Revisiting Recommendations: From Customers to Manufacturers

by

Shailendra Agarwal

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

September 2013

Abstract

Recommender systems exploit user feedback over items they have experienced for making recommendations of other items that are most likely to appeal to them. However, users and items are but two of the three types of entities participating in this ecosystem of recommender systems. The third type of entities are the manufacturers of the products, and users are really their customers. Traditional recommender systems research ignores the role of this third entity type and exclusively focuses on the other two. What might item producers bring to recommender systems research? Their objectives are related to their business and are captured by questions such as "what kind of (new) products should I manufacture that will maximize their popularity?" These questions are not asked in a vacuum: manufacturers have constraints, e.g., a budget. The idea is that the user feedback data (e.g., ratings) capture users' preferences. The question is whether we can learn enough intelligence from it, so as to recommend new products to manufacturers that will help meet their business objectives.

We propose the novel problem of new product recommendation for manufacturers. We collect real data by crawling popular e-commerce websites, and model cost and popularity as a function of product attributes and their values. We incorporate cost constraints into our problem formulation: the cost of the new products should fall within the desired range while maximizing the popularity. We show that the above problem is NP-hard and develop a pseudo-polynomial time algorithm for the recommendations generation. Finally, we conduct a comprehensive experimental analysis where we compare our algorithm with several natural heuristics on three real data sets and perform scalability experiments on a synthetic data set.

Preface

This thesis is the outcome of my collaborative research with my colleague Dr. Amit Goyal and my supervisor Dr. Laks V.S. Lakshmanan. Below listed are the contributions of each of the collaborators, including me.

Dr. Laks V.S. Lakshmanan was involved in the inital motivation of the problem solved here. He was present in many of the brainstorming sessions and gave various suggestions to formulate the problem, to solve the problem and to conduct experiments to test the proposed algorithm. He was also involved in refining the content and presentation of the work.

Dr. Amit Goyal was present in all the discussions involving formally defining the problem, proposing solutions to the problem and designing experiments. He proposed various improvements to the algorithms. He looked at some of the previous literature in the field. He was also involved in improving the content and presentation of the work.

Along with Dr. Laks V.S. Lakshmanan, I was involved in the initial brainstorming that led to the formulation of this problem. I looked at most of the previous literature in the field. I, along with both collaborators, defined the problem formally. I was responsible for crawling the real data from multiple e-commerce websites. I implemented the regression-based cost and popularity modeling. I proposed the initial algorithm to solve the problem and conducted expeirments to test the algorithm. I was responsible for keeping track of the work done during the project and summarizing it.

Table of Contents

List of Tables

List of Figures

Glossary

LOOCV	Leave-one-out Cross-validation
OPDP	Optimal Product Design Problem
MCKP	Multiple-choice Knapsack Problem
RMSE	Root Mean Square Error
RS	Recommender Systems
RECMAN	RECommendation for MANufacturers

Acknowledgments

I would like to sincerely thank my supervisor Dr. Laks V.S. Lakshmanan for his continuous support and encouragement throughout my study period. He was always excited to talk about new ideas and was a great help in completing this thesis project. I would also like to thank Dr. Amit Goyal for the fun and insightful discussions we had during the course of this project. My thanks go to Dr. Rachel Pottinger for patiently reading this thesis and providing suggestions for improvement. I would like to thank Dr. Ruben H. Zamar, Dr. Matias Salibian-Barrera and Fred for the various stimulating discussions during the project. Lastly, thanks to all my friends, labmates and family for always being there.

Chapter 1

Introduction

Recommender Systems (RS) exploit user feedback of items to make recommendations of other items that are most likely to appeal to them. In the ecosystem of a RS, three types of entities participate – customers (users), products (items), and manufacturers (item producers). Research in RS has mostly focused on the customer perspective of the recommendation problem: how can we recommend high quality products to customers in a personalized way? This is usually done by predicting a user's likely rating on a product she has not experienced before, by leveraging users' feedback on products they have experienced [17]. There is an important, "flip" side to this problem: we still recommend products but the target population no longer are the customers who buy the product, rather the target population now are the manufacturers who sell the product. The perspective of this third entity type, item manufacturers, has heretofore largely been ignored.

What might item producers bring to RS research? Their objectives are related to their business and are captured by questions such as "what kind of (new) products should I manufacture that will maximize the popularity?" Here, a *new* product refers to a non-existent product. E.g., a smartphone company may wish to know how to "tune" the design of new products it would like to launch in the market so that they achieve a high (online) popularity. A similar remark applies to the manufacturers of other durable goods like video gaming consoles. These business questions are not asked in a vacuum: manufacturers have constraints, e.g., a budget. The idea is that the user feedback data (e.g., ratings) capture users' pref-

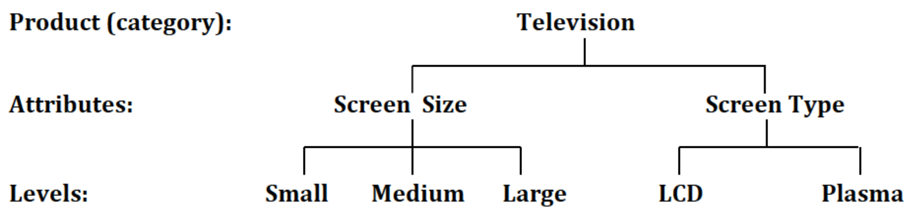


Figure 1.1: Example.

erences. The question we ask is whether we can learn enough intelligence from it, e.g., customers' preference for products' attribute values, so as to recommend new products to manufacturers that will help meet their business objectives.

New product recommendations can be valuable to manufacturers: careful design of a new product before it is launched is crucial for the product's success. Marketing and launching a new product is expensive and critical to brand management [11]. Once launched, it is difficult to roll back. Kotler et al. [11] identify bad product design as one of the most frequent reasons for the failure of a new product. Motivated by this, we bring a "dual" perspective, i.e., the business perspective, to the long-studied customer-centric RS problem.

A product (or product category) possesses a number of attributes where each attribute can have one of several possible values (also called *levels*). For instance, consider the (simplified) example shown in Figure 1.1. In this example, the television category has two attributes – `Screen Size` and `Screen Type`. Screen Size has three values – Small, Medium and Large, while Screen Type has two values – LCD and Plasma. A product is designed by picking exactly one value for each attribute, e.g., (Screen Size = Medium, Screen Type = LCD) is one out of six feasible products. In real life, there may be multiple attributes, each having a large number of possible values. The objective is to design one or more new products maximizing their predicted popularity.

In this work, we focus on durable products. We collect data on three product categories – camera, television and laptop – by crawling two popular e-commerce platforms – www.shopping.com and www.bestbuy.com. Our first task is to build predictive models for cost and popularity of products, where cost is the price per unit of the product mentioned on the e-commerce website and popularity is the number of customer ratings received by the product. A major challenge in building predictive models is that these data sets suffer from acute user sparsity, that is, most of the users rate very few products. Indeed the user sparsity issue is intrinstic to these types of data sets: one user is unlikely to buy multiple (expensive) products of the same kind (e.g., many cameras) and hence, may not review multiple similar products. Thus, individual user preferences for product attributes cannot be learned reliably. Consequently, instead of learning per-attribute user preferences, and then estimating a new product's popularity, we predict the popularity by directly estimating the impact of various attribute values on popularity. Similarly, we build a model for predicting the cost of a new product as a function of the attribute values that make it up.

Since the decision of which product to launch may be driven by other external considerations such as availability of expertise, engineering feasibility, etc., there is a need for flexibility in the product designs recommended. To accommodate this, we propose the problem of recommending top-*k* new items to the manufacturers. Thus, similar to the *k* most appealing product recommendations to customers in RS, we recommend top-*k* new products (*k* is a manufacturer specified parameter) to manufacturers, out of which it can choose one or more products based on technical feasibility, product diversity and other engineering concerns.

Cost is a key factor in designing new products [18, 21], as it directly influences the quality of (attribute values in) the product. As Hauser et al. [5] note, "Price plays a special role in consumer preferences. It is not a feature of a product per se, but rather that which the consumer pays in return for features." Moreover, marketing decisions, such as which market segment to target, depend on the cost of the product. Thus, in view of the importance of cost of a product, we allow the manufacturer to specify a cost range such that the cost of the new products, should satisfy the specified range, while maximizing the popularity.

In marketing research literature, the problem of optimal product design has been extensively studied. Conjoint analysis is the popular approach that is followed. The data is collected via conducting user surveys, and then, users' preferences are modeled based on the responses. A detailed comparison with that problem appears in Chapter 2.

In summary, we formulate new product REClommendation for MANufacturers

(RECMAN) problem as follows: Given the attributes and the possible values that a product (category) can have, lower bound C_{lb} and upper bound C_{ub} on cost, and a parameter k, recommend k new products with top-k highest (predicted) popularity, such that the (predicted) cost of each of these new products lies between the cost bounds. We make the following contributions.

- We propose a novel problem of new product recommendation for manufacturers, capturing an interesting business perspective of recommender systems. Our formulation incorporates cost constraints and asks for top-k new product designs that achieve the highest popularity with cost falling in a specified range (Chapter 3).

- We explore various regression models to predict cost and popularity of product and find that linear regression performs the best among them (Chapter 4).

- We show that RECMAN is NP-hard. We develop a pseudo-polynomial time dynamic programming algorithm whose running time is polynomial in the value of the cost upper bound. We show that our algorithm is optimal (Chapter 5).

- We conduct a comprehensive set of experiments on three real data sets and compare our algorithm with natural heuristics. The results show that our algorithm outperforms the alternatives by considerable margins. Furthermore, our algorithm scales very well w.r.t. both running time and memory usage Chapter 6.

Related work is presented in Chapter 2, while we conclude the thesis in Chapter 7.

Chapter 2

Related Works

A popular approach in RS is collaborative filtering, which seeks to predict the expected rating a given user may provide to an item which she hasn't rated before. The top items w.r.t. predicted ratings are then recommended to the user. Collaborative filtering approaches can be classified into memory-based (e.g., User-based and Item-based) and model-based (e.g., Matrix Factorization). We recommend Chapters 4 and 5 of [17] as excellent surveys. As we have described above, while RS research mostly focuses on recommending relevant items to customers, we propose a novel problem – recommending designs of new items to the manufacturers, building and launching which, their popularity could be maximized.

