A Novel Predictive Modeling Framework: Combining Association Rule Discovery With EM Algorithm

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Zhonghua Jiang

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

George Karypis

February, 2013

Acknowledgements

This thesis could not have been accomplished without the guidance, support, love and friendship of many people. My deepest thanks to my wife Qian Dong, whose love, support, and sacrifice made this work possible. I would like to thank my son, Karson, and my daughter, Kyla. They are my source of courage and power in the face of difficulties. I am indebted to my parents in law, Mr. Jiachun Dong and Mrs. Chuanzhi Sheng, who helped us to get through difficult times. I would also like to thank my parents, Mr. Songhe Wu and Mrs. Jinlan Lu. Thank you, Mom and Dad, for your silent support, so I can fly freely.

I have been very fortunate to have Professor George Karypis as my mentor and academic advisor. Professor George Karypis gave me an open and encouraging working environment in which I could explore various research opportunities. In the same time, he kept a close watch on my research activities to ensure my research is in the right direction. George, your guidance, support, patient and encouragement will always be remembered and appreciated. I am indebted to Professor George Karypis and Professor Clay Carter for their financial support through my graduate study. I would also like to thank Professor Stergios Roumeliotis, Professor Rui Kuang, and Professor Charles Geyer for the time they spent serving on my thesis committee.

Thanks are also due to the cooperative staff at the Department of Computer Science, the Digital Technology Center, and the Minnesota Supercomputing Institute at the University of Minnesota for providing research and computing facilities.

Dedication

To my beloved wife, Qian Dong, my son, Karson, and my daughter, Kyla.

Abstract

Building predictive models and finding patterns are two fundamental problems in data mining. This thesis focuses on making contributions to these two areas.

In recent years, there have been increasing efforts to apply association rule mining to build predictive models, which have resulted in the areas of Associative Classification (AC) and Associative Regression (AR). The first major contribution of this thesis is a novel predictive modeling framework that can be applied to build both AC and AR models. The resulting classification/regression model is called ACEM/AREM. ACEM/AREM derives a set of classification/regression rules by: (i) applying an instance based approach to mine itemsets which form the rules' left hand side, and (ii) developing a probabilistic model which determines, for each mined itemset, the corresponding rule's parameters. The key contributions of ACEM/AREM include the probabilistic model that is able to capture interactions among itemsets and an expectation and maximization (EM) algorithm that is derived to learn rule parameters. The extensive experimental evaluation shows that the EM optimization can improve the predictive performance dramatically. We also show that ACEM/AREM can perform better than some of the state of the art classification/regression models.

The second major contribution of this thesis is the development of effective pruning methods that lead to efficient algorithms for two pattern mining problems. The first pattern mining problem is the instance based itemset mining of ACEM/AREM. ACEM/AREM utilizes an Instance-Based Itemset Miner (IBIMiner) algorithm to discover best itemsets for each training instance. IBIMiner incorporates various methods to bound the quality of any future extensions of the itemset under consideration. Our experiments show that these bounds allow IBIMiner to considerably prune the size of the search space. The second pattern mining problem is the extention of association rule mining to the dyadic datasets. These are the datasets where the features are naturally partitioned into two groups of distinct types. Traditional association rule mining methods employ metrics (e.g., confidence) that fail to distinguish the two types of features. We address this problem by proposing a new metric called dual-lift that captures the interaction between features. Based on that, we formulate a constraint pattern mining problem, which is solved by an efficient algorithm that pushes various constraints deeply into the rule mining process.We apply the dual-lift mining formulation to some real world applications and show some interesting results.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In many areas of science, business, and other environments, the vast amount of data needs to be turned into meaningful knowledge. Data mining and Knowledge Discovery in Databases is the process of discovering and extracting interesting and useful knowledge from the relevant sets of data. Predictive modeling and pattern mining are two important data mining tasks that are applicable to a wide range of application domains. Predictive modeling is a supervised learning task where the goal is to build models to predict class attributes (or target values) for new instances. Classification and regression are two well known predictive modeling areas. For classifications, the class attribute is a categorical variable, and for regressions, the class attribute is a continuous variable. Pattern mining is a data mining task of extracting interesting correlations, frequent patterns, associations or casual structures hidden in the data. It is an unsupervised technique, where no class attribute is involved. Association rule mining is a special type of pattern mining problems, in which a pattern is of the form of an association rule, which consists of a left hand side (LHS) feature set and a right hand side (RHS) target value(s).

This thesis makes contributions to both areas of predictive modeling and pattern mining. We address the following two problems: (i) how to build predictive models using association rules to achieve good predictive performances, and, (ii) how to develop efficient algorithms for pattern mining problems with hard constraints. The methods that we developed apply to datasets whose instances are described by a set of features that are present. Such datasets occur naturally in market basket transactions (features represent the set of products purchased) or bag-of-word modeling of documents (features correspond to the set of words in the document). We will refer to these features as items. Note that other types of datasets can be converted to the above format via discretization techniques [1].

1.1 Associative Classification and Regression

Recent years have seen increasing efforts in applying association rule mining to build predictive models, which have resulted in the branch of Associative Classification (AC) and Associative Regression (AR) modeling. Several studies have provided empirical evidence that AC/AR models can outperform tree-based and rule-induction based models. The good performance of the AC/AR models can be attributed to the fact that, by using a bottom-up approach to rule discovery (either via frequent itemset mining or instance-based rule mining), they can discover better rules than the traditional heuristic-driven top-down approaches.

In general, an AC/AR model consists of a set of classification/regression rules. A classification/regression rule has a LHS, which is a set of features (or an itemset), and a RHS, which is a target value. The target value is either a class label or a numeric value depending on whether the task is classification or regression. A classification/regression rule can be used to predict a test instance, if the test instance contains the set of features which form the LHS of the rule. In that case, the rule simply predicts its RHS for the test instance. However, to achieve better performance, it is beneficial to combine multiple rules together for prediction. For this, each rule is associated with an importance weight, and the prediction is determined by the combination of multiple rules, where each rule is weighted by its importance weight. The process of building an AC/AR model can generally be divided into two major components. The first component is to discover a set of itemsets that form the LHS of the final rules. The second component is to learn rule parameters, e.g., the rule's importance weight and the rule's RHS for the regression problem.

In this thesis, we present a novel AC/AR modeling framework that combines the instance based approach for itemset discovery with an Expectation and Optimization (EM) [2] algorithm for learning rule parameters. We call the resulting AC and AR models as ACEM and AREM [3, 4]. The idea of the instance based itemset mining is to discover the best itemsets for each training instances and combine them together into the final set of itemsets. The advantage of this approach is to ensure that all training instances are covered by the mined itemsets, which in the mean while can be of high qualities. To learn the rule parameters, ACEM/AREM develops a probabilistic model that captures the interactions of the various itemsets. An EM optimization framework is then applied to learn rules' parameters. This is a major deviation after existing approaches that do not take into account complex interactions among itemsets. Our experimental evaluation shows that the EM optimization approach ACEM/AREM takes can greatly improve the predictive performance on almost all the datasets. In addition, we show that ACEM/AREM outperforms several state of the art classification/regression models.

1.2 Efficient Algorithms For Pattern Mining

In addition to the ACEM/AREM model, the second main contribution of this thesis is that of developing efficient pattern mining algorithms for problems with hard constraints. We focused on two specific problems within this broad topic. The first problem is the instance based itemset mining of ACEM/AREM [4]. The second problem is the extension of association rule mining to the dyadic datasets [5]. We discuss the details of these two problems in the rest of this section.

1.2.1 Instance Based Itemset Mining

A key component of the AC/AR models is the approach they use for discovering a small number of high quality itemsets for building their predictive models. A straightforward way of achieving this is to employ a two-step approach, where the first step applies an efficient association itemset mining algorithm (e.g., Apriori or FP-growth) to discover the complete set of frequent itemsets, and the second step applies various itemset selection strategies (e.g., database sequential coverage or instance based approach) to select a subset of the itemsets that lead to high quality itemsets. Because the final set of selected itemsets is often orders of magnitude smaller than the initial set of itemsets, the drawback of this two-step approach is that it performs a large amount of unnecessary computations. To address this drawback, alternative approaches have been developed that discover the final set of itemsets directly. These approaches utilize various pruning strategies to reduce the part of the search space that needs to be explored, leading to considerable runtime reductions. Development of effective pruning strategies is a non-trivial task because they depend on the itemset selection criteria used by the AC/AR approaches and they often lead to itemset mining problem instances that are not downward closed.

AREM employs an instance-based approach to select the itemsets. It reduces the number of itemsets that need to be mined and then discarded by using an efficient Instance-Based Itemset Miner (IBIMiner), which directly discovers the final set of itemsets. Developing efficient itemset mining approaches for ACEM/AREM is challenging because the quality metrics that it uses to select the final set of itemsets are not downward closed. However, given the current itemset under consideration, we present novel methods for finding the upper bound of the quality of any of its future extensions. These upper bounds are used by the IBIMiner algorithm to prune the search space within a database projection-based itemset mining approach. Our experimental evaluation shows that the IBIMiner algorithm can be an order of magnitude faster than the baseline approach and that it scales linearly with the size of the database.

1.2.2 Mining Association Rules With Dyadic Interactions

The dyadic dataset is the type of data where feature variables can be naturally partitioned into two groups of distinct types. For example, the Vaccine Adverse Event Reporting System (VAERS) [6] can be viewed as a dyadic dataset for the vaccine adverse reaction detection problem. In this problem, the target variable is the adverse reactions, the first type of features is vaccines and the second type of features is the historical medical conditions, such as, medications the vaccine recipient was taking, pre-existing physician diagnosed allergies and birth defects, and any illness at the time of vaccination.

We focus on the association rule mining problem on the dyadic dataset. For this problem, an association rule consists of a LHS, which contains two types of features, and a RHS, which contains a set of target variables. The traditional association rule mining algorithm typically utilize confidence or lift to measure the quality of the rule.

However, these measures fail to distinguish the different types of features that form the rule's LHS. Instead, we propose a new measure called dual-lift that captures interactions between two feature sets. An association rule with high dual-lift can be interpreted as that the interaction between its two types of LHS features leads to the improvement of prevalence of the rule's RHS target set. Using the vaccine adverse reaction detection as an example, the dual-lift measure identifies rules such that the adverse reaction is likely to be caused by the interaction between vaccines and the patient's medical conditions. To our best knowledge, this type of rules has not been studied before. And it may provide new insights into patterns hidden within the data.

We formulate the association rule mining problem for dyadic dataset in the constraint pattern mining's framework. In addition to the minimum dual-lift constraint, which is used to ensure the quality of the rule, we apply the minimum support constraint to ensure the prevalence of the rule and the non-redundancy constraint to remove redundant rules. To solve this rule mining problem, the standard two-step approach can be applied, in which the first step is to identify all frequent itemsets, and the second step is to construct all possible association rules and remove the ones that violate the constraints. However, this approach is computationally very inefficient as it discovers a large amount of rules that are discarded later. We develop a Dual-Lift Miner (DLiftMiner) algorithm that discovers the final set of rules directly. DLiftMiner applies similar strategies as IBIMiner to prune the search space so that the running time can be reduced dramatically.

1.3 Chapter Descriptions

The remainder of this thesis is organized as follows. Chapter 2 presents the related work in this area. Then we present the proposed ACEM and AREM modeling framework and the corresponding experimental evaluations in Chapter 3. After that, in Chapter 4, we present the IBIMiner algorithm for efficiently mining the instance based itemsets of ACEM/AREM. Next in Chapter 5, we present the efficient algorithm that we develop to mine the association rules with dyadic interactions. Finally Chapter 6 concludes this thesis.

Chapter 2

Background and Related Work

2.1 Associative Classification/Regression Models

Our proposed ACEM/AREM model determines the set of rules by first mining the LHS itemsets and then learning rule parameters. This is in contrast to most state of the art AC/AR models in which rule parameters are assigned to straightforward values (e.g., confidence for classification rules and mean value for regression rules), which are computed during the itemset mining process. For these models, the itemset mining step effectively mines the set of rules used by the model, and thus can also be referred to as rule mining or rule discovery.

2.1.1 Classification/Regression Rule Mining

Most AC/AR models apply a two step approach for rule mining. The first step mines a preliminary set of rules satisfying some constraints. This step typically generates a large number of rules of various qualities. Using them directly for making prediction will have poor performances. For this, a second pruning step is applied to select a small subset of high quality rules for building the classification/regression model.

Step 1: Preliminary Rule Mining

The preliminary rule discovery step for most AC models is to mine all classification rules satisfying minimum support and minimum confidence requirements. This is usually accomplished by first finding all frequent rules, that is, rules satisfying minimum support requirement, and then removing those that violate the minimum confidence requirement. There are several methods developed in literature that are applied by AC models for finding the set of frequent rules: Apriori candidate generation[7], frequent pattern (FP) growth [8], and vertical layout intersection [9]. Apriori candidate generation is an approach that utilizes the "downward-closure" property to produce new candidate itemsets based on frequent itemsets found in the previous level. The "downward-closure" property ensures that all subsets of a frequent itemset must be frequent as well. The Classification By Association rules (CBA [10]) algorithm is one of the first AC algorithms that applies the Apriori candidate generation to find frequent rules. After its introduction, many other algorithms adopt its approach, for example, the Association Rule Categorizer By Category (ARC-BC [11]) and the Association Rule Categorizer for All Categories (ARC-AC [12]). The FP growth approach applies the divide and conquer technique for rule mining. It builds a compact representation called the frequent pattern tree (FP-tree) for the training database, in which infrequent items can be removed efficiently. At least three AC models apply FP growth for mining the set of frequent patterns: Classification based on Multiple Association Rules (CMAR [13]), Live and Let Live (L^3 [14]), and L^3_G ([15]). Vertical layout is the approach to represent the database as a group of items where each item is followed by its list of transaction ids. Supports of frequent itemsets can be computed easily as the intersection of transaction ids. The Multi-class Classification based on Association Rules (MCAR [16]) and Multi-class Multi-label Associative Classification (MMAC [17]) are two methods applying the vertical layout intersection for the rule mining.

There are another group of AC models that mine the set of classification rules satisfying a different set of criteria other than the combination of support and confidence requirements. An AC model called Classification Based on Predictive Association Rules (CPAR [18]), improves upon the FOIL ([19]) strategy for generating rules. The FOIL algorithm follows a heuristic approach which mines the set of rules sequentially. The FOIL algorithm finds the item that yields the largest FOIL-gain to be added to the current rule. Once a rule is mined, FOIL removes all training instances associated with it. CPAR differs from FOIL in that it does not remove instances associated with the rule all at once; instead, weights of instances are reduced by a multiplying factor. The Association based Decision Tree (ADT [20]) is the approach which mines the set of rules using the confidence threshold only. It explores so called "existential upward-closure" property of confidence for pruning the search space. An improved CBA model called CBA (2) [21] employs the multiple class supports by assigning a different support threshold to each class based on the class distribution.

The Regression Based on Association Rules (RBA [22]) model is the only AR model that we are aware of. The preliminary regression rule mining step for RBA is to simply apply the Apriori candidate generation method [7] to generate a set of frequent itemsets. Then, for each generated itemset, the corresponding regression rule's RHS is the mean of target variables of training instances that contain the itemset.

Step 2: Rule Pruning

Pruning is aimed to reduce the size of classifiers by removing redundant and misleading rules. The CMAR [13] algorithm adopts χ^2 testing for rule pruning in the discovery step. A number of methods apply the redundant rule pruning (CMAR [13], ADT [20], ARC-AC [12], ARC-BC [11]). A rule is redundant if there is some general rule with a higher rank. Methods CBA [10], CMAR [13], MCAR [16], MMAC [17], CBA (2) [21] and the AR model RBA [22] apply the database sequential coverage technique. In database sequential coverage, rules are first ranked. And then for each rule of the ranked list, a pass over the database is to find instances that match that rule. These matching instances are then removed. This process continues until all training instances are removed. The remaining rules are pruned. The pessimistic error estimation, which was proposed as the post-pruning for decision tree algorithms such as C4.5 and See5 [23], is also applied in AC models CBA [10], ADT [20], and CBA (2) [21]. Finally, there is a special pruning approach called lazy pruning applied in L^3 and L^3_G. The basic idea of lazy pruning is that the pruning should be limited to only "negative" rules that lead to incorrect classification. It starts with a ranked list of rules. Each training instance is taken in turn to find the first matching rule in the ranking list. Once all training instances are considered, rules wrongly classified instances are discarded and their covered instances are put into the new cycle. This process is repeated until all instances are classified correctly. Another innovation of lazy pruning is that the rules that are not used in the above process are kept instead of pruned to form a level two rule set which are used for making predictions.

