each bootstrap sample
5. Compute the score for each point z_{uv} in $\mathcal{U}$ as the variance of the outputs of the N models at z_{uv}
6. Set $\mathcal{T} = b$ points from $\mathcal{U}$ with the highest scores
7. Obtain labels for points in $\mathcal{T}$ and add to $\mathcal{L}$
8. Update the global model M based on the updated $\mathcal{L}$
9. Until suitable stopping criterion

Figure 6.1: Pseudo-code for active learning with an ensemble of global models (EVAL).

6.5 Active Learning with Multiple Localized Models

In this section, we devise active learning approaches tailored to leverage the multiple local models induced by SCOAL on DyaC data.

6.5.1 BlockRank: Local Error Based Active Learning

As discussed in Section 5.3 our objective is to identify unlabeled instances for which the current model's prediction exhibits the largest error. A unique property of localized modeling approaches is that they identify homogeneous neighborhoods, each consisting of a set of instances with similar maps of the independent variables to the target value. SCOAL obtains this by simultaneously partitioning the data matrix Z into co-clusters, and inducing a local model from the data in

each co-cluster. In this section, we propose to estimate the model's error for an unlabeled instance by the model's *average* prediction error for "similar" instances, drawn from the same customer/product input space. We estimate the benefit of acquiring an unlabeled instance x by the fit of the local model representing x, estimated over labeled instances from the co-cluster that x belongs to. Specifically, we estimate the model's fit as the mean squared error within the corresponding co-cluster. For SCOAL with linear regression models, the MSE for co-cluster (g, h) is given by

$$\frac{1}{\sum_{u:\rho(u)=g}\sum_{v:\gamma(v)=h} w_{uv}} \sum_{u:\rho(u)=g}\sum_{v:\gamma(v)=h} w_{uv}(z_{uv} - \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}})^2. \qquad (6.1)$$

The fits of different local models are likely to differ across co-clusters, particularly for a heterogeneous dataset. Models induced from noisy and sparse regions of the input space are likely to exhibit poorer fit and the overall prediction model can be improved by acquiring instances from these regions. In contrast, it would be beneficial to avoid spending resources to acquire additional training instances from regions of the input space where the current local models perform well. We refer to this acquisition policy as BlockRank.

Figure 6.2 presents the BlockRank active learning algorithm, using the SCOAL technique to learn a collection of localized models. The BlockRank policy estimates the average prediction error in each local region and acquires instances from the region(s) with the highest average error. The utility of acquiring each unlabeled instance in $\mathcal{U}$ is estimated by the mean squared error of the co-cluster to which the unlabeled instance belongs. At each phase, a subset of b instances from $\mathcal{U}$ with the

highest scores are selected to be labeled. Note that all the unlabeled instances in a given co-cluster share the same score. Hence, when b is smaller than the number of unlabeled instances in a selected co-cluster, b unlabeled instances are drawn from the co-cluster uniformly at random [1]. At each acquisition phase, the acquired instances are added to the training set $\mathcal{L}$, and the iterative SCOAL algorithm is applied to the augmented training dataset, initialized with the previous co-clustering (ρ, γ). The co-cluster assignments and the co-cluster models are hence updated based on the augmented training data.

Algorithm: BlockRank
Input: $Z_{m\times n}$, $W_{m\times n}$, covariates, sets $\mathcal{L}, \mathcal{U}$

1. Run SCOAL on $\mathcal{L}$ to get co-clustering (ρ, γ) and co-cluster models $\{\boldsymbol{\beta}\}$)
2. Repeat
3. $\quad \forall z_{uv} \in \mathcal{U}$, assign score = fit of the corresponding local model (i.e., score = co-cluster MSE (Equation 6.1))
4. $\quad$ Set $\mathcal{T} = b$ points from $\mathcal{U}$ with the highest scores
5. $\quad$ Obtain labels for points in $\mathcal{T}$ and add to $\mathcal{L}$
6. $\quad$ Update local models using the newly labeled points (i.e., run SCOAL on $\mathcal{L}$ initialized with previous co-clustering to update (ρ, γ) and $\{\boldsymbol{\beta}\}$)
7. Until suitable stopping criterion

Figure 6.2: Pseudo-code for the BlockRank algorithm, using SCOAL to induce localized models.

As described in Section 3.5, SCOAL is not restricted to linear least squares models and can use any model or loss function that is better suited to the data. The

[1] In Section 6.5.2 we examine a policy for ranking the instances within a co-cluster so as to refine the selection of informative instances.

BlockRank active learning procedure is directly applicable to these settings. Hence BlockRank can also be applied to classification problems by learning local classification models using SCOAL. In the classification scenario, the fit of a co-cluster model can be evaluated by the classification accuracy or the loss function of the classifier, e.g., log-loss. Additionally, we would like to highlight that the BlockRank active learning policy can be applied to any modeling technique that induces multiple localized models on DyaC data. This is achieved by simply adapting the computation of the fit of a local model (Equation 6.1) based on the specific modeling technique. In Section 6.6, we explore BlockRank as an active learning policy for simultaneous *co-segmentation* and learning (described in Chapter 7), which is a localized model based predictive modeling technique for bi-modal data with one of the modes having an implicit ordering, e.g., time.

An important practical advantage of the structured manner in which the BlockRank strategy selects instances to be labeled is that it is easily actionable. Since the instances to be queried are restricted to certain co-clusters, their labels can be efficiently obtained [2]. Specifically, for predicting customer-product preferences, our policy selects to query a related subset of customers regarding their choices for a related subset of products (i.e., belonging to the same co-cluster). This enables a retail business, for instance, to focus on customers within a particular geographical region and a particular brand of products. On the other hand, a global model based active learning strategy selects individual customer-product combinations in

[2]Note that the assumption here is that it is cheaper to get a single customer to rate multiple products than having multiple customers rate different products.

an unstructured manner from the entire customer-product base. It might be more cost effective acquire feedback from a closely related set of customers on multiple products simultaneously rather than querying individual customer-product pairs.

Finally, because active learning policies are sequential, it is desirable to incrementally revise the models at each phase as new instances are acquired, rather than learn new models from scratch. For linear regression models, for instance, this can be achieved by updating the QR factorization of the coefficient matrix to reflect the addition and deletion of data points [39].

6.5.2 Hybrid Approach

The BlockRank policy estimates the benefit of acquiring the labels of instances from a given co-cluster, and thus does not differentiate among prospective acquisitions within the same co-cluster. It may be possible to yield a more refined strategy by further ranking prospective acquisitions within a co-cluster. Any active learning strategy can be applied to rank instances in each co-cluster, giving rise to a hybrid approach. For instance, the EVAL policy (Section 5.3) for actively learning a global model, can be used to compute the score associated with unlabeled instances in any given co-cluster. In the resulting hybrid approach, the scoring step (line 3) of the algorithm in Figure 6.2 is modified to obtain only an initial ranking of the points in $\mathcal{U}$ via Equation 6.1, which assigns all unlabeled instances in a given co-cluster the same score. The EVAL algorithm, described in Figure 6.1, is then applied to refine the ranking of prospective acquisitions within each co-cluster, based on their current estimation variance. b instances with the highest estimation variance from

the co-cluster(s) with the highest BlockRank score(s) are then selected for labeling. This hybrid policy is more likely to yield an outcome that differs from BlockRank when the number of instances to be acquired from a co-cluster is (substantially) smaller than the number of unlabeled instances in the co-cluster. When most or all prospective acquisitions from a given co-cluster are to be selected, ranking these instances will have little effect.

6.5.3 Hierarchical BlockRank

Since learning multiple local models entails the tuning of a larger set of parameters as compared to a single global model, a single model is likely to yield a lower generalization error initially, when the size of the training dataset is small. The increased complexity of multiple local models is likely to be increasingly more beneficial as more labeled instances are acquired. While active learning policies typically assume a fixed model complexity, we propose to dynamically adapt the complexity of the induced predictive model as more labeled instances are acquired. Specifically, the BlockRank algorithm assumes that the number of local models, and hence the number of co-clusters in the case of SCOAL, is determined *a priori* and remains the same through the active learning process. We propose to increase the number of local models induced if the increase in complexity improves generalization performance. Indeed, the choice of model complexity and the selection of instances to acquire next are not independent; representing the data by the most suitable model will improve the selection of training data and vice versa. Sugiyama et al. [104] made a similar observation and demonstrated that generalization perfor-

mance can be improved by simultaneously optimizing the model and the selection of input points. We now outline a hierarchical active learning approach that begins with a single global model and adaptively increases the number of models as more training data becomes available, hence representing the data at an increasingly fine level of resolution.

The Hierarchical BlockRank active learning policy we propose is based on an efficient, top-down, "bisecting", greedy model selection algorithm. The policy begins with a single co-cluster ($k = 1$ and $l = 1$), i.e., a single global model, and increases the number of co-clusters and corresponding local models as more training data becomes available, provided the increase in complexity also yields a lower validation set error. The Hierarchical BlockRank policy is derived by modifying the BlockRank active learning procedure, described in Figure 6.2, to perform a model selection step every T iterations. This step attempts to increase the number of co-cluster models if this yields a lower validation set error. The new set of local models is then used for active learning until the next invocation of the model selection procedure. The details of the model selection procedure, which increases the number of models by suitably splitting a row/column cluster, are given in Section 3.7. The pseudo-code is illustrated in Figure 3.3. Since Hierarchical BlockRank uses the previous co-clustering and set of models, which fit the data reasonable well, to initialize each subsequent run of SCOAL, it is likely to achieve a better local optimum than random initialization and hence find a better final solution.

6.6 Experimental Evaluation

The proposed active learning policies are evaluated on the following three problems:

1. Predicting the number of units of a product purchased by a household on the ERIM Marketing dataset described in Section 3.9.1. Such datasets are used, for example, by retailers to predict customer purchase trends. As discussed in [112], the prediction of consumer purchases can be significantly improved if information on customers-product purchases from other retailers can be acquired through third party information providers at a cost. In this setting, active learning selects customer-product pairs to obtain purchase data for, such that the most accurate predictive model can be learnt for a given acquisition cost.

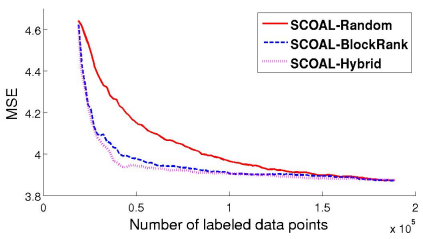
2. Predicting user-movie ratings on the MovieLens Dataset described in Section 3.9.2. While the availability of a large number of user-movie ratings is critical for making accurate predictions, many users are reluctant to provide such information without costly incentives. We will employ the proposed active learning policies to select the most informative user-movie pairs to query, so as to bring about maximum improvement in the predictive performance for a given acquisition cost.

3. Predicting household expenditure on the ERIM Customer Time dataset described in Section 7.7.1. In this setting, active learning aims at identifying informative household-weeks pairs to query, so as to improve the prediction of

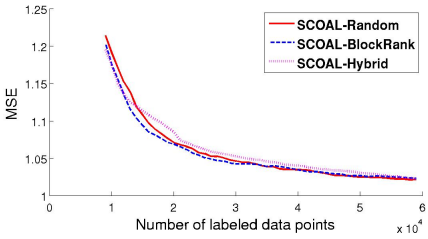
weekly household spending, for a given information acquisition cost. Since one of the modes of this dyadic dataset is time, a more suitable strategy than SCOAL to learn localized models is simultaneous co-segmentation and learning as seen in Chapter 7. The BlockRank approach discussed in Section 6.5.1 is readily extended to simultaneous co-segmentation and learning. The score for unlabeled instances at the co-cluster level is computed according to Equation 6.1 and the active learning procedure in Figure 6.2 is directly applicable to this setting.

We begin by examining the BlockRank and Hybrid approaches, which use the learnt set of localized models to evaluate the value of prospective acquisitions. SCOAL and the simultaneous co-segmentation and learning approach learn $k \times l$ independent linear models on the data and hence have a large number of parameters to be tuned. A sufficiently large number of training points are thus required to learn a stable initial set of local models, without running into empty co-cluster and coefficient matrix ill-conditioning issues. Later we examine the Hierarchical BlockRank approach, which begins with a single model and adaptively increases modeling complexity as more labeled data is available.

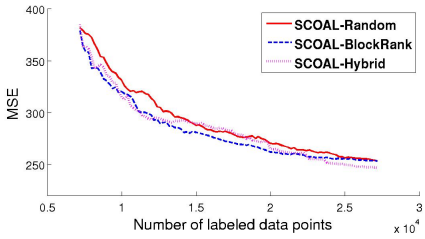
For the three datasets, Figure 6.3 compares the performance of BlockRank and Hybrid (using EVAL at the co-cluster level) with the Random baseline approach. Random uses a collection of localized models learnt using the SCOAL algorithm, but selects the data points to be labeled uniformly at random from the unlabeled set $\mathcal{U}$. EVAL with SCOAL as the "global" model is computationally very



(a) ERIM Marketing Data



(b) MovieLens



(c) ERIM Customer Time Data

Figure 6.3: Comparison of the BlockRank and Hybrid active learning approaches with randomly selecting points to be labeled.

expensive and did not scale to this experimental setup. On the three datasets, the local models used are ridge regression [45] models. All techniques begin with an initially labeled training dataset, after which labeled points are successively input to the techniques based on their respective selection strategies. The x-axis represents the number of labeled points available for induction. As detailed in Figure 6.2 the labeled points are acquired in batches of a fixed size b. We use a batch size b of 1700 for the ERIM Marketing data, 1000 for MovieLens and 200 for the ERIM

Customer Time dataset. The ridge regression parameter (λ), selected by cross validation, is set to 0.98 for the ERIM Marketing data, 0.78 for MovieLens and 1.64 for the ERIM Customer Time data. The number of row and column clusters are set to $k = 3, l = 2$ for ERIM Marketing, $k = 2, l = 2$ for MovieLens and $k = 5, l = 2$ for ERIM Customer Time. Mean squared error on a held out test set (10% of the data) is plotted on the y-axis. Each point on the plot is averaged over 10 random 90-10 % train-test splits of the data. The error bars are relatively small and are omitted for visibility.