Our work is partly inspired by the Optimal Product Design Problem (OPDP), studied in the fields of Marketing and Operations Research. The objective here is to design a new product that maximizes the number of users who buy it. In conjoint analysis, which is the state of the art for this problem [3, 20], the utility of a user *u* for a product is assumed to be additive, that is, sum of her utilities for the values of different attributes of the product. If *u*'s utility for a new product is higher than the utility of the status quo product that *u* owns, then *u* is assumed to buy the new product. OPDP is known to be NP-hard [9] and several heuristics have been proposed [20]. While the overall objective of OPDP is same as RECMAN, on a higher level, there are several major differences between the two problems in terms of input, constraints and the consequent solution approach. (i) In OPDP, the data used is collected via conducting user surveys (see [3]), which are expensive and time-consuming, while the input to RECMAN is the easily available RS data i.e., ratings data collected from e-commerce websites. (ii) In OPDP, individual user preferences are modeled. Whereas, as we show in Chapter 3, the durable products data sets have inherent acute user sparsity (each user buys and, hence, rates very few items) and hence it is not possible to model individual user preferences reliably. Accordingly, we look at each product as a whole and estimate the cost and popularity of a new product as a function of product attributes of a given category by aggregating the available social signals in the form of user ratings. (iii) Unlike in OPDP, our problem admits cost constraints and aims to output top-*k* new products instead of just one.

Das et al. [2] study the problem of web item design. They use collaborative tagging data from users to identify new products that are expected to attract maximum number of distinct desirable tags. Even though our problem, at a very high level, bears some similarities to theirs, there are several key differences. Their objective function is different: they take a set of desirable tags as input and try to maximize the number of *distinct* desirable tags. On the other hand, our objective is to maximize product popularity. The different objective functions make way for different solution approaches for the two problems. In particular, their approach cannot be applied for recommending products that maximize number of ratings since their framework ignores repeated tagging on the same item and only considers distinct tags. Besides, unlike them, we incorporate cost constraints as input, thus providing additionally flexibility to manufacturers.

Considerable work has been done on extracting customer opinions, by applying text mining techniques [1, 6, 16]. We recommend Chapter 5 of [13] for an excellent survey. Archak et al. [1], in addition to mining customer opinions, apply regression analysis to model sales (of products) as a function of opinions of product attributes. For instance, one of their results is: a better camera has more positive impact on sales (in category Camera and Photo) than a better battery. Our problem is fundamentally different than that of opinion mining: our goal is to build new products, to maximize the popularity. We do it by identifying the actual attribute-values of products (instead of opinions). e.g., in our analysis, we find that a camera with a memory stick is more popular than a camera with built-in memory. Note that “memory stick” and “built-in” are two values of the attribute “memory type”.

Miah et al. [14] study the problem of finding top-k attributes of a product that a seller should highlight in a marketplace such that it attracts maximum visibility. The problems are significantly different as their goal is not to design a new product, but to maximize the visibility of an existing product in a marketplace. Secondly, they do not consider the identification of attribute values. Rather, the attribute values are already known.

Real world product design is a more complex process involving product engineering, advertising and other managerial decisions. Nevertheless, RECMAN tends to provide a starting point to the decision makers regarding the set of attributes values that are most important for maximizing popularity, given a cost constraint.

Chapter 3

Problem Definition

3.1 Data Sets

Before we define the problem statement formally, we explore the characteristics of the data sets we collected. We perform our study on 3 types of durable products – television, camera and laptop. To collect the data, we crawled two popular platforms – www.shopping.com for cameras and www.bestbuy.com for televisions and laptops. From each data set, we built 3 tables as follows. The first table corresponds to the *ratings log* where each tuple $\langle u, p, r \rangle$ represents that user u rates product p with rating r. The second table corresponds to *products' attributes data* where each tuple $\langle p, a, l \rangle$ indicates that in product p, attribute a has value (also called level) l. Finally, the third table records the cost (sale price) of each product as $\langle p, c \rangle$. Table 3.1 summarizes statistics of the raw crawled data. For instance, in category television, the raw data contains 334 products, 9682 users and 9884 ratings. Each product (television) has 27 attributes, including both numerical and categorical.

3.2 Issue of User Sparsity

In the original RS problem, a user is recommended k items by predicting her likely ratings for items she has yet not rated. To do this, user preferences are captured using her previous interactions with the system. For example, Matrix Factoriza-

Statistic	Television	Camera	Laptop
Number of products	334	969	304
Number of users	9682	5317	2317
Number of ratings	9884	6013	2353
Number of attributes	27	90	16
Numerical	6	27	7
Categorical	21	63	9

Table 3.1: Raw Data Set Statistics

tion [10] represents each user by a latent factor vector which is learned based on her prior ratings. So, a natural approach to solve RECMAN would be to build a similar model for each user to predict whether the particular user would rate the product, following which we can return the top-*k* products that are rated by maximum number of users. The argument below shows the problem encountered with such an approach and paves the way for the possible approach.

In Table 3.2, we present the frequency distribution of ratings from a user perspective. The table shows, that for every product category, the number of users who rated more than 1 product is a tiny fraction of the number of users who rated exactly 1 product and this number decays extremely fast as number of ratings increases. As we argued before, this behavior is intrinsic to durable products: it is highly unlikely that one single user will buy (and rate) more than one or two products from a "single product category". The average numbers of ratings per user are 1.02, 1.13 and 1.01 respectively for television, camera and laptop categories. We refer to this as the *user sparsity* issue. Similar user sparsity issue has also been witnessed in other data sets like www.amazon.com and www.resellerratings.com, across a variety of product categories – Books, Music, DVD, electronics, computers etc. [7, 22]. Indeed, *individual* user preferences learnt from such data sets would severely suffer from overspecialization and overfitting and, hence, would be highly unreliable. Therefore, we formulate RECMAN in an alternative way that can mitigate the user sparsity issue.

Instead of looking at individual user preferences, we look at each product as a whole. Figure 3.1 shows the frequency distribution of ratings from product perspective. The plots show that the sparsity in this case is much less compared to

Number	Number of users		
of ratings	Television	Camera	Laptop
1	9499	4997	2289
2	168	218	24
3	13	50	2
4	1	19	0
5	0	12	2
$\geq$6	1	21	0

Table 3.2: Frequency Distribution of User Ratings

user sparsity. For instance, in television, 151 products received 5 or more ratings. Similarly, in categories camera and laptop, the number of products which received 5 or more ratings are 288 and 88, respectively. The average numbers of ratings received per product are 30.4, 6.2 and 7.9 respectively. We further prune the products that have received very few ratings (< 5). This gives us the average numbers of ratings per product as 66.7, 19 and 26.8, corresponding to the three data sets. These numbers suggest that if we look at the product as a whole, and try to model cost and popularity as functions of attribute values of products, we would be able to avoid the issue of overfitting.

3.3 Our Problem Formulation

The objective of RECMAN is to recommend top-*k* products such that the popularity, expressed as number of ratings they receive are maximized. Let ρ denote a product (or product category). With m, we denote the number of attributes it can have. An attribute can assume several values (or levels). Then, a product is built by selecting exactly one value for each attribute. We handle a realistic setting where manufacturers may have cost constraints, by permitting parameters for lower bound (denoted as C_{lb}) and upper bound (denoted as C_{ub}). The new products should achieve the maximum popularity while leaving the cost of each of the products within the specified cost bounds.

In case of RS, to recommend relevant products to users, the number of products that need to be explored are only the products that exist in the system. In other words, search space is linear in case of RS. By contrast, in RECMAN, we must

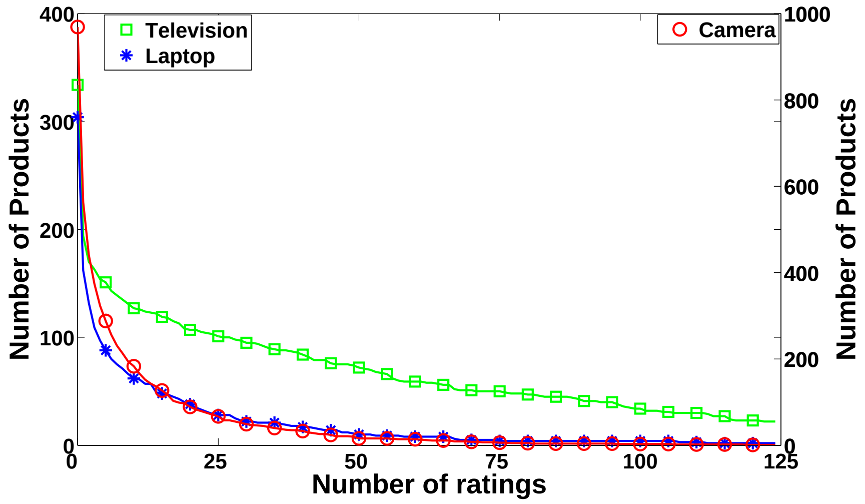


Figure 3.1: Frequency Distribution of Ratings from Product Perspective. Left Y-axis: Television and Laptop; Right Y-axis: Camera.

recommend *new product designs* that don't exist in the system. A product is built by selecting exactly one value for each attribute, thus making the search space exponential. We refer to this space of product designs as Ω. Its important to bear this in mind to understand the notion of scale for new product designs. Any algorithm, to be useful, must scale w.r.t. the size of this huge search space. Notice that the number of products in the data set has no direct bearing on the scale issue.

To define the problem, for now, we assume that we are given an oracle $\textsc{Cost}(\rho)$ that predicts the cost of the product ρ, and another oracle $\textsc{Popularity}(\rho)$ that predicts the popularity of ρ. Further, we assume that cost (also popularity) of a product can be determined by its attributes and hence can be modeled as a function of the attributes. In Chapter 4, we explore various regression models to estimate cost and popularity, and report our findings. We next formally state RECMAN.

Problem 1. *Given a product design space Ω of m attributes and their possible levels (values), lower bound C_{lb} and upper bound C_{ub} on cost, and a parameter k, design k new products $\mathscr{P} \subseteq \Omega$, each by selecting exactly one level for each attribute, such that $\forall \rho \in \mathscr{P} : [C_{lb} \leq \textsc{Cost}(\rho) \leq C_{ub}$ & $\nexists \rho' \in (\Omega \setminus \mathscr{P}) : (C_{lb} \leq \textsc{Cost}(\rho') \leq C_{ub} \& \textsc{Popularity}(\rho') > \textsc{Popularity}(\rho))]$.*

Chapter 4

Modeling Cost and Popularity

We first report the steps taken to clean the data. Then, we describe the models explored for implementing cost (and popularity) oracles and compare their performance. Further, we define $\textsc{Cost}(\rho)$ and $\textsc{Popularity}(\rho)$, based on linear regression, the model which performs the best among those we explored. Finally, we give a quantitative description of some of the attribute levels that are strong predictors of cost (and popularity).