Combining Steps 1 & 2: Mining Final Rule Set Directly

There is one unique category of AC/AR models that combine the preliminary rule discovery step with the pruning step. The idea is to push rule pruning into the discovery process so that the search space can be reduced dramatically and the final set of rules can be mined directly. The IBIMiner algorithm of ACEM/AREM is based on this idea. That is, it combines the frequent itemset mining together with the instance based itemset pruning. Harmony [24] is an AC model whose mining algorithm is the closest to ours, since it also takes an instance-centric view and directly mines, for each instance, one of the highest confidence classification rules. Harmony utilizes an upper bound of confidence to prune the search space. IBIMiner extends Harmony's mining algorithm in two perspectives. First, for the classification rules, IBIMiner derives a tighter bound than Harmony for pruning. Second, IBIMiner can also be applied for mining regression rules. There are a few other less relevant AC models in this category that we briefly summarize in the following. In the direct discriminative pattern mining approach (DDPMine [25]), branch and bound algorithm is developed to mine the best rule with the maximum information gain one at a time. FARMER [26] finds interesting rule groups for the microarray databases. In [27], the authors propose to discover top-k covering rule groups for each row of a gene expression dataset.

2.1.2 Making Predictions

The prediction of a test instance is based on a set of rules whose LHS itemsets are contained in the test itemset. However, instead of using all these rules, a few high quality rules are typically selected. In order to make such a selection, rules need to be ranked first. The most commonly used strategy for rule ranking by AC models is the "support, confidence and cardinality method", in which a rule is ranked higher if it has higher confidence, or higher support if confidence is the same, or smaller number of items if both confidence and support are the same. This is the strategy used in CBA

[10], CMAR [13], CPAR [18], ARC-BC [11], MCAR [16] and CBA (2) [21]. In L^3 and L^3_G, more specific rules (those with larger number of items) instead of general rules are ranked higher. ADT [20], L^3 [14], and L^3_G [15] also use the items lexicographical order as the tie breaker. In MMAC [17], the class distribution is explored as the further tie breaker for ranking. The simplest approach for making predictions is to find the rule with the highest precedence in the ranked list that covers the test instance. This single rule prediction approach is applied in CBA [10], ADT [20], L^3, L^3_G and CBA (2) [21]. Another approach that is more commonly used is to split rules matching a test instance into groups by class labels. Then compute the average score based on top rules in the ranked list for each group. The class of the group with the highest score is used as the prediction.

For the RBA [22] model, the variance of the rule is used for ranking (the smaller the variance, the higher the rank). During prediction, the top k matching rules are selected and the predicted target value is the weighted average of RHS of selected rules. Three weighting schemes are developed in RBA: (1) *equal*, where rules are equally weighted, (2) *supp*, where rules are weighted by their supports, and (3) *inv-var*, where the rule's weight is inverse proportional to its variance.

Our ACEM/AREM model applies similar prediction strategies as the above methods. The major difference is that we rank rules by their importance weights, which are determined by the probabilistic model.

2.1.3 Lazy Models

What we have discussed so far is the type of models that are Eager. They discover a small subset of classification/regression rules which are then used later for prediction. Lazy models postpone rule mining until the testing phase. They only mine the set of rules specific to each test instance. This strategy is applied in models Lazy Associative Classifier (LAC [28]) and Lazy Associative Classification using Information gain (LACI [29]).

2.2 Other Rule Based Models

One of the advantages of the association rule based predictive model is that it is descriptive enough to be easily interpreted and comprehended by end users. Tree based and rule induction based models are another two groups of descriptive models. Tree based models such as C4.5 [23] and rule induction models such as IREP ([30]) FOIL ([19]) and RIPPER ([31]) derive local sets of rules in the greedy manner. The classification and regression tree (CART) [32] partitions the input space into smaller, rectangular regions, and assigns the average of the target variables as the predicted value to each region. Cubist [33] is a rule based algorithm and fits a linear regression model to each of the regions.

Boosting [34] is a technique to build ensemble models by training each new model to emphasize the training instances that previous models misclassified. Boosted regression trees have shown to be arguably the best algorithms for web-ranking. In fact, all winning teams of the Yahoo learning to rank challenge [35] used boosted regression tree in one form or another.

2.3 Modeling & Pattern Mining For Dyadic Dataset

We have seen a few works [36, 37, 38] that focus on developing predictive models or mining interesting patterns from the dyadic dataset. For example, in [36], the authors present a model-based co-clustering algorithm that interleaves clustering and construction of predictive models to iteratively improve both cluster assignment and fit of the models. In [38], the authors mine a set of closed discriminative dyadic sequential patterns from a database of sequence pairs each belonging to one of the two classes. Note that this work does not handle a set of features for each dyadic component, and in this sense, their problem is simpler than ours.

2.4 Constraint Pattern Mining

The algorithm we develop for the association rule mining with dyadic interactions can be seen as an instance of the constraint pattern mining problem [39] [40]. This research area is different from the rule mining for AC/AR models in that the goal is to discover meaningful and complex patterns hidden in the dataset instead of building predictive models.

The key idea of the constraint pattern mining is to be able to push various constraints deep into the mining process so that they can be utilized to prune the search space. The specific strategies for pruning are dependent on the properties of constraints. Existing works in literature have categorized constraints into various categories. A constraint (e.g., minimum support constraint) is anti-monotone [7, 41] if for any pattern S not satisfying the constraint, none of the super-patterns of S can satisfy the constraint. This type of constraints satisfies the downward closure property, which defines a border, beyond which the pattern space can be pruned. Similarly, a constraint is monotone if for any pattern S satisfying the constraint, every super-pattern of S also satisfies the constraint. The first work of monotone constraint is in [42], in which Brin *et al* show an upward closure property of correlation measure: if a set of items S is correlated, so is every super set of S. This upward closure property also defines a border in the pattern space. One only needs to mine patterns on this border as any patterns beyond it satisfies the constraint automatically and thus need not be enumerated. Another category is the succinct constraint first proposed in [41]. For the succinct constraint, one can directly generate precisely all and only those patterns satisfying the constraints by using a member generating function [41], which does not require generation and testing of itemsets not satisfying the constraint. In [43], authors exploit algorithms that utilize FP-tree data structure to incorporate succinct constraints for pruning. They further categorize the succinct constraint into three subclasses. And their pruning strategies are based on the observation that for succinct constraint, items can be splitted into mandatory groups and optional groups. The convertible constraint is introduced in [44]. It is defined to be the constraints that can be converted to either anti-monotone or monotone constraints when items are sorted according to certain global order. Similar pruning strategies that have been developed for anti-monotone and monotone constraints can be adapted to the convertible constraints.

The rest of the constraints that cannot be categorized into the above categories are referred to as *tough* constraints [44]. The tough constraints require a different group of pruning strategies depending on the characteristics of specific constraints. In [45], a framework proposed for pushing tough constrains is based on the concept of finding a witness, i.e., an itemset such that, by testing whether it satisfies the constraint, one can deduce the information about properties of other itemsets, which can then be exploited to prune the search space. In [46], a new class of tough constraint called *Loose Anti-Monotone* constraint is introduced. If an itemset X with $|X| > 2$ satisfies the *Loose Anti-Monotone* constraint, there exists an item $i \in X$ such that $X - \{i\}$ also satisfies the constraint. The pruning strategy in [46] is based on the fact that a transaction can be deleted if it is not superset of at least one frequent k-itemset satisfying the constraint at any iteration $k \geq 2$. LPMiner [47] utilizes smallest valid extension (SVE) property for pruning when dealing with the constraint that the support decreases as a function of the itemset length. The SVE property is that given a particular infrequent itemset, there exists a minimum length that its super itemset must have in order to become frequent.

There exists a special type of tough constraint called boundable constraint [39], in which the pruning methods are based on finding the upper bound of the metric used in the problem. The approach we take in this thesis for pattern mining is largely based on this idea. There are a few other works which explore this same idea. For instance, in [48], authors use the chi-square as the association rule's quality metric in the hope of capturing the correlation between rule's LHS and RHS. To prune the search space, the upper bound of chi-square is derived based on convexity of chi-square function. In [49], authors focus on the problem of finding all the strongly correlated item pairs where the correlation is measured by the Pearson's correlation coefficient. To prune the search space, they find that the upper bound of Pearson's correlation has the conditional monotone property. In [50], authors mine association rules satisfying a new measure called improvement in addition to the traditional measures (i.e., support and confidence). They also develop methods to find the upper bound of improvement to reduce the search space. Similar ideas are applied in the direction of subgroup mining [51], where the tight optimistic estimates are calculated [52] [53] for some commonly used quality functions to reduce the search space. Note that the pattern mining problems we solve in this thesis use metrics (i.e., dual-lift for dyadic dataset mining and itemset quality functions for instance based mining) that are different from the above works. In addition, the bounding methods we develop are also different from the methods used in existing literature.

Chapter 3

Associative Predictive Modeling Based On EM Algorithm

3.1 Notations And Definitions

Let the data set $\mathcal{D}_0 = \{(\tau_i, y_i)|i = 1, 2, ..., N\}$ be a set of N instances. The instance (with index) i is a tuple (τ_i, y_i), where τ_i is a set of items (or, an itemset), and y_i is a target variable. The target variable is a numeric value for regression and a class label for classification. Given an itemset x, and an instance (τ_i, y_i), we say, x is contained in (τ_i, y_i), or, (τ_i, y_i) contains x, if $x \subseteq \tau_i$. The (absolute) support of itemset x, denoted as s_x, is the number of instances in $\mathcal{D}_0$ that contain x. The relative support of x is $s_x^r = s_x/|\mathcal{D}_0|$. Itemset x is frequent if $s_x \geq s_0$ ($s_x^r \geq s_0^r$), where $s_0 > 0$ ($s_0^r > 0$) is the user specified parameter.

For the regression problem, we define the mean (μ_x) and standard deviation (σ_x) of itemset x as computed from the set of target variables from instances in $\mathcal{D}_0$ that contain x. A regression rule is of the form $r_x : x \to \alpha_x$, where the rule's LHS x is an itemset,and the rule's RHS α_x is the target value predicted by this rule. Each regression rule is associated with a numeric weight w_x, which is used as the importance weight when combining multiple rules together for making predictions.

A classification rule is of the form $x \to y$, where the rule's LHS x is an itemset, and the rule's RHS y is a class label predicted by this rule. Each classification rule's LHS itemset x is associated with a numeric importance weight w_x. Each classification rule $x \rightarrow y$ is associated with a quality metric $q_{x\rightarrow y}$. The rule's confidence, denoted as conf$(x \rightarrow y)$, is defined as the ratio of the number of instances that contain both x and y to the number of instances that contain x only.

3.2 AREM: An Associative Regression Model

Our proposed AREM model consists of a set of regression rules $\mathcal{R} = \{r_x : x \rightarrow \alpha_x\}$. Each regression rule in set $\mathcal{R}$ is also associated with an importance weight w_x. Given a test instance with itemset τ, AREM predicts its target variable $\hat{y}$ as follows. First, it identifies the set of rules $\mathcal{R}_\tau = \{r_{x_1}, \ldots, r_{x_m}\} \subseteq \mathcal{R}$ whose LHS are subsets of τ (i.e., $(x_i \rightarrow \alpha_{x_i}) \in \mathcal{R}_\tau$ if $x_i \subseteq \tau$), then it eliminates from $\mathcal{R}_\tau$ all but the k rules that have the highest w_{x_i} values among them. This set of rules, denoted by $\mathcal{R}_\tau^k$, is then used to predict the target variable using the formula

$$\hat{y} = \frac{\sum_{r_{x_i} \in \mathcal{R}_\tau^k} w_{x_i} \alpha_{x_i}}{\sum_{r_{x_i} \in \mathcal{R}_\tau^k} w_{x_i}}, \tag{3.1}$$

which is nothing more than the average of the RHS of the k rules weighted by their corresponding w_{x_i} values. To handle the case that the test itemset τ may not contain any rule's LHS from $\mathcal{R}$, we insert the empty rule $\emptyset \rightarrow \alpha_\emptyset$ into $\mathcal{R}$. This rule's RHS and weight are learned together with the rest of the rules within our modeling framework.

The model training of AREM is the process of deriving the set of rules in $\mathcal{R}$. It consists of two major components: (i) how to discover the set of itemsets $\mathcal{X}$ which form the LHS of rules in $\mathcal{R}$ (i.e., $\mathcal{X} = \{x | r_x \in \mathcal{R}\}$), and (ii) how to assign for each itemset $x \in \mathcal{X}$ two numeric values α_x and w_x to form the rule $x \rightarrow \alpha_x$ with its importance weight w_x. We explain these two components in the remaining of this section.

3.2.1 Instance Based Itemset Mining For AREM

Our approach for itemset discovery is to require the itemsets in $\mathcal{X}$ to satisfy the following three constraints. First, to ensure the prevalence of the itemset $x \in \mathcal{X}$, the number of instances containing x should be larger than some threshold value s_0. In other words, we require each itemset $x \in \mathcal{X}$ to be frequent, satisfying $s_x \geq s_0$. Second, the training instances should be fully covered by the itemsets in $\mathcal{X}$, that is, for each training instance, we should be able to find a non-empty subset of itemsets from $\mathcal{X}$ that are contained in that instance. Third, we prefer high quality itemsets that are likely to achieve better predictive performances, when they are used as the LHS of the set of rules in $\mathcal{R}$.

We utilize the instance based approach for itemset discovery to address these constraints. The basic idea is to discover a set of high quality frequent itemsets for each instance and then combine them together into the final set of itemsets. Specifically, denote $\mathcal{F}$ be the complete set of frequent itemsets such that for any $x \in \mathcal{F}$, we have $s_x \geq s_0$. For each training instance i, let $\mathcal{F}_i \subseteq \mathcal{F}$ be the set of frequent itemsets which are contained in that instance, i.e., $x \in \mathcal{F}_i$ if $x \subseteq \tau_i$. Our approach is that for each instance i, we identify K itemsets from $\mathcal{F}_i$ to form a new set $\mathcal{X}_i \subseteq \mathcal{F}_i$, such that combining them together gives the final set of itemsets, i.e., $\mathcal{X} = \cup_i \mathcal{X}_i$. In this way, the above first and second constraints are satisfied. To satisfy the third constraint, we need to make sure that the itemsets in $\mathcal{X}_i$ should have higher qualities than other itemsets in $\mathcal{F}_i$. For this, we need a way to evaluate the qualities of itemsets in $\mathcal{F}_i$. This will be an instance based quality metric that measures the quality of an itemset $x \in \mathcal{F}_i$ from instance i's perspective. We assume the quality metric is of the following form $Q(x, y_i)$ so that its dependency on instance i is captured by the target variable y_i. If we are given such a quality function, we sort itemsets $x \in \mathcal{F}_i$ by $Q(x, y_i)$ from large to small and then pick the top K itemsets to form $\mathcal{X}_i$.

We derive three quality functions based on the itemset x and target variable y_i. The first quality function is based on the intuition that an itemset is good if the set of training instances containing it have consistent target values. Or, equivalently, we prefer the itemset x whose standard deviation σ_x is small. We denote this quality function as Q^σ, which is formally defined as

$$Q^\sigma(x, y_i) = \frac{1}{\sigma_x}. \tag{3.2}$$

Note that we take the inversion so that we can uniformly treat itemsets with higher qualities as being better. However, the function Q^σ has the limitation that it does not capture the instance's target variable y_i, which can lead to the sub-optimal set of itemsets. We propose two additional quality functions to address this limitation.