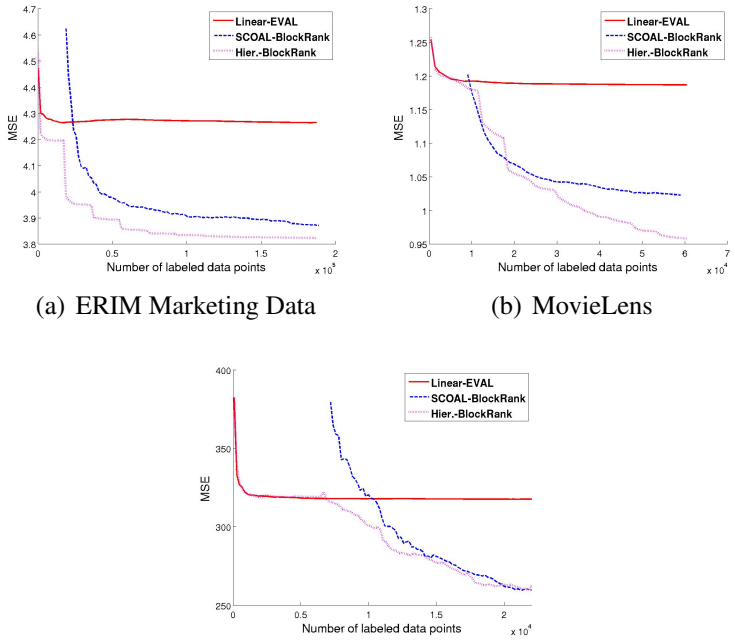
Table 6.1 displays the results of significance tests, similar to those conducted in [78], comparing the BlockRank, Hybrid and Random approaches on the three datasets. In comparing approaches A and B, the "significant" column indicates whether the average error across all the points on the learning curve of A is significantly less than that of B according to a paired t-test ($\alpha = 0.05$). The p-value for the test and the confidence interval on the difference of the population means ($A - B$) are also reported. One can observe that the BlockRank approach is significantly better than Random and none of the corresponding confidence intervals include 0.

Based on Figure 6.3 and Table 6.1, it is evident that on all three datasets BlockRank outperforms Random, highlighting the potential of selecting points to be labeled based on local model fit. It is useful to consider improvements in predictive performance within the context of the application so as to gauge its practical value. For example, recent work by Koren [65] shows that the achievable MSE values on the Netflix dataset lie in a narrow range and that even very small improvements (2-10%) in MSE translate into significant improvements in the quality

A vs. B	significant	p-value	confidence interval
ERIM Marketing Data			
BlockRank vs. Random	yes	10^{-27}	(-0.262, -0.201)
Hybrid vs. Random	yes	10^{-29}	(-0.274, -0.213)
BlockRank vs. Hybrid	yes	10^{-8}	(0.007, 0.015)
MovieLens Data			
BlockRank vs. Random	yes	0.0006	(-0.005, -0.001)
Hybrid vs. Random	yes	0.0009	(0.0016, 0.006)
BlockRank vs. Hybrid	yes	10^{-10}	(-0.009, -0.005)
ERIM Customer Time Data			
BlockRank vs. Random	yes	10^{-28}	(-10.004, -7.752)
Hybrid vs. Random	yes	10^{-17}	(-7.933, -5.311)
BlockRank vs. Hybrid	yes	0.0002	(-3.433,-1.078)

Table 6.1: Results of significance tests comparing the BlockRank, Random and Hybrid approaches on the ERIM and MovieLens datasets.

of the top K recommended movies. The improvement in MSE of BlockRank over Random is in this range for all three datasets. The consistent 5-7% improvement of BlockRank over Random on the ERIM Customer Time data (Figure 6.4(c)) and the similar improvements BlockRank obtains for the ERIM Marketing data are both substantial for inventory planning, and marketing campaigns which rely on consumer's projected buying behavior. Finally, recall that we also aimed to examine whether the Hybrid approach can improve upon the BlockRank policy performance by employing variance-based estimation to rank prospective acquisition within each co-cluster. As shown in Figure 6.3, the results suggest that the Hybrid policy does not offer consistent performance improvement. While it slightly improves upon BlockRank's performance on the ERIM Marketing dataset, it may also perform worse than uniform selection.

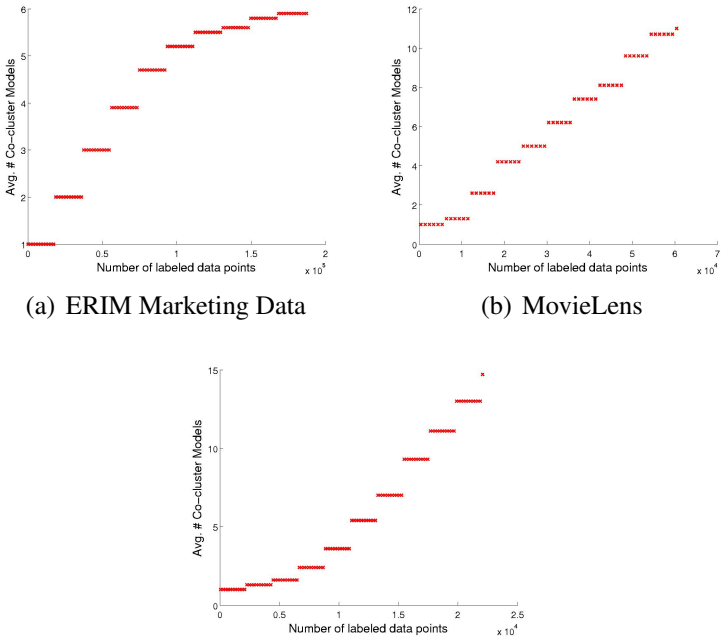


(a) ERIM Marketing Data (b) MovieLens

(c) ERIM Customer Time Data

Figure 6.4: Comparison of the Hierarchical BlockRank, EVAL and BlockRank active learning approaches.

Figure 6.4 compares Hierarchical BlockRank with the EVAL approach (Section 5.3). Recall, that in contrast to Hierarchical BlockRank, EVAL is based on a traditional active learning procedure which induces a single global model to represent the data and employs an ensemble of global models to estimate the variance of the model output for each prospective acquisition. For the sake of comparison, we also include BlockRank in the plots. Note that the BlockRank policy begins with a comparatively larger initial training set since it has a larger set of parameters to tune, i.e., $k \times l$ the number of parameters of a single global regression model, which EVAL and Hierarchical BlockRank use initially. Hierarchical BlockRank in-



(a) ERIM Marketing Data (b) MovieLens

(c) ERIM Customer Time Data

Figure 6.5: Average # of co-cluster models for Hierarchical BlockRank.

vokes the model selection procedure at 10 evenly placed points along the x-axis. In the empirical evaluation we present here, the model selection procedure is invoked after every 18700 label acquisitions for the ERIM Marketing dataset, and after every 6000 and 2200 acquisitions for the MovieLens and the ERIM Customer Time dataset, respectively. The changes in model complexity can be observed in the plot of the average number of co-cluster models for Hierarchical BlockRank versus the number of labeled data points in Figure 6.5. Note that the number of co-cluster models is computed as $k \times l$. Each point on the plot is the number of models averaged over 10 random train-test data splits. The ridge regression parameter (λ), for the linear models in the EVAL technique is selected by cross validation and is

0.84, 0 and 0.8 for the three datasets respectively. The parameters of the BlockRank approach are set in the same way described above. We also compared Hierarchical BlockRank with a hierarchical modeling approach that selects acquisitions uniformly at random and found Hierarchical BlockRank to be significantly better on the ERIM datasets, while there was no significant difference on the MovieLens dataset (plot not included in Figure 6.4 to avoid clutter).

One can observe that the local model based approaches perform significantly better than EVAL when sufficient labeled data is available, while EVAL with a single global model is better with a small training dataset. It is important to note that Hierarchical BlockRank effectively exploits the advantages of both. Since Hierarchical BlockRank starts with a single global model, its performance early on is close to that of EVAL. As more training data is available, the number of co-cluster models gradually increases as illustrated in Figure 6.5 and the predictive performance improves. One can actually observe on the ERIM Marketing dataset that the sharp falls in the MSE of Hierarchical BlockRank in Figure 6.4(a) are well coordinated with an increase in the number of models in Figure 6.5(a). These results also highlight the benefit of an adaptive number of local models versus the fixed number of local models in the BlockRank approach. In particular, on the ERIM Marketing dataset, the number of co-cluster models increases quickly at first and eventually saturates at around 6 models (Figure 6.5(a)). Although BlockRank also uses 6 co-cluster models on this dataset, it does not do as well, suggesting the benefit of the improved initialization technique used by Hierarchical BlockRank.

Figure 6.4 also provides interesting insights into the nature of the structure in the data learnt by a single global model versus the collection of local models constructed by SCOAL. One can notice that the global model captures the linear structure in the data very quickly and attains a reasonable accuracy on the test set with relatively few training examples. In contrast, the performance of BlockRank with a very small amount of training data is not as good, since the available data is not sufficient to fit the multiple localized models. However, as more training data is provided, the global model is unable to exploit the availability of data to improve its performance, while BlockRank continues to fit the data better by capturing more complex local structures. Hierarchical BlockRank gives us the advantages of both techniques by adapting the modeling resolution as the labeled data increases.

Comments on empirical results. The proposed local models based active learning approaches do dramatically better than active learning with a global model on the three datasets used for experimentation. However, since BlockRank obtains different improvements over Random across the three datasets, as observed in Figure 6.3, it is useful to gain some insights to help understand the cause for these differences. Contributing to this variation in performance is the variation of prediction error across co-clusters, which is most significant for the ERIM Marketing dataset. On this dataset, some co-clusters have a much better fit than the others, making the scores computed by the BlockRank policy for prospective acquisitions from different co-clusters very informative. On the other hand, for the MovieLens dataset, the prediction errors do not differ substantially across co-clusters. Thus, similar to uniform random acquisitions, the benefit from different prospective ac-

quisitions is estimated to be roughly the same. To demonstrate this difference in variation, one can sort the unlabeled points (prospective acquisitions) by their estimated scores, i.e., the fit of the corresponding co-cluster models, in ascending order. We use a lift measure to evaluate the obtained ranking. The lift is computed as the ratio of the prediction error on the entire set of unlabeled points to the prediction error of a certain fraction of the ranked unlabeled points. Figure 6.6 compares the lift at different fractions of the ranked unlabeled points for the MovieLens and ERIM datasets. It is clear that the lift curve is substantially steeper for ERIM than MovieLens, which implies that the estimated BlockRank score is a better indicator of the prediction error for unlabeled points in ERIM. Hence, on the ERIM Marketing dataset, by selecting the points with the highest BlockRank scores, we are able to identify and learn from the points with the largest prediction errors. This identification of the worst performing points is not as effective on MovieLens using our criterion of co-cluster model fit, which is why the improvement over a random selection of points is not as much.

6.7 Conclusions and Limitations

This chapter presented and empirically evaluated new active learning policies that exploit multiple local regression models learned by SCOAL on heterogeneous, DyaC data. Such data are widely used, such as for estimating consumers preferences, projecting demand, and predicting click-through rates for online advertisements. The empirical evaluation in Section 6.6 demonstrates that the novel idea of performing active learning with a collection of localized models is promising on

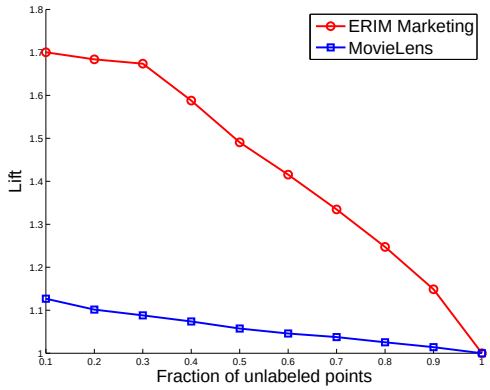


Figure 6.6: Lift at different fractions of the ranked unlabeled points on the ERIM Marketing and MovieLens datasets. The unlabeled pool consists of 20 % of the data matrix entries. Each point on the plot is averaged over 10 random splits of the data into the training set and the unlabeled set.

a variety of real life datasets. Moreover, an important practical advantage of our approach is that it is more actionable and cost effective than classical active learning techniques. Our proposed active learning policies are computationally efficient and hence scalable to large datasets. BlockRank uses only the learned prediction model for accessing the benefit of prospective acquisitions. No additional ensemble of models has to be created as in EVAL.

The BlockRank policy uses the error of a local model on training data points as an estimate of the model's generalization error. The training error is a reasonable proxy for the error on unseen data points only if a certain minimum number of training points have been observed. With a very small number of training points a local model will overfit the training data and have very low training error; the training error will be zero in the extreme case of fewer training points than the number of model parameters. The generalization performance will however be poor. A lim-

itation of BlockRank is that it requires the initial labeled set $\mathcal{L}$ to be large enough to avoid this situation. This implies that with SCOAL, there should be enough labeled data to start with so that none of the $k \times l$ local models overfit the training data and the training MSE is a reasonable proxy for fit of each local model. This was one of the motivations behind developing the hierarchical SCOAL approach discussed in Section 6.5.3, which works with a smaller sized initial labeled set.

The BlockRank approach currently makes a hard, greedy choice regarding which region of the input space to acquire points from. This might result in certain regions being relatively under explored. It will be interesting to explore a randomized version of the BlockRank policy, wherein the local model fits are used to obtain a probability distribution on prospective acquisitions from different regions. Each active learning iteration can select points for labeling by randomly sampling from this distribution. The hybrid approach employs estimation-variance-based active learning (EVAL) to rank prospective acquisitions within a SCOAL co-cluster. We find that EVAL does not yield consistent performance in this setting. One possible drawback of this approach is that it may be more likely to select outliers. The risk of acquiring outliers can be reduced by using at the co-cluster level, an active learning technique that is more robust to outliers [91].

Chapter 7

Simultaneous Co-segmentation and Predictive Modeling of Temporal Marketing Data

7.1 Introduction

Several marketing problems involve prediction of customer purchase behavior and forecasting future preferences. We consider predictive modeling of large scale, dyadic or multi-modal temporal marketing data, for instance, datasets consisting of customer spending behavior over time. Such datasets are characterized by variability in purchase patterns across different customer subgroups and shifting trends in behavior over time, which pose challenges to any predictive technique. Consider the concrete example of a dataset consisting of the number of dollars spent by a set of customers at a particular store per week, tracked over n consecutive weeks. The response variable is the dollar amount, which is not known for several customer-week combinations. Customers and weeks are described by covariates that represent the independent variables. The covariates include customer demographics, descriptive features for each week and features associated with a customer-week pair, e.g., whether the customer was exposed to advertising or could avail of special discounts during the week. In this scenario, we consider the following three problems: (i) predicting the missing dollar amounts for customer-week combinations, (ii) predicting the amount spent by all the customers in the $(n+1)^{\text{th}}$

week, and (iii) predicting the expenditure of a new customer over the n weeks.

The typical approach to solving the above problems is to construct a single prediction model that specifies a map between the covariates and the response variable. This model might do well for a small and homogenous group of customers over a few weeks. However, in real applications, such datasets are typically very large and heterogeneous. It is expected that for a diverse population of customers, the factors influencing purchase decisions and hence spending patterns are different in different customer subgroups. Moreover, customer purchase behavior is not static and changes with time.

To address these issues, practitioners often adopt a "divide and conquer" strategy that clusters customers into relatively homogeneous groups, and then model each customer group independently. The time axis can also be segmented into intervals and a different set of customer models can be learnt in each time segment. This two step sequential process, in which the partitioning is done *a priori*, independent of the modeling, is however not optimal. A more sophisticated approach is to simultaneously learn the segmentation and a collection of local models, one in each "homogeneous" segment. The SCOAL technique, described in Chapter 3 is not directly applicable to this problem. When applied to a dyadic dataset with a time mode, SCOAL clusters the time axis, hence permuting the time points and losing important sequential and temporal information. This might still be reasonable if the objective is just prediction, but typically, interpretability of the models is also important. For example, in a marketing study it might be interesting to understand how the influence of covariates on customer purchase changes with time. This

formed a major motivation for developing the approach presented in this chapter.