4.1 Data Set Pre-processing

The raw data set (Table 3.1) contains both numerical and categorical attributes. We replace missing values of numerical attributes by the mean of known numerical values of that attribute, and ignore missing values of categorical attributes [19]. We employ K-means Clustering with number of clusters as 3, to discretize the numerical attributes into 3 clusters[1]. Next, we binarize all attributes to derive features for regression analysis. For instance, for the attribute Screen Type in the example shown in Figure 1.1, there are two levels – LCD and Plasma; we create two binary features for it – ScreenType_LCD and ScreenType_Plasma.

To avoid overfitting, we take the following measures, on all three data sets. First, we prune all the products that received few (< 5) ratings and very large num-

[1] We also tried K-means clustering with 5 clusters on numerical attributes. We obtained similar results in both cases, although numerical values varied slightly. For brevity, we present results only for 3 clusters.

Statistic	Television	Camera	Laptop
Number of products	136	250	75
Number of ratings	7702	3612	1525
Number of attributes	13	34	15
Total # of levels	34	85	27

Table 4.1: Pruned Data Set Statistics

ber of ratings (top 2.5% of ratings received). Second, we remove the products which are too costly (top 2.5%) or too cheap (bottom 2.5%). We remove the (binarized) features that occur rarely (< 10 times). Finally, we do feature selection using minimal-redundancy-maximal-relevance criterion [15]. Table 4.1 summarizes the statistics of the pruned data.

4.2 Regression Analysis

We explore the following regression models (see [4]) for modeling cost and popularity.

Linear Regression. It assumes a linear relationship between the features and the expected output values of $\textsc{Cost}(\cdot)$ and $\textsc{Popularity}(\cdot)$. We constrain the coefficients of regression to be non-negative since both cost and popularity assume only non-negative values.[2]

Poisson Regression. This is a generalized linear model that assumes linear relationship between input and logarithm of the output, and returns only non-negative output, as required by our problem formulation.

Regression Trees. The idea here is to recursively partition the feature space based on the value of features. At each step, the remaining set of training data is divided into two classes based on the value of the feature under consideration. This continues until a certain depth, calculated using various heuristics, following which a model such as regression is fit, within each leaf of the tree [4].

We use MATLAB implementations of linear regression (`lsqnonneg`), poisson regression (`glmfit`) and regression trees (`classregtree`). We perform Leave-

[2]For the sake of completeness, we also performed regression analysis with unconstrained coefficients. As expected, the results were worse than for the non-negative case.

Model used	Television	Camera	Laptop
Linear Regression	343.5	320.6	293.5
Poisson Regression	360.4	526.6	235.3
Regression Tree	458.8	423.1	255.3
Range of cost ($$)	140 - 4300	50 - 2549	350 - 1900

Table 4.2: RMSE in Predicting the Cost of Product

one-out Cross-validation (LOOCV) (see chapter 4 of [19]) to compare various models. In LOOCV, the test set contains only one product and the training set contains the remaining products. Thus, the total number of models built equals total number of products. A key benefit of using LOOCV is that it uses the maximum data to train the model, which is especially helpful in cases when the data does not contain many samples, as in our case.

4.2.1 Comparison

Table 4.2 and Table 4.3 show the Root Mean Square Error (RMSE) associated with cost and popularity prediction using the three models. To provide a context, we also include the range of cost and popularity under various categories in the table. For example, in television category, the cost varies from $140 to $4300 (after pre-processing) and popularity varies from 5 to 172 (note that we removed products which received less than 5 ratings, as part of pre-processing). Consider the prediction of cost (Table 4.2): linear regression outperforms others in television and camera categories, with RMSE values 343.5 and 320.6, respectively. Even though linear regression is dominated by other models in laptop category, the gap is marginal. Besides, it has significantly less error on the other two data sets. In case of prediction of popularity (Table 4.3), linear regression with RMSE values 41.9, 8.6 and 16.7 is clearly better than the other two methods on all three datasets. Hence, we take linear regression as the winner.

We further look into the error characteristics of linear regression and see if they follow intuitions. The results are presented in Figure 4.1, in terms of *percentage*

Model used	Television	Camera	Laptop
Linear Regression	41.9	8.6	16.7
Poisson Regression	54.3	11.0	80.9
Regression Tree	51.1	10.3	19.1
Range of popularity	5 - 172	5 - 44	5 - 66

Table 4.3: RMSE in Predicting the Popularity of Product

prediction error, which is computed as

$$\frac{|\text{predicted value} - \text{actual value}|}{\text{actual value}} \times 100$$

Figure 4.1a shows the distribution of percentage error in cost prediction. It shows that the frequency of percentage error decreases as the percentage error increases. Thus, we can expect reasonable prediction accuracy, with high error occurring very few times. Similar results for popularity are shown in Figure 4.1b. One immediate observation, differentiating Figure 4.1a and Figure 4.1b is that the prediction of popularity is not as accurate as that of cost. One possible reason is that, unlike cost which depends mostly on attributes of a product, popularity may depend on various other factors like quality of implementation of various attributes, marketing, overall consumer experience, emotional attachment of consumers to the brand, etc. Hence, the task of predicting popularity is more challenging than predicting cost.

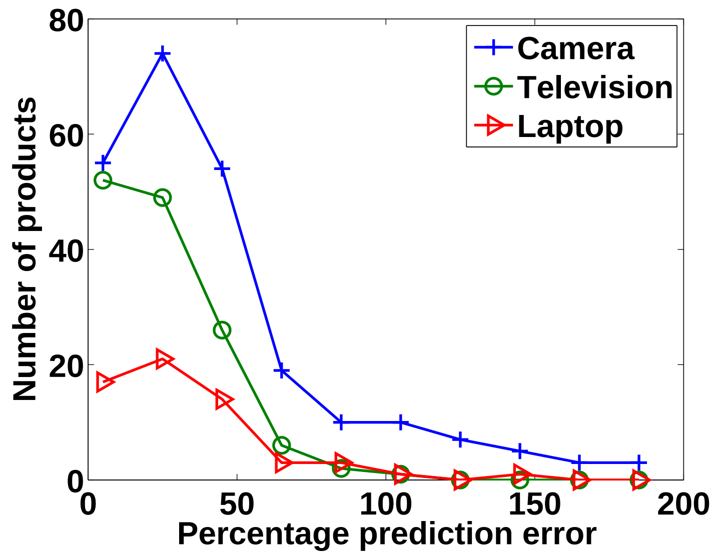
4.2.2 Cost and Popularity Oracles

Having shown that linear regression is best able to model the cost and popularity, here, we define the predicted cost as

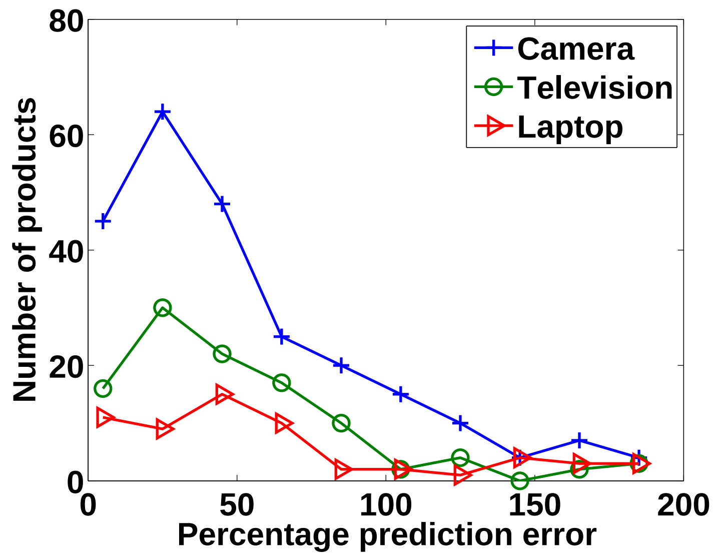
$$\textsc{Cost}(\rho) = \sum_{(a:l)\in\rho} \textsc{Cost}(a,l) \tag{4.1}$$

where $(a : l) \in \rho$ means that the attribute a has level l in the product ρ. Similarly, we define the predicted popularity as

$$\textsc{Popularity}(\rho) = \sum_{(a:l)\in\rho} \textsc{Popularity}(a,l) \tag{4.2}$$



(a) Cost



(b) Popularity

Figure 4.1: Frequency Distribution of Percentage Prediction Error (a) Cost (b) Popularity

Here, $\textsc{Cost}(a,l)$ ($\textsc{Popularity}(a,l)$) is the coefficient of the binarized feature that corresponds to level l found in the linear regression model of $\textsc{Cost}$ (.) ($\textsc{Popularity}$ (.)), and, effectively, denotes the impact of level l on the overall cost (popularity) of product. For simplicity, we refer to $\textsc{Cost}(a,l)$ ($\textsc{Popularity}(a,l)$) as cost (popularity) of l. Note that we have constrained both $\textsc{Cost}(a,l)$ and $\textsc{Popularity}(a,l)$ to be non-negative, so that the $\textsc{Cost}(\rho)$ and $\textsc{Popularity}(\rho)$ are also non-negative.

4.3 Strong Predictors

Table 4.4 shows the top 10 attribute-level pairs that are strong predictors of the cost and popularity (top 10 highest regression coefficients) of television and camera category, respectively. Recall that we binarize the attribute-value pairs, hence, one attribute may appear multiple times in both tables. Most of the results match our intuitions, for example, in television category, high dollar savings on a product implies that the cost of product would be high and is a good indicator of cost. Similarly, large size of screen significantly increases the cost of television. Ethernet port is present only in high cost televisions and brand LG mostly has high priced televisions compared to Dynex and Insignia.

Looking at popularity modeling, in the television category, we find that brand is the most important predictor of popularity. Also note that, many attribute-level pairs that have high cost don't have high popularity. This is because the majority of population chooses products that have neither very high nor very low cost. Such products have medium dollar savings and 2 HDMI inputs, instead of high dollar savings and 4 HDMI inputs which have higher cost. We further investigate the televisions of brand Dynex, which has the highest popularity. We find that these are only available at two stores: www.bestbuy.com and www.futureshop.ca. Since our television data set was crawled from www.bestbuy.com, it is naturally biased towards the popularity of this brand.