We derive the second quality function by considering the itemset $x \in \mathcal{F}_i$ to have the high quality if the instances which contain x are close to the instance i. We measure the closeness between instances j and i as $(y_j - y_i)^2$. Averaging this quantity for all instances j which contain x gives the Mean Squared Error (MSE):

$$MSE(x, y_i) = \frac{1}{s_x} \sum_{x \subseteq \tau_j} (y_j - y_i)^2 = (\mu_x - y_i)^2 + \sigma_x^2. \tag{3.3}$$

So, we propose the following:

$$Q^m(x, y_i) = \frac{1}{\sqrt{MSE(x, y_i)}} = \frac{1}{\sqrt{(\mu_x - y_i)^2 + \sigma_x^2}}. \tag{3.4}$$

The third quality function we derive is from the probabilistic point of view. We treat each itemset x as a probabilistic model with parameters μ_x and σ_x. This model can be used to generate the target variable y_i. Given y_i, we can treat the probability of y_i being generated by the model x as the itemset x's quality metric from the instance i's perspective. That is, the higher the probability of y_i is, the better quality x has. The simplest probabilistic model of x is the Normal distribution $\mathcal{N}(y_i|\mu_x, \sigma_x^2)$. So we propose the quality function based on $\mathcal{N}(y_i|\mu_x, \sigma_x^2)$ as

$$Q^n(x, y_i) = \frac{1}{\sigma_x} e^{-\frac{(\mu_x - y_i)^2}{2\sigma_x^2}}, \tag{3.5}$$

where we have dropped the constant term $\sqrt{2\pi}$.

Once the quality function Q is chosen to be one of the three candidates: Q^σ, Q^m and Q^n, and the two parameters s_0 and K are specified, our instance based approach defines the set of itemsets in $\mathcal{X}$. We leave the discussion of how to efficiently mine this set of itemsets in Chapter 4.

3.2.2 The Probabilistic Model For Determining α_x And w_x

We address the problem of how to determine α_x and w_x for each itemset $x \in \mathcal{X}$ to form the regression rule $(x \to \alpha_x) \in \mathcal{R}$ with weight w_x. To determine the rule's RHS, a straightforward approach is to use the mean value μ_x of the itemset x. However, a closer investigation reveals that this approach has the drawback of ignoring the potentially important itemset-itemset interactions. Indeed, when computing μ_x for $x \in \mathcal{X}_i$, the target variable y_i is used directly which implicitly assumes that y_i is fully contributed by x while ignoring the effects of other itemsets in $\mathcal{X}_i$ on y_i. To determine the rule's weight, the simplest approach is to assign $w_x = 1$ so that rules are equally weighted during prediction. Other approaches as adopted by RBA [22] include $w_x = s_x$ and $w_x = 1/\sigma_x^2$. However, these approaches have the drawback of being completely decoupled from the method for determining α_x. We propose a probabilistic model to address the above drawbacks. In this model, the target variable y_i is the combined contribution from all itemsets in $\mathcal{X}_i$, so that the itemset-itemset interactions are taken into account. Moreover, we treat α_x and w_x as model parameters which are then learned consistently in a unified framework. We elaborate the details of the probabilistic model in the following.

Consider an arbitrary training instance (τ, y). The goal of the probabilistic model is to specify the probability of target variable y given τ, i.e., $P[y|\tau]$. We want to relate this quantity to the set of itemsets in $\mathcal{X}$. To this end, we treat itemset x as a random variable that takes values in $\mathcal{X}$ and write $P[y|\tau]$ as

$$P[y|\tau] = \sum_x P[y, x|\tau] = \sum_x P[y|\tau, x]P[x|\tau],$$

where $P[y|\tau, x]$ is the probability of generating the target variable y given τ and x, which is generated from τ with probability $P[x|\tau]$. Our goal then becomes to specify $P[y|\tau, x]$ and $P[x|\tau]$ and relate them to α_x and w_x.

In order to specify $P[y|\tau, x]$, we first assume the conditional independency $P[y|\tau, x] = P[y|x]$. This is, we assume that once the itemset x is known, the probability of y is not dependent on τ, which simplifies our model so that the dependency of τ is fully captured in $P[x|\tau]$. Given that, we then model $P[y|x]$ as a Normal distribution whose mean is the RHS of the rule $x \to \alpha_x$ and standard deviation is β_x. That is,

$$P[y|x] = \mathcal{N}(y|\alpha_x, \beta_x^2). \tag{3.6}$$

Next, we specify $P[x|\tau]$ by considering how AREM makes predictions. In order to simplify this discussion we ignore the fact that AREM picks the top k rules (i.e., it uses the set of rules in $\mathcal{R}_\tau^k$) and assume that it predicts the target value by using all the rules in $\mathcal{R}_\tau$. Specifically, Equation 3.1 now becomes

$$\hat{y} = \frac{\sum_{r_{x_i} \in \mathcal{R}_\tau} w_{x_i} \alpha_{x_i}}{\sum_{r_{x_i} \in \mathcal{R}_\tau} w_{x_i}} = \sum_x \alpha_x \frac{I_{x \subseteq \tau} w_x}{\sum_{x' \subseteq \tau} w_{x'}}, \tag{3.7}$$

where $I_{x \subseteq \tau}$ is the indicator function which takes value 1 (0) when $x \subseteq \tau$ is true (false).

From the probabilistic modeling point of view, we predict the target variable as the expected value of y given τ, that is,

$$\hat{y} = E[y|\tau] = \sum_{x} E[y|\tau, x]P[x|\tau]. \tag{3.8}$$

From Equation 3.6, we get $E[y|\tau, x] = \alpha_x$. To specify $P[x|\tau]$, we compare Equation 3.7 with 3.8, and get

$$P[x|\tau] = \frac{I_{x \subseteq \tau} w_x}{\sum_{x' \subseteq \tau} w_{x'}}. \tag{3.9}$$

To summarize, we have reached a two step model $P[y, x|\tau] = P[y|x]P[x|\tau]$. In the first step, a regression rule's LHS $x \in \mathcal{X}$ is generated based on τ with probability $P[x|\tau]$ given by Equation 3.9. In the second step, the target variable y is generated by x with probability $P[y|x]$ given by Equation 3.6.

Our proposed probabilistic model uses α_x, β_x and w_x for $\forall x \in \mathcal{X}$ as parameters. We present how we learn these parameters in an EM framework in Section 3.2.3.

3.2.3 The EM Framework For Learning α_x and w_x

Denote by $\boldsymbol{\theta} = \{\alpha_x, \beta_x, w_x | x \in \mathcal{X}\}$ the complete set of parameters of the probabilistic model we proposed in Section 3.2.2. The maximum likelihood estimation of $\boldsymbol{\theta}$ given the training data set is to maximize

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i} \log\left(P[y_i|\tau_i, \boldsymbol{\theta}]\right) = \sum_{i} \log\left(\sum_{x_i} P[y_i, x_i|\tau_i, \boldsymbol{\theta}]\right), \tag{3.10}$$

where we have introduced x_i to denote the itemset generated by our probabilistic model for instance i. The difficulty of this optimization problem comes from the summation inside the logarithmic function. This is due to the existence of the hidden variables x_i, which are not directly observable from the training data set. We apply the EM algorithm that is presented in Chapter A to solve this problem.

EM algorithm is an iterative optimization technique. In the following, we add a subscript t to all model parameters to denote the parameters used by EM algorithm at iteration t. For each iteration t, EM algorithm finds the updated set of parameters $\boldsymbol{\theta}_{t+1}$ given the current parameter estimations $\boldsymbol{\theta}_t$. According to the Equation A.1, this can be accomplished by maximizing the function

$$\mathcal{Q}(\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t) = \sum_i \sum_{x_i} P[x_i|\tau_i, y_i, \boldsymbol{\theta}_t] \log(P[y_i, x_i|\tau_i, \boldsymbol{\theta}_{t+1}]),^{1} \tag{3.11}$$

This optimization problem is much easier than the original one for Equation 3.10, due to the fact that the logarithmic function is now inside the summation. The EM algorithm at iteration t is splitted into an E-step which computes $\pi_{i,x_i,t} = P[x_i|\tau_i, y_i, \boldsymbol{\theta}_t]$ and an M-step which optimizes $Q(\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t)$ given $\pi_{i,x_i,t}$. After each iteration, the log-likelihood function $\mathcal{L}$ is guaranteed to be increased, that is, $\mathcal{L}(\boldsymbol{\theta}_{t+1}) \geq \mathcal{L}(\boldsymbol{\theta}_t)$.

At iteration $t = 0$, we initialize the weight $w_{x,0}$ to one and $\alpha_{x,0}$, $\beta_{x,0}$ to the mean and standard deviation of x in $\mathcal{D}_0$, i.e., $w_{x,0} \leftarrow 1$, $\alpha_{x,0} \leftarrow \mu_x$ and $\beta_{x,0} \leftarrow \sigma_x$.

For the E-step, we first apply Bayes' Theorem so that

$$\pi_{i,x_i,t} = P[x_i|\tau_i, y_i, \boldsymbol{\theta}_t] = \frac{P[y_i|\tau_i, x_i, \boldsymbol{\theta}_t]P[x_i|\tau_i, \boldsymbol{\theta}_t]}{P[y_i|\tau_i, \boldsymbol{\theta}_t]} \propto P[y_i|x_i, \boldsymbol{\theta}_t]P[x_i|\tau_i, \boldsymbol{\theta}_t]. \tag{3.12}$$

According to Equations 3.9 and 3.6, we have

$$P[y_i|x_i, \boldsymbol{\theta}_t]P[x_i|\tau_i, \boldsymbol{\theta}_t] \propto \mathcal{N}(y_i|\alpha_{x_i,t}, \beta^2_{x_i,t}) w_{x_i,t} I_{x_i \subseteq \tau_i}.$$

Combining these two Equations, we get

$$\pi_{i,x_i,t} = \frac{\mathcal{N}(y_i|\alpha_{x_i,t}, \beta^2_{x_i,t}) w_{x_i,t} I_{x_i \subseteq \tau_i}}{\sum_{x' \subseteq \tau_i} \mathcal{N}(y_i|\alpha_{x',t}, \beta^2_{x',t}) w_{x',t}}. \tag{3.13}$$

For the M-step, we split $P[y_i, x_i|\tau_i, \boldsymbol{\theta}_{t+1}]$ as $P[y_i|x_i, \boldsymbol{\theta}_{t+1}]P[x_i|\tau_i, \boldsymbol{\theta}_{t+1}]$, so that $\mathcal{Q} = \mathcal{Q}_1 + \mathcal{Q}_2$, where $\mathcal{Q}_1$ contains only $\alpha_{x,t+1}$, $\beta_{x,t+1}$ and $\mathcal{Q}_2$ contains only $w_{x,t+1}$.

Next, we optimize $\mathcal{Q}_1$ which is given by

$$\mathcal{Q}_1 = \sum_i \sum_{x_i \subseteq \tau_i} \pi_{i,x_i,t} \log(P[y_i|x_i, \boldsymbol{\theta}_{t+1}]).$$

By changing the order of summation, we can write $\mathcal{Q}_1 = \sum_x \mathcal{Q}_x$, where

$$\mathcal{Q}_x = \sum_{i:x \subseteq \tau_i} \pi_{i,x,t} \log(P[y_i|x, \boldsymbol{\theta}_{t+1}]). \tag{3.14}$$

One can see that different itemsets are decoupled from each other, so we only need to solve $\mathcal{Q}_x$ for $\forall x \in \mathcal{X}$. Observe that $\mathcal{Q}_x$ is nothing but the weighted version of the

[1] The Equation 3.11 is slightly different from the Equation A.1 due to that all probabilities are conditional on τ_i in this problem.

log-likelihood function of model $P[y|x, \boldsymbol{\theta}_{t+1}] = \mathcal{N}(y|\alpha_{x,t+1}, \beta^2_{x,t+1})$, where the weights are given by $\pi_{i,x,t}$ for instance i. The solution is straightforward:

$$\alpha_{x,t+1} = \frac{\sum_{i:x\subseteq\tau_i} \pi_{i,x,t} y_i}{\sum_{i:x\subseteq\tau_i} \pi_{i,x,t}}, \tag{3.15}$$

and,

$$\beta^2_{x,t+1} = \frac{\sum_{i:x\subseteq\tau_i} \pi_{i,x,t}(y_i - \alpha_{x,t+1})^2}{\sum_{i:x\subseteq\tau_i} \pi_{i,x,t}}. \tag{3.16}$$

In Equations 3.15 and 3.16, the parameters $\alpha_{x,t+1}$ and $\beta_{x,t+1}$ are the *weighted* mean and standard deviation where the weight of instance i at iteration t is given by $\pi_{i,x,t}$. This weighting mechanism can help to remove the outlier instance whose $\pi_{i,x,t}$ is small.

Now, we optimize $\mathcal{Q}_2$ which is given by

$$\mathcal{Q}_2 = \sum_i \sum_{x_i\subseteq\tau_i} \pi_{i,x_i,t} \log(P[x_i|\tau_i, \boldsymbol{\theta}_{t+1}]). \tag{3.17}$$

By plugging Equation 3.9 into $\mathcal{Q}_2$, and taking the derivative, we get

$$\frac{\partial \mathcal{Q}_2}{\partial w_{x,t+1}} = \sum_{i:x\subseteq\tau_i} \left(\frac{\pi_{i,x,t}}{w_{x,t+1}} - \frac{1}{\sum_{x'\subseteq\tau_i} w_{x',t+1}}\right).$$

One can see that different weights $w_{x,t+1}$ are coupled in the above equation. So the exact analytical solution becomes impossible. To ensure the simplicity and computational efficiency of our approach, we make an approximation here by replacing $t+1$ by t in the second term of RHS. Then by setting the derivative to zero, we get

$$\frac{w_{x,t+1}}{w_{x,t}} = \frac{\sum_{i:x\subseteq\tau_i} \pi_{i,x,t}}{\sum_{i:x\subseteq\tau_i} \frac{w_{x,t}}{\sum_{x'\subseteq\tau_i} w_{x',t}}}. \tag{3.18}$$

From Equations 3.15, 3.16 and 3.18, we see that $\pi_{i,x}$ plays the key role of relating parameters α_x and β_x to weights w_x, so that they can interact with each other and be optimized consistently.

Finally, we note that AREM introduces a parameter M which controls the number of EM-steps. After the EM algorithm is completed, the rule's RHS and weight are finalized to be $\alpha_{x,M}$ and $w_{x,M}$.

3.3 ACEM: An Associative Classification Model

In this section, we extend the AREM modeling framework developed in Section 3.2 to solve the classification problem, which results in the Associative Classification based on EM algorithm, or, ACEM model.

Similar to AREM, the ACEM model also consists of a set of itemsets $\mathcal{X}$. For each itemset $x \in \mathcal{X}$, we create a set of classification rules $x \to y$ for each possible class label y from the training dataset. The union of all these classification rules forms the complete rule set $\mathcal{R}$, which we use for making predictions. Note that unlike the regression rule, the classification rule cannot be indexed by x because each x are assigned with multiple class labels y as the RHS. We introduce a numeric weight w_x to capture the importance of the itemset x, and another numeric parameter $q_{x\to y}$ to capture the importance of the class label y given the rule's LHS itemset x.

Given a test instance with itemset τ, ACEM predicts its target variable $\hat{y}$ as follows. First, it identifies the set of itemsets $\mathcal{X}_\tau \subseteq \mathcal{X}$ which are subsets of τ (i.e., $x \in \mathcal{X}_\tau$ if $x \subseteq \tau$), then it eliminates from $\mathcal{X}_\tau$ all but the k itemsets that have the highest weights w_x. This resulting set of itemsets, denoted by $\mathcal{X}_\tau^k$, is then used to predict the target variable. Unlike AREM, ACEM cannot use the Equation 3.1 for prediction since class labels cannot be added. The approach we take for ACEM is to compute an aggregated score for each possible class label y, and then select the one with the highest score. Formally, for each class label y, we form a set of classification rules $\{x \to y | x \in \mathcal{X}_\tau^k\} \subseteq \mathcal{R}$. Then we use the quantities $q_{x\to y}$ for these rules together with the itemset importance weights $w_x \forall x \in \mathcal{X}_\tau^k$ to derive a score for label y as

$$\frac{\sum_{x\in\mathcal{X}_\tau^k} q_{x\to y} w_x}{\sum_{x\in\mathcal{X}_\tau^k} w_x}.$$

Then, the predicted class label is given by

$$\hat{y} = \arg\max_y \frac{\sum_{x\in\mathcal{X}_\tau^k} q_{x\to y} w_x}{\sum_{x\in\mathcal{X}_\tau^k} w_x}. \tag{3.19}$$

To handle the case that the test itemset τ may not contain any itemsets from $\mathcal{X}$, we follow the same strategy as AREM and insert the empty set into $\mathcal{X}$.