This chapter presents a comprehensive framework to solve prediction problems of the nature described above. As before, we represent the data as a matrix/tensor, whose entries are instances of the response variable. In case of bi-modal data, we use a matrix with rows representing the unordered mode, e.g., customers and columns representing the ordered mode, e.g., time. The three prediction problems we deal with now translate into: (i) predicting missing response variable values in the matrix, i.e., predicting a missing entry in an existing row and column of the matrix, (ii) predicting the response variable for the next consecutive point along the ordered mode, which implies predicting a new column of the matrix, and (iii) predicting the response variable values for a new entity along the unordered mode, which corresponds to predicting a new row of the matrix.

We first propose a simultaneous co-segmentation and learning approach that segments the "time" mode and clusters along the other mode to partition the response variables in the data matrix into a grid of blocks (co-clusters). Predictive models that relate the covariates to the corresponding response variables are concurrently learned for each (evolving) homogeneous partition. block. Similar to SCOAL, the partitioning and modeling are carried out simultaneously through an iterative algorithm that provably converges to the local optimum of a suitable cost function. We also present a model selection procedure to determine the appropriate number of time segments and clusters along the other mode.

As we will see in Section 7.4, to make predictions for a future time interval or a new entity along the ordered mode (e.g., a new customer), our approach

uses only a suitable subset of the co-cluster models. Simultaneous co-segmentation and learning helps in this case by identifying these co-cluster models and hence allowing us to consider only some data points, belonging to the same cluster, to do the forecast. We illustrate the effectiveness of our approach through detailed experimentation on the challenging ERIM marketing dataset.

Apart from making predictions based on temporal marketing data, our approach provides interpretable and actionable predictive models that can be used to gain valuable insights into patterns of customer behavior. Identified seasonal trends can be used to plan future advertising strategy and design promotional campaigns to maximize sales. We zoom into the details of the co-cluster models obtained on the ERIM dataset and make interesting observations in Section 7.7.1.

7.2 Related Work

A traditional strategy for analyzing temporal changes is to segment a time period into relatively homogeneous intervals and model each interval separately. This strategy has been adopted by Himberg et al. [49] to infer the context of a user from a context data sequence, which can then be used to achieve context awareness in mobile devices. Lingras et al. [70] focus on the analysis of changes in customer cluster characteristics of supermarket customers over a 24 week period. A modified SOM based clustering technique that uses properties of rough sets is applied to 6 month-long time segments to study the temporal migration of customers across clusters. Another interesting approach is presented in [42], where the authors propose a dynamic customer relationship management system to analyze the temporal

transition of customer segments. This evolution of customer segments is modeled by a Markov chain, which is used to evaluate the effectiveness of marketing strategies over time. However, as far as we know there has not been any work on developing a simultaneous segmentation and modeling framework for the prediction problems that we address in this chapter.

Time series forecasting has been a topic of extensive research in several fields. While most classical techniques such as ARIMA [13] focus on a single series of observations, our approach is designed for multiple series, where the similarity between series can be exploited for prediction. An example of modeling co-evolving time sequences is MUSCLES (multi-sequence least squares) [110], where a value in a time sequence is modeled as a linear combination of the values of the same as well as other time sequences within a certain window size. Our framework is distinguished from time series forecasting since we utilize neighborhood information through clustering and aim to solve a wider set of prediction problems.

"Concept drift" deals with learning in domains where the target concept depends on a hidden context that changes with time. Widmer et al. proposed an incremental algorithm FLORA [108] that maintains a hypothesis consistent with examples in a window that slides over the input stream. By forgetting old examples and including new ones, the hypothesis is updated to track concept drift. Stable concepts are stored to be reused in case of seasonal or recurring patterns. The temporal inductive transfer model, proposed by Forman [33] adapts to changing concepts but at the same time exploits previously learned concepts. Rather than using the data in a suitably sized window to train the current prediction model,

this approach proposes using the outputs of previously learned models as additional features in the current model. In contrast, our approach builds separate models in each time segment, while simultaneously learning the segment boundaries.

7.3 Problem Definition

For the sake of concreteness, in the following discussion, we refer to the entities along the unordered mode as "customers" and those along the ordered mode as "time intervals". An extension to multi-modal datasets is presented in Section 7.6. Let m be the total number of customers and n the total number of time intervals (e.g. weeks). The data is represented as an $m \times n$ matrix Z of customers versus time, with cells z_{ij} representing the corresponding real valued response variable, e.g., the shopping expense of customer i in week j, and $\mathbf{x_{ij}}$ representing the associated vector of covariates. The covariate vector is typically composed of the customer attributes, $\mathbf{c_i}$, the features of the time interval, $\mathbf{t_j}$ as well as annotations associated with the customer-time pair, $\mathbf{a_{ij}}$. The aim is to simultaneously cluster the rows (customers) and segment the columns (time intervals) into a grid of k row clusters and l column segments, such that the response variables within each co-cluster of customers and contiguous time intervals can be well represented by a single, common model.

Formally, let ρ be a mapping from the m rows to the k row clusters and γ be a mapping from the n columns to the l column segments. Note that while there are no restrictions on ρ, γ is constrained to ensure that the ordering of the time intervals

is retained. Specifically, γ is of the form

$$\begin{aligned}\gamma(i) &= 1 \quad 1 \leq i \leq b_1 \\ &= 2 \quad b_1 + 1 \leq i \leq b_2 \\ &= . \quad . \\ &= l \quad b_{l-1} + 1 \leq i \leq n,\end{aligned} \tag{7.1}$$

where $b_1 < b_2 < ... < b_{l-1}$ are the boundary elements of the l time segments. For linear regression models, we want to find a "co-segmentation" defined by (ρ, γ) and the associated $k \times l$ models that minimize the following objective function

$$\sum_{g=1}^{k} \sum_{h=1}^{l} \sum_{u:\rho(u)=g} \sum_{v:\gamma(v)=h} w_{uv}(z_{uv} - \hat{z}_{uv})^2, \tag{7.2}$$

where z_{uv} is the original value in row u, column v of the matrix, with associated weight w_{uv} and $\hat{z}_{uv} = \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}}$. $\boldsymbol{\beta^{gh}}$ denotes the coefficient vector of the model associated with the co-cluster that z_{uv} is assigned to.

7.4 Simultaneous Co-segmentation and Linear Regression

A co-segmentation (ρ, γ), that decreases the cost function (7.2) can be obtained by a simple iterative algorithm that alternately updates co-cluster models, row clusters and column segments. The algorithm begins by initializing (ρ, γ). The row cluster assignments are initialized either randomly or by a suitable clustering procedure and the column segment assignments are obtained by selecting $l-1$ time

intervals that partition the time axis into l equally sized segments[1].

Given the initial cluster and segment assignments (ρ, γ), the co-cluster models are constructed, i.e., the coefficient vector $\boldsymbol{\beta}$ is learnt for each co-cluster, as in the SCOAL algorithm described in Section 3.3. The cluster and segment reassignment steps are based on the key observation that the objective function is the squared error summed over all the elements of the matrix and can hence be expressed as a sum of row or column losses. As in the SCOAL algorithm, for a given column segmentation and model parameter sets, each row is assigned to the row cluster that minimizes the row error. Such row cluster updates hence decrease the objective function and improve the clustering solution.

Column Segmentation Update by Dynamic Programming. In contrast, the column segmentation update step cannot simply assign every column to the closest column segment as it needs to ensure that the time intervals (columns) are not permuted. The column segment assignment problem can be formulated as follows. For a given row clustering and a set of model parameters, there is a certain cost involved in assigning each column to a column segment. Note that the column cost is the regression error and is computed similar to row error. The column segmentation problem aims at finding the column assignment satisfying order constraints, which minimizes the total cost, i.e., the objective function (7.2). Dynamic programming is applicable to this problem since any subsequence of segments of an optimal assignment is also optimal for that subsequence.

[1] A divisive segmentation approach, which results in better initialization is proposed in Section 7.5.

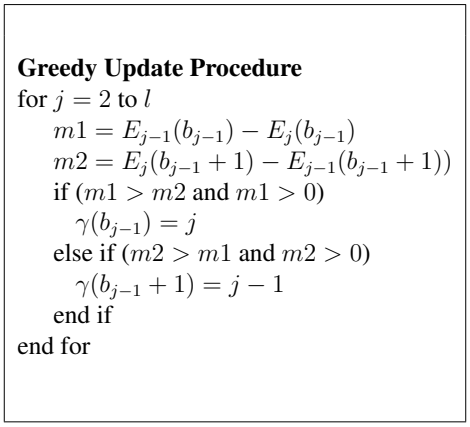


Figure 7.1: Pseudo-code for the greedy update procedure for column segmentation

With the dynamic programming approach, the time complexity of this step is $O(mnl + ln^2)$, where $O(mnl)$ is the complexity of computing the column errors. For large n this can be very expensive and hence slow. In such scenarios, randomized variants of the dynamic programming strategy such as Global Iterative Replacement can be applied. These variants are linear in the size of the input data and obtain close to optimal segmentation in considerably lesser time [49].

Column Segmentation Update by Greedy Local Search. We also propose a greedy local search strategy as a more efficient and simple alternative to dynamic programming based approaches. The greedy strategy incrementally adjusts the segment boundaries in such a way that the objective function is reduced.

An outline of the greedy update procedure is as follows. Let the j^{th} time segment be denoted by S_j, spanning time intervals $b_{j-1} + 1$ through b_j (from equation(7.1)). Let $E_{j-1}(b_{j-1})$ denote the column error of time interval b_{j-1} (the last time interval of segment S_{j-1}), and $E_j(b_{j-1})$ the column error of this time interval if assigned to time segment S_j. Time interval b_{j-1} is "closer" to segment S_j than

S_{j-1} if $E_j(b_{j-1}) < E_{j-1}(b_{j-1})$ and is a candidate for being moved to S_j. Similarly, $E_j(b_{j-1}+1)$ and $E_{j-1}(b_{j-1}+1))$, are the errors of the first time interval of time segment S_j and the error if it is assigned to time segment S_{j-1}. Time interval $b_{j-1}+1$ can be moved to segment S_{j-1} if it reduces the error. From among these two candidate moves, the one that brings about maximum reduction in error is performed, updating the boundary between segments S_{j-1} and S_j. This procedure is repeated for each time segment boundary, resulting in time complexity $O(lm)$ for the column segmentation update step. The pseudo-code for this procedure is illustrated in Figure 7.1.

The algorithm alternates between the row cluster and column segment reassignment and model update steps until convergence. Optionally, the cluster and segmentation update steps can be repeated several times and in arbitrary order until both row cluster and column segment memberships converge, since every step reduces the objective function. Close to convergence, it is more efficient to incrementally update the regression models by updating the QR factorization of the coefficient matrix [39] rather than retraining from scratch.

The overall algorithm is described in Figure 7.2. Step 1 reduces the objective function because of the optimality of the pseudo-inverse solution to linear regression, steps 2(a) and 2(b) are greedy updates and directly reduce the objective function. The objective function hence decreases in every iteration. Since this function is bounded from below by zero, the algorithm is guaranteed to converge to a local minimum.

Predicting response variables: After convergence, the final co-cluster as-

Algorithm ModelCSeg
Input: $Z_{m\times n}$, $W_{m\times n}$, covariates
Output: Co-segmentation (ρ, γ) and co-cluster models $\boldsymbol{\beta}$'s
1. Begin with random assignments (ρ, γ)
2. Repeat
Step 1: Update co-cluster models
3. for $g = 1$ to k do
4. for $h = 1$ to l do
5. Train a linear regression model with all the training
6. samples $(\mathbf{X_{uv}}, z_{uv})$ in co-cluster (g, h), with
7. associated weights w_{uv}, to obtain an updated $\boldsymbol{\beta^{gh}}$.
8. end for
9. end for
Step 2(a): Update ρ
10. Assign each row to the row cluster that
11. minimizes the row error
12. for $u = 1$ to m do
13. $\rho(u) = argmin_g \sum_{h=1}^{l} \sum_{v:\gamma(v)=h} w_{uv}(z_{uv} - \hat{z}_{uv})^2$
14. end for
Step 2(b): Update γ
15. Update column segments using the
16. dynamic programming or greedy procedure
Until Convergence
17. return (ρ, γ) and $\boldsymbol{\beta}$'s

Figure 7.2: Pseudo-code for simultaneous co-segmentation and regression

signments and models are used to solve the three prediction problems as follows.

- **Missing value prediction.** Let z_{uv} be a missing matrix entry that has been assigned to row cluster g and column cluster h. $\mathbf{x_{uv}}$ is the associated covariate vector and $\boldsymbol{\beta^{gh}}$ represents the model parameters of the linear regression model of the assigned co-cluster. z_{uv} is predicted as $\hat{z}_{uv} = \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}}$.

- **Prediction for future time interval.** Predicting the response variable for all the customers in the next (future) time interval implies predicting a new

column $v = n + 1$ to the extreme right of matrix Z. The new time interval is assumed to belong to the last time segment $\gamma(n + 1) = l$ and its entries are predicted using the models in this time segment, i.e., $\hat{z}_{uv} = \boldsymbol{\beta}^{\boldsymbol{gl}^T}\mathbf{x_{uv}}$, where $u = 1..m$, $g = \rho(u)$ and $v = n + 1$.

- **Prediction of new customer responses.** Predicting the response variable values for a new customer, with customer attributes c^{new}, involves predicting a new row of Z. First, the new customer is assigned to the row cluster of his nearest neighbor, computed based on similarity of the customer attributes. The models in the selected row cluster and column segments corresponding to the time intervals are then used to predict the new z values.

7.5 Model Selection

The simultaneous co-segmentation and learning (meta)-algorithm (ModelCSeg) described in Section 7.4 requires the number of row clusters k and the number of column segments l as an input and results in $k \times l$ co-cluster models. With a motivation similar to that of the SCOAL model selection algorithm, described in Section 3.7 we develop a similar model selection algorithm for ModelCSeg. As in the SCOAL case, the algorithm is a top-down "bisecting" greedy algorithm that begins with $k = 1$ and $l = 1$ and then iteratively tries to increase the number of row and column clusters as long as the validation set error reduces. The detailed steps are as follows. (ρ, γ) denotes the row cluster and column segment assignments and M the set of co-cluster models.

1. Run the simultaneous co-segmentation and learning algorithm (ModelCSeg) with $k = 1$ and $l = 1$ and initialize (ρ, γ) and M.

2. Try to split a row cluster. Select the row cluster with maximum average error on the validation set. Split the cluster into two by assigning half the rows with the largest row errors to a new row cluster and update ρ accordingly. Use this co-clustering (ρ, γ), with k increased by 1 to initialize a new run of ModelCSeg and compute the validation set error. If the error is lower than the error before splitting the row cluster, accept the split otherwise revert to the previous co-clustering and set of models.