In camera category, cameras belonging to Nikon D and Canon EOS family lines are expensive. Further, cameras with HDMI interface are costlier as opposed to cameras with USB interface and cameras having interchangeable lens cost high. For popularity modeling, popularity of cameras with memory stick is higher than cameras with compact flash card. This again supports our observation, from television category, that majority of population chooses products that have neither very high cost (compact flash card) nor very low cost (built-in memory), but goes for something in between (memory stick).

Cost		Popularity	
Attribute	Level	Attribute	Level
Dollar savings*	High	Brand	Dynex
Screen size*	Large	Brand	Insignia
Dollar savings*	Medium	Refresh rate	120 Hz
Refresh rate	240 Hz	Dollar savings*	Medium
Brand	LG	Refresh rate	240 Hz
Refresh rate	600 Hz	Dollar savings*	High
#HDMI inputs	4	#HDMI inputs	2
Ethernet port	Present	Brand	Samsung
Screen size*	Medium	Vert. resolution	768 pixel
USB port	Present	Screen size*	Medium

(a) Television

Cost		Popularity	
Attribute	Level	Attribute	Level
Family line	Nikon D	Memory type	Memory stick
Family line	Canon EOS		
Interface type	HDMI	Compression mode	Basic
Interchangeable lens	Yes	Resolution*	Low
Width*	Medium	Height*	Medium
32 MB Memory card	Present	Battery type	Lithium
Memory type	Compact flash card type II	Max. movie length*	Medium
		Viewfinder	Optical
Compression type	TIFF	Built-in memory size*	Large
Software	Absent	32 MB Memory Card	Present
Memory type	Memory stick	LCD screen size*	Small

(b) Camera

Table 4.4: Top 10 Attribute-level Pairs in Cost and Popularity Modeling, in decreasing order (a) Television (b) Camera. * denote Numerical Attributes.

Chapter 5

Algorithm

5.1 Complexity of RECMAN

Next, we analyze the complexity of RECMAN. We show that it is NP-hard and is closely related to the Multiple-choice Knapsack Problem (MCKP) [8]. MCKP is a generalization of the classical knapsack problem. In classical knapsack, given a set of n items, where each item $j \in [1,n]$ is associated with a profit p_j and a weight w_j, the goal is to pick a set of items such that their total weight is under a given bound W, and the total profit is maximum. In MCKP, the items are partitioned into m lists (or classes), and exactly one item must be selected from each list such that the total cost is under W and the profit is maximum.

Theorem 1. RECMAN *as defined in Problem 1 such that* $\textsc{Cost}(\cdot)$ *and* $\textsc{Popularity}(\cdot)$ *are defined as in Equation 4.1 and Equation 4.2, respectively, is NP-hard.*

Proof. We reduce MCKP to RECMAN. Given an instance $\mathscr{I}$ of MCKP, create an instance $\mathscr{G}$ of RECMAN as follows. Set $k = 1$ and $C_{lb} = 0$ in instance $\mathscr{G}$. For each list L_i in instance $\mathscr{I}$, create an attribute a_i in instance $\mathscr{G}$. Next, for each item x in a list L_i in $\mathscr{I}$, create a level l_x for the corresponding attribute a_i in $\mathscr{G}$. Set $\textsc{Cost}(a_i, l_x) = w_{ix}$ where w_{ix} is the weight of item x in list L_i. Similarly, set $\textsc{Popularity}(a_i, l_x) = p_{ix}$, where p_{ix} is the profit of item x in list L_i. Finally, set $C_{ub} = W$.

Considering only top-1 product design ($k = 1$), a new product ρ in instance $\mathscr{G}$

is a tuple of attribute-level pairs, which in turn corresponds to a set of items S in instance $\mathscr{I}$, exactly one from each list. Clearly, the total weight of S is same as $\textsc{Cost}(\rho)$ and likewise, the profit of S is same as $\textsc{Popularity}(\rho)$. Hence ρ is a solution to RECMAN iff the corresponding set of items S is a solution to MCKP, showing RECMAN is NP-hard. $\square$

5.2 An Optimal Algorithm

One may naturally believe that since the problem is NP-hard, we may need to settle for an approximation algorithm or resort to heuristics. Interestingly, this is not the case with RECMAN. Even though RECMAN is NP-hard, we show that the algorithm that we develop in this section is both optimal and scalable, for real world data sets in our context. Its running time is linear in the upper bound of the cost range of the product, i.e., C_{ub}. In experiments, we show that our algorithm scales well on large data sets even when the cost upper bound C_{ub} is as high as 100,000 \$. The algorithm is an extension of a dynamic programming algorithm proposed for MCKP [8]. For ease of exposition, we first describe the original algorithm adapted to our context.

5.2.1 Sketch of Original Algorithm

It maintains a table T of size $(m+1) \times (C_{ub}+1)$ where each entry in the table contains the popularity of the optimal *partial* product. If costs are real numbers, they are rounded off to the nearest integer to facilitate table construction [8]. The row index of the table corresponds to an attribute of the product and the column index corresponds to the cost. All indices start with 0. We initialize row 0 with appropriate base cases and then span the entire table T, one row at a time, starting from index $[1,0]$. At any step, corresponding to index $[i,j]$, build the optimal partial product such that its cost is exactly j and it contains the attributes $\{1,\cdots,i\}$. The idea is that the optimal partial products for row i can be built recursively from the optimal partial products for row $i-1$. When the table is built completely, pick the product at index $T[m,j]$ such that $j \leq C_{ub}$ and $T[m,j]$ is maximum. Finally, the levels of attributes of the optimal product are obtained from a backtracking process where, starting from the last row at index $[m,j]$, the optimal level for each attribute is used to arrive to the previous row. This is done until levels of all attributes are extracted.

5.2.2 Our Algorithm

Our problem differs from MCKP in two respects. (i) First, instead of only one optimal product, we are interested in top-*k* optimal products. (ii) Second, in addition to upper bound on cost, we also have a lower bound on it. Hence, RECMAN is a generalization of MCKP, and we adapt the above framework according to our needs. However, extension of the original algorithm to handle the top-*k* requirement (instead of top-1) is non-trivial as now we need to ensure that (i) no redundant products are formed (ii) no product among the top-*k* optimal products is missed. In particular, instead of maintaining one optimal partial product at index $[i, j]$ in table T, we maintain a list of at most k partial products.

To the best of our knowledge, this is the first work that proposes top-*k* solutions to MCKP, while the existing works only consider top-1 solution [8]. The well-known Lawler's procedure [12] could be used to obtain a top-*k* algorithm. It is a general technique to enumerate top-*k* solutions to any optimization problem, given an algorithm for the optimal solution. It's worth noting that if we use Lawler's procedure combined with top-1 algorithm for MCKP, the running time complexity would be much higher than that of our top-*k* algorithm. We will revisit it after describing our algorithm. *Thus, this is a significant algorithmic contribution in itself.*

The overall process is presented in Algorithm 1. We represent a (partial) product using a vector $\texttt{p} = \langle \texttt{attr}, \texttt{cost}, \texttt{depth}, \texttt{popularity}, \texttt{level}, \texttt{flag} \rangle$, where `p.attr` is the row index, `p.cost` is the column index and `p.depth` is the depth or rank of (partial) product p among top-*k* partial products at the corresponding index $[\texttt{p.attr}, \texttt{p.cost}]$. `p.popularity` stores the popularity of the partial product `p`, while `p.level` keeps the optimal level of the attribute `p.attr`. The variable `flag` is employed to save temporary information. We first initialize index $[0,0]$ of table T by inserting an empty product there with popularity 0 (line 1). Then, using Algorithm 2, we span the table, one row at a time, starting from index $[1,0]$ and build the list of top-*k* partial products (lines 2-4). Finally, we backtrack to get the top-*k* optimal product profiles, using Algorithm 3 (line 5). In this section, the term product profile refers to the set of attributes and the corresponding values of the product.

Algorithm 1 Overall process

```
1: Initialize [0,0] of table $T$ by inserting an empty product with popularity 0.
2: for $i \leftarrow 1 : m$ do
3:     for $j \leftarrow 0 : C_{ub}$ do
4:         Build top-$k$ partial products at [i,j] (Algorithm 2).
5: Backtrack to get back the top-$k$ product profiles (Algorithm 3).
```

We describe Algorithm 2 with the help of an example, shown in Figure 5.1, where we revisit the toy example from the introduction. Cost and popularity information for both attributes Screen Size and Screen type for category television are presented in Figure 5.1a. Figure 5.1b shows the dynamic programming table T. The cost bounds C_{lb} and C_{ub} are set to 50 and 80, respectively, while k is set to 3. Table T has $m+1=3$ rows and potentially $C_{ub}+1=81$ columns. The table is sparse and we show only the columns that have at least one filled entry. Each cell can have up to $k=3$ partial products.

Algorithm 2 Build top-*k* products for index $[i,j]$

```
1: for $depth \leftarrow 1 : k$ do
2:     Initialize a new product p with p.attr $\leftarrow i$, p.cost $\leftarrow j$ and p.depth $\leftarrow$ $depth$.
3:     $S \leftarrow \emptyset$.
4:     for each $level$ corresponding to attribute p.attr do
5:         Let $p'$ be the product such that $p'$.attr $=$ p.attr $-1$, $p'$.cost $=$ p.cost $-$ $\textsc{Cost}$(p.attr, $level$), $p'$.depth $= 1 + |\{p'' | p''$.attr $=$ p.attr, $p''$.cost $=$ p.cost, $p''$.level $= level$, $p''$.depth $<$ p.depth$\}|$.
6:         $p'$.flag $\leftarrow level$; $S \leftarrow S \cup \{p'\}$.
7:     If $S = \emptyset$, $break$.
8:     $p'_{best} \leftarrow \arg\max_{p' \in S}(p'$.popularity $+ \textsc{Popularity}$(p.attr, $p'$.flag)).
9:     p.level $\leftarrow p'_{best}$.flag.
10:    p.popularity $\leftarrow p'_{best}$.popularity $+ \textsc{Popularity}$(p.attr, p.level).
```

For each index $[i,j]$ and each value of `depth`, Algorithm 2 builds a new product `p`. It starts by initializing `p` (line 2). In S (initialized in line 3), we store

	Screen Size			Screen Type	
	Small	Medium	Large	Plasma	LCD
Cost	20	40	40	20	40
Popularity	30	70	40	30	50