The model training of ACEM is the process of deriving the set of rules in $\mathcal{R}$ with associated rule parameters being properly learned. It consists of two major components:

(i) discover the set of itemsets $\mathcal{X}$ which form the LHS of the rules in $\mathcal{R}$, and (ii) determine rule parameters for each itemset $x \in \mathcal{X}$. For itemset x, the rule parameters to be determined include w_x and $q_{x \to y}$ for each possible class label y.

3.3.1 Instance Based Itemset Mining For ACEM

ACEM applies the same instance based itemset mining approach as AREM. Specifically, let $\mathcal{F}$ be the complete set of frequent itemsets. For each training instance i, $\mathcal{F}_i \subseteq \mathcal{F}$ is the set of frequent itemsets contained in that instance. Our approach is to select K itemsets from $\mathcal{F}_i$ to form a new set $\mathcal{X}_i$, so that the final set of itemsets is $\mathcal{X} = \cup_i \mathcal{X}_i$. To derive the set $\mathcal{X}_i$, we sort itemsets in $\mathcal{F}_i$ by some quality function $Q(x, y_i)$ from large to small and then select the top K itemsets.

For the classification problem, the quality function $Q(x, y_i)$ is much simpler due to that y_i is now a class label. We choose the most common metric for evaluating the quality of the rule $x \to y_i$: confidence. Specifically, we define

$$Q^c(x, y_i) = \text{conf}(x \to y_i). \tag{3.20}$$

3.3.2 The Probabilistic Model For Determining $q_{x \to y}$ And w_x

The naive choice for $q_{x \to y}$ is the rule's confidence conf$(x \to y)$. However, just like the case of using μ_x for α_x in AREM, using conf$(x \to y)$ for $q_{x \to y}$ shares the same drawback of ignoring the potentially important itemset-itemset interactions. To address this drawback and to be able to learn w_x together with $q_{x \to y}$, we extend the probabilistic model developed in Section 3.2.2 to the classification problem.

Recall that for AREM, we derived the probability $P[y|\tau]$ for instance (τ, y) as

$$P[y|\tau] = \sum_x P[y, x|\tau] = \sum_x P[y|\tau, x]P[x|\tau] = \sum_x P[y|x]P[x|\tau],$$

where we have substituted the conditional independency assumption $P[y|x, \tau] = P[y|x]$. We use the same equation for the classification problem.

First, we need to specify the conditional probability $P[y|x]$. In fact, this is what $q_{x \to y}$ is designed for:

$$P[y|x] = q_{x \to y}. \tag{3.21}$$

Next, to specify $P[x|\tau]$, we follow the same strategy of making sure the prediction method derived from the probabilistic model is consistent with the method we presented earlier in Equation 3.19. We first ignore the fact that ACEM picks the top k itemsets and assume that it predicts the target value by using all the itemsets in $\mathcal{X}_\tau$. Equation 3.19 now becomes

$$\hat{y} = \arg\max_y \frac{\sum_{x \in \mathcal{X}_\tau} q_{x\to y} w_x}{\sum_{x \in \mathcal{X}_\tau} w_x} = \arg\max_y \sum_x q_{x\to y} \frac{I_{x\subseteq\tau} w_x}{\sum_{x'\subseteq\tau} w_{x'}}. \tag{3.22}$$

From the probabilistic model's perspective, we can predict the class label by the maximum likelihood method:

$$\hat{y} = \arg\max_y P[y|\tau] = \arg\max_y \sum_x P[y|x]P[x|\tau]. \tag{3.23}$$

From Equation 3.21, we get

$$\hat{y} = \arg\max_y P[y|\tau] = \arg\max_y \sum_x q_{x\to y} P[x|\tau]. \tag{3.24}$$

By comparing Equations 3.22 with 3.24, we find that $P[x|\tau]$ should satisfy the same formula as in Equation 3.9 for the regression problem.

3.3.3 The EM Framework For Learning $q_{x\to y}$ and w_x

For ACEM, we denote the set of model parameters as $\boldsymbol{\theta} = \{w_x, q_{x\to y} | \forall x \in \mathcal{X}, \forall y\}$. Before we proceed to the problem of optimizing Equation 3.11 for the EM iteration t, we initialize our model parameters for $t = 0$: $w_{x,0} \leftarrow 1$ and $q_{x\to y,0} \leftarrow \text{conf}(x \to y)$. The reason that we initialize $q_{x\to y}$ to be $\text{conf}(x \to y)$ is because both of these two quantities are used to model $P[y|x]$. The confidence as a simpler estimation which ignores itemset-itemset interactions should be a reasonable starting point for $q_{x\to y}$.

Observe that the only part that is different between the the probabilistic model of ACEM (Section 3.3.2) and the probabilistic model of AREM (Section 3.2.2) is the method for specifying $P[y|x]$. For AREM, it is specified by Equation 3.6 and for ACEM, it is specified by Equation 3.21. As a consequence, a large part of our EM algorithm derived for AREM in Section 3.2.3 can be used for ACEM without modifications.

For the E-step, we want to compute $\pi_{i,x_i,t}$. For this, we plug Equations 3.21 and 3.9 into the Equation 3.12 and get

$$\pi_{i,x_i,t} \propto P[y_i|x_i,\boldsymbol{\theta}_t]P[x_i|\tau_i,\boldsymbol{\theta}_t] \propto q_{x_i\to y_i,t} w_{x_i,t} I_{x_i\subseteq\tau_i}.$$

After normalization, we get

$$\pi_{i,x_i,t} = \frac{q_{x_i \to y_i,t} w_{x_i,t} I_{x_i \subseteq \tau_i}}{\sum_{x' \subseteq \tau_i} q_{x' \to y_i,t} w_{x',t}}. \tag{3.25}$$

The M-step has been splitted into two problems of optimizing $\mathcal{Q}_x$ in Equation 3.14 and $\mathcal{Q}_2$ in Equation 3.17, respectively. Optimizing $\mathcal{Q}_x$ gives us the solution of $q_{x \to y,t+1}$. And optimizing $\mathcal{Q}_2$ gives us the solution of $w_{x,t+1}$. Due to that the model specifications for $P[x|\tau]$ (Equation 3.9) are the same for ACEM and AREM models. The problems of optimizing $\mathcal{Q}_2$ are also the same, so that Equation 3.18 is also valid for the classification problem.

So we only need to solve the optimization problem for $\mathcal{Q}_x$. Plugging Equation 3.21 into the Equation 3.14 gives

$$\mathcal{Q}_x = \sum_{i:x \subseteq \tau_i} \pi_{i,x,t} \log\left(q_{x \to y_i,t+1}\right).$$

To optimize $\mathcal{Q}_x$ in the above equation, we need to take into account the constraints on $q_{x \to y,t+1}$:

$$0 \leq q_{x \to y,t+1} \leq 1, \tag{3.26}$$

and,

$$\sum_y q_{x \to y,t+1} = 1. \tag{3.27}$$

We ignore the constraint in Equation 3.26 for now. For the constraint in Equation 3.27, we introduce the Lagrangian multiplier λ_x and construct the new objective function

$$\mathcal{Q}'_x = \sum_{i:x \subseteq \tau_i} \pi_{i,x,t} \log\left(q_{x \to y_i,t+1}\right) - \lambda_x \sum_y q_{x \to y,t+1} + \lambda_x.$$

We have

$$\frac{\partial \mathcal{Q}'_x}{\partial q_{x \to y,t+1}} = \frac{\sum_{i:x \subseteq \tau_i \wedge y_i = y} \pi_{i,x,t}}{q_{x \to y,t+1}} - \lambda_x.$$

Setting the above Equation to zero gives

$$q_{x \to y,t+1} = \frac{\sum_{i:x \subseteq \tau_i \wedge y_i = y} \pi_{i,x,t}}{\lambda_x}. \tag{3.28}$$

By plugging Equation 3.28 into the Equation 3.27, we have the solution for the multiplier:

$$\lambda_x = \sum_{i:x \subseteq \tau_i} \pi_{i,x,t}. \tag{3.29}$$

Table 3.1: Data Set Summary

Data Set	# of instances	# of items	density (%)[a]	# of trials[b]
BestBuy.wdep	10k	1428	2.3	20
BestBuy.swf	10k	962	1.7	20
CitySearch.wdep	10k	1554	2.3	20
CitySearch.swf	10k	996	2.2	20
Yelp.wdep	10k	2231	2.3	20
Yelp.swf	10k	1442	2.1	20
MovieLens	10k	66	33.3	20
Airline	10k	676	1.6	20
Socmob	1156	44	11.4	200
Pollen	3848	17	23.5	50
Spacega	3107	24	25.0	60

[a] The density captures how sparse the data set is. It is the percentage of non-zero entries of the data matrix, where rows are instances and columns are items. [b] Number of trials the data set is randomized and then splitted into 80% training set, 10% validation set and 10% testing set.

Now, we get the solution for $q_{x \to y, t+1}$ by combining Equations 3.28 and 3.29:

$$q_{x \to y, t+1} = \frac{\sum_{i: x \subseteq \tau_i \wedge y_i = y} \pi_{i,x,t}}{\sum_{i: x \subseteq \tau_i} \pi_{i,x,t}}. \tag{3.30}$$

Note that by Equation 3.30, the constraint in Equation 3.26 is automatically satisfied. So the Equation 3.30 gives the solution we needed for $q_{x \to y, t+1}$. In this equation, if $\pi_{i,x,t} = 1$, $q_{x \to y, t+1}$ simply reduces to $\text{conf}(x \to y)$. So we can interpret $q_{x \to y, t+1}$ as the generalized confidence of the classification rule $x \to y$, in which instances are weighted according to $\pi_{i,x,t}$.

Finally, we note that ACEM introduces a parameter M which controls the number of EM-steps. After the EM algorithm is completed, the rule's parameters are finalized to be $q_{x \to y, M}$ and $w_{x,M}$.

3.4 Experimental Design For AREM

To evaluate our AREM model, we design two sets of experiments. First, we evaluate how the predictive performances are affected by using different itemset mining methods (instance based approach and database sequential coverage approach). We also study the effectiveness of the EM optimization by comparing it to the straightforward approach for determining rule's parameters. After that, we compare the AREM model against the other state of the art regression models.

3.4.1 Data Sets

Table 3.1 summarizes the set of datasets that we use to evaluate the predictive performances of various models.

Text Reviews Data The first six data sets are randomly sampled from user reviews downloaded from three websites: BestBuy [54], CitySearch [55], and Yelp [56]. Each instance corresponds to the review of a product where the target variable to predict is the user's rating which ranges from one to five. The review text is parsed and a set of features, or items, is extracted. We constructed two types of features: wdep and swf. For wdep, the Stanford dependencies [57] between words in each sentence are extracted. Each dependency is a triplet containing the name of the relation, the governor and the dependent. We replace nouns in the dependency with the wildcard. For swf, words in the review text are extracted after applying stemming and removing stop words. We remove the infrequent items whose relative supports are less than 0.5%.

MovieLens The MovieLens [58] dataset is a movie rating dataset, where the target variable is the movie's rating which ranges from one to five. Features include users' demographic information and 19 movie genres, each of which is a binary variable indicating whether the movie belongs to the genre or not.

Airline The Airline data set is downloaded from DataExpo09 competition [59]. The Airline data set describes flight arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008. We randomly sampled $10,000$ instances out of the 2008 data set. We chose the arrival delay, normalized to have mean zero and variance one, as the target variable to predict. Input features include month, day of month, day of week, scheduled departure hour, scheduled arrival hour, carrier, origin, destination, scheduled elapsed time discretized into 11 intervals, departure delay discretized into 11 intervals, and distance discretized into 10 intervals.

Socmob The last three data sets are downloaded from CMU StatLib [60]. For Socmob, the counts for son's current occupation, normalized to have mean zero and variance one, is selected as the target variable. Input features include father's occupation, son's occupation, family structure, race, and son's first occupation discretized into 6 intervals.

Pollen Pollen is a synthetic dataset about the geometric features of pollen grains. We chose Density, normalized to have mean zero and variance one, as the target variable. Other four variables RIDGE, NUB, CRACK, and WEIGHT are discretized into 5, 4, 5, and 3 intervals, respectively.

Spacega Spacega contains election data including spatial coordinates on 3107 US counties. We chose ln(VOTES/POP), normalized to have mean zero and variance one, as the target variable. Other variables POP, EDUCATION, HOUSES, INCOME, XCOORD and YCOORD are discretized into 4, 3, 4, 4, 4 and 5 intervals respectively.

3.4.2 Different Versions of AREM Models

We compare the predictive performances of four itemset mining algorithms, three of which are instance based approach with qualities Q^{σ}, Q^{m} and Q^{n}, respectively and the fourth is the RBAMine which stands for the RBA's database sequential coverage mining approach. Each of these four algorithms takes two input parameters: (i) K as the number of top itemsets mined for each instance in the instance based approach and the number of itemsets each instance is covered by in the sequential coverage approach, and (ii) s_0^r as the minimum relative support which defines the set of frequent itemsets. We choose K to be one to five. and the smallest s_0^r to be 0.5%. We try different values of s_0^r above 0.5% for different datasets.

Notice that, due to its two step approach, RBAMine can not finish within reasonably amount of time on wdep datasets with the support threshold low enough to get satisfactory performances. We apply a work around approach which allows us to lower the support to as low as 0.5% on these datasets. To describe this approach, recall that RBAMine applies an error checking step which removes itemsets which does not reduce the training errors. Without this error checking step, RBAMine will be a simple sequential coverage method which will be equivalent to an instance based approach with

Table 3.2: *mean*(MSE) for Direct Method for Assigning Rule Parameters (I)

Mining Method	k	BestBuy		CitySearch		Yelp	
		wdep	swf	wdep	swf	wdep	swf
RBAMine	1	1.131	1.143	1.272	1.184	1.120	1.073
	5	1.083	0.970	1.136	1.023	1.084	1.028
	10	1.077	**0.937**	1.093	**0.993**	1.068	1.009
	15	1.075	**0.932**	1.076	**0.985**	1.061	**0.997**
	20	1.074	**0.930**	1.066	**0.983**	**1.055**	**0.993**
Q^σ	1	1.195	1.182	1.442	1.381	1.143	1.123
	5	1.178	1.153	1.342	1.292	1.114	1.107
	10	1.171	1.132	1.297	1.259	1.109	1.100
	15	1.167	1.122	1.277	1.248	1.107	1.097
	20	1.163	1.113	1.267	1.241	1.105	1.094
Q^m	1	1.181	1.205	1.434	1.441	1.186	1.188
	5	1.097	1.045	1.080	1.166	1.111	1.055
	10	1.054	1.021	1.033	1.075	1.081	1.025
	15	1.042	1.014	**1.018**	1.049	1.063	1.014
	20	1.037	1.012	**1.012**	1.038	**1.054**	1.009
Q^n	1	1.181	1.205	1.434	1.441	1.186	1.188
	5	1.085	1.058	1.128	1.226	1.110	1.058
	10	1.041	1.033	1.057	1.122	1.075	1.027
	15	**1.029**	1.026	1.037	1.087	**1.056**	1.015
	20	**1.024**	1.022	1.028	1.071	**1.047**	1.008

parameter K and quality function Q^σ. Now, with this error checking step, some of the top K itemsets are removed and some of the itemsets beyond top K are selected. It is reasonable to believe that most of the selected itemsets will still be within top $\tilde{K} > K$ for each instance. So our work around approach is to run our instance based miner (which will explained in detail in Chapter 4) with $\tilde{K} = 10$ and quality Q^σ to generate a set of itemsets, which are then processed by the RBAMine algorithm. We have observed that this strategy generates almost the same set of itemsets as RBAMine.