3. Try to split a column cluster. Select the column segment with maximum average error on the validation set. Split the column segment into two contiguous segments, each one containing half of the time intervals and update γ. Use this co-clustering (ρ, γ), with l increased by 1 to initialize a new run of ModelCSeg. Follow a procedure similar to that in Step 2 to decide whether to accept the split.

4. Perform the best split. If both the row and column cluster splits decrease the error on the validation set, select the one that decreases the error the most and accordingly update the current co-clustering (ρ, γ) and models M.

5. If neither a row or column cluster split reduces the error, and k and l are the same as at the end of the previous iteration, terminate and use the current (ρ, γ) to initialize a final run of ModelCSeg. Otherwise loop back to Step 2.

7.6 Extensions

Extension to Other Regression and Classification Models. Similar to the SCOAL algorithm, the algorithm presented in Section 7.4 actually suggests a meta-algorithm that can be extended to other models such as ridge regression, regression with the L_1 norm or logistic regression by appropriately modifying the objective function.

Extension to Tensor Data.

The simultaneous co-segmentation and learning algorithm can also be extended to the general N mode tensor setting, with one of the modes (say mode N without loss of generality), having a temporal ordering. The aim is to assign entities along each of the first $N-1$ modes to clusters, and the entities along mode N to segments, such that all the response variables within each co-cluster (sub-tensor) can be represented by a common prediction model. The problem formulation and objective function for the N-mode case are a natural extension of the 2-mode case. Since the objective function is still additive over the individual tensor elements, the iterative algorithm in Figure 7.2 can be readily extended to achieve a locally optimal solution. The cluster update steps cycle through the modes, treating the unordered modes like rows and the ordered modes like columns in the bi-modal case. As in the case of bi-modal data, the algorithm is guaranteed to converge to a locally optimal solution. The model selection procedure described in Section 7.5 can also be easily extended to the tensor setting.

7.7 Experimental Evaluation

We evaluated our approach on two datasets that we created from the ERIM household panel data[2] described in detail in Section 3.9.1. One of the files in the ERIM database records the shopping visits of households in Sioux Falls, South Dakota at a number of local stores over 50 weeks. We used this information to create a bi-modal dataset (discussed in Section 7.7.1) that contains the total dollars spent by households in each week at the most frequently visited store across all households and weeks. The tensor extension of our approach is evaluated in Section 7.7.2 on a 3-mode dataset that includes the total dollars spent by households in each week at the five most frequently visited stores. [3]

7.7.1 Bi-modal ERIM Marketing Dataset

Algorithm	Train Err.	Train R-sq.	Test Err.	Test Err. Orig.	Test R-sq.
Global Model	0.470 (0.001)	0.528	0.479 (0.004)	325.466 (2.452)	0.528
Cluster Models	0.444 (0.001)	0.467	0.453 (0.003)	307.914 (2.193)	0.553
CC	0.692 (0.002)	-	0.756 (0.007)	513.417 (4.474)	-
SCOAL	0.264 (0.001)	0.699	0.345 (0.003)	234.497 (2.174)	0.66
ModelCSeg-Dyn	0.267 (0.001)	0.690	0.332 (0.003)	225.537 (2.012)	0.670
ModelCSeg-Greedy	0.264 (0.001)	0.688	0.338 (0.004)	229.422 (2.709)	0.663
ModelSel-Dyn	0.284 (0.003)	0.709	0.329 (0.005)	223.715 (3.388)	0.667
ModelSel-Greedy	0.287 (0.003)	0.696	**0.329 (0.005)**	**223.078 (3.539)**	**0.673**

Table 7.1: Comparison of prediction error on the bi-modal ERIM dataset with 80% training data and 20% test data, averaged over 10 random test-train splits.

The data preprocessing steps we took were guided by the data selection procedure used by Seetharam et al. [95]. We selected households that made purchases

[2]URL: http://research.chicagogsb.edu/marketing/databases/erim/

[3]Both datasets available at http://www.ece.utexas.edu/~deodhar/timeSeriesData

in at least 5 out of the 50 weeks and had weekly expenditures less than $300, resulting in a set of 1595 households. More concretely, the data is represented as a matrix of 1595 households by 50 weeks, with the total number of dollars spent by a household in a week as the response variable. Covariate information includes 5 household attributes - income, number of residents, male head employed, female head employed and total expense, 1 "week" attribute - the expenditure across all households and the number of store visits made by households in each week and the amount spent by households in the previous week. The covariates and the matrix entries are standardized to make them comparable across all household-week pairs. Modeling this dataset is challenging since the data matrix is sparse, with 58.63% of the values being 0 and the distribution of the total dollars spent across all household-week pairs is very skewed.

In order to evaluate the missing value prediction capability of our approach, the matrix entries are randomly split into a training set and a set of missing (test) entries. The quality of the predictions made for the missing entries is measured using mean squared error. As expected, the prediction performance of different algorithms varies with the amount of training versus test data. Table 7.1 zooms into the case where 80% of the data is used for training to predict the remaining 20%, while Figure 7.3 provides a snapshot of the performance for a wide range of training-test fractions. The baseline model we compare with is a single linear regression model for all household-week combinations, referred to as Global Model. We also compare with a two-step, sequential clustering and modeling approach (Cluster Models) that first clusters the households based on demographic informa-

tion and then fits a regression model in each household cluster. CC is the partitional Bregman co-clustering algorithm [5] with squared Euclidean distance. CC simultaneously clusters the households and weeks to find homogeneous co-clusters that minimize the distance of the entries within the co-cluster to the co-cluster mean[4]. We also compare with the SCOAL approach. Note that both CC and SCOAL cluster the time axis and could result in permuting the weeks. ModelCSeg-Dyn and ModelCSeg-Greedy are the simultaneous co-segmentation and learning techniques with the dynamic programming and greedy column segmentation approaches respectively. CC, SCOAL, ModelCSeg-Dyn and ModelCSeg-Greedy are run with $k = 10$ and $l = 5$ and Cluster Models is run with $k = 10$. ModelSel-Dyn and ModelSel-Greedy incorporate the model selection strategy described in Section 7.5. Test Err. and Test Err. Orig. are the test set errors on the standardized data and the original, unstandardized data obtained by back transforming the standardized data. The standard errors are included in parentheses. The train R-sq. is the R-square averaged across all the co-cluster linear regression models, which accesses the model fit on the training data, while the test R-sq. quantifies the fit of the learnt models to the test data. Note that a larger R-sq. implies a better fit.

The results in Table 7.1 show that ModelCSeg-Dyn and ModelCSeg-Greedy do a lot better than a single global model, Cluster Models and co-clustering, indicating that simultaneously learning multiple models along with partitioning the dataset substantially improves predictive performance. ModelCSeg also improves upon SCOAL, which confirms the intuition that making use of temporal information

[4]This corresponds to base 2 of the Bregman co-clustering algorithm [5].

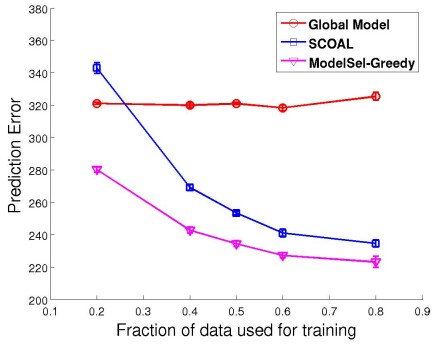


Figure 7.3: Comparison of Global Model, SCOAL and ModelSel-Greedy at varying training/test data fractions. Each point is averaged over 10 random test-train splits.

leads to a better local optimum. The model selection strategies ModelSel-Greedy and ModelSel-Dyn, further reduce the prediction error by tuning the number of co-clusters and avoiding overfitting. An overall trend we observed in all the comparative experiments is that there is either no significant difference in the performance of simultaneous co-segmentation and learning with the greedy or dynamic programming column segmentation update or the dynamic programming approach does only slightly better.

Figure 7.3 compares the prediction error of the Global Model, SCOAL and ModelSel-Greedy approaches for different train-test splits of the data ranging from 20% training data and 80% test data to 80% training and 20% test. It is clear that the single global model is too simple to be able to accurately represent the data and hence has very high error, irrespective of the fraction of training data available. In contrast, with more training data ModelSel-Greedy and SCOAL do better since the models learnt represent the data better. The difference in performance between SCOAL and ModelSel-Greedy is more dramatic at low fractions of training data, where SCOAL overfits the data, while ModelSel-Greedy exploits the time ordering

and also tunes the k and l values to achieve better results. The number of row and column clusters identified by ModelSel-Greedy range from around $k = 2, l = 2$ at 20% training data to $k = 5, l = 2$ at 80% training data. This indicates that there exists significant heterogeneity along both, the customer and time axes, which when explicitly modeled improves predictive performance.

Analysis of Co-cluster Models In addition to accurately predicting missing values, the simultaneous co-segmentation and learning approach provides interpretable models for each co-cluster, which give insights into the factors that influence purchase behavior across customers and time. Here we illustrate the clustering result of a run of ModelSel-Greedy with 60% of the data used for training, consisting of 4 customer clusters and 3 time segments, i.e., a total of 12 co-clusters. Table 7.2 displays the average dollars spent per customer per week, within each co-cluster.

	Time Seg. 1	Time Seg. 2	Time Seg. 3
Cust. Group 1	14.94	14.89	14.47
Cust. Group 2	9.34	9.53	9.78
Cust. Group 3	26.25	21.77	20.37
Cust. Group 4	22.21	20.20	20.25

Table 7.2: Mean dollars spent per customer per week in each co-cluster.

By comparing the models across the 12 co-clusters, we observed that in customer clusters 1, 3 and 4 the average # dollars spent reduces from time segment 1 to 3, with time segment 3 having the least # dollars spent. In these customer clusters, the number of store visits per week covariate also decreases across the time segments and is strongly predictive in the co-cluster models, with high, positive coefficients with very low p-values. We speculate that an external economic factor influences the number of store visits to produce this effect. Customer cluster 2

differs in behavior with the average dollars spent slightly increasing across time segments. This large cluster consists of 599 households with the least average income, number of residents and total grocery expenditure. It has the lowest dollar amount spent across all the time segments, with the total grocery expenditure being the most significant predictor in the corresponding co-cluster models. In contrast, customer cluster 3 is a small cluster consisting of the highest earning group of households, with the largest fraction of employed male heads and largest grocery expenditure. This cluster has the highest dollar amount spent across all time segments.

Table 7.3 displays the linear regression coefficients for a single global model and 2 interesting co-clusters (Cust. Group 3-Time Seg. 3, Cust. Group 2-Time Seg. 2) identified by ModelSel-Greedy. The values in parentheses are the p-values from the significance tests of the coefficients. In Cust. Group 3-Time Seg. 3 one can observe that the # store visits is the most dominant factor. Another strongly influential factor is the # dollars spent in the previous week, which has a negative coefficient. In Cust. Group 2-Time Seg. 2 the most influential factor is the total household grocery expenditure. One can see that these details regarding the heterogeneity across customers and time cannot be easily inferred from the single global model.

Table 7.4 evaluates the ability of the algorithms to predict the total number of dollars spent by each household in the 50^{th} time interval, assuming complete information for the previous 49 intervals. The test error is the squared error, averaged over all the households. The values displayed are averages over 10 trials and k and l are set to 20 and 3 respectively. One can observe that the simultaneous co-segmentation and learning techniques perform better than Global Model and

Attribute	Global	Cust. Group 3, Time Seg. 3	Cust. Group 2, Time Seg. 2
intercept	0.00 (0.4)	1.21 (0)	-0.12 (0)
income	0.02 (0)	0.03 (0)	0.01 (0.49)
# members	0.05 (0)	-0.03 (0)	0.02 (0.04)
M head emp.	0.01 (0)	0.01 (0.14)	0.01 (0.41)
F head emp.	0.02 (0)	-0.03 (0)	0.00 (0.87)
total expense	0.36 (0)	0.15 (0)	0.34 (0)
weekly expense	0.03 (0)	-0.01 (0.02)	0.02 (0.05)
# store visits	0.52 (0)	2.61 (0)	0.29 (0)
$ in prev. week	-0.06 (0)	-0.05 (0)	-0.06 (0)

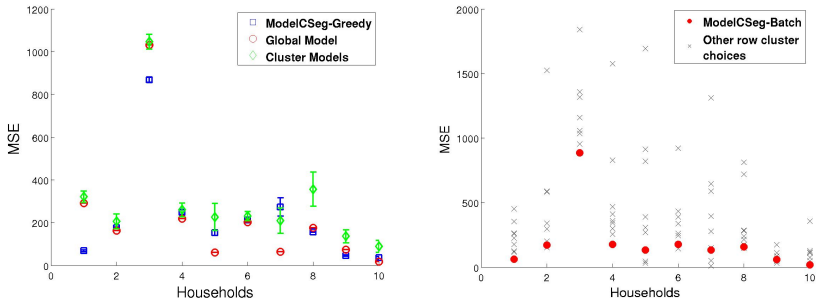
Table 7.3: Coefficients of the global model and of 2 co-cluster models.

Algorithm	Train Err.	Test Err.	Test Err. Orig.
Global Model	0.471 (0)	0.546 (0)	370.63 (0)
Cluster Models	0.434 (0.001)	0.499 (0.001)	338.67 (0.817)
ModelCSeg-Dyn	0.257 (0)	0.371 (0.004)	251.719 (2.426)
ModelCSeg-Greedy	0.258 (0)	0.374 (0.004)	254.142 (2.848)
ModelSel-Greedy	0.284 (0.002)	**0.367 (0.005)**	**249.198 (3.146)**

Table 7.4: Comparison of prediction error for a future time interval on the bi-modal ERIM dataset.

Cluster Models.

Finally, we address the problem of predicting the total dollars spent by a set of entirely new households over each of the 50 weeks. In order to evaluate our approach on this task, a set of randomly selected households was held out for prediction and the rest of the data was used for training. Figure 7.4(a) displays the prediction error of different approaches for each household from a held-out set of size 10, averaged over 50 weeks. ModelCSeg-Greedy is run with $k = 10$ and $l = 3$ and Cluster Models with $k = 10$. While the errors vary substantially across households, overall, one can observe that the performance of ModelCSeg-Greedy is comparable to that of Global Model and slightly better than Cluster Models. The



(a) (b)

Figure 7.4: (a)Comparison of the average prediction error of Global Model, ModelCSeg-Greedy and Cluster Models on 10 held-out households.(b)Average error for the household cluster selected by ModelCSeg-Greedy compared to all possible choices of clusters.

Global Model does better than ModelCSeg-Greedy for some households since it makes predictions based on all the data, which might be better than using only local information if the new household is not well represented by any single set of household cluster models or the inappropriate set of models is selected.