(a) Cost and Popularity of various Levels of both Attributes

Attribute (Row)		Cost				
		0	20	40	60	80
Row 0	depth=1	0 : Empty	-	-	-	-
	depth=2	-	-	-	-	-
	depth=3	-	-	-	-	-
Screen Size (Row 1)	depth=1	-	30 : Small	70 : Medium	-	-
	depth=2	-	-	40 : Large	-	-
	depth=3	-	-	-	-	-
Screen Type (Row 2)	depth=1	-	-	60 : Plasma	100 : Plasma	120 : LCD
	depth=2	-	-	-	80 : LCD	90 : LCD
	depth=3	-	-	-	70 : Plasma	-

(b) Dynamic Programming Table

	Product			
	Screen Size	Screen Type	Cost	Popularity
top-1	Medium	LCD	80	120
top-2	Medium	Plasma	60	100
top-3	Large	LCD	80	90

(c) Top-3 Products

Figure 5.1: Television with two Attributes - Screen Size and Screen Type. $k = 3$, $C_{lb} = 50$ and $C_{ub} = 80$ (a) Cost and Popularity of various Levels of both Attributes (b) Dynamic Programming Table (c) Top-3 Products

the set of partial products of row $i-1$ that help us in building p. We look at each level that the product p may possibly assume for the current attribute p.attr (line 4). To calculate the optimal popularity, we look at every relevant product p$'$ from the previous row (line 5), store it in S (line 6), and later, pick the one which provides the maximum popularity (line 8). For a product p$'$ to be relevant, clearly it should belong to row p.attr $-$ 1. Moreover, its cost should be exactly p.cost $-$ $\textsc{Cost}$(p.attr, $level$) so that adding level $level$ for attribute p.attr to p$'$ would make it equivalent to p. Recall that $\textsc{Cost}$(p.attr, $level$) is the cost of the level $level$ for attribute p.attr which can be determined with a cost oracle, e.g., using linear regression. To ensure that we do not form redundant products, we take the partial product p$'$ that has yet not been combined with level $level$ to build a top-k partial product for the current index $[i, j]$. At the same time, we want p$'$ to be optimal among all partial products satisfying these conditions. Hence, p$'$.depth $=1$ plus the number of partial products that have previously been formed at [p.attr, p.cost] using level $level$ (line 5). If there exists such a product, we add it to S (line 6). In flag, we store the corresponding level (line 6). Next, the chosen level is saved in p.level (line 9) which is used in the subsequent backtracking process, discussed below. Finally, the popularity of p is calculated (line 10).

Once the table T is built, we can identify the top-k products w.r.t. maximum popularity by following Algorithm 3. We look at each product p such that p.attr $= m$ and p.cost $\in [C_{lb}, C_{ub}]$ and take the top-k among them (line 1). Note that, till now, we only have the top-k popularity (in Figure 5.1b, the top-3 popularity are 120, 100 and 90), and to get the complete product profile, we use the backtracking process. We store the product profiles in a set Q, such that each product profile q $\in Q$ contains m entries of the form $\langle a, l \rangle$, denoting the level l for attribute a for product q (shown in Figure 5.1c). Q is initially empty (line 2). We analyze one product p among top-k products at a time (line 4) and initialize an empty product profile for it (line 5). The product profile is built in lines 6-9, where the goal is to determine the levels of the attributes that were being used to form the product in context. To this end, we backtrack one row at a time and re-discover the partial products. Let p$'$ denote such a partial product. Clearly, p$'$.attr $=$ p.attr $-$ 1 as we move up, one row at a time. Recall that we store the optimal level in p.level, and hence, p$'$.cost $=$ p.cost $-$ $\textsc{Cost}$(p.level). Finally, there may be multiple partial products $\mathtt{p}'$ that satisfy these conditions. Similar to Algorithm 2, we want $\mathtt{p}'.\mathtt{depth}$ such that no redundant products are formed and, at the same time, $\mathtt{p}'$ is optimal at its depth (line 8).

Algorithm 3 Backtrack

```
1: $\mathscr{P} \leftarrow$ k-arg $\max_{\mathtt{p}|\mathtt{p.attr}=m,\mathtt{p.cost}\in[C_{lb},C_{ub}]} \mathtt{p.popularity}$.
2: $Q \leftarrow \emptyset$.
3: while $\mathscr{P} \neq \emptyset$ do
4:    $\mathtt{p} \leftarrow \mathscr{P}.pop()$.
5:    Initialize an empty product profile $\mathtt{q}$.
6:    while $\mathtt{p} \neq \emptyset$ do
7:       $\mathtt{q}.push(\langle \mathtt{p.attr}, \mathtt{p.level} \rangle)$.
8:       Let $\mathtt{p}'$ be the product such that $\mathtt{p}'.\mathtt{attr} = \mathtt{p.attr} - 1$, $\mathtt{p}'.\mathtt{cost} = \mathtt{p.cost} - \textsc{Cost}(\mathtt{p.level})$, $\mathtt{p}'.\mathtt{depth} = 1 + |\{\mathtt{p}''|\mathtt{p}''.\mathtt{attr} = \mathtt{p.attr}, \mathtt{p}''.\mathtt{cost} = \mathtt{p.cost}, \mathtt{p}''.\mathtt{level} = \mathtt{p.level}, \mathtt{p}''.\mathtt{depth} < \mathtt{p.depth}\}|$.
9:       $\mathtt{p} \leftarrow \mathtt{p}'$.
10:   $Q \leftarrow Q \cup \{\mathtt{q}\}$.
```

Note that, we can identify a position of a partial product in table T by a smaller-sized tuple $\langle \mathtt{attr}, \mathtt{cost}, \mathtt{depth} \rangle$, which contains only the first three elements of the entire product tuple. In the example, we can see that while building the partial product at position $\langle 2, 60, 2 \rangle$ (filled rectangle in Figure 5.1b), S consists of partial products at positions $\langle 1, 20, 1 \rangle$ and $\langle 1, 40, 2 \rangle$ (ovals in Figure 5.1b). Note that, S does not contain the partial product at $\langle 1, 40, 1 \rangle$ since it has already been used to construct the product at $\langle 2, 60, 1 \rangle$. The value of `flag` for partial products at $\langle 1, 20, 1 \rangle$ and $\langle 1, 40, 2 \rangle$ would correspond to levels LCD and Plasma, respectively. The level chosen by the algorithm corresponding to position $\langle 2, 60, 2 \rangle$ is LCD, because $\mathtt{p}'_{best}$ would be at position $\langle 1, 20, 1 \rangle$ (filled oval in Figure 5.1b).

We next show that our algorithm is optimal.

Theorem 2 (Optimality)**.** *The algorithm described above correctly finds the optimal solution to* RECMAN.

Proof. For brevity, we only show the proof that the popularity found by the algorithm is the correct optimal popularity. The correctness of Algorithm 3 in finding the optimal product profile is straightforward. The proof is by induction.

Base Case: The base case corresponds to row index i being 0. Index $[0,0]$ contains the empty product (with no attributes) with cost and popularity 0. Any other index $[0,j]$ for $j \in 1,\ldots,C_{ub}$ doesn't contain any product, as the cost of an empty product cannot be more than 0. This is trivially correct.

Induction Step: Assume the optimality for row $i-1$, i.e., the algorithm correctly discovers all the top-k optimal products corresponding to row $i-1$. Consider any index $[i,j]$, for any j. Let S_i^a be the set of partial products at index $[i,j]$ found by our algorithm, and S_i^* be the top-k optimal products at index $[i,j]$. Assume there exists a product $x \in S_i^* : x \notin S_i^a$, that is, x is among the top-k optimal products at index $[i,j]$, but our algorithm didn't find it. To be precise, $x.\texttt{attr} = i$, $x.\texttt{cost} = j$. Let x' be the partial product we obtain after removing attribute i from x. Then, $x'.\texttt{attr} = i-1$, $x'.\texttt{cost} = x.\texttt{cost} - \textsc{Cost}(x.\texttt{attr}, x.\texttt{level})$ and $x'.\texttt{popularity} = x.\texttt{popularity} - \textsc{Popularity}(x.\texttt{attr}, x.\texttt{level})$. Let S_{i-1}^* be the top-k optimal products at index $[i-1, x'.\texttt{cost}]$. and S_{i-1}^a be the products identified by our algorithm.

Three possible cases arise: (i) x' does not exist, in which case x does not exist (which is a contradiction). (ii) x' exists but $x' \notin S_{i-1}^*$. In this case, we can obtain k better products (in terms of popularity) than x at index $[i,j]$, by adding level $x.\texttt{level}$ for attribute i (to products in S_{i-1}^*). Hence, $x \notin S_i^*$, which is a contradiction. (iii) x' exists and $x' \in S_{i-1}^*$. In this case, because of the induction hypothesis which assumes the optimality for row $i-1$, all products in S_{i-1}^*, including x', must have been identified by our algorithm. As x' is already discovered by our algorithm, then it must have been already considered by the algorithm to build top-k products at index $[i,j]$. Since x is not found by the algorithm, for every product $\texttt{p} \in S_i^a$, we must have $\texttt{p.popularity} \geq x.\texttt{popularity}$ and $|S_i^a| = k$. Hence, x cannot be in S_i^*. Again, this is a contradiction. This completes the proof. □

Complexity of the Algorithm. It can be shown that space complexity of our algorithm is $O(C_{ub} \cdot k \cdot m)$, while time complexity is $O(C_{ub} \cdot k \cdot \sum_{i \in [1,m]} L_i)$, where L_i is the number of levels for attribute i. In comparison, if we use Lawler's procedure [12] along with the optimal (top-1) algorithm for MCKP, then space complexity would be $O(C_{ub} \cdot m)$ and time complexity would be $O(C_{ub} \cdot k \cdot (\sum_{i \in [1,m]} L_i)^2)$. Our algorithm runs faster than Lawler's procedure, but uses more memory. Hence, we provide an alternative method to find top-*k* solutions of MCKP, that trades space for time.

Chapter 6

Experiments

The goals of our experiments is two-fold. (1) Evaluating our algorithm by comparing it to other intuitive and natural heuristics. Studying the effect of parameter k on the popularity achieved and, finally, showing the scalability of our algorithm, by running it on a larger (synthetic) dataset (Section 6.1), and (2) Reporting the configuration of the best product generated by our algorithm (Section 6.2).