To evaluate the effectiveness of the EM optimization (EMOpt) of our AREM model. We compare it against the straightforward approach (Direct) which assigns rule's RHS as the mean μ_x of the itemset and the rule's weight $w_x = 1$. Other weighting schemes as applied by RBA are not presented here because we did not see advantages of those methods. The Direct method makes prediction by selecting k rules with the smallest variances from those rules whose itemsets are contained in the test instance. For our

Table 3.3: *mean*(MSE) for Direct Method for Assigning Rule Parameters (II)

Mining Method	k	MovieLens	Airline	Socmob	Pollen	Spacega
RBAMine	1	1.257	**0.676**	0.852	0.513	**0.507**
	5	1.241	0.713	0.595	0.565	0.528
	10	1.236	0.722	0.595	0.566	0.543
	15	1.231	0.722	0.596	0.566	0.544
	20	1.230	0.722	0.596	0.566	0.544
Q^{σ}	1	1.277	0.688	0.876	0.514	0.519
	5	1.249	**0.678**	0.741	**0.505**	**0.510**
	10	1.240	**0.685**	0.674	**0.505**	0.519
	15	1.237	0.686	0.664	**0.505**	0.526
	20	1.236	0.686	0.671	**0.505**	0.527
Q^{m}	1	1.281	0.692	0.795	**0.508**	0.514
	5	1.251	**0.681**	0.510	**0.500**	**0.503**
	10	1.228	0.696	0.**465**	0.540	**0.508**
	15	**1.220**	0.692	**0.457**	0.596	**0.511**
	20	**1.213**	0.694	**0.457**	0.599	0.520
Q^{n}	1	1.282	0.689	0.803	0.509	0.516
	5	1.247	**0.678**	0.605	**0.499**	**0.506**
	10	1.225	0.693	0.495	0.539	**0.508**
	15	**1.217**	0.692	0.489	0.598	**0.509**
	20	**1.213**	0.694	0.483	0.608	0.516

experimental study, we choose the parameter k for both Direct and EMOpt to be 1, 5, 10, 15 and 20. The rationale of choosing these values comes from the following: if k is too large, AR models' strength of being interpretable essentially disappears; on the other hand, if k is too small, the performance may not be satisfactory. We choose the maximum k value to be 20 as a compromise from these two extreme case considerations. In addition, EMOpt also takes another parameter M, which controls the number of EM steps. Most of our datasets only need a few EM steps to get the best performance.

3.4.3 State Of The Art Regression Models

For model comparison's purpose, we focus on descriptive models and select several state of the art tree-based and rule-based regression models. The support vector regression (SVR) [61] is an exception. It is included because it is one of the best known and standard models for regression.

Table 3.4: *mean*(MSE) for EMOpt Method for Assigning Rule Parameters (I)

Mining Method	k	BestBuy		CitySearch		Yelp	
		wdep	swf	wdep	swf	wdep	swf
RBAMine	1	1.098	1.055	1.235	1.075	1.132	1.085
	5	1.017	0.796	1.018	0.866	1.061	0.905
	10	1.014	0.795	0.997	0.856	1.034	0.900
	15	1.013	0.793	0.997	0.863	1.028	0.901
	20	1.011	0.793	0.996	0.867	1.028	0.902
Q^σ	1	1.150	1.180	1.327	1.205	1.147	1.132
	5	1.044	0.820	1.058	0.970	1.053	0.922
	10	1.010	0.781	1.017	0.895	1.038	0.897
	15	1.013	0.788	0.999	0.871	1.026	0.883
	20	1.020	0.791	0.981	0.865	1.014	0.879
Q^m	1	1.209	1.235	1.453	1.258	1.186	1.195
	5	0.930	0.777	0.988	0.858	1.025	0.902
	10	0.868	**0.729**	0.886	0.790	0.934	0.844
	15	**0.849**	**0.739**	**0.856**	**0.782**	**0.910**	0.835
	20	**0.844**	0.746	**0.851**	**0.775**	**0.907**	**0.827**
Q^n	1	1.212	1.267	1.488	1.243	1.186	1.194
	5	0.934	0.789	0.992	0.863	1.043	0.896
	10	0.868	0.741	0.895	0.790	0.946	0.838
	15	**0.849**	0.742	**0.856**	**0.781**	0.924	**0.826**
	20	**0.842**	0.746	**0.846**	**0.773**	0.920	**0.821**

SVR We use libsvm [62] for *SVR*, and use only the linear kernel. Model parameters tuned are: C and ϵ, where ϵ is the size of ϵ-insensitive tube, and C controls the model complexity.

CART$_k$ This group of models contain the Classification And Regression Tree (CART) [32] and the Boosted Regression Tree [34] where CART of fixed size is acting as the weak learners. So, $CART_k$ stands for CART being boosted k times [63]. We tuned three parameters for $CART_k$: *depth*, *leaf* and *lrate*, where *depth* is the maximum depth of the tree, *leaf* is the minimum number of leaf samples of the tree, and *lrate* is the learning rate of the gradient boosting method.

CUBIST$_k$ Cubist [33] is a rule based algorithm which has the option of building committee models. The number of members in the committee is captured in k. We

Table 3.5: *mean*(MSE) for EMOpt Method for Assigning Rule Parameters (II)

Mining Method	k	MovieLens	Airline	Socmob	Pollen	Spacega
RBAMine	1	1.257	0.676	0.484	0.520	0.509
	5	1.247	0.668	0.459	0.491	**0.480**
	10	1.244	0.668	0.462	0.490	**0.479**
	15	1.236	0.668	0.462	0.490	**0.480**
	20	1.233	0.668	0.462	0.490	**0.480**
Q^{σ}	1	1.275	0.681	0.505	0.519	0.510
	5	1.244	0.671	0.391	**0.485**	0.486
	10	1.229	0.672	0.398	**0.482**	**0.479**
	15	1.230	0.673	0.398	**0.481**	**0.477**
	20	1.230	0.673	0.397	**0.481**	**0.476**
Q^{m}	1	1.281	0.692	0.431	0.508	0.527
	5	1.251	0.676	**0.300**	0.500	0.504
	10	1.238	0.659	**0.293**	**0.482**	0.491
	15	**1.224**	**0.651**	**0.293**	**0.480**	**0.477**
	20	**1.220**	**0.650**	**0.292**	**0.480**	**0.474**
Q^{n}	1	1.282	0.689	0.475	0.509	0.531
	5	1.247	0.673	**0.302**	0.499	0.514
	10	1.235	0.659	0.303	**0.483**	0.497
	15	**1.221**	**0.653**	0.304	**0.480**	**0.482**
	20	**1.215**	**0.647**	0.305	**0.480**	**0.477**

tuned two binary parameters for $CUBIST_k$: *UB* (unbiased), and *CP* (composite). Parameter *UB* instructs CUBIST to make each rule approximately unbiased. Parameter *CP* instructs CUBIST to construct the composite model.

RBA$_k$ We implemented the RBA model following [22]. Here k is the number of top ranked rules used for prediction. We tuned two parameters for RBA_k: s_0 and *weight*, where s_0 is the minimum support threshold, and *weight* is the weighting scheme used for prediction, which can take three values *supp*, *inv-var* and *equal*.

AREM$_k$ We compare the above models against $AREM_k$, where k is the number of rules used for prediction. For this comparison, we use the instance based itemset mining approach with quality function Q^m, which will be shown to be one of the best choices.

For the above models (except *SVR*), we set $k = 1, 5, 10, 15, 20$ for the same interpretation considerations as explained above.

Table 3.6: EMOpt v.s. Direct: win-tie-loss

Mining Methods	Q^m	Q^n	Q^σ	RBAMine
EMOpt v.s. Direct	10-1-0	10-1-0	9-2-0	9-2-0

[a] It is a tie if $|\mu_{m_1-m_2}| < \sigma_{m_1-m_2}$.

3.4.4 Evaluation

We used the Mean Squared Error (MSE) between the actual and predicted target variable's values as the performance metric. For each (model, data) pair, we first identified a set of parameter configurations that was likely to achieve the best performance. The model was then trained on the training set and MSE was calculated on the validation set for each of the parameter configurations. Then we selected the parameter configuration that gives the best MSE on the validation set, and computed the corresponding MSE on the testing set. This process is repeated for the number of trials shown in Table 3.1. Finally, we reported the average MSE on all testing trials.

For the regression problem, the significance of the MSE difference between two models is data dependent. To properly evaluate our model, we employ a statistics based approach which is explained in the following. For a given data set, in order to compare model m_1 to model m_2, we take into account the distribution of the MSE values computed on multiple testing trials for each model. Let μ_1, σ_1, n_1 (μ_2, σ_2, n_2) be the mean, standard deviation and the number of observations of the set of MSE values for model m_1 (m_2), respectively. We introduce $\mu_{m_1-m_2} = \mu_2 - \mu_1$ and $\sigma_{m_1-m_2} = \sqrt{\sigma_1^2/n_1 + \sigma_2^2/n_2}$. The quantity $\mu_{m_1-m_2}/\sigma_{m_1-m_2}$ is used in statistical testing [64] for the comparison of two population means. Under the null hypothesis that two population means are the same, $\mu_{m_1-m_2}/\sigma_{m_1-m_2}$ can be assumed to have the Normal distribution $\mathcal{N}(0,1)$. So the more deviated from zero this quantity is, the more likely that two models are performing differently.

Table 3.7: Mining Methods Comparison: win-tie-loss

Method	Q^m v.s. Q^n	Q^m v.s. Q^σ	Q^m v.s. RBAMine
EMOpt	1-10-0	8-3-0	9-2-0
Direct	2-9-0	8-3-0	4-4-3
Method	Q^n v.s. Q^σ	Q^n v.s. RBAMine	Q^σ v.s. RBAMine
EMOpt	8-3-0	10-1-0	3-8-0
Direct	9-2-0	4-4-3	1-3-7

[a] It is a tie if $|\mu_{m_1-m_2}| < \sigma_{m_1-m_2}$.

3.5 Experimental Results For AREM

3.5.1 Predictive Performances Of Different Mining Algorithms And EM Optimization

Tables 3.2, 3.3, 3.4 and 3.5 present the mean value of MSE for different itemset mining algorithms with two approaches (Direct and EMOpt) for assigning rule parameters. We make a few comparisons as follows.

We show the effectiveness of the EM algorithm by comparing EMOpt against Direct for each of the four itemset mining algorithms. For each of the models under comparison, we first select k value which gives the smallest MSE, and then compute the $\mu_{m_1-m_2}/\sigma_{m_1-m_2}$ for each dataset. We count it as a tie if $|\mu_{m_1-m_2}|/\sigma_{m_1-m_2} < 1$. Otherwise we count it as a win or loss depending on the sign of $\mu_{m_1-m_2}$. Finally, we accumulate the counts of wins, ties and losses over all datasets and present the results in Table 3.6. Table 3.6 clearly shows that the EM algorithm improves the performance dramatically for all itemset mining methods on almost all the datasets.

Next, we compare different pairs of itemset mining algorithms for EMOpt and Direct, respectively. We follow the same steps discussed above and compute the win-tie-loss which is then shown in Table 3.7. From Table 3.7, we can rank the itemset mining algorithms for the Direct method as $Q^m \geq Q^n \geq$ RBAMine $> Q^\sigma$. For the EMOpt method, we get the ranking $Q^m \approx Q^n > Q^\sigma >$ RBAMine. We see that Q^m and Q^n are better than Q^σ and RBAMine in both cases. This is reasonable because Q^m and Q^n take into account the instance's target variable when selecting best instance based itemsets. It is also interesting to see that the additional error checking step of RBAMine improves the performance (RBAMine $> Q^\sigma$) when the Direct method is applied. However, this

Table 3.8: Comparing $AREM_k$ To Other Models (I): $mean$(MSE)

model	BestBuy		CitySearch		Yelp	
	wdep	swf	wdep	swf	wdep	swf
SVR	0.890	0.826	**0.859**	0.845	**0.881**	0.837
$CART_1$	1.116	0.904	1.336	1.020	1.149	1.137
$CART_5$	0.922	0.815	1.022	0.876	0.997	0.985
$CART_{10}$	0.918	0.793	0.984	0.845	0.947	0.952
$CART_{15}$	0.885	0.783	0.959	0.834	0.915	0.908
$CART_{20}$	0.857	0.777	**0.850**	0.825	0.901	0.864
$CUBIST_1$	0.989	0.930	1.135	1.048	1.080	1.024
$CUBIST_5$	1.021	0.975	1.133	0.992	1.071	1.014
$CUBIST_{10}$	1.034	0.987	1.131	0.993	1.070	1.014
$CUBIST_{15}$	1.042	0.986	1.136	0.993	1.081	1.010
$CUBIST_{20}$	1.042	0.988	1.134	0.997	1.081	1.012
RBA_1	1.139	1.031	1.291	1.145	1.123	1.102
RBA_5	1.008	0.913	1.092	0.977	1.040	0.996
RBA_{10}	0.982	0.903	1.033	0.946	1.027	0.972
RBA_{15}	0.972	0.899	1.010	0.938	1.022	0.964
RBA_{20}	0.968	0.897	1.000	0.937	1.015	0.963
$AREM_1$	1.209	1.235	1.453	1.258	1.186	1.195
$AREM_5$	0.930	0.777	0.988	0.858	1.025	0.902
$AREM_{10}$	0.868	**0.729**	0.886	0.790	0.934	0.844
$AREM_{15}$	**0.849**	**0.739**	**0.856**	**0.782**	0.910	**0.835**
$AREM_{20}$	**0.844**	0.746	**0.851**	**0.775**	0.907	**0.827**

error checking step can hurt the performance ($Q^\sigma >$ RBAMine) if we apply the EM algorithm afterwards.

3.5.2 Comparing AREM To The State Of the Art Regression Models

The average MSE for the discussed set of models on the various data sets are shown in the Tables 3.8 and 3.9, where the best results have been highlighted. In order to compare $AREM_k$ against the rest of the models, we compute the quantity $\mu_{m_1-m_2}/\sigma_{m_1-m_2}$ for these model pairs. For this purpose, we choose the k values which give the smallest MSE for the four k-dependent models ($AREM_k$, $CART_k$, RBA_k and $CUBIST_k$). These results are shown in Table 3.10. Note that $CART_1$ is the standard CART model, in contrast to $CART_k$ which stands for the boosted regression tree. For easy comparison,

Table 3.9: Comparing $AREM_k$ To Other Models (II): $mean$(MSE)

model	MovieLens	Airline	Socmob	Pollen	Spacega
SVR	1.234	**0.643**	0.535	**0.469**	**0.480**
$CART_1$	1.228	**0.649**	0.440	0.488	0.488
$CART_5$	**1.201**	**0.640**	0.349	0.482	**0.480**
$CART_{10}$	**1.199**	**0.642**	0.349	0.482	**0.481**
$CART_{15}$	**1.198**	**0.640**	0.349	0.483	**0.482**
$CART_{20}$	**1.195**	**0.640**	0.341	0.483	**0.484**
$CUBIST_1$	1.237	0.658	0.363	0.501	0.491
$CUBIST_5$	1.257	0.663	0.367	0.500	0.495
$CUBIST_{10}$	1.260	0.664	0.370	0.500	0.492
$CUBIST_{15}$	1.260	0.665	0.369	0.499	0.493
$CUBIST_{20}$	1.260	0.665	0.369	0.500	0.493
RBA_1	1.291	0.730	0.522	0.508	0.522
RBA_5	1.244	0.684	0.564	0.497	0.493
RBA_{10}	1.238	0.691	0.595	0.497	0.496
RBA_{15}	1.235	0.691	0.605	0.497	0.496
RBA_{20}	1.235	0.691	0.605	0.497	0.496
$AREM_1$	1.281	0.692	0.431	0.508	0.527
$AREM_5$	1.251	0.676	**0.300**	0.500	0.504
$AREM_{10}$	1.238	0.659	**0.293**	0.482	0.491
$AREM_{15}$	1.224	0.651	**0.293**	0.480	**0.477**
$AREM_{20}$	1.220	**0.650**	**0.292**	0.480	**0.474**

we derive the win-tie-loss from Table 3.10 and present them in Table 3.11.