Recall from Section 7.4, that to make predictions for a new household, ModelCSeg-Greedy first assigns the household to the row cluster of a household from the training set it is most similar to. It would be interesting to evaluate how well this heuristic of selecting the most suitable row cluster from among the available row clusters does in terms of prediction accuracy. Figure 7.4(b) compares for each held-out household, the prediction error with the cluster selected by ModelCSeg-Greedy, indicated by a filled circle to the errors that would be achieved by selecting each of the other row clusters. The crosses specify the errors for the other 9 choices of row cluster assignment (since ModelCSeg-Greedy was run with $k = 10$). This figure highlights that for all households, the errors achieved by the

ModelCSeg-Greedy heuristic are very close to those obtained by the best choice of prediction models.

7.7.2 Tensor ERIM Marketing Dataset

Algorithm	Train Err.	Train R-sq.	Test Err.	Test Err. Orig.	Test R-sq.
Global Model	0.554 (0.003)	0.444	0.559 (0.004)	166.359 (1.271)	0.443
Cluster Models	0.519 (0.002)	0.453	0.525 (0.005)	156.061 (1.365)	0.478
SCOAL	0.386 (0.002)	0.582	0.478 (0.003)	142.084 (0.990)	0.524
ModelCSeg-Dyn	0.392 (0.002)	0.588	0.463 (0.003)	137.661 (0.782)	0.537
ModelCSeg-Greedy	0.391 (0.003)	0.59	0.465 (0.003)	138.402 (0.906)	0.537
ModelSel-Greedy	0.416 (0.004)	0.604	**0.447 (0.003)**	**132.942 (0.88)**	**0.552**

Table 7.5: Comparison of prediction error on the tensor ERIM dataset with 60% training data and 40% test data, averaged over 10 random test-train splits.

Algorithm	Train Err.	Test Err.	Test Err. Orig.
Global Model	0.541 (0)	1.299 (0)	386.255 (0)
Cluster Models	0.506 (0.001)	1.251 (0.001)	372.172 (0.185)
ModelCSeg-Dyn	0.39 (0)	**1.126 (0.005)**	**334.965 (1.36)**
ModelCSeg-Greedy	0.391 (0)	1.131 (0.004)	336.306 (1.244)
ModelSel-Greedy	0.402 (0.002)	1.14 (0.003)	338.957 (0.747)

Table 7.6: Comparison of prediction error for a future time interval on the tensor ERIM dataset.

We now evaluate our approach on the 3-mode tensor ERIM dataset, consisting of 1240 households, 50 weeks and 5 stores, with each tensor entry indicating the total number of dollars spent by a household in the corresponding store visit. The data was preprocessed similar to the bi-model ERIM dataset by selecting households that made purchases at each of the 5 stores in at least 5 out of the 50 weeks and had weekly expenditures less than $300. In addition to the covariates mentioned in Section 7.7.1, each store is described by its type i.e. grocery or drug, and

its total earnings over the 50 week period.

Table 7.5 displays the error of different algorithms on the missing value prediction task, with 60% of the data used for training and 40% missing. SCOAL, ModelCSeg-Dyn and ModelCSeg-Greedy are run with 10 household clusters and 5 time segments, while the stores are not clustered. Consistent with the results in Section 7.7.1, on this dataset as well, ModelSel-Greedy does significantly better than Global Model and Cluster Models and also outperforms SCOAL and ModelCSeg, which tend to overfit.

Table 7.6, deals with the task of predicting the number of dollars spent by each household at each store in the last time interval, given complete data for all the previous intervals. The results are averaged over 10 trials and k and l are set to 10 and 3 respectively. Even in this scenario, the simultaneous co-segmentation and learning approaches show significantly lower prediction error than Global Model and Cluster Models, with ModelCSeg-Dyn achieving the best performance.

Chapter 8

Decoupled SCOAL: An Approach for A Less Constrained Problem Decomposition

8.1 Motivation

The SCOAL approach partitions the data matrix into a grid of blocks (co-clusters) as described in Chapter 3. While this structured partitioning technique allows for efficient and scalable algorithms, it may be too constrained in some situations. SCOAL assumes that the heterogeneity across the space of the customer-product cross-product can be accounted for by models at roughly the same level of granularity. This might be restrictive in scenarios where the heterogeneity is non-uniform across this space. For instance, a dataset can have very high heterogeneity in some regions where a larger number of models are required, while other regions can be adequately represented at a coarser granularity. We address this by proposing Decoupled SCOAL (D-SCOAL), a greedy, bisection algorithm that allows for a more flexible partitioning of the customer-product space. D-SCOAL begins with a single co-cluster and recursively splits it if doing so improves prediction accuracy, resulting in a more general partitioning of the data matrix.

Figure 8.1 illustrates the nature of the partitioning obtained by D-SCOAL and contrasts it with rigid block structure imposed by SCOAL. D-SCOAL allows

certain regions to be finely decomposed into a large number of co-clusters, where several local models are learnt, whereas other regions can be represented by larger co-clusters and fewer models. Note that similar to SCOAL, in case of D-SCOAL as well, all the data points within each block are represented by a common block/co-cluster model.

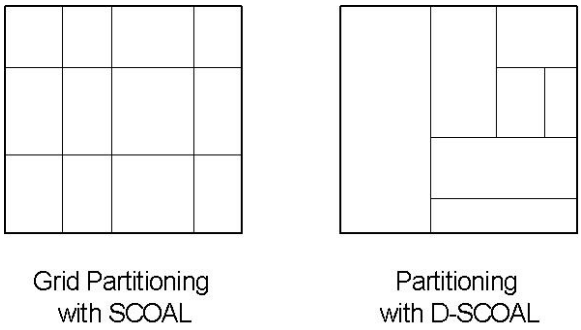


Figure 8.1: Partitioning structure of SCOAL versus D-SCOAL.

8.2 Related Work

The most closely related piece of work is Hartigan's paper [43] on direct clustering, one of the earliest works on co-clustering a data matrix. Hartigan's proposed approach to simultaneously cluster rows and columns imposes the requirement that the resulting co-clusters form a partition of the data matrix, and the row and column clusters respectively form tree structures, i.e., any two row (column) clusters are either disjoint or one contains the other. Direct clustering looks for homogeneous clusters, where all the values are similar (zero variance in the ideal case). The clustering algorithm starts with the entire matrix as a single cluster and then iteratively splits the existing clusters along the rows/columns. All possible

row/column splits for each cluster are evaluated and the one that brings about the maximum improvement in the homogeneity objective function is executed. Only those splits that retain the tree structure of row/column clusters are considered.

While direct clustering is completely unsupervised, the primary objective of D-SCOAL is predictive modeling and the partitioning is based on prediction error rather than cluster homogeneity. Like direct clustering, D-SCOAL requires the co-clusters to form a partition of the data matrix, however the row and columns clusters are not constrained to form tree structures. Thus, D-SCOAL allows for more flexibility in the co-clustering structure.

D-SCOAL does not select the best of all possible row/column splits as done in direct clustering since this would require re-computation of the co-cluster predictive models several times and be computationally very expensive. The D-SCOAL algorithm chooses between splitting along the rows/columns independently in each co-cluster as described in Section 8.3. Hence, different co-clusters can be processed in parallel, making our approach very efficient.

8.3 Decoupled SCOAL Algorithm

The D-SCOAL algorithm begins with a single co-cluster and a single global model and similar to the SCOAL model selection algorithm described in Section 3.7, uses a validation set to tune the number of co-cluster models. The first step is to split the single initial co-cluster along the rows or columns. The split that improves the validation set error the most is selected. The splitting procedure is then recursively applied to each of the two co-clusters obtained in the first splitting

step. The recursion base case is when neither a row/column split of a co-cluster results in improving the validation set performance. The current co-cluster model is returned at that point. The algorithm hence works in a top-down, greedy manner, increasing the number of co-clusters as long as it improves the "local" validation error. The pseudo-code for the procedure to split a given co-cluster is illustrated in Figure 8.2. ρ and γ denote row and column cluster assignments. D-SCOAL operates by invoking **split-co-cluster** with the input as the entire data matrix.

From the D-SCOAL algorithm one can observe that each split is local in nature and optimizes the validation set error in a localized region. Since the global cost is a convex combination of the local costs, this reduces the global cost as well. Note that the algorithm is greedy at each step; every split is final and cannot be undone. However, a big advantage of the recursive splitting procedure in D-SCOAL is that the overall problem is naturally decomposed into multiple, completely decoupled sub-problems. This allows for more parallelization than SCOAL. For instance, after a co-cluster is split, each of the sub-co-clusters can be processed completely independent of each other in parallel. This cannot be done in the case of SCOAL, where due to a global objective function such decoupling is not possible.

8.4 Experimental Results

Table 8.1 compares the performance of D-SCOAL with SCOAL and M-SCOAL on the ERIM and MovieLens datasets. The results are averaged over 10 90-10 % train-test data splits. The number of co-clusters for each modeling technique is also displayed; for M-SCOAL and D-SCOAL the range of the number of co-clusters

Algorithm	ERIM Dataset	ERIM Dataset (low valued entries)	MovieLens Dataset
SCOAL # co-clusters	15.807 (0.49) k=4,l=4	3.965 (0.044) k=4,l=4	0.943 (0.004) k=4,l=4
M-SCOAL # co-clusters	15.032 (0.416) k=1-2,l=4-6	3.832 (0.035) k=1-2,l=4-6	0.935 (0.005) k=3-5,l=3-6
D-SCOAL # co-clusters	15.123 (0.568) 17-32	3.785 (0.050) 19-34	0.96 (0.007) 18-42

Table 8.1: Mean squared error of D-SCOAL vs. SCOAL on the ERIM and MovieLens datasets, averaged over 10 90-10 % train-test data splits.

over the 10 runs is displayed. Note that while the number of row and column clusters is input to SCOAL, M-SCOAL and D-SCOAL automatically identify the most suitable number of co-clusters from the input data.

8.4.1 Discussion on the Results.

From Table 8.1 one can observe that D-SCOAL performs better than SCOAL on both the ERIM datasets. In the ERIM data, there is more heterogeneity along the columns (products) than the rows (customers). This is evident from the SCOAL model selection algorithm, which tends to split column clusters more often than row clusters. Another possible reason for this heterogeneity is that the products are known to be from 6 different categories. On the other hand, the customer base is more homogeneous, evident from the fact that the customer attribute coefficients in the SCOAL co-cluster models are low and do not vary as much across models as compared to the product attribute coefficients (Table 3.8). D-SCOAL hence tends to split the product mode into a large number of clusters (25), with very less splitting along the rows, which does better than the 4 row and 4 column cluster partition-

ing enforced by SCOAL. Additionally, D-SCOAL allows customer heterogeneity to be modeled at different granularities in different product clusters, which is not the case with SCOAL. Our intuition is that this kind of non-uniform heterogeneity structure exists in the ERIM dataset, since in the D-SCOAL runs we observed that row cluster splits occur only for only a few, select product clusters. The flexibility in decomposition provided by D-SCOAL is hence advantageous on the ERIM data, and the grid based SCOAL approach is restrictive. A possible reason for the D-SCOAL performing better than M-SCOAL (which also allows column clusters to be split more than row clusters) on the ERIM dataset with low valued entries is that each of the D-SCOAL problems being solved are simpler (always only 2 co-clusters) than running SCOAL on the entire data matrix. The chances of overfitting are hence lower, resulting in better generalization.

However, D-SCOAL does not perform as well as SCOAL on the MovieLens data. A possible explanation is that unlike ERIM, there is no difference in the heterogeneity across users and movies in the MovieLens dataset. SCOAL model selection returns almost equal number of user and movie clusters for each run. The D-SCOAL runs also show that neither row or column cluster splits are favored in particular, as in the ERIM datasets. The data heterogeneity seems to be more uniform in this dataset, rather than it varying in different user/movie groups. Imposing the SCOAL grid structure may actually be helpful in this case and optimizing a global cost function may be more beneficial that solving local problems.

Procedure: split-co-cluster(cc)
Input: Co-cluster cc **with predictive model** M
Output: Collection of models $\{M\}$ **within co-cluster** cc
1. Compute $V =$ error on validation points in cc.
2. **Try row cluster split**
3. Split the cluster into two by assigning half the rows with the largest row errors to a new row cluster.
4. Update ρ accordingly.
5. Use this co-clustering (ρ, γ), with $k = 2, l = 1$ to initialize a run of SCOAL.
6. Compute error V_r on validation points in cc.

7. **Try column cluster split**
8. Split the cluster into two by assigning half the columns with the largest column errors to a new column cluster.
9. Update γ accordingly.
10. Use this co-clustering (ρ, γ), with $k = 1, l = 2$ to initialize a run of SCOAL.
11. Compute error V_c on validation points in cc.

12. **if (**$V_r > V$**) and (**$V_c > V$**)**
13. Return current co-cluster model M.
14. **else**
15. if ($V_r < V_c$)
16. Apply row split to obtain co-clusters cc_{r1} and cc_{r2}.
17. $\{M_{r1}\}$ = split-co-cluster(cc_{r1})
18. $\{M_{r2}\}$ = split-co-cluster(cc_{r2})
19. $\{M\} = \{M_{r1}\} + \{M_{r2}\}$
20. else
21. Apply column split to obtain co-clusters cc_{c1} and cc_{c2}.
22. $\{M_{c1}\}$ = split-co-cluster(cc_{c1})
23. $\{M_{c2}\}$ = split-co-cluster(cc_{c2})
24. $\{M\} = \{M_{c1}\} + \{M_{c2}\}$

25. Return $\{M\}$

Figure 8.2: Pseudo code of the recursive procedure to split an input co-cluster.

Chapter 9

Parallel SCOAL: A Distributed Implementation Using Map-Reduce

9.1 Introduction

Several industry scale applications involve very large datasets. For instance, Google, Amazon and Yahoo! process several terabytes of data daily. With hardware being relatively inexpensive, the way to go is to parallelize computation across hundreds of CPU's to scale up to the data volume. Hence, a lot of attention has recently been devoted to approaches for distributed processing of large-scale data. For instance, a dataflow implementation of Bregman co-clustering presented at KDD09 [22] was shown to give speed ups of several orders of magnitude over serial implementations on the Netflix problem.

Map-Reduce [24], an effective paradigm for coordinating parallel computing has attracted a lot of attention from both academia and industry. It provides a clean and easy to use distributed programming abstraction. Several commonly used machine learning algorithms have been recently reimplemented under the Map-Reduce paradigm [19]. This also includes a Map-Reduce version of the EM algorithm for PLSI [23], which has been effectively applied to Google News personalization. Another example is that of DisCo: a distributed co-clustering technique

based on Map-Reduce [84].

SCOAL is most suitable for modeling large-scale datasets, where there is large heterogeneity to require multiple models and sufficient data is available to learn the model parameters. For instance, industry applications like online advertising and product recommendation where SCOAL is a good fit involve large volumes of data. There is hence a compelling need to develop a scalable implementation of SCOAL. An important motivation for developing a parallel version of SCOAL is that since it distributes computation across multiple machines it is not necessary to have all the data on a single machine, and therefore the algorithm can scale to much larger datasets than a serial SCOAL implementation.