6.1 Evaluating our Algorithm

6.1.1 Comparison of Algorithms w.r.t. Popularity

In Theorem 2, we proved that our algorithm outputs the top-k optimal product recommendations. Here, we compare the algorithm with other natural heuristics, empirically. Before we study the effect of k, we compare the following algorithms and heuristics, for the case when $k = 1$. Recall that our algorithm is exact and hence, we expect the popularity achieved from it to be an upper bound. The aim of this comparison is to look at the difference in popularity achieved by the exact algorithm and various heuristics. We use `attr(p)` to denote the level assumed by the product `p` for the attribute `attr`. We let E denote the set of existing products in a given data set.

OPTIMAL: This is our algorithm as described in Section 5.2. Note that costs (linear regression coefficients) of individual levels in general are real-valued. We round them off to turn them into integers with no significant loss of accuracy in modeling the cost of product.

RANDOM: We generate 100,000 random products by randomly selecting a level corresponding to each attribute. Based on its cost, each product is binned into one of the cost ranges and average popularity is calculated for each bin. Note that the popularity returned might not correspond to an actual product, rather it is the popularity that one can expect when a product is chosen at random. We take this as a baseline.

EXISTING: Given the cost bounds, this method selects an *existing* product in the data set that has cost within the bounds and has maximum popularity. That is, a "new" product ρ is chosen such that

$$\rho = \underset{\mathtt{p} \in E : \textsc{Cost}(\mathtt{p}) \in [C_{lb}, C_{ub}]}{\arg\max} \textsc{Popularity}(\mathtt{p})$$

The idea is to see how the optimal product found by our algorithm compares in terms of popularity with the best existing product whose cost falls in the range $[C_{lb}, C_{ub}]$.

GREEDY: This is a relatively sophisticated heuristic. For each attribute `attr`, a level is selected such that

$$\mathtt{attr}(\rho) = \underset{\mathtt{level}}{\arg\min}\, \textsc{Cost}(\mathtt{attr}, \mathtt{level})$$

For all levels l (except $\mathtt{attr}(\rho)$) corresponding to attribute `attr`, the *efficiency* is defined as Efficiency$(\mathtt{attr}, l) =$

$$\frac{\textsc{Popularity}(\mathtt{attr}, l) - \textsc{Popularity}(\mathtt{attr}, \mathtt{attr}(\rho))}{\textsc{Cost}(\mathtt{attr}, l) - \textsc{Cost}(\mathtt{attr}, \mathtt{attr}(\rho))}$$

At any step, $\mathtt{attr}(\rho)$ is replaced by another level l corresponding to `attr` that has the current maximum efficiency. Further, efficiency of the remaining levels of `attr` is updated. This is done until $\textsc{Cost}(\rho)$ does not exceed C_{ub}. Popularity for all products which have cost within the cost range are noted and the maximum popularity becomes the greedy solution. It may happen that during this process, COST(ρ) jumps directly from below C_{lb} to above C_{ub}. In this case, instead of maximum efficiency, the level corresponding to lower efficiencies are considered until a product is formed within the cost range or no unseen level is left. In the latter case, greedy heuristic returns the null product.

In all experiments, we set the size of the cost range to be $C_{ub} - C_{lb} = 100$, and C_{ub} is the cost shown on the plots. Work done would be almost the same even if the size of cost range were varied for a given value of the cost upper bound.

In Figure 6.1, Figure 6.2 and Figure 6.3, we compare various methods described above w.r.t. popularity achieved, where the popularity are predicted using linear regression, for television, camera and laptop, respectively. The results show that our OPTIMAL algorithm beats other algorithms, including GREEDY, by a comfortable margin, on all 3 data sets. For instance, in television category (Figure 6.1), for the cost bound of 1900-2000, OPTIMAL achieves the popularity of 174, an improvement by 30% over GREEDY, which achieves 134. For the same cost bound, EXISTING and RANDOM achieve the popularity of 67 and 96, respectively. Hence, for the same cost bound, there is significant room for improvement in the products currently existing in the market. Note that, at certain cost bounds, popularity achieved by GREEDY is the same as popularity achieved by OPTIMAL. Similarly, in camera category (Figure 6.2), for the cost bound of 1900-2000, while the popularity achieved using OPTIMAL algorithm is 50, it is 37 from using the GREEDY algorithm. Interestingly, for laptop (Figure 6.3), we observe that GREEDY performs very well, almost as good as OPTIMAL, for most cost bounds. Overall, OPTIMAL outperforms other methods and GREEDY is the next choice.

6.1.2 Variation in Popularity of Top-*k* Products

Figure 6.4 shows the variation in the popularity of top-*k* products, for different values of cost (i.e., C_{ub}), in category television. As expected, the popularity decreases as the value of k increases. Interestingly, the decrease in popularity is slow. For instance, when the cost is 1500, the popularity of the top-1 product is 185 while the popularity of 100th product is 162, i.e. the average decrease in popularity is only 0.23 for unit increase in k. This observation suggests that there is not much of a difference in popularity among top products (in the real data sets we tested), strengthening our hypothesis that it is important to recommend multiple products to the marketing decision maker as she can look at variety of products, all with high popularity, and choose one from among them.

6.1.3 Scalability

The three key factors affecting the running time and space complexity of our algorithm are: Cost upper bound C_{ub}, number of products returned k and the number of levels in each attribute. Note, in Figure 6.2, the maximum possible cost of any camera is 2700, which is not representative of all real-world products. Thus, to cover most actual products in real life, we create a SYNTHETIC dataset by scaling the costs in category camera (camera is the biggest of all three real data sets). The scaling is done so that the sum of the cost coefficients of the maximum cost levels of each attribute becomes 100,000, i.e., the maximum cost of possible products is 100,000, which, we believe, covers the cost of most actual products in real life. We also add dummy attributes-level pairs to double the number of attribute-level pairs in the original camera data set. This is done to show the scalability of the algorithm w.r.t. number of feasible products. We show results in Figure 6.5 and Figure 6.6. Figure 6.5 shows the results for five different costs (C_{ub}) between 0 and 100,000, at equal intervals, and three different values of k (note the two y-axes in the figure). We vary k over 1, 10 and 100. It is clear that the algorithm is scalable both in running time (Figure 6.5a) and memory (Figure 6.5b). For example, for $C_{ub} = 50,000$, the time taken for k=1, 10 and 100 is just 5, 6 and 13 seconds, respectively. Similarly, the memory used is 0.5, 0.8 and 3.0 GB, respectively. Thus, the increase in time and memory is *not* huge even when the number of products recommended is increased by orders of magnitude.

In Figure 6.6, we plot the time and memory performance against the number of feasible products. The plot has two Y-axes, on the left we have running time and on the right, we have memory usage. We fixed $C_{ub} = 100,000$ and $k = 100$. To vary the number of feasible products, we remove ten attributes at a time from the SYNTHETIC data set. As we can see, the algorithm scales well with respect to number of feasible products and finishes within 25 seconds using only 5 GB memory even when the number of feasible products is 10^{25}.

To summarize, OPTIMAL algorithm, as expected, provides the upper bound on the popularity that can be achieved by any algorithm, given the cost bound. More interestingly, there is considerable difference in the popularity achieved by OPTIMAL and all other heuristics. Not only it is optimal, it is scalable, w.r.t. both running time and memory usage. Finally, our experiments show that the top-*k* products for *k* up to 100 have close value for popularity. Thus, recommending multiple top products, instead of just one, is important as it provides flexibility to the manufacturer to select the product they'd like to launch based on external considerations.

6.2 Recommended Product Designs

In Table 6.1, we report the top-10 attributes (w.r.t. popularity) and the levels for the best product returned by OPTIMAL in categories television and camera, respectively. The cost bound has been fixed to 900-1000. These results would convey various interesting insights to the manufacturer, e.g., refresh rate of most popular television is 120 Hz, as opposed to other higher refresh rates. Similarly, televisions with LCD flat-panels are more popular than LED flat-panels. In the camera category, cameras with memory stick and CMOS sensor type are more popular than cameras with built-in memory and CCD sensor type. This is especially enabled by the fact that our algorithm is highly scalable, allowing the manufacturer to vary the cost bounds at will and seeing what kind of recommendations come out.