Tables 3.10 and 3.11 show that AREM is performing better than all competing methods. In particular, it is much better (with very high $\mu_{m_1-m_2}/\sigma_{m_1-m_2}$) than RBA_k and the standard tree-based ($CART_1$) and rule-based ($CUBIST_k$) models. When comparing against the more competitive methods $CART_k$ and SVR, the number of wins still dominates the number of losses, even under the very conservative decision criteria ($|\mu_{m_1-m_2}| \geq 3\sigma_{m_1-m_2}$). It is also interesting to observe that AREM performs almost uniformly well on the review data sets, but not as uniform on the rest of the data sets. Given that the review data sets have much larger number of items (see Table 3.1), we think this is an indication that AREM is more suitable for high-dimensional and sparse data sets. Finally, from Tables 3.8 and 3.9, we can see how different k values affect the AREM's performance. When $k = 1$, the performance is not satisfactory. This is

Table 3.10: Comparing $AREM_k$ To Other Models: $\mu_{m_1-m_2}/\sigma_{m_1-m_2}$

Data set	$CART_k$	SVR	RBA_k	$CART_1$	$CUBIST_k$
BestBuy.wdep	0.64	2.27	5.47	12.35	7.16
BestBuy.swf	3.01	5.21	9.03	9.83	9.88
CitySearch.wdep	−0.07	0.59	8.95	30.67	16.82
CitySearch.swf	4.01	5.81	13.45	17.24	14.47
Yelp.wdep	−0.50	−2.30	9.55	18.90	11.30
Yelp.swf	3.25	1.08	13.74	29.16	14.99
MovieLens	−1.52	0.82	0.85	0.47	1.00
Airline	−0.94	−0.66	3.01	−0.09	0.71
Socmob	2.56	10.56	9.31	6.92	3.72
Pollen	0.18	−1.94	3.01	1.24	3.36
Spacega	0.24	0.24	0.71	0.57	0.63

Table 3.11: Comparing $AREM_k$ To Other Models: win-tie-loss

comparing criteria[a]\model	$CART_k$	SVR_k	RBA_k	$CART_1$	$CUBIST_k$
$\lvert\mu_{m_1-m_2}\rvert \geq \sigma_{m_1-m_2}$	4-6-1	5-4-2	9-2-0	8-3-0	9-2-0
$\lvert\mu_{m_1-m_2}\rvert \geq 2\sigma_{m_1-m_2}$	4-7-0	4-6-1	9-2-0	7-4-0	8-3-0
$\lvert\mu_{m_1-m_2}\rvert \geq 3\sigma_{m_1-m_2}$	3-8-0	3-8-0	9-2-0	7-4-0	8-3-0

[a] It is a tie if $|\mu_{m_1-m_2}| < n\sigma_{m_1-m_2}$.

not surprising because our probabilistic model is optimized for large number of itemsets. However, as k becomes sufficiently large (15 or 20), the performance improves considerably and remains quite stable.

3.6 Experimental Design For ACEM

3.6.1 Datasets

We evaluate the performance of our ACEM model on twenty data sets summarized in Table 3.12. The density column captures the sparsity of the datasets. It is computed as the percentage of non-zero entries of the dataset matrix, where the row corresponds to an instance and the column corresponds to an item. For repeated evaluation, each dataset is randomized multiple trials as summarized in the last column. Note that we create more trials for smaller datasets.

Table 3.12: Dataset Summary

data	# of instances	# of items	density	# of classes	# of trials
BestBuy	10000	962	1.7%	5	20
CitySearch	10000	996	2.2%	5	20
Yelp	10000	1442	2.1%	5	20
adult	48842	95	14.6%	2	4
anneal	898	49	14.9%	5	200
balance	625	20	20%	3	300
breast[65]	699	90	10%	2	300
chess	28056	40	15%	18	10
connect4	10000	125	33.6%	3	20
credit	1000	57	16.2%	2	200
diabetes	768	77	10.4%	2	200
kr-vs-kp	3196	37	22.2%	2	60
led7	10000	7	65.7%	10	20
letRecog	20000	76	21.1%	26	10
nursery	12960	27	29.6%	5	20
pageBlocks	5473	39	25.6%	5	40
segment	2310	119	8.6%	7	80
spambase	4601	142	5.5%	2	40
tic-tac-toe	958	27	33.3%	2	200
vowel	990	100	10%	11	200

Text Review Datasets

The first three data sets are randomly sampled from user reviews downloaded from three websites: *BestBuy* [54], *CitySearch* [55], and *Yelp* [66]. Each instance corresponds to the review of a product where the target variable to predict is the user's rating which ranges from one to five. Words in the review text are extracted as features, or items, after applying stemming and removing stop words. We remove features with relative support less than 0.5%.

UCI Datasets

The remaining seventeen datasets are from the UCI machine learning repository [67]. We discretize continuous attributes into a set of intervals. Combining them with the additional categorical attribute values gives us the complete set of items for the dataset, which are summarized in the third column of Table 3.12.

Table 3.13: ACEM Model Comparison (I): average accuracy

model	BestBuy	CitySearch	Yelp	adult	anneal	balance	breast
SVM	**0.655**	**0.494**	**0.425**	**0.850**	**0.966**	**0.910**	**0.967**
BCAR	0.626	0.463	0.406	0.835	0.951	0.868	**0.964**
$CPAR_1$	0.610	0.454	0.380	0.789	0.912	0.779	0.954
$CPAR_5$	0.604	0.448	0.381	0.785	0.911	0.747	0.953
$CPAR_{10}$	0.603	0.446	0.377	0.784	0.911	0.746	0.952
$CPAR_{15}$	0.603	0.441	0.375	0.784	0.911	0.746	0.952
$CPAR_{20}$	0.603	0.439	0.374	0.784	0.911	0.746	0.952
$HARMONY_1$	0.627	0.432	0.383	0.834	0.880	0.844	0.947
$HARMONY_5$	0.626	0.446	0.393	0.836	0.869	0.890	0.952
$HARMONY_{10}$	0.627	0.456	0.396	0.844	0.876	0.890	0.949
$HARMONY_{15}$	0.627	0.460	0.402	0.836	0.877	0.890	0.949
$HARMONY_{20}$	0.628	0.461	0.404	0.844	0.877	0.890	0.949
$ACNoEM_1$	0.631	0.438	0.387	0.834	0.878	0.849	0.950
$ACNoEM_5$	0.623	0.453	0.388	0.836	0.847	0.888	0.951
$ACNoEM_{10}$	0.623	0.456	0.386	0.836	0.846	0.895	0.952
$ACNoEM_{15}$	0.623	0.457	0.385	0.836	0.836	0.895	0.951
$ACNoEM_{20}$	0.623	0.458	0.385	0.836	0.836	0.894	0.951
$ACEM_1$	0.631	0.436	0.383	0.835	**0.959**	0.830	0.951
$ACEM_5$	0.643	0.475	0.406	0.844	**0.960**	0.890	**0.958**
$ACEM_{10}$	0.645	**0.486**	0.415	**0.848**	**0.963**	0.892	**0.959**
$ACEM_{15}$	**0.646**	**0.490**	**0.416**	**0.855**	**0.964**	0.892	**0.960**
$ACEM_{20}$	**0.649**	**0.490**	**0.418**	**0.855**	**0.964**	0.892	**0.960**

3.6.2 Models

We compare the predictive performance of ACEM against the support vector machine (SVM) [68] and three state of the art AC models: Harmony [24], BCAR [69], and CPAR [18]. In addition, we evaluate the performance of the EM optimization by comparing ACEM to model ACNoEM, which is derived from ACEM with the EM optimization being turned off.

***ACEM$_k$*, *ACNoEM$_k$* & *HARMONY$_k$*.** These three models all apply the instance based approach for rule discovery. ACEM and ACNoEM use the same set of classification rules, while the difference is the approach used to estimate rule parameters. ACEM applies the EM algorithm for learning $q_{x \to y}$ and w_x. ACNoEM simply assigns $q_{x \to y}$ to be the confidence of rule $x \to y$ and w_x to one. Another difference between ACEM and

Table 3.14: ACEM Model Comparison (II): average accuracy

model	chess	connect4	credit	diabetes	kr-vs-kp	led7	letRecog
SVM	0.441	0.760	**0.734**	**0.742**	**0.971**	**0.735**	**0.736**
$BCAR$	0.434	0.767	**0.732**	0.727	0.920	0.669	0.660
$CPAR_1$	0.377	0.728	0.700	**0.737**	0.743	0.661	0.607
$CPAR_5$	0.341	0.700	0.698	0.729	0.735	0.659	0.601
$CPAR_{10}$	0.334	0.677	0.694	0.728	0.735	0.659	0.596
$CPAR_{15}$	0.334	0.674	0.694	0.728	0.735	0.659	0.596
$CPAR_{20}$	0.334	0.674	0.694	0.728	0.735	0.659	0.596
$HARMONY_1$	0.339	0.752	0.711	0.698	0.915	0.467	0.611
$HARMONY_5$	0.368	0.767	**0.725**	0.708	0.929	0.618	0.663
$HARMONY_{10}$	0.374	**0.770**	**0.727**	0.708	0.931	0.673	0.673
$HARMONY_{15}$	0.374	**0.771**	**0.728**	0.708	0.934	0.718	0.674
$HARMONY_{20}$	0.374	**0.769**	**0.729**	0.707	0.926	0.718	0.670
$ACNoEM_1$	0.338	0.753	0.712	0.699	0.917	0.431	0.611
$ACNoEM_5$	0.364	**0.770**	**0.725**	0.706	0.920	0.432	0.656
$ACNoEM_{10}$	0.368	**0.771**	**0.728**	0.709	0.929	0.428	0.678
$ACNoEM_{15}$	0.365	**0.774**	**0.729**	0.707	0.931	0.424	0.686
$ACNoEM_{20}$	0.367	**0.774**	**0.729**	0.708	0.932	0.420	0.688
$ACEM_1$	0.548	0.756	0.699	0.724	**0.975**	**0.735**	0.710
$ACEM_5$	**0.568**	**0.771**	0.713	**0.735**	**0.978**	**0.732**	**0.730**
$ACEM_{10}$	**0.568**	**0.775**	0.722	**0.737**	**0.979**	**0.733**	**0.733**
$ACEM_{15}$	**0.568**	**0.777**	**0.726**	**0.736**	**0.979**	**0.732**	**0.734**
$ACEM_{20}$	**0.568**	**0.776**	**0.727**	**0.738**	**0.979**	**0.732**	**0.735**

ACNoEM is the approach used to select top k itemsets for prediction. ACEM selects k itemsets with highest weights w_x. However, for ACNoEM, this approach does not apply because the itemsets' weights are all the same. To fix this problem, we first select the target variable y which gives the highest $q_{x\to y}$ for itemset x and use the corresponding $q_{x\to y}$ as a metric for the quality of the itemset. So, during prediction, top k itemsets with the highest $\max_y q_{x\to y}$ will be selected. HARMONY mines the same set of itemsets as ACEM and ACNoEM. However, for each mined itemset x, it does not use all possible ys as the rule's RHS. In fact, only those y, for which there exists at least a training instance with target variable being y and x being one of its top k itemsets, are selected. Our experiments will show that this subtle difference has no significant impact on the performance. These three models share the same parameters s_0^r and K, where s_0^r is the minimum relative support of the rule's LHS and K is the number of top itemsets mined for each instance. We set $s_0^r = 0.5\%, 1.0\%, 3.0\%, 5.0\%$ and $K = 1, 2, 3, 4, 5$ for all datasets. ACEM has an additional parameter M, i.e., the number of EM steps, which we optimize to get the best performance.

SVM. We use the libsvm [62] implementation for linear SVM. In libsvm, the multiclass classification is handled by the one-vs-one scheme. We tune the parameter C to get the best performance for SVM.

BCAR. We downloaded the implementation of BCAR from `http://isoft.postech.ac.kr/~ywyoon/BCAR/usage.html`. One of the key ideas of BCAR is to set the minimum support and confidence to low values but limit the maximum rule length to 2 or 3 as suggested in [69] to avoid the massive number of generated rules. So we set the minimum support to 0.5% and confidence to 0.1. And we set the maximum rule length to 3 for all UCI datasets and to 2 for the text review datasets, due to that there were too many rules with length 3 for the review datasets. Then we tune the parameter coverage threshold so that BCAR can achieve the best performance.

CPAR. We downloaded the implementation of CPAR from `http://cgi.csc.liv.ac.uk/~frans/KDD/Software`. We set the total weight factor to 0.05 and gain similarity ratio to 99%. These are the settings used in [18]. Our experience is that the performance is insensitive to the weight factor, and lowering gain similarity is likely to hurt the performance. Then we tune another two parameters minimum gain and weight decay factor to achieve the best performance.

In addition to the above parameters, ACEM, ACNoEM, Harmony, and CPAR share another parameter k which is the number of rules used for making predictions. We choose k to be 1, 5, 10, 15 and 20 for all three models. The rationale of choosing the maximum k as 20 is: if k is too large, these models' strength of being interpretable essentially disappears; on the other hand, if k is too small, the performance may not be satisfactory.

Table 3.15: ACEM Model Comparison (III): average accuracy

model	nursery	pageBlocks	segment	spambase	tic-tac-toe	vowel
SVM	0.930	**0.928**	0.942	**0.921**	0.985	0.807
$BCAR$	0.916	**0.919**	**0.952**	0.873	0.957	**0.917**
$CPAR_1$	0.862	0.916	0.898	0.911	0.818	0.616
$CPAR_5$	0.853	0.916	0.884	0.908	0.764	0.595
$CPAR_{10}$	0.853	0.916	0.881	0.906	0.762	0.594
$CPAR_{15}$	0.853	0.916	0.881	0.905	0.762	0.594
$CPAR_{20}$	0.853	0.916	0.881	0.905	0.762	0.594
$HARMONY_1$	0.942	0.913	0.930	0.891	**0.994**	0.764
$HARMONY_5$	0.948	0.913	0.935	0.899	**0.998**	0.795
$HARMONY_{10}$	0.940	0.913	0.935	0.898	**0.997**	0.797
$HARMONY_{15}$	0.937	0.913	0.936	0.899	**0.997**	0.796
$HARMONY_{20}$	0.937	0.913	0.937	0.899	**0.997**	0.799
$ACNoEM_1$	0.941	0.909	0.931	0.899	**0.995**	0.770
$ACNoEM_5$	0.939	0.910	0.935	0.900	**0.997**	0.796
$ACNoEM_{10}$	0.942	0.910	0.935	0.898	**0.998**	0.803
$ACNoEM_{15}$	0.939	0.910	0.935	0.897	**0.997**	0.803
$ACNoEM_{20}$	0.938	0.910	0.935	0.897	**0.997**	0.803
$ACEM_1$	**0.982**	**0.922**	0.934	0.909	**0.997**	0.762
$ACEM_5$	**0.985**	**0.922**	0.938	**0.925**	**0.998**	0.792
$ACEM_{10}$	**0.985**	**0.922**	0.939	**0.925**	**0.999**	0.800
$ACEM_{15}$	**0.985**	**0.922**	0.939	**0.926**	**0.999**	0.800
$ACEM_{20}$	**0.985**	**0.922**	0.939	**0.925**	**0.999**	0.799

3.6.3 Model Evaluation

We use the accuracy, i.e., the ratio of test instances being predicted correctly, as the performance metric. We randomly split each dataset into 80% training set, 10% validation set and 10% testing set. The model is trained on training set and then evaluated on validation set. After this, we select the parameters which give the best performance on validation set and evaluate the model (with selected parameters) on testing set. This process is repeated certain number of trials as summarized in Table 3.12. Finally, we compute the average accuracy from all testing trials. For model comparison's purpose, we consider a model is better if its (average) accuracy is higher by more than 0.01, that is, there are more than 1% of test instances that are predicted correctly by this model than the model that is compared to.

Table 3.16: ACEM Model Comparison: win-tie-loss

Model	Compared To	win-tie-loss
$ACEM_k$	SVM	4-15-1
$ACEM_k$	$HARMONY_k$	12-8-0
$ACEM_k$	$BCAR$	14-4-2
$ACEM_k$	$ACNoEM_k$	13-7-0
$ACEM_k$	$CPAR_k$	17-3-0
SVM	$HARMONY_k$	13-4-3
SVM	$BCAR$	13-6-1
SVM	$ACNoEM_k$	14-3-3
SVM	$CPAR_k$	18-2-0
$HARMONY_k$	$BCAR$	7-7-6
$HARMONY_k$	$ACNoEM_k$	2-17-1
$HARMONY_k$	$CPAR_k$	13-4-3
$BCAR$	$ACNoEM_k$	8-6-6
$BCAR$	$CPAR_k$	14-5-1
$ACNoEM_k$	$CPAR_k$	11-5-4

3.7 Experimental Results For ACEM

We present our results for various models evaluated on the twenty datasets in Tables 3.13, 3.14 and 3.15. The best performing results for each dataset are highlighted. For easier model comparison, we present, for each pair of models under study, the win-tie-loss accumulated for all datasets in Table 3.16. For models ACEM, ACNoEM, HARMONY, and CPAR, we select the k value with the highest accuracy for the purpose of comparing with other models.