There is massive scope for parallelization in the SCOAL algorithm that can be exploited by a distributed computing environment. The major bottleneck in each iteration of the SCOAL algorithm is the computation of distances of every row and column from every row and column cluster respectively, which is then used by the row and column cluster update steps (detailed steps in Section 3.2). However, one can observe that the distance computation can be done independently for every entry in the data matrix and then aggregated by either rows or columns to obtain row/column distances from the candidate clusters. Hence, this computation intensive component of the SCOAL iteration is inherently embarrassingly parallel. Similarly, the update of the prediction models within each co-cluster can also be parallelized across co-clusters. In this chapter, we describe our implementation of Parallel SCOAL, which parallelizes these main components of the SCOAL algorithm using the Map-Reduce framework. Parallel SCOAL is developed using

Hadoop (hadoop.apache.org), an open source Java (Apache) implementation of Map-Reduce.

While the Parallel SCOAL implementation exploits the obvious parallelism within each step of SCOAL and gives good speed-ups as illustrated in Section 9.5, the parallelism is still restricted by the fact that Parallel SCOAL is constrained to guarantee the same outcome as a serial implementation. Hence, the bottleneck created by the SCOAL iterations and the steps within each iteration being executed serially still remains. The main challenge in addressing this bottleneck is the global objective function that SCOAL guarantees to minimize to a local optimum. This global cost is responsible for all concurrent jobs to be synchronized at the end of each step, resulting in sequential execution of the individual SCOAL steps. An interesting question raised by this limitation is whether the minimization of a global cost function can be traded-off for an asynchronous, more parallel version of SCOAL. Exploration of a fully asynchronous parallel implementation or one with limited asynchrony and intelligent decomposition of the problem are areas of future work. The Decoupled SCOAL algorithm presented in Chapter 8 is one such promising direction. It would be worth implementing D-SCOAL on a suitable distributed computing platform to demonstrate its scalability to large-scale data. In general, it would be interesting to explore the range of trade-offs between parallelism and modeling accuracy in a systematic manner.

9.2 Background: Map-Reduce

Map-Reduce is described in the original paper by Dean and Ghemawat [24]. Here we present a quick overview and illustrate the programming model for easy exposition of the Parallel SCOAL algorithm developed in Section 9.3. Map-Reduce is a programming model and associated implementation for automatic parallelization. It enables computation to be scaled transparently to any number of machines, abstracting away the details of parallel execution from the programmer. The programmer is exposed to a simple, clean abstraction and does not have to be concerned with the details of fault tolerance, data distribution and load balancing.

The Map-Reduce paradigm primarily draws from functional programming. Any computation under this paradigm is decomposed into a Map and a Reduce operation, which are executed by the Mapper and Reducer functions respectively. The inputs and outputs of the two functions are key-value pairs, denoted as `<k,v>`. The functions are specified as follows:

```
Map: <k_in, v_in> -> <k_int, v_int>
Reduce:  <k_int, <v_int>> -> <k_out, v_out>
```

Records from the data source, e.g., from files or from a database, are input to the Map function as key-value pairs. The Map function produces an intermediate key and associated value (`<k_int, v_int>`) from the input. After the Map operation is done, all the intermediate values for a given output key are combined together into a list. The Reduce function takes as input the intermediate key and

the list of its corresponding values denoted as <v_int>. The intermediate values are then combined, finally resulting in an output key-value pair (<k_out, v_out>). Multiple Map/Reduce jobs can be executed in parallel. However, the bottleneck is the shuffle operation that groups together the intermediate values associated with the same key after the Map phase to prepare the input to the Reduce phase. Hence, the Reduce phase cannot start until the Map phase is completely finished.

9.2.1 Example

The following example pseudo-code of a Map-Reduce program [24] to obtain word frequencies for a collection of documents illustrates the use of the Map and Reduce functions. The Map function takes as input the document name as the key and a string with the document contents as the value. Each word and the value 1 are output as the intermediate key-value pairs. The Reduce function adds all the intermediate values associated with a key to emit the word along with its frequency. The advantage of Map-Reduce is that the same program can be effortlessly run on a large cluster of machines.

```
Map(String input_key, String input_value):
  // input_key: document name
  // input_value: document contents
  for each word w in input_value:
    EmitIntermediate(w, 1);

Reduce(String output_key, Iterator intermediate_values):
```

```
  // output_key: a word
  // output_values: a list of counts
  int result = 0;
  for each v in intermediate_values:
    result += v; //frequency
 Emit(output_key, result);
```

9.3 Parallel SCOAL with Map-Reduce

In this section we present our design for a parallel implementation of SCOAL using Map-Reduce. Recall that the SCOAL algorithm iterates over three steps until convergence (i) row cluster re-assignment (ii) column cluster re-assignment (iii) model building, i.e., learning prediction models in each co-cluster. The independence of the several different operations performed in each step makes massive parallelization possible. Most of the time in the execution of the serial SCOAL algorithm is spent in the computation of row and column distances from row/column clusters. However, one can observe that the distance can be computed independently for each cell of the data matrix. The row/column distance can then be computed by appropriately grouping together entries in the same row/column. This forms the basis of the parallel SCOAL algorithm.

Each entry in the data matrix along with its covariate vector and row/column cluster assignment is bundled into a tuple. The data can now be thought of as a collection of tuples that is operated on by the three steps of the SCOAL algorithm iteratively. A "group by" operation can be done to collect tuples for a

row/column/co-cluster as required by the three steps. The motivation behind this design is that decomposing the problem by tuples (individual data points) is more fine grained than decomposing it by rows or columns as in [84], resulting in better parallelism. The model coefficient vectors $\{\beta\}$ are maintained as global parameters. They are globally broadcast to the Map-Reduce jobs that use them, i.e., the ones for the row and column cluster re-assignment steps. Figure 9.1 gives an overview of the parallel SCOAL algorithm. Each iteration is composed of three Map-Reduce jobs chained together, one for each of the three steps described above. The details of each Map-Reduce job are as follows:

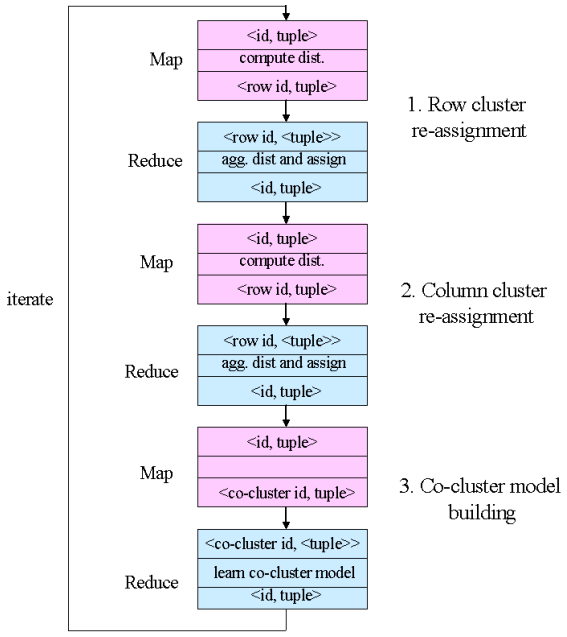


Figure 9.1: Overview of the Parallel SCOAL Map-Reduce implementation.

9.3.1 Row Cluster Re-assignment

Map function. The input to the Map function is a key (tuple id) and value (tuple with response z_{ij} and covariates $\mathbf{x_{ij}}$) pair. The Map function computes the distance of the entry z_{ij} from each of the k possible row clusters that row i could be assigned to. The row id and the tuple along with the vector of k distances are output as the intermediate key and value respectively.

Reduce function. The input to the Reduce function is a row id, which is the key and a list of values (tuples with distance vectors) for the entries in the corresponding row. The Reduce function simply sums up the distances of the individual row elements, to get the total distance of the row from k possible row clusters. The row is then assigned to the row cluster with the minimum distance. The input tuples are updated with the new row cluster assignment and pairs of tuple ids and tuples are output.

Figure 9.2 provides the pseudo-code for the Map and Reduce jobs, assuming linear regression co-cluster models.

9.3.2 Column Cluster Re-assignment

The Map and Reduce functions are symmetric to the Row Cluster Re-assignment step. The Map function outputs a column id and the tuple along with the vector of l distances from possible column clusters as the intermediate key and value respectively. The output of the Reduce function is the set of tuple ids and tuples associated with a column, updated with the new column cluster assignment.

```
Map(TupleId t_id, Tuple t)
    for each row cluster i = 1 : k
        dist_vect(i) = (t.z − β_i^T t.x)^2
    Emit(t.row_id, (t, dist_vect))

Reduce(RowId row_id, Iterator tuples_list)
    //initialize distance vector of size k to all zeros
    Initialize dist_vect(1 : k) = 0

    //aggregate distance vectors over all tuples in the row
    for each v in tuples_list
        dist_vect += v.dist_vect

    //assign row to closest cluster
    [min_dist, min_index] = min(dist_vect)

    new_cluster = min_index
    //update tuple row cluster assignments to new_cluster
    for each v in tuples_list
        v.tuple.row_clust = new_cluster
        Emit(v.tuple_id, v.tuple)
```

Figure 9.2: Pseudo-code for the Map-Reduce row cluster re-assignment step.

```
Map(TupleId t_id, Tuple t)
    Emit(t.cocluster_id, t)

Reduce(CoclusterId cocluster_id, Iterator tuples_list)
    //learn co-cluster model
    β = LinearRegression(tuples_list)
    obj_fn = 0
    for each t in tuples_list
        obj_fn += (t.z − β^T t.x)^2

    Emit(cocluster_id, (β,obj_fn))
```

Figure 9.3: Pseudo-code for the Map-Reduce co-cluster model building step.

9.3.3 Model Building

Map function. The input to the Map function is a tuple id, tuple pair. The function itself is an identity functions, which simply emits the co-cluster id as the intermediate key and the tuple as the intermediate value.

Reduce function. The input to the Reduce function is a co-cluster id and a list of all the tuples in the co-cluster. The main job of the Reduce function is to learn the co-cluster model from the tuples. In case of linear regression models, this involves solving a least squares problem and obtaining a new set of co-cluster specific betas. The new model coefficients and the objective function value for the co-cluster are globally broadcast.

Figure 9.3 provides the pseudo-code for the Map and Reduce jobs of the co-cluster model building step. A global synchronization occurs after the execution of the three steps, which (i) updates the global $\{\beta\}$ vectors and (ii) computes the overall objective function value by aggregating over the co-cluster objective function

values. If convergence has not been reached, the next iteration is initiated.

9.4 Parallel SCOAL Model Selection with Map-Reduce

In this section we discuss the steps involved in parallelizing the SCOAL model selection algorithm described in Section 3.7. Each step in the algorithm is decomposed into Map and Reduce operations, and the overall algorithm consists of these multiple Map-Reduce jobs chained together. The detailed steps are as follows:

1. Generate validation set.

   This steps selects a certain fraction (f) of the training data as the validation set on which the number of clusters is tuned.

   Map. For each input tuple, a random number between 0 and 1 is selected. If the number is $< f$ the tuple is marked for validation.

   Reduce. Reduce is the identity function.

2. Run SCOAL with $k = 1, l = 1$ using the parallel SCOAL algorithm described in Section 9.3.

3. Compute validation set error.

   Map. For each input tuple, if it is part of the validation set, its error with the current co-cluster assignment and current models is computed. The co-cluster id and the tuple along with the error are output as the intermediate key-value pair.

Reduce. The errors of the tuples within the co-cluster, associated with the input co-cluster id are summed to obtain the co-cluster validation error. The co-cluster id and co-cluster error are output.

After the Reduce phase has completed, all the co-cluster errors are added to get the total validation error.

4. Try to split a row cluster.

   (a) Compute average row cluster errors.

   Map. Given the tuple id and tuple as input, if the tuple is part of the validation set, the Map function outputs the row cluster id and the tuple, which includes the corresponding validation error for that entry.

   Reduce. For the input row cluster id, the Reduce step computes the average error of all the validation tuples in the row cluster, i.e., the input list of values. The row cluster id and the average cluster validation error are output.

   After the Reduce phase completes, the row cluster with the largest average validation error is selected to be split.

   (b) Compute average errors for the rows belonging to the row cluster to be split.

   Map. Given the tuple id and tuple as input, if the tuple is part of the validation set and part of the row cluster to be split, the Map function outputs the row id and the tuple along with the corresponding validation error.

Reduce. For the input row id, the Reduce step computes the average error of all the validation tuples in the row (input list of values), and outputs the row id and the average row error.

After the completion of all the Reduce jobs, the median of the average row validation errors is computed as the splitting threshold t.

(c) Split selected row cluster.

Map. If the input tuple belongs to the row cluster to be split and the average validation error of the row it belongs to is greater than the threshold t, the row cluster assignment for the tuple is updated to reflect its assignment to a new row cluster.

Reduce. Reduce is the identity function.

(d) Run Parallel SCOAL initialized with the current cluster assignments and k incremented by 1.

(e) Compute the validation set error with a Map-Reduce job similar to that in Step 3.

5. Try to split a column cluster. The steps here are symmetric to those for splitting a row cluster.

6. Apply the best split and go back to step 3. If there is no reduction in validation error, stop and return the current model.

9.5 Experiments

9.5.1 Setup

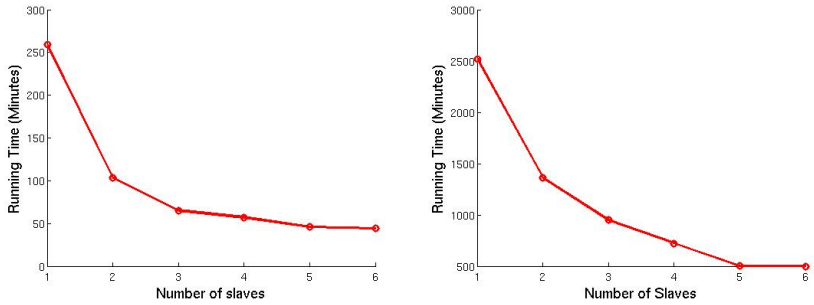
After testing the parallel SCOAL implementation in pseudo-distributed mode on a single machine, it was tested in a true distributed mode on the Amazon Elastic Compute Cloud (EC2) http://aws.amazon.com/ec2/. EC2 provides an easy to use interface to do so along with Hadoop-EC2 configuration and monitoring tools. Several pre-made and packaged Hadoop-EC2 images are also publicly available. A suitable image can be selected via the Hadoop-EC2 tools and loaded on the EC2 machines. Our experiments were carried out on the default small instance EC2 machines with the following specifications: 1.7 GB memory, 1 virtual core, 160 GB instance storage, 32-bit platform and moderate I/O performance.

9.5.2 Datasets

The datasets used for experimentation were the MovieLens dataset described in Section 3.9.2 consisting of 100K user-movie ratings and the larger MovieLens1M dataset with 1 million ratings for 3900 movies by 6040 users (http://www.grouplens.org/node/73). Both these datasets are actually small by Hadoop standards, resulting in a large filesystem overhead compared to actual computation. However, they are sufficient for a preliminary test of the scalability of the parallel SCOAL implementation. The correctness of the parallel implementation was checked by comparing the results of 10 fold cross-validation with SCOAL and SCOAL model selection on the MovieLens100K dataset with those obtained from the Matlab implementation.