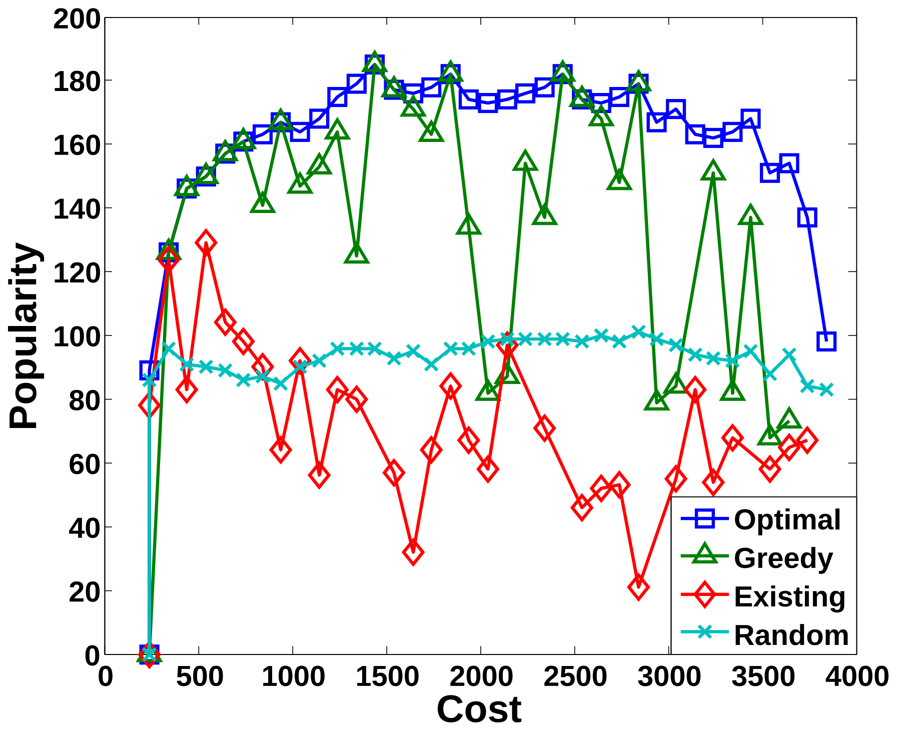


Figure 6.1: Comparison of Algorithms w.r.t. Popularity in Television Category

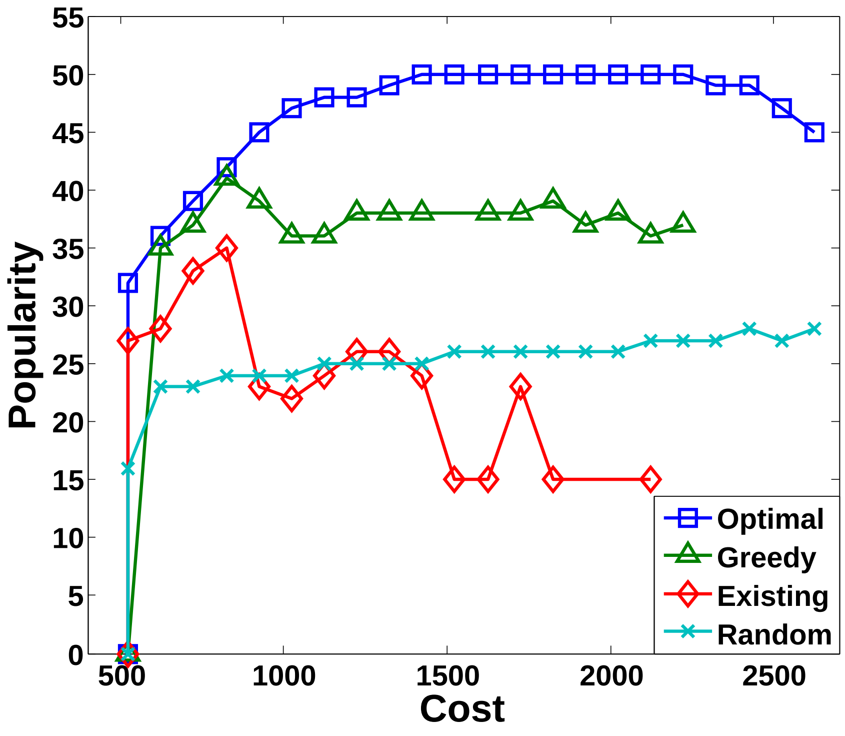


Figure 6.2: Comparison of Algorithms w.r.t. Popularity in Camera Category

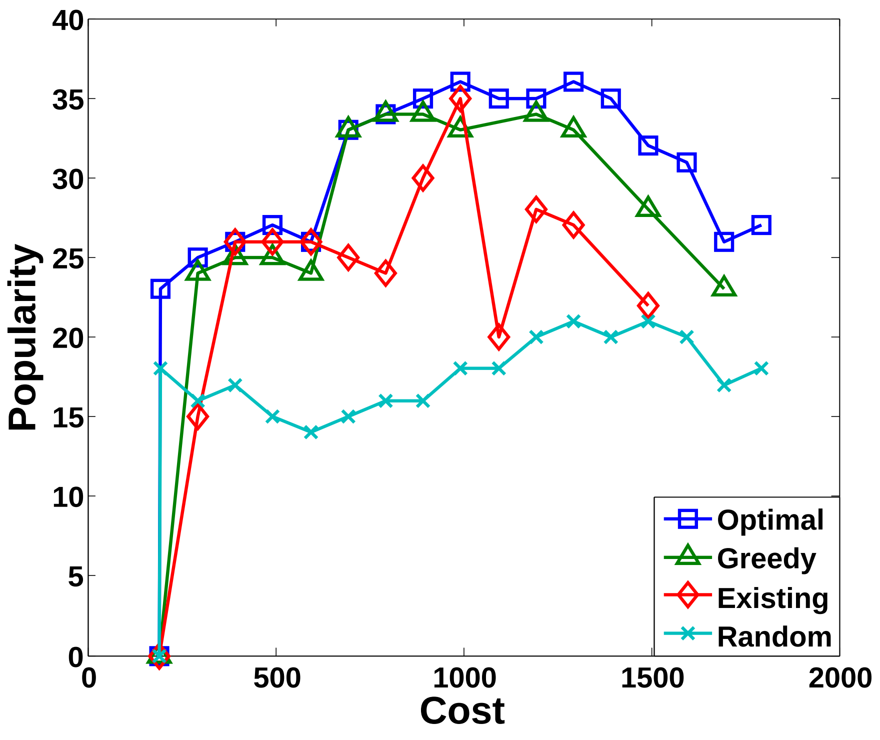


Figure 6.3: Comparison of Algorithms w.r.t. Popularity in Laptop Category

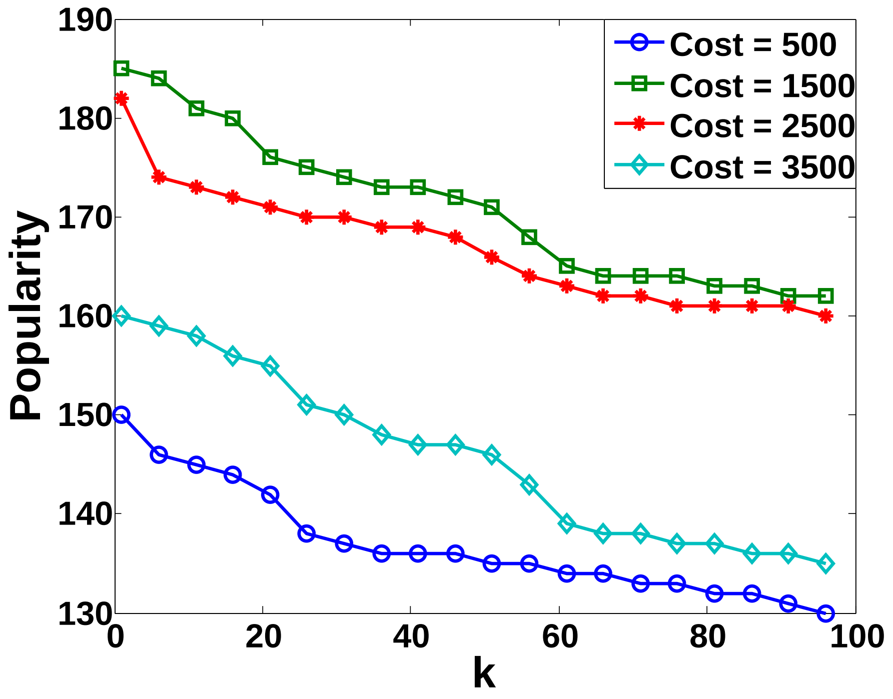
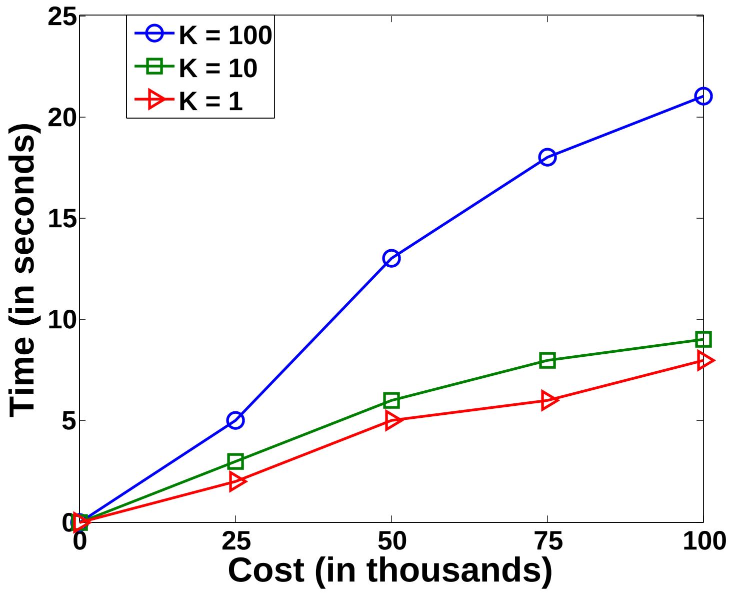
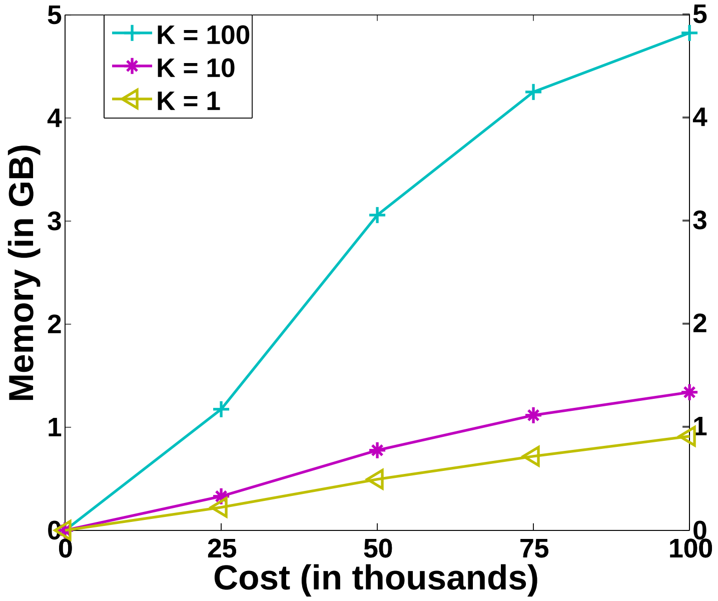


Figure 6.4: Popularity of Top-100 Products for four different Costs in Television Category



(a) Time



(b) Memory

Figure 6.5: Running Time and Memory Usage with respect to Cost Upper Bound and *k* on SYNTHETIC Data Set (a) Time (b) Memory

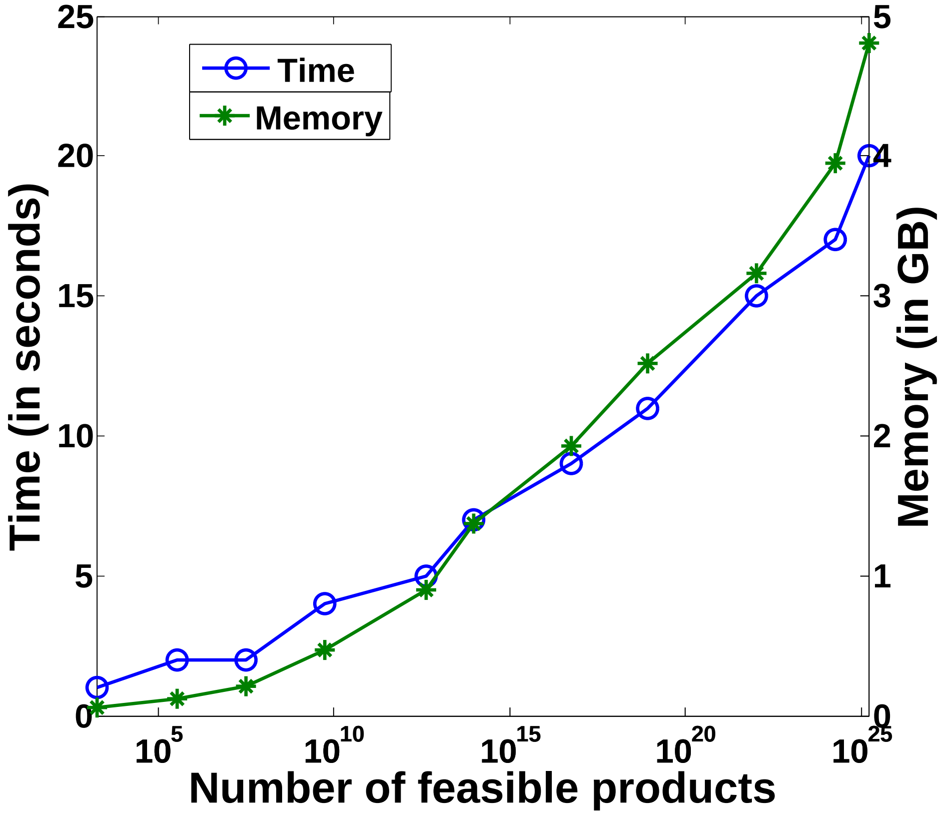


Figure 6.6: Running Time and Memory Usage with respect to Number of Feasible Products on SYNTHETIC Data Set. X-axis is Logarithmic.