We draw a number of conclusions by observing the win-tie-loss counts shown in Table 3.16. First, the effectiveness of our EM optimization is shown to be significant on 13 datasets by comparing $ACEM_k$ against $ACNoEM_k$. Second, the two instance based classification rule mining based models (ACNoEM and Harmony) without EM optimization are performing very similarly. Third, our ACEM model is shown to perform much better than the state of the art AC models (Harmony, BCAR, and CPAR) on majority of our datasets. Finally, ACEM outperforms SVM on four datasets and is performing worse than SVM on only one dataset. On the remaining fifteen datasets, ACEM and SVM perform similarly, which may seem discouraging at first glance. However, we want to point out that ACEM, as an AC model, is much easier to be interpreted by end users than SVM. In general, the penalty of being more interpretable is to sacrifice performance, as is the case for the other three AC models. However, ACEM overcomes this penalty and can achieve similar and sometimes even better performances than SVM.

Chapter 4

Instance Based Itemset Mining

The first component of AREM and ACEM modeling is to mine the set of high quality itemsets $\mathcal{X}$. We have explained the instance based approach that defines this set of itemsets $\mathcal{X}$, without studying how to mine them from the algorithmic point of view. This itemset mining problem is the topic of this chapter.

The naive method for mining the set of itemsets $\mathcal{X}$ is a two step approach, in which the first step is to apply the standard frequent itemset mining algorithm (e.g., Apriori [7] or FP-growth [8]) to mine the complete set of frequent itemsets $\mathcal{F}$, and the second step is to apply the instance based strategy to mine $\mathcal{X}_i \subseteq \mathcal{F}_i \subseteq \mathcal{F}$ for each training instance i so that the final set of itemsets is $\mathcal{X} = \cup_i \mathcal{X}_i$. However, the frequent itemset mining step can be computationally very expensive for dense datasets, and a huge number of itemsets can be generated which not only occupies large disk space but also presents challenges for the following instance based step. For this, we develop the Instance Based Itemset Miner (IBIMiner) outlined in Algorithm 1 for directly mining the instance-based sets of itemsets $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N$, without having to first generate the complete set of frequent itemsets. Note that our proposed IBIMiner can be used for both regression and classification's itemset mining problems. We only need to treat the classification's case as providing a different quality function, which has to be handled separately.

Algorithm 1 Instance Based Itemset Miner

```
Input: quality function $Q$, prefix itemset $x_p$, conditional database $\mathcal{D}_{x_p}$, instance index set $\mathcal{I}_{x_p}$, instance $i$'s
    current set of itemsets $\mathcal{X}_i$ for $\forall i$
Output: updated set $\mathcal{X}_i$ for $\forall i$
 1: prune infrequent items in $\mathcal{D}_{x_p}$
 2: prune support equivalence items in $\mathcal{D}_{x_p}$
 3: compute $\mu_{x_p}$, $\sigma_{x_p}$ or conf($x_p \rightarrow y$) $\forall y$ for the regression or classification problems, respectively.
 4: for each instance $i$ in $\mathcal{I}_{x_p}$ do
 5:     $Q(x_p, y_i) \leftarrow$ quality of itemset $x_p$ for instance $i$
 6:     $Q_i^{thr} \leftarrow$ quality threshold for instance $i$
 7:     if $Q(x_p, y_i) \geq Q_i^{thr}$ then
 8:         update $\mathcal{X}_i$ by adding the new itemset $x_p$
 9:     else
10:         $B_i^{x_p} \leftarrow$ upper bound of qualities $Q(x_p \cup x, y_i)$ for any frequent itemsets $x_p \cup x$ (i.e., $s_{x_p \cup x} \geq s_0$) to be
            discovered from $\mathcal{D}_{x_p}$ such that $x_p \cup x \subseteq \tau_i$.
11:         if $B_i^{x_p} < Q_i^{thr}$ then
12:             remove $i$ from $\mathcal{I}_{x_p}$
13:         end if
14:     end if
15: end for
16: if $|\mathcal{I}_{x_p}| > 0$ then
17:     while there exists more items in $\mathcal{D}_{x_p}$ do
18:         $m \leftarrow$ next item in $\mathcal{D}_{x_p}$, next prefix itemset $x'_p \leftarrow x_p \cup \{m\}$
19:         $\mathcal{I}_{x'_p} \leftarrow$ indices in $\mathcal{I}_{x_p}$ whose instances in $\mathcal{D}_{x_p}$ contain $m$
20:         if $|\mathcal{I}_{x'_p}| > 0$ then
21:             $\mathcal{D}_{x'_p} \leftarrow$ instances in $\mathcal{D}_{x_p}$ which contain $m$
22:             prune item $m$ from $\mathcal{D}_{x'_p}$
23:             run Instance Based Itemset Miner with $Q$, $x'_p$, $\mathcal{D}_{x'_p}$, $\mathcal{I}_{x'_p}$ and $\mathcal{X}_i \forall i$
24:         end if
25:         prune item $m$ from $\mathcal{D}_{x_p}$
26:     end while
27: end if
```

4.1 IBIMiner Algorithm

The IBIMiner algorithm follows the depth-first approach and grows the current itemset by adding to it one item at a time [70]. However, besides frequency, it also employs various methods to determine if any extensions of the current itemset can potentially lead to itemsets that are better than those discovered thus far, and use this information to terminate the temset generation early. This can dramatically reduce the total number of itemsets that need to be considered and lead to considerable reductions in runtime.

The prefix itemset x_p is the current itemset. The conditional Database $\mathcal{D}_{x_p}$ consists of the subset of instances from $\mathcal{D}_0$ which contain x_p. The instance index set $\mathcal{I}_{x_p}$ contains indices of a subset of instances in $\mathcal{D}_{x_p}$. These are the set of instances i that we need to update their corresponding $\mathcal{X}_i$, which contains the current set of top K quality itemsets for instance i, mined from the itemsets enumerated so far. We initialize x_p to be the empty set, $\mathcal{D}_{x_p}$ to be $\mathcal{D}_0$, $\mathcal{I}_{x_p}$ to be the set of indices of all training instances, and $\forall i$, $\mathcal{X}_i$ to be the empty set. The goal of the IBIMiner algorithm is to enumerate all itemsets of the form $x_p \cup x$, where x is an itemset from $\mathcal{D}_{x_p}$, and update set $\mathcal{X}_i$ for all instances i in $\mathcal{I}_{x_p}$. Given the initialization above, when the algorithm completes, we only need to take the union of all $\mathcal{X}_i$ to get set $\mathcal{X}$.

For the current itemset x_p, whose μ_{x_p}, σ_{x_p} or conf$(x_p \to y)$ are computed for regression or classification problem in line 3, the algorithm iterates through each instance i in $\mathcal{I}_{x_p}$ (line 4) and attempts to update $\mathcal{X}_i$ by adding x_p to it (lines 5-8). It first computes the quality $Q(x_p, y_i)$ of x_p (line 5) and then computes the quality threshold Q_i^{thr} for instance i (lines 6). The threshold Q_i^{thr} is taken to be negative infinity if $|\mathcal{X}_i| < K$. Otherwise, it is the lowest quality of the itemsets in $\mathcal{X}_i$. Next if $Q(x_p, y_i) \geq Q_i^{thr}$, x_p is added to $\mathcal{X}_i$ (line 8), which replaces the lowest quality itemset in $\mathcal{X}_i$ if $|\mathcal{X}_i| \geq K$.

We apply three pruning strategies to aggressively reduce the search space: infrequent items pruning (line 1), support equivalence items pruning (line 2), and quality upper bound pruning (lines 10-12). The quality upper bound pruning tries to reduce the size of the index set $\mathcal{I}_{x_p}$ by finding the upper bound $B_i^{x_p}$ of the qualities of frequent itemsets $x_p \cup x$ to be mined from $\mathcal{D}_{x_p}$ such that $x_p \cup x \subseteq \tau_i$ (line 10). If $B_i^{x_p} < Q_i^{thr}$, we can be sure that $\mathcal{X}_i$ is finalized, so i can be removed from $\mathcal{I}_{x_p}$ (line 12).

The remaining of the algorithm is to iterate through each item m in $\mathcal{D}_{x_p}$ and add it to the current prefix itemset x_p to form the new prefix $x'_p \leftarrow x_p \cup \{m\}$ (line 18), and then construct the new conditional database $\mathcal{D}_{x'_p}$ (line 21) and index set $\mathcal{I}_{x'_p}$ (line 19), and call the IBIMiner algorithm recursively (line 23). Due to the quality upper bound pruning strategy (line 10), it may be the case that $\mathcal{I}_{x'_p}$ is empty while $\mathcal{D}_{x'_p}$ is not. In that case, there is no need to call the IBIMiner algorithm on $\mathcal{D}_{x'_p}$, and the rest of the search space is pruned away.

4.1.1 Infrequent Items Pruning

Infrequent items pruning is based on the anti-monotone property of the support measure [7]. If a pattern is infrequent, any of its super patterns has to be infrequent. Consider an item m whose support computed in conditional database $\mathcal{D}_{x_p}$ is less than s_0. The support of m in $\mathcal{D}_{x_p}$ is the support of itemset $x_p \cup \{m\}$ in the original database $\mathcal{D}_0$. So we have $s_{x_p \cup \{m\}} < s_0$. Any itemset to be discovered that contains m is of the form

$x_p \cup x \cup \{m\}$. Due to $s_{x_p \cup x \cup \{m\}} \leq s_{x_p \cup \{m\}} < s_0$, we conclude that all these itemsets of the form $x_p \cup x \cup \{m\}$ can be pruned, which is equivalent to removing m from $\mathcal{D}_{x_p}$.

4.1.2 Support Equivalence Items Pruning

The support equivalence item m is defined as the item whose support computed in the conditional database $\mathcal{D}_{x_p}$ is equal to the size of database $\mathcal{D}_{x_p}$. From this definition, m appears in all instances of $\mathcal{D}_{x_p}$, so that $x \cup \{m\}$ is contained in the same set of instances in $\mathcal{D}_{x_p}$ as itemset x. Equivalently, $x_p \cup x \cup \{m\}$ is contained in the same set of instances in $\mathcal{D}_0$ as $x_p \cup x$. So the quality of the itemset $x_p \cup x$ is the same as the quality of $x_p \cup x \cup \{m\}$. The support equivalence items pruning suggests that these items m should be pruned, since the discovered itemsets can get shorter and thus have better generalization without losing qualities.

4.1.3 Quality Upper Bound Pruning

In line 10, the algorithm tries to find an upper bound of the quality $Q(x_p \cup x, y_i)$ for any x in $\mathcal{D}_{x_p}$ satisfying that: (1) itemset $x_p \cup x$ is frequent (i.e., $s_{x_p \cup x} \geq s_0$) and (2) itemset $x_p \cup x$ is in $\mathcal{F}_i$ (i.e., $x_p \cup x \subseteq \tau_i$). We need to solve this bounding problem for four quality functions Q^σ, Q^m, Q^n and Q^c. Even though this bounding problem is formulated in the original database $\mathcal{D}_0$, we note that it can be solved completely in the conditional database $\mathcal{D}_{x_p}$. This follows by observing that (i) the itemset quality $Q(x_p \cup x, y_i)$ and support $s_{x_p \cup x}$ are only dependent on the instances containing $x_p \cup x$ which are in $\mathcal{D}_{x_p}$, and (ii) τ_i is in $\mathcal{D}_{x_p}$ (due to i is in $\mathcal{I}_{x_p}$) so that the constraint $x_p \cup x \subseteq \tau_i$ can be simplified to $x \subseteq \tau_i$ (due to $x_p \subseteq \tau_i$). Because of these, to simplify the notation, we drop the prefix itemset x_p in the rest of the discussions of this section. And we formulate the problem formally in an arbitrary database $\mathcal{D}$ (which stands for $\mathcal{D}_{x_p} \forall x_p$) in the following:

Problem 4.1.1. *The problem of finding the quality upper bound in database $\mathcal{D}$ with respect to the instance $(\tau, y) \in \mathcal{D}$ is to find quantity $\mathcal{B}(\tau, y)$ so that for any $x \subseteq \tau$ satisfying $s_x \geq s_0$, we have $\mathcal{B}(\tau, y) \geq Q(x, y)$.*

Instance Group's Formulation

To solve Problem 4.1.1, we need to evaluate $Q(x, y)$ for different choices of itemsets x. However, the number of all possible itemsets can be exponential. We need approaches that can find the quality upper bound, without enumerating all possible itemsets x. Our strategy is to map the itemset x to the set of instances containing x. Specifically, for any instance i in $\mathcal{D}$, define $I_i = 1$ if $x \subseteq \tau_i$ and $I_i = 0$ otherwise. In this way, x is mapped to N binary variables $I_1, \dots, I_N$. Problem 4.1.1 can be transformed to a bounding problem involving only this set of new variables.

However, our first attempt in applying the above strategy did not give us satisfactory performance. Then we realized that there was a simple yet very effective constraint among these I_is that we did not take into account: for any instances i and j with $\tau_i = \tau_j$, we always have $I_i = I_j$ regardless of what x is. This motivates us to group instances with the same itemsets together into instance groups:

Definition 4.1.2. *We partition all the instances of database $\mathcal{D}$ into a set of instance groups (τ_g, Y_g), where instance group g contains all instances whose itemsets are τ_g, and Y_g is a set of target variables for these instances. Let s_g to be the number of instances in group g, i.e., $s_g = |Y_g|$. For the regression dataset, define (μ_g, σ_g) to be the (mean, standard deviation) of target variables for instances in group g. For the classification dataset, define s_g^y to be the number of instances in instance group g whose class labels are y.*

For any instance group g, we define $I_g = 1$ if $x \subseteq \tau_g$ and $I_g = 0$ otherwise, so that the original N variables I_is are now mapped to a smaller number of binary variables I_gs. Next, we focus on reformulating different parts of Problem 4.1.1 in terms of the new variables (I_gs).

Instance Group's Formulation Of $s_x \geq s_0$ And $x \subseteq \tau$ From Definition 4.1.2, the support s_x of itemset x can be rewritten as

$$s_x = \sum_i I_i = \sum_g \sum_{i:\tau_i=\tau_g} I_i = \sum_g \sum_{i:\tau_i=\tau_g} I_g = \sum_g I_g s_g, \tag{4.1}$$

so that the support constraint $s_x \geq s_0$ is equivalent to

$$\sum_g I_g s_g \geq s_0. \tag{4.2}$$

Now we consider $x \subseteq \tau$. Notice that (τ, y) is itself a training instance and belongs to some group g_0, so that $\tau = \tau_{g_0}$. From the constraint $x \subseteq \tau$, we have $x \subseteq \tau_{g_0}$ so that,

$$I_{g_0} = 1. \tag{4.3}$$

Instance Group's Formulation Of $Q^\sigma(x, y)$ Next, we reformulate (the upper bound) of the three quality functions in terms of I_gs. From $Q^\sigma(x, y) = 1/\sigma_x$, we only need to reformulate (the lower bound) of σ_x for the quality function Q^σ. From Definition 4.1.2, we have

$$\begin{aligned}
\sigma_x^2 = \frac{1}{s_x} \sum_i I_i (y_i - \mu_x)^2 & \\
= \quad \frac{1}{s_x} \sum_g I_g \sum_{i:\tau_i = \tau_g} (y_i - \mu_x)^2 & \\
= \quad \frac{1}{s_x} \sum_g I_g \sum_{i:\tau_i = \tau_g} [(y_i - \mu_g)^2 + (\mu_g - \mu_x)^2] & \\
\geq \quad \frac{1}{s_x} \sum_g I_g \sum_{i:\tau_i = \tau_g} (y_i - \mu_g)^2 = \frac{1}{s_x} \sum_g I_g s_g \sigma_g^2. & \qquad (4.4)
\end{aligned}$$

By plugging Equation 4.1 into Equation 4.5, we get

$$\sigma_x^2 \geq \frac{\sum_g I_g s_g \sigma_g^2}{\sum_g I_g s_g}. \tag{4.5}$$

In order to find the upper bound of $Q^\sigma(x, y)$, we only need to find the lower bound of the right hand side (RHS) of Equation 4.5.