9.5.3 Scalability Tests

Figure 9.4 shows the runtime of Parallel SCOAL on clusters with an increasing number of slave machines [1] on the two MovieLens datasets. The number of Map and Reduce jobs is set to 10. On both datasets, one can observe that increasing the number of slave machines gives good speed-up, e.g., with two slaves rather than a single one, the running time is cut by half. The runtimes flatten out at around 5 machines, possibly due to an increasing overhead cost of the Hadoop framework, which results in diminishing marginal returns from a larger number of slaves.



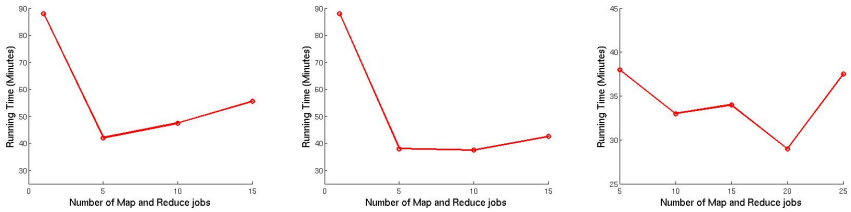
(a) MovieLens100K (b) MovieLens1M

Figure 9.4: Running time in minutes vs. number of slaves in the cluster.

9.5.4 Performance Tuning

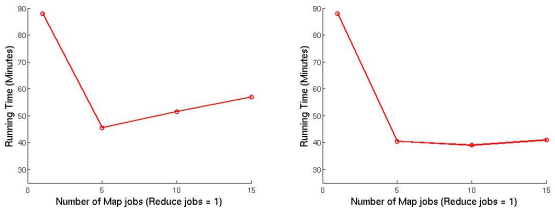
While the results presented in Figure 9.4 serve as a proof of concept of the scalability of Parallel SCOAL, there is scope for further speed-up by tuning several parameters of the Hadoop framework. This section studies the impact of the number of Map and Reduce jobs on performance along with its interaction with the number

[1] A Hadoop cluster is comprised of a single master and multiple slave machines.



(a) 5 slaves (b) 10 slaves (c) 20 slaves

Figure 9.5: Running time in minutes vs. number of Map and Reduce jobs.



(a) 5 slaves (b) 10 slaves

Figure 9.6: Running time in minutes vs. number of Map jobs, with one Reduce job.

of machines.

Figure 9.5 displays the results of varying the number of Map and Reduce jobs on the MovieLens100K dataset on EC2 clusters with 5, 10 and 20 slaves respectively. Under the Hadoop framework, the user can only specify the minimum number of Maps jobs spawned (thus providing a hint to the framework regarding the number of Map jobs), rather than the actual total number. For the results in Figure 9.5, the number of Map and Reduce jobs executed are equal and vary from 1 to 25, depending on the number of slaves in the cluster.

In contrast, the number of Reduce jobs to be spawned can be specified exactly. Figure 9.6 displays the results of varying the number of Map jobs, while keeping the number of Reduce jobs fixed at 1, on the MovieLens100K dataset on

EC2 clusters with 5 and 10 slaves. The actual number of Map jobs spawned are displayed in the figure.

From Figure 9.5, one can observe that the run time reaches a minimum when the number of Map and Reduce jobs spawned is very close to the number of slave machines. A similar trend is observed in Figure 9.6. Therefore, a rough guideline is to set the number of Map and Reduce jobs to the number of machines in the cluster, or slightly higher, so as to fully utilize the computing power of all the machines. Moreover, from the run times in the two sets of figures (Figure 9.5 and 9.6), one can observe that the parallelization of the Map operations is more critical than that of the Reduce operations. For a fixed number of Map jobs, having more then one Reduce job improves the run time by a relatively small amount. This is due to the fact that the Map jobs do the most expensive operations of computing row/column distances from their respective cluster candidates, which comprises the major computational bottleneck. Because of these expensive operations, tuning the number of Map jobs is more critical to the overall run time.

In addition to the number of Map and Reduce jobs, there are several other options that can be tuned. For instance, another option that impacts performance is the file replication factor. Currently we use the default replication factor of 3. Since our files are small, a larger replication factor could be beneficial, since a larger number of Map jobs can then work on local copies of the data.

9.6 Limitations and Future Work

1. As described in Section 9.1, parallelism is currently exploited only within each step of each SCOAL iteration. Due to dependencies across the three steps, they have to be executed sequentially. Moreover, due to a global objective function, which has to be computed at the end of each iteration, all concurrent jobs have to be synchronized at the end of the model update step. This implies that the iterations have to be executed serially to achieve the same results as the sequential SCOAL algorithm. An interesting research direction is to modify the SCOAL algorithm so that synchronization at each iteration is not required. Exploration of a fully asynchronous parallel implementation or one with limited asynchrony and intelligent decomposition of the problem are areas of future work.

2. The parallelism in the model update step is dependent on the number of co-clusters since each Reduce job operates on all the entries within a co-cluster. In the extreme case of only one co-cluster, all the data points are processed by a single Reduce job, on a single machine, resulting in a memory bottleneck. While more parallelism can be obtained with a larger number of co-clusters, most real world datasets are typically well modeled by co-clusters in the order of tens. Increasing the number of co-clusters will cause the co-cluster models to overfit due to lack of adequate data to tune the model parameters.

3. Although the relative speed-up achieved by increasing the number of machines in the cluster is good (Figure 9.4), the absolute runtimes are still high.

This is due to the fact that Hadoop itself is known to be very slow and the overhead introduced by HDFS is large. It would be worthwhile exploring other, faster implementations of Map-Reduce such as Phoenix, a shared memory implementation http://mapreduce.stanford.edu/.

Chapter 10

Conclusions and Future Directions

Simultaneous co-clustering and modeling is a promising framework that generalizes co-clustering, collaborative filtering and traditional segment-wise modeling. The SCOAL approach that we have presented forms a powerful and versatile framework, useful for solving a broad range of classification/regression problems. SCOAL shows promising results and out-performs alternative techniques on prediction problems involving heterogeneous, DyaC data. This approach is useful not only in terms of prediction accuracy, as results on challenging real life datasets illustrate, but also gives interpretable and actionable models. The SCOAL strategy has been further extended to address the challenges associated with large, noisy datasets by automatically detecting and eliminating non-informative data points and outliers, and modeling only the most relevant regions of the data. In addition, SCOAL has been utilized to develop novel strategies for ranking, active learning and modeling temporal data. Finally, all of the above proposed techniques are made applicable to large-scale industry level problems by developing a scalable, parallel version of SCOAL based on Map-Reduce.

There are several aspects of the SCOAL framework that can be explored further. The following are interesting directions for future research:

1. Robust error functions.

   Most of our experiments use SCOAL with linear regression models. However, as seen in Section 3.9 linear least-squares models are susceptible to outliers and a few outliers could skew the model results. It will be interesting to investigate the application of SCOAL with non-linear models or robust error functions, which will help overcome the limitations of linear models in the presence of outliers.

2. Bayesian SCOAL.

   The co-clustering produced by the SCOAL approach is inherently partitional in nature. Each row and column belongs exactly to one row and column cluster respectively. This might be restrictive in certain scenarios where an entity is naturally associated with more than one cluster. For instance a customer might display purchase patterns characteristic of multiple customer clusters, with varying degrees. In this case, it may be more appropriate to allow the customer to be a member of multiple clusters rather than forcing it into a single cluster. A similar problem in the context of Bregman co-clustering was recently addressed by Shan and Banerjee [98]. The Bayesian Co-clustering model that they propose can be extended to SCOAL as well, with Dirichlet distributions as priors over row and column cluster memberships and a suitable generative model for each co-cluster. A variational inference technique can be used for parameter estimation.

   Another advantage of this Bayesian formulation is that the reliability of a lo-

cal model's prediction can be determined based on the corresponding predictive distribution. This will provide a statistically sound method for estimating the variance associated with a prediction. It will be interesting to compare a ranking of model predictions based on their reliability using this technique with our proposed heuristic approach, discussed in Section 5.

3. Sparse co-cluster models and feature selection.

   As discussed in Chapter 3, SCOAL has $k \times l$ models, each with a number of parameters equal to the number of covariates. SCOAL is hence susceptible to overfitting when not enough data is available to tune the model parameters. In Section 3.6 we addressed this issue by suitable model selection, and regularization/smoothing. The overfitting problem is particularly important in feature-rich domains, where the number of covariates is very large. Feature selection is another strategy for preventing overfitting in this scenario, which when carried out at the level of each co-cluster will result in sparse co-cluster models. Modeling co-clusters with lasso or Bridge regression models [36], which perform an implicit feature selection and produce parsimonious models is a possible solution. Feature selection is also advantageous from an interpretation point of view since it will help identify different sets of covariates being predictive in different "customer-product" subsets. It will be useful to apply this technique to a suitable feature-rich dataset and analyze its overall performance and resulting co-cluster models.

   Another interesting question to look at is the interaction between model selection and feature selection, i.e., how many models local should be learnt

and how complex should each one be? While these two problems are typically addressed independently, in the context of the SCOAL framework it may be advantageous to optimize the number of models and model complexity jointly.

4. SCOAL with additional structural information.

   The SCOAL framework assumes that the entities along the two modes of a dyadic dataset are independent in a structural sense, i.e., they can be permuted and grouped together without any constraints into a set of row/column clusters such that the total prediction error is minimized. In Chapter 7 we proposed a modification to SCOAL to model dyadic data where one of the modes is time and is hence constrained to sequential time-wise ordering. In general, the entities along a mode could have other structural constraints, e.g., a fixed ordering imposed by the domain or they could be related via a taxonomy. In certain domains, additional information relating the entities to one another could also be available, e.g., a graph structure on the entities. We would like to generalize SCOAL to exploit any such structural information available about the entities to learn prediction models that are more accurate and also interpretable. The following are motivating examples of applications that could potentially benefit from this approach:

   Spatial constraints on entities. Certain marketing problems could involve spatially ordered modes such as geographical location. It will be interesting to explore applications of the simultaneous co-segmentation and modeling

approach to this setting to model location specific trends. As described in Section 1.2.1, dyadic data with covariates arises in ecology applications with species and sites/locations forming the two modes respectively. The sites are inherently related via a 2-D spatial structure. This information could be useful for regularizing local co-cluster models. For example, intuitively, one expects the models for sites that are spatially close to be similar. Their parameters can be constrained to be similar by imposing a penalty on the distance between their parameter vectors.

Taxonomy on products. Another problem described in Section 1.2.1 is CTR prediction for online advertising, where SCOAL can be used to model the dyadic dataset of users and ad categories. The ad categories are typically organized as a hierarchical taxonomy. In general, a taxonomy may be available on the "'products" arising in other applications as well. For instance, documents, web-pages or movies may be related via an ontology or products via a catalog. It is often the case that some leaves of the taxonomy do not have enough data to learn accurate prediction models. For example, certain ad categories or sets of web pages receive very low traffic or certain rare movies receive very few user ratings. In the CTR prediction problem, the taxonomy on the categories imposes a certain structure on the input space that can be exploited to decompose it more effectively. A problem similar to this has been addressed by Pandey *et al.* [83] by formulating it as a multi-arm bandit problem with dependencies between arms imposed by the structure of the taxonomy. It will be interesting to study ways to partition the problem

space based on the taxonomy and understand its impact on the performance of the local models within the SCOAL formulation. A preliminary approach is to partition the set of categories hierarchically in a top-down manner, beginning with a single partition including all the categories and then driving down the hierarchy until finer splits no longer improve overall model performance. Since some categories may not individually have sufficient data to train local models, the hierarchical approach will allow systematic pooling of data from related categories to learn better models. Another idea is to perform smoothing of local model parameters based on the taxonomy structure. The intuition is that the models for categories that are close in the taxonomy will be similar, and hence forcing their parameters to be similar will be an effective means of regularization. Given that the data in this domain is very sparse, this will avoid overfitting and result in a more generalizable set of models.

Social network information for customers. In recent times, the increasing popularity of online social networks, e.g., Facebook, Twitter, Buzz, has made available a rich collection of data about customers besides their purchasing preferences. A social network structure including the customers could be available in the form of a customers by customers graph. This provides information about links between customers, which could potentially have an impact on their purchasing tendencies. For example, a customer may be very likely to purchase a product that his trusted friends have purchased. Recently, social network data has been used by Ma *et al.* [74] as an additional data

source in the problem of predicting user-item ratings. They incorporate user trust information into the latent factor model proposed by Salakhutdinov and Mnih [94] and illustrate that it improves the accuracy of predicting ratings on the Epinions dataset. How to best utilize the social network or any graph structure in general for predictive modeling with SCOAL is a challenging problem. Going a step further, the links in the social network could also be annotated. For instance, the network could be a email/IM graph where the links are described by the topic of the conversation. It is possible that this information can be systematically used to improve prediction accuracy. Another interesting research direction is to explore the reverse problem of predicting links and associated topics in the social network of users, given their purchase data and covariate information.