Attribute	Levels
Brand	Dynex, Sony, Insignia, LG, Samsung
Refresh rate	120 Hz, 60 Hz, 240 Hz, 600 Hz
Dollar savings*	Medium, Low, High
#HDMI inputs	2, 3, 4
Television type	LCD flat-panel, LED flat-panel
Weight*	Small, Large
Height*	Small, Large
Mount bracket pattern	200mm X 200mm, 400mm X 400mm
Max. hori. resolution	1366 pixel, 1920 pixel
Width*	Large, Small

(a) Television

Attribute	Levels
Memory type	Memory stick, Built-in, Compact flash card type I, Compact flash card type II, IBM microdrive, SD card, SDHC card
Compression modes	Basic, Fine, Normal, Uncompressed
Resolution*	Low, Medium, High
Battery type	Lithium, 2XAA, Li-lion
Max. movie length*	Short, Long
LCD screen size*	Small, Medium, Large
Max. image resolution*	Medium, Low, High
Height*	Small, Medium, Large
16 MB Memory Card	Present, Absent
Image sensor type	CMOS, CCD

(b) Camera

Table 6.1: Top 10 Attributes (in decreasing order of Popularity) of the Best Product (a) Television (b) Camera. Right columns show all possible Levels for the particular Attribute and Level present in the Best Product is underlined. *denote Numerical Attributes.

Chapter 7

Conclusions and Future Work

Previous RS research almost exclusively focuses on generating recommendations to customers. In this thesis, we direct the attention of the community to an important third entity type of this ecosystem – the product manufacturers, and propose a novel problem from their business perspective: new product recommendations for manufacturers. Our problem is to recommend top-*k* new products that maximize the popularity while satisfying the cost constraints given by the manufacturer. We explored various regression models to predict cost and popularity of product. We show that the problem is NP-hard and develop a scalable pseudopolynomial time algorithm, by exploiting the connection to MCKP. We perform a comprehensive empirical analysis on 3 real data sets, and show that our exact algorithm outperforms various natural heuristics by a significant margin in terms of the popularity achieved, while remaining highly scalable.

This work opens up a rich area for future research. Below, we briefly discuss some interesting directions.

Alternative Objectives. We studied RECMAN with the objective of maximizing popularity, expressed in terms of number of ratings in the data sets we could gather. If data sets with page views or store sales information could be made available, they will be valuable in further validating our research. Indeed, alternative objectives can be formulated and optimized in place of popularity: e.g., maximizing the number of satisfied users (say as measured by the ratings/reviews provided) or maximizing the company's profit [20], given manufacturer's profit margins. Our framework is general enough to accommodate the optimization of different objective functions, with minimal changes to the approach.

Constraints. Not every attribute that is a strong predictor of the popularity or cost of a product is necessarily actionable for a manufacturer. E.g., we found that `brand` is one of the most important attributes in predicting popularity. But a manufacturer cannot just assume a top-selling brand, rather this really implies building a good reputation for your brand is important. Similarly, real-world product design often has technical feasibility constraints: e.g., a very light-weight laptop with a DVD-ROM and very large storage (hard disk) may not be feasible. It is important to extend our product recommendation algorithm for manufacturers to incorporate such constraints. The aim of this thesis is not to replace the manufacturers, or their engineering design experts, rather the idea is to recommend promising candidates for new product designs to the manufacturers, whose expertise is still required for postprocessing the set of designs recommended. As an example of this postprocessing, keeping abreast of the technological advances in the field is essential for ultimately deciding on the new product(s) that a manufacturer may want to launch.

Cost and Popularity Oracles. We presented some approaches for building cost and popularity oracles in Chapter 4. There is considerable room for improvement in their prediction accuracy. Part of the challenge in improving the accuracy is the availability of relevant data. It would be desirable to include additional information such as quality of implementation of various attributes, marketing, overall consumer experience, etc. as input to the cost/popularity prediction oracles. Further, it might be a good idea to normalize the popularity of a product based on the length of time it has been present in the market, to compensate for the fact that products that have been in the market longer have an unfair advantage over recent products, in terms of their popularity achieved.

Bibliography

[1] N. Archak, A. Ghose, and P. G. Ipeirotis. Show me the money!: deriving the pricing power of product features by mining consumer reviews. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 56–65, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-609-7. doi:10.1145/1281192.1281202. URL http://doi.acm.org/10.1145/1281192.1281202. → pages 6

[2] M. Das, G. Das, and V. Hristidis. Leveraging collaborative tagging for web item design. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 538–546, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7. doi:10.1145/2020408.2020493. URL http://doi.acm.org/10.1145/2020408.2020493. → pages 6

[3] P. Green, A. Krieger, and Y. Wind. Thirty years of conjoint analysis: Reflections and prospects. In Y. Wind and P. Green, editors, *Marketing Research and Modeling: Progress and Prospects*, volume 14 of *International Series in Quantitative Marketing*, pages 117–139. Springer US, 2004. ISBN 978-0-387-24308-5. doi:10.1007/978-0-387-28692-1_6. URL http://dx.doi.org/10.1007/978-0-387-28692-1_6. → pages 5

[4] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. 2001. → pages 13

[5] J. Hauser and V. Rao. Conjoint analysis, related modeling, and applications. In Y. Wind and P. Green, editors, *Marketing Research and Modeling: Progress and Prospects*, volume 14 of *International Series in Quantitative Marketing*, pages 141–168. Springer US, 2004. ISBN 978-0-387-24308-5. doi:10.1007/978-0-387-28692-1_7. URL http://dx.doi.org/10.1007/978-0-387-28692-1_7. → pages 3

[6] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi:10.1145/1014052.1014073. URL http://doi.acm.org/10.1145/1014052.1014073. → pages 6

[7] N. Jindal and B. Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 219–230, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-927-2. doi:10.1145/1341531.1341560. URL http://doi.acm.org/10.1145/1341531.1341560. → pages 9

[8] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer, 2004. ISBN 978-3-540-40286-2. → pages 20, 21, 22

[9] R. Kohli and R. Krishnamurti. Optimal product design using conjoint analysis: Computational complexity and algorithms. *Eur. Journal of Operational Research*, 40(2):186 – 195, 1989. ISSN 0377-2217. doi:10.1016/0377-2217(89)90329-9. URL http://www.sciencedirect.com/science/article/pii/0377221789903299. → pages 5

[10] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8), 2009. ISSN 0018-9162. doi:10.1109/MC.2009.263. URL http://dx.doi.org/10.1109/MC.2009.263. → pages 9

[11] P. Kotler, G. Armstrong, V. Wong, and J. Saunders. *Principles of Marketing*. Financial Times Prentice Hall, 2008. ISBN 9780273711568. → pages 2

[12] E. L. Lawler. A procedure for computing the *k* best solutions to discrete optimization problems and its application to the shortest path problem. *Manage. Sci.*, 18(7):pp. 401–405, 1972. ISSN 00251909. URL http://www.jstor.org/stable/2629357. → pages 22, 27

[13] B. Liu. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012. → pages 6

[14] M. Miah, G. Das, V. Hristidis, and H. Mannila. Determining attributes to maximize visibility of objects. *IEEE Trans. on Knowl. and Data Eng.*, 21

(7):959–973, July 2009. ISSN 1041-4347. doi:10.1109/TKDE.2009.72. URL http://dx.doi.org/10.1109/TKDE.2009.72. → pages 7

[15] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, 2005. ISSN 0162-8828. doi:10.1109/TPAMI.2005.159. → pages 13

[16] A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 339–346, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi:10.3115/1220575.1220618. URL http://dx.doi.org/10.3115/1220575.1220618. → pages 6

[17] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011. ISBN 978-0-387-85819-7. → pages 1, 5

[18] C. Schön. On the optimal product line selection problem with price discrimination. *Manage. Sci.*, 56(5):896–902, May 2010. ISSN 0025-1909. doi:10.1287/mnsc.1100.1160. URL http://dx.doi.org/10.1287/mnsc.1100.1160. → pages 3

[19] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. ISBN 0321321367. → pages 12, 14

[20] S. Tsafarakis and N. Matsatsinis. Designing optimal products: Algorithms and systems. In J. Casillas and F. Martnez-Lpez, editors, *Marketing Intelligent Systems Using Soft Computing*, volume 258 of *Studies in Fuzziness and Soft Computing*, pages 295–336. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15605-2. doi:10.1007/978-3-642-15606-9_19. URL http://dx.doi.org/10.1007/978-3-642-15606-9_19. → pages 5, 41

[21] X. J. Wang, J. D. Camm, and D. J. Curry. A branch-and-price approach to the share-of-choice product line design problem. *Manage. Sci.*, 55(10): 1718–1728, Oct. 2009. ISSN 0025-1909. doi:10.1287/mnsc.1090.1058. URL http://dx.doi.org/10.1287/mnsc.1090.1058. → pages 3

[22] S. Xie, G. Wang, S. Lin, and P. S. Yu. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages

823–831, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1462-6.
doi:10.1145/2339530.2339662. URL
http://doi.acm.org/10.1145/2339530.2339662. → pages 9