Bibliography

[1] N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *Proc. 15th Intl. Conf. on Machine Learning (ICML)*, pages 1–9, 1998. 6.2

[2] D. Agarwal and B. Chen. Regression-based latent factor models. In *KDD '09*, pages 19–28, 2009. 1.2.1, 2.2.1

[3] D. Agarwal, B. Chen, and P. Elango. Spatio-temporal models for estimating click-through rate. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 21–30, 2009. 1.2.1

[4] D. Agarwal and S. Merugu. Predictive discrete latent factor models for large scale dyadic data. In *Proc. KDD '07*, pages 26–35, 2007. 1.2, 2.2.1, 3.4

[5] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *JMLR*, 8:1919–1986, 2007. 2.3, 2.3.1, 3.8.1.1, 4, 3.9.1.3, 2, 4.1, 4.2, 4.3, 5.5, 7.7.1, 4

[6] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Jl. Machine Learning Research (JMLR)*, 6:1705–1749, October 2005. 3.4

[7] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *In ICML '04*, pages 65–72, 2004. 2.2

[8] T. Baumann and A. Germond. Application of the kohonen network to short-term load forecasting. In *ANNPS*, pages 407–412, 1993. 1.1, 2.1.1.1

[9] C. Bishop and C. Quazaz. Bayesian inference of noise levels in regression. In *ICANN '96*, pages 59–64, 1996. 5.1, 5.2

[10] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 5.2, 5.3, 5.9.1

[11] M. Blum, R. Floyd, V. Pratt, R. Rivest, and R. Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461, 1973. 5.5.1

[12] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984. 2.1.1.2

[13] P.J. Brockwell and R.A. Davis. *Introduction to Time Series and Forecasting*. Springer-Verlag, New York, 2002. 7.2

[14] G. Cawley, N. Talbot, and O. Chapelle. Estimating predictive variances with kernel ridge regression. In *Machine Learning Challenges '06*, pages 56–77, 2006. 5.2

[15] Y. Chen, D. Pavlov, and J. Canny. Large-scale behavioral targeting. In *KDD '09*, pages 209–218, 2009. 1.2.1

[16] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proc. ICMB '00*, pages 93–103, 2000. 2.3

[17] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum squared residue co-clustering of gene expression data. In *Proc. SDM '04*, 2004. 2.3

[18] C. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16:41–46, 1970. 5.2

[19] C. Chu, S. Kim, Y. Lin, Y. Yu, G. Bradski, A. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *NIPS '06*, pages 281–288, 2006. 9.1

[20] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15:201–221, 1994. 6.1, 6.2

[21] D. Cohn, Z. Gharamani, and M. Jordan. Active learning with statistical models. *Artificial Intelligence Research*, 4:129–145, 1996. 6.2

[22] Srivatsava Daruru, Nena M. Marin, Matt Walker, and Joydeep Ghosh. Pervasive parallelism in data mining: dataflow solution to co-clustering large and sparse netflix data. In *KDD '09*, pages 1115–1124, 2009. 9.1

[23] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW '07*, pages 271–280, 2007. 9.1

[24] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008. 1.3, 9.1, 9.2, 9.2.1

[25] M. Deodhar and J. Ghosh. A framework for simultaneous co-clustering and learning from complex data. *Dept of Elec. and Comp. Engg., Univ. of Texas at Austin, IDEAL-2007-08*, page Downloadable from http://www.lans.ece.utexas.edu/papers/techreports/deodhar07Coclust.pdf, 2007. 3.9.1.4

[26] M. Deodhar, G. Gupta, J. Ghosh, H. Cho, and I. Dhillon. A scalable framework for discovering coherent co-clusters in noisy data. In *ICML '08*, 2008. 1.3, 5.1, 5.5

[27] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *Proc. KDD '03*, pages 89–98, 2003. 2.3

[28] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. 2.1.3

[29] M. Djukanovic, B. Babic, D. Sobajic, and Y. Pao. Unsupervised/supervised learning concept for 24-house load forecasting. *IEE Proceedings-Generation, Transmission and Distribution*, 140:311–318, 1993. 1.1, 2.1.1.1

[30] M. A. Munson et al. The ebird reference dataset. *Tech. Report, Cornell Lab of Ornithology and National Audubon Society*, 2009. 1.2.1

[31] V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972. 6.1, 6.2

[32] C. Ferri and J. Hernandez-Orallo. Cautious classifiers. In *ROCAI '04*, pages 27–36, 2004. 5.2

[33] G. Forman. Tackling concept drift by temporal inductive transfer. In *SIGIR '06*, pages 252–259, 2006. 7.2

[34] J. Fox. *Applied Regression Analysis, Linear Models, and Related Methods*. Sage Publications, 1997. 5.3

[35] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997. 6.2

[36] J. Friedman. Fast sparse regression and classification. Technical report, Stanford University, 2008. 3.6.1.2, 3

[37] K. Fukumizu. Statistical active learning in multilayer perceptrons. *IEEE Transactions on Neural Networks*, 11:17–26, 2000. 6.2

[38] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM*, pages 625 – 628, 2005. 1.2, 2.3, 2.3.1, 3.4, 4.1

[39] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, Harcourt Brace and Company, London, 1981. 3.3, 6.5.1, 7.4

[40] A. Godfrey, L. McAlister, and M. Saar-Tsechansky. A product segmentation model and its relationship to customer segmentation models. *Ph.D. dissertation, McCombs School of Business, Univ. of Texas at Austin*, 2007. 4.4, 4.4

[41] R. Grover and V. Srinivasan. A simultaneous approach to market segmentation and market structuring. *Journal of Marketing Research*, pages 139 – 153, 1987. 2.1.1.1, 3.9.1, 3.9.1.5

[42] S. Ha, S. Bae, and S. Park. Customer's time-variant purchase behavior and corresponding marketing strategies: an online retailer's case. *Comput. Ind. Eng.*, 43(4):801–820, 2002. 7.2

[43] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972. 8.2

[44] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *Advances in Neural Information Processing Systems*, 10, 1998. 2.1.3

[45] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001. 3.5, 3.6.1.1, 3.6.1.2, 6.6

[46] A. Henneguelle, J. Ghosh, and M. M. Crawford. Polyline feature extraction for land cover classification using hyperspectral data. In *Proc. IICAI-03*, pages 256–269, Dec 2003. 2.1.3

[47] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237, 1999. 1.2

[48] T. Heskes. Practical confidence and prediction intervals. In M.I. Jordan M.C. Mozer and T. Petsche, editors, *Advances in Neural Information Processing Systems-9*, pages 176–82. MIT Press, 1997. 5.1, 5.2, 5.3

[49] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H. Toivonen. Time series segmentation for context recognition in mobile devices. In *ICDM '01*, pages 203–210, 2001. 7.2, 7.4

[50] T. Hoffman and J. Puzicha. Latent class models for collaborative filtering. In *Proc. IJCAI '99*, 1999. 2.1.2.3

[51] Z. Huang. Selectively acquiring ratings for product recommendation. In *Proc. International Conference for Electronic Commerce '07*, pages 379–388, 2007. 6.2

[52] P. J. Huber. *Robust Statistics*. Wiley, New York, 1981. 3.10

[53] R. Jin and L. Si. A bayesian approach toward active learning for collaborative filtering. In *UAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 278–285, 2004. 6.2

[54] J.Leonard, M.Kramer, and L.Ungar. Using radial basis functions to approximate a function and its error bounds. *IEEE Transactions on Neural Networks*, 3:4:624–627, July 1992. 5.1, 5.2

[55] J. Johnston. *Econometric Methods*. New York: McGraw-Hill, 1963. 5.2, 5.3

[56] M.I. Jordan and R.A. Jacobs. Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994. 2.1.2.1

[57] M.I. Jordan, R.A. Jacobs, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991. 1.3, 2.1.2.1

[58] Rebecka Jornsten and Bin Yu. Simultaneous gene clustering and subset selection for sample classification via mdl. *BMC Bioinformatics*, 19(9):1100–1109, 2003. 2.1.1.1

[59] T. Kanamori and H. Shimodaira. Active learning algorithm using the maximum weighted log-likelihood estimator. *Journal of Statistical Planning and Inference*, 116(1):149 – 162, 2003. 6.2

[60] J. Kiefer. Optimum experimental designs. *Journal of the Royal Statistical Society*, 21:272–304, 1959. 6.2

[61] B. Kim and P. Rossi. Purchase frequency, sample selection, and price sensitivity: The heavy-user bias. *Marketing Letters*, pages 57 – 67, 1994. 3.9.1

[62] B. Kim and M. Sullivan. The effect of parent brand experience on line extension trial and repeat purchase. *Marketing Letters*, pages 181 – 193, 1998. 3.9.1, 6.1

[63] J. Kleinberg and E. Tardos. *Algorithm Design*. Pearson Education, 2006. 5.5.1

[64] R. Kohavi, R. Longbotham, D. Sommerfield, and R. Henne1. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1):140–181, 2009. 6.1

[65] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08*, pages 426–434, 2008. 2.2.1, 6.6

[66] Yehuda Koren. Collaborative filtering with temporal dynamics. In *KDD '09*, 2009. 2.2.1

[67] S. Kumar, J. Ghosh, and M. M. Crawford. Best-bases feature extraction algorithms for classification of hyperspectral data. *IEEE Trans. Geoscience and Remote Sensing*, 39(7):1368–79, 2001. 2.1.3

[68] W. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, 2003. 3.2.2

[69] D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proc. 17th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 3–12, 1994. 6.2

[70] P. Lingras, M. Hogo, M. Snorek, and C. West. Temporal analysis of clusters of supermarket customers: conventional versus interval set approach. *Inf. Sci. Inf. Comput. Sci.*, 172(1-2):215–240, 2005. 7.2

[71] Xiaoxing Liu, Arun Krishnan, and Adrian Mondry. An entropy-based gene selection method for cancer classification using microarray data. *BMC Bioinformatics*, 6:76, 2005. 2.1.1.1

[72] Larisa Lokmic and Kate A. Smith. Cash flow forecasting using supervised and unsupervised neural networks. *IJCNN*, 06:6343, 2000. 1.1, 2.1.1.1

[73] Z. Lu, D. Agarwal, and I. Dhillon. A spatio-temporal approach to collaborative filtering. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 13–20, 2009. 2.2.1

[74] H. Ma, H. Yang, M. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM '08*, pages 931–940, 2008. 4

[75] D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992. 6.2

[76] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Trans. on Computational Biology and Bioinformatics*, 1(1):24–45, 2004. 2.1.2.2, 2.3, 2

[77] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, London, 1983. 3.4

[78] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. An expected utility approach to active feature-value acquisition. In *ICDM '05*, pages 745–748, 2005. 6.6

[79] W. Moe and P. Fader. Modeling hedonic portfolio products: A joint segmentation analysis of music compact disc sales. *Journal of Marketing Research*, pages 376 – 385, 2001. 2.1.1.1, 3.9.1.5

[80] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. MIT Press, 1998. 3.4, 3.4

[81] A. Nix and A. Weigend. Estimating the mean and variance of the target probability distribution. In *Neural Networks '94*, pages 55–60, 1994. 5.2

[82] K. Oh and I. Han. An intelligent clustering forecasting system based on change-point detection and artificial neural networks: Application to financial economics. In *HICSS-34*, volume 3, page 3011, 2001. 1.1, 2.1.1.1

[83] S. Pandey, D. Agarwal, D. Chakrabarti, and V. Josifovski. Bandits for taxonomies: A modelbased approach. In *In In Proc. of the SIAM International Conference on Data Mining*. SDM, 2007. 4

[84] S. Papadimitriou and J. Sun. Disco: Distributed co-clustering with map-reduce: A case study towards petabyte-scale end-to-end mining. In *ICDM '08*, pages 512–521, 2008. 9.1, 9.3

[85] T. Pietraszek. On the use of ROC analysis for the optimization of abstaining classifiers. *Mach. Learn.*, 68(2):137–169, 2007. 5.2

[86] J. R. Quinlan. Learning with continuous classes. In *Proc. 5th Australian Joint Conference on Artificial Intelligence*, pages 343–348. World Scientific, 1992. 2.1.1.2

[87] J. J. Rowland R. Burbidge and R. D. King. Active learning for regression based on query by committee. *Lecture Notes in Computer Science*, 4881:209–218, 2007. 6.2, 6.4

[88] V. Ramamurti and J. Ghosh. On the use of localized gating in mixtures of experts networks. In *(invited paper), SPIE Conf. on Applications and Science of Computational Intelligence, SPIE Proc. Vol. 3390*, pages 24–35, Orlando, Fl., April 1998. 1.3, 2.1.2.1

[89] T. RayChaudhuri and L. Hamey. Minimisation of data collection by active learning. In *In IEEE ICNN*, 1995. 6.2, 6.4

[90] T Reutterer. Competitive market structure and segmentation analysis with self-organizing feature maps. *Proceedings of the* 27^{th} *EMAC Conference*, pages 85 – 115, 1998. 2.1.1.1, 3.9.1.5

[91] N. Roy and A. K. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th Intl. Conf. on Machine Learning (ICML)*, pages 441–448, Williams Collge, USA, 2001. 6.2, 6.7

[92] N. Rubens and M. Sugiyama. Influence-based collaborative active learning. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 145–148, 2007. 6.2

[93] M. Saar-Tsechansky and F. Provost. Active sampling for class probability estimation and ranking. *Machine Learning*, 54(2):153–178, 2004. 6.2

[94] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS '07*, 2007. 2.2.1, 4

[95] P.B. Seetharaman, A. Ainslie, and P.K. Chintagunta. Investigating household state dependence effects across categories. *Journal of Marketing Research*, pages 488 – 500, 1999. 3.9.1, 6.1, 7.7.1

[96] H. S. Seung, M. Opper, and H. Smopolinsky. Query by committee. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 287–294, Pittsburgh, USA, 1992. 6.2, 6.4

[97] A. Sfetsos and C. Siriopoulos. Time series forecasting with a hybrid clustering scheme and pattern recognition. *Systems, Man and Cybernetics, Part A, IEEE*, 34:399– 405, 2004. 1.3, 2.1.1.2

[98] H. Shan and A. Banerjee. Bayesian co-clustering. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 530–539, 2008. 2

[99] A. Sharkey. On combining artificial neural networks. *Connection Science*, 8(3/4):299–314, 1996. 1.1

[100] D. Shepard. *The New Direct Marketing: How to Implement A Profit-Driven Database Marketing Strategy*. McGraw-Hill, 1999. 5.1

[101] D. Shrestha and D. Solomatine. Machine learning approaches for estimation of prediction interval for the model output. *Neural Networks*, 19(2):225–235, 2006. 5.2

[102] M. Sugiyama. Active learning in approximately linear regression based on conditional expectation of generalization error. *Journal of Machine Learning Research*, 7:141–166, 2006. 6.2

[103] M. Sugiyama and S. Nakajima. Pool-based active learning in approximate linear regression. *Jl. of Machine Learning*, 75(3):249–274, 2009. 6.2

[104] M. Sugiyama and N. Rubens. Active learning with model selection in linear regression. In *Proc. Siam Conference On Data Mining '08*, pages 514–529, 2008. 6.2, 6.5.3

[105] R. Tibshirani. A comparison of some error estimates for neural network models. *Neural Computation*, 8:152–163, 1996. 5.2

[106] Y. Wang and I. H. Witten. Inducing model trees for continuous classes. In *In Proc. of the 9th European Conf. on Machine Learning*, pages 128–137, 1997. 2.1.1.2, 3

[107] M. Wedel and J. Steenkamp. A clusterwise regression method for simultaneous fuzzy market structuring and benefit segmentation. *Journal of Marketing Research*, pages 385 – 396, 1991. 2.1.2.2, 2.3

[108] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69–101, 1996. 7.2

[109] D. Wiens. Robust weights and designs for biased regression models: Least squares and generalized m-estimation. *Journal of Statistical Planning and Inference*, 83(2):395 – 412, 2000. 6.1, 6.2

[110] B. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *ICDE*, pages 13–22, 2000. 7.2

[111] Shuai Zhang, Daniel Neagu, and Catalin Balescu. Refinement of clustering solutions using a multi-label voting algorithm for neuro-fuzzy ensembles. In *ICNC*, pages 1300–1303, 2005. 2.1.1.2

[112] Z. Zheng and B. Padmanabhan. Selectively acquiring customer information: A new data acquisition problem and an active learning based solution. *Management Science*, 52(5):697–712, 2006. 1

Vita

Meghana Deodhar completed her undergraduate education from the Govt. College of Engineering, Pune (India) and graduated in June 2004 with the Bachelors of Engineering degree in Computer Engineering. She received her masters degree in May 2006 from the ECE department at UT Austin. She began her Ph.D. at UT Austin in August 2006 and was advised by Dr. Joydeep Ghosh, director of the Intelligent Data Exploration and Analysis Lab. Her industry experience includes internships at Kosmix, eBay, Yahoo! and Amazon between 2006 and 2009. Her research interests are co-clustering, prediction problems in complex domains and data mining applications in bioinformatics and marketing.

Permanent address: 20 Pushpak Park, Baner Road, Aundh, Pune
411007, India.

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.