Instance Group's Formulation Of $Q^m(x, y)$ A naive approach can be derived by replacing $(\mu_x - y)^2$ with zero in the definition of $Q^m(x, y)$ (Equation 3.4). After this, we get $Q^m(x, y) \leq 1/\sigma_x = Q^\sigma(x, y)$. So the approach we developed for $Q^\sigma(x, y)$ can also be used for finding the upper of $Q^m(x, y)$. However, this approach ignores the target variable dependent term, which which once being included can improve the bound greatly. From Equation 3.4, in order to find the upper bound of $Q^m(x, y)$, we only need to find the lower bound of $MSE(x, y)$. So we reformulate $MSE(x, y)$ into the instance group's representation:

$$\begin{aligned}
MSE(x, y) &= \frac{1}{s_x}\sum_i I_i(y_i - y)^2 \\
&= \frac{1}{s_x}\sum_g I_g \sum_{i:\tau_i=\tau_g}(y_i - y)^2 \\
&= \frac{1}{s_x}\sum_g I_g \sum_{i:\tau_i=\tau_g}[(y_i - \mu_g)^2 + (\mu_g - y)^2] \\
&= \frac{\sum_g I_g s_g[\sigma_g^2 + (\mu_g - y)^2]}{\sum_g I_g s_g}. \qquad (4.6)
\end{aligned}$$

In order to find the upper bound of $Q^m(x, y)$, we only need to find the lower bound of the RHS of Equation 4.6.

Instance Group's Formulation Of $Q^n(x, y)$ Similarly, a naive approach can be derived by replacing $(\mu_x - y)^2$ with zero in the definition of $Q^n(x, y)$ (Equation 3.5). After this, we get $Q^n(x, y) \leq 1/\sigma_x = Q^\sigma(x, y)$. So the approach we developed for $Q^\sigma(x, y)$ can also be used for finding the upper of $Q^n(x, y)$. However, this approach did not give us satisfactory performance due to that Q^σ as an upper bound of Q^n is not tight enough. To find a tighter upper bound, we notice that Q^m in Equation 3.4 can be rewritten as

$$Q^m(x, y) = \frac{1}{\sigma_x}\frac{1}{\sqrt{1 + (\mu_x - y)^2/\sigma_x^2}}. \qquad (4.7)$$

By comparing Q^m in Equation 4.7 to Q^n in Equation 3.5, we observe the following similarities: (i) the first term $1/\sigma_x$ is the same, and (ii) the second term is dependent on the same quantity $(\mu_x - y)^2/\sigma_x^2$. These similarities motivate us to consider the relationship between the two quality functions. To simplify the notation, let $a \leftarrow (\mu_x - y)^2/\sigma_x^2$. We have

$$(Q^n(x, y))^2 = \frac{1}{\sigma_x^2 e^a} \leq \frac{1}{\sigma_x^2(1 + a)} = (Q^m(x, y))^2,$$

where we have utilized $e^a \geq 1 + a$ for $a \geq 0$. This derivation tells us that $Q^m(x, y)$ is an upper bound of $Q^n(x, y)$. In fact, $Q^m(x, y)$ is also a tighter upper bound of Q^n than

Q^σ due to $Q^m(x,y) \le Q^\sigma(x,y)$. The fact that Q^m is an upper bound of Q^n implies that the same instance group's representation in Equation 4.6 can be used for finding the upper bound of Q^n.

Instance Group's Formulation Of $Q^c(x,y)$ From $Q^c(x,y) = \mathrm{conf}(x \to y)$ and the definition of the confidence, we have

$$Q^c(x,y) = \frac{s_x^y}{s_x},$$

where

$$s_x^y = \sum_i I_i I_{y_i=y} = \sum_g \sum_{i:\tau_i=\tau_g} I_g I_{y_i=y} = \sum_g I_g s_g^y.$$

Combined with the Equation 4.1, we get

$$Q^c(x,y) = \frac{\sum_g I_g s_g^y}{\sum_g I_g s_g}. \tag{4.8}$$

In summary, two bounding constraints in Problem 4.1.1 are now reformulated as Equations 4.2 and 4.3. The upper bounding problem of the four quality functions are also transformed into the lower bounding problem of the RHS of Equations 4.5 and 4.6, and the upper bounding problem of the RHS of Equation 4.8. So Problem 4.1.1 can now be fully represented in the instance group's notation. In addition, notice that the RHS of Equations 4.5, 4.6 and 4.8 are of the same form:

$$\sigma_x^2 \ge \frac{\sum_g I_g v_g}{\sum_g I_g h_g}, \text{ with } v_g \leftarrow s_g \sigma_g^2,\ h_g \leftarrow s_g,$$

$$MSE(x,y) = \frac{\sum_g I_g v_g}{\sum_g I_g h_g}, \text{ with } v_g \leftarrow s_g[\sigma_g^2 + (\mu_g - y)^2],\ h_g \leftarrow s_g,$$

and,

$$Q^c(x,y) = \frac{\sum_g I_g v_g}{\sum_g I_g h_g}, \text{ with } v_g \leftarrow s_g^y,\ h_g \leftarrow s_g.$$

So, we only need to focus on finding the lower and upper bound of the function of the generic form $\sum_g I_g v_g / \sum_g I_g h_g$.

In the above derivations, the values of variables I_gs are determined by the itemset x. Now, we decouple this connection to allow I_gs to change freely except for satisfying

Equations 4.2 and 4.3. It is this decoupling that avoids the need of searching the exponential space of x. To sum things up, we only need to focus on solving the following problem:

Problem 4.1.3. *Given $h_g > 0$ and v_g for each instance group g, our goal is to find the lower and upper bound of the quantity*

$$\frac{v_{g_0} + \sum_{g \neq g_0} I_g v_g}{h_{g_0} + \sum_{g \neq g_0} I_g h_g}, \tag{4.9}$$

for any $I_g \in \{0,1\}$ $(g \neq g_0)$ satisfying $h_{g_0} + \sum_{g \neq g_0} I_g h_g \geq s_0$.

Note that the instance group g_0 is isolated to be always present due the constraint in Equation 4.3.

The Geometric Formulation And Solutions

We find that it is much easier to present the solution and proof to the Problem 4.1.3 in the geometric representation, which we explain in the following. For each instance group g, let $\mathbf{e}_g$ be a two-dimensional vector such that its horizontal component $e_{g,h} = h_g$ and vertical component $e_{g,v} = v_g$. Let $\mathbf{E} = \mathbf{e}_{g_0} + \sum_{g \neq g_0} I_g \mathbf{e}_g$. We have that our objective function in Equation 4.9 is the slope of $\mathbf{E}$, which is denoted as $||\mathbf{E}|| = E_v/E_h$. In addition, the constraint in Equation 4.2 becomes a constraint on the horizontal component of $\mathbf{E}$: $E_h \geq s_0$. So the Problem 4.1.3 can translated into a problem of finding the lower and upper bound of the slope of vector $\mathbf{E}$ satisfying $E_h \geq s_0$. This problem is formally defined as:

Problem 4.1.4. *Let $\{\mathbf{e}_g | g = 1, \ldots, n\}$ be a sequence of 2-D vectors defined above, satisfying $e_{g,h} > 0$ (i.e., $s_g > 0$) and $\sum_g e_{g,h} \geq s_0$ (i.e., $|\mathcal{D}| \geq s_0$). Given $1 \leq g_0 \leq n$, the problem is to find the lower and upper bound of $||\mathbf{E}||$, where $\mathbf{E} = \mathbf{e}_{g_0} + \sum_{g \neq g_0} I_g \mathbf{e}_g$ for any $I_g \in \{0,1\}$ $(g \neq g_0)$ satisfying $E_h \geq s_0$.*

The solution to Problem 4.1.4 is described in the following:

Theorem 4.1.5. *Assume the notations introduced for Problem 4.1.4. Without loss of generality, we assume they are ordered as*

$$||\mathbf{e}_1|| \leq \cdots \leq ||\mathbf{e}_g|| \leq \cdots \leq ||\mathbf{e}_n||,$$

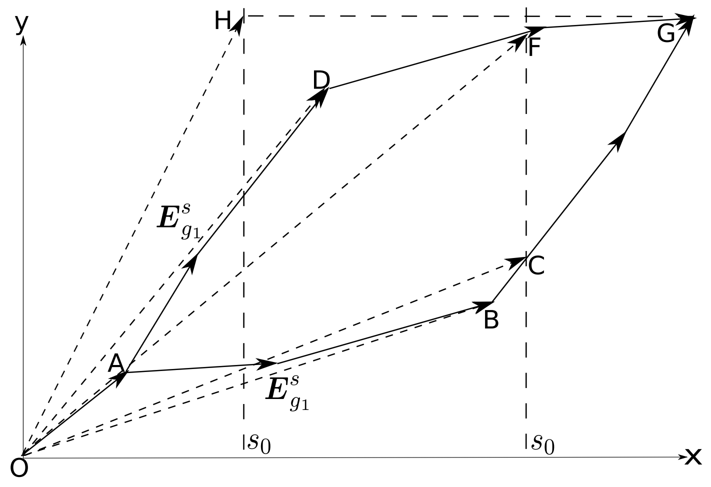

Figure 4.1: Quality Lower/Upper Bound: the complete version

for the lower bounding problem or ordered as

$$||\mathbf{e}_1|| \geq \cdots \geq ||\mathbf{e}_g|| \geq \cdots \geq ||\mathbf{e}_n||,$$

for the upper bounding problem. Define a new vector sequence

$$\mathbf{E}^s_{g'} = \mathbf{e}_{g_0} + \sum_{g \neq g_0 \wedge g \leq g'} \mathbf{e}_g,$$

for $g' = 0, 1, 2, \ldots, g_0 - 1, g_0 + 1, \ldots, n$. *Let* g_1 *to be the first index* g' *whose vector's slope in this sequence is not higher (for the lower bounding problem) or not lower (for the upper bounding problem) than the next vector's slope if it exists, and the last index otherwise. If* $E^s_{g_1,h} \geq s_0$*, define* $\mathbf{E}_c = \mathbf{E}^s_{g_1}$*. Otherwise, find* $g_c > g_1$*,* $g_c \neq g_0$ *and* $0 < \delta_c \leq 1$ *such that*

$$\mathbf{E}_c = \mathbf{E}^s_{g_1} + \sum_{g_1 < g < g_c \wedge g \neq g_0} \mathbf{e}_g + \delta_c \mathbf{e}_{g_c}$$

satisfies $E_{c,h} = s_0$*. We have* $||\mathbf{E}_c||$ *is the solution to problem 4.1.4 for both the lower bounding and the upper bounding problem.*

Proof. See Appendix B.3. □

Theorem 4.1.5 is graphically shown in Figure 4.1, which contains five instance groups. The vector OA represents $\mathbf{e}_{g_0}$. The convex region ABGDA is formed by connecting $\mathbf{e}_g$

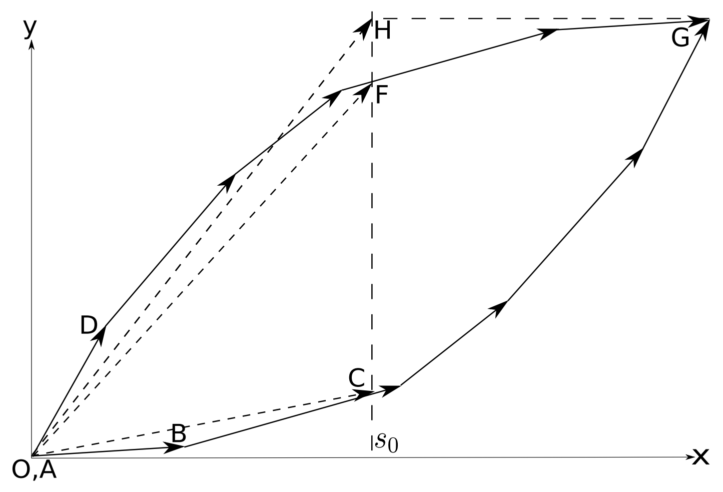

Figure 4.2: Quality Lower/Upper Bound: the relaxed version

($g \neq g_0$) sequentially in the increasing order (to form ABC) and decreasing order (to form ADG) of slopes. The minimum support vertical line s_0 splits this region into two parts. The right part of ABGDA is the region which $\mathbf{E}$ can fall into. The vector OB and OD is the $\mathbf{E}^s_{g_1}$ defined in Theorem 4.1.5 for the lower bounding and upper bounding problems, respectively. The slope of OB (OD) represents the global lower (upper) bound for slopes of vectors that can fall into the convex region ABGDA. When s_0 is smaller than the horizontal components of OB (OD), the slope of OB (OD) is the solution we are looking for. Otherwise, we can improve the bound, by adding more vectors into $\mathbf{E}^s_{g_1}$ until the horizontal component becomes s_0. The resulting vectors are represented by OC and OF for the lower bounding and upper bounding problems, respectively.

Theorem 4.1.5 provides a complex solution to Problem 4.1.4. The computational complexity may slow down the mining algorithm, even though its bound may be tight enough. Next, we present a relaxed solution to Problem 4.1.4 by relaxing the constraint $x \subseteq \tau$ or $I_{g_0} = 1$. The relaxed solution is easier to compute, but the bound itself is worse than the complete solution of Theorem 4.1.5. In the end, which bounding strategy (relaxed or complete) gives a shorter running time is data dependent.

Theorem 4.1.6. *Assume the notations introduced for Theorem 4.1.5. Find index g_r and $0 < \delta_r \leq 1$ such that*

$$\mathbf{E}_r = \mathbf{e}_1 + \mathbf{e}_2 + \cdots + \delta_r \mathbf{e}_{g_r}$$

satisfies $E_{r,h} = s_0$. We have, $||\mathbf{E}_r||$ is the (relaxed) solution to problem 4.1.4 for both the lower bounding and the upper bounding problem.

Proof. See Appendix B.2. □

Theorem 4.1.6 is graphically shown in Figure 4.2. Due the absence of instance group g_0, the point O is now overlapping with the point A. The convex region ABGDA is formed in the same way as in Figure 4.1, except that all vectors $\mathbf{e}_g$ are involved for this time. The two vectors OB and OD, which represents $\mathbf{E}^s_{g_1}$ in Theorem 4.1.5 for the lower bounding and upper bounding problems, now simply become the vector $\mathbf{e}_g$ with the lowest and highest slopes, respectively. Because of this, $\mathbf{E}^s_{g_1}$ is not participating in Theorem 4.1.6 anymore, which simplifies the algorithm. The solutions of Theorem 4.1.6 are represented by slopes of OC and OF for the lower bounding and upper bounding problems, respectively.

The following theorem states that the bound is improved by taking into account the constraint $x \subseteq \tau$ or $I_{g_0} = 1$:

Theorem 4.1.7. *Let $\mathbf{E}_r$ and $\mathbf{E}_c$ be defined in theorems 4.1.6 and 4.1.5, respectively. We have*

$$||\mathbf{E}_r|| \leq ||\mathbf{E}_c||,$$

for the lower bounding problem, and,

$$||\mathbf{E}_r|| \geq ||\mathbf{E}_c||,$$

for the upper bounding problem.

Proof. See Appendix B.4. □

Intuitively, Theorem 4.1.7 follows from the observation that the convex region ABGDA in Figure 4.1 falls inside the region ABGDA in Figure 4.2.

Methods For Finding Quality Upper Bounds

In the remaining of this section, we summarize the methods we have developed so far for computing the upper bounds of the four quality functions.

To find the upper bounds for qualities $Q^\sigma(x, y)$ and $Q^m(x, y)$ ($Q^n(x, y)$), we first find the lower bounds of σ_x^2 and $MSE(x, y)$, respectively. Then the upper bounds are the inverse of the squared root of the computed lower bounds. To compute the lower bounds, we define $v_g \leftarrow s_g\sigma_g^2$, $h_g \leftarrow s_g$ and $v_g \leftarrow s_g[\sigma_g^2 + (\mu_g - y)^2]$, $h_g \leftarrow s_g$ respectively. And then we apply Theorems 4.1.5 and 4.1.6 to the RHS of Equations 4.5 and 4.6 respectively. We denote the methods for computing the upper bound of Q^σ (Q^m and Q^n) be $\mathcal{B}_c^{sig}$ and $\mathcal{B}_r^{sig}$ ($\mathcal{B}_c^{mse}$ and $\mathcal{B}_r^{mse}$), which applies Theorems 4.1.5 and 4.1.6, respectively.

However, there is a computational limitation of the methods $\mathcal{B}_c^{mse}$ and $\mathcal{B}_r^{mse}$. In these two approaches, we need to sort instance groups $\mathbf{e}_g$ by $||\mathbf{e}_g|| = \sigma_g^2 + (\mu_g - y)^2$. The problem is that each instance may potentially have a different y, and thus have a different ordering of instance groups. We have to sort instance groups for each instance, which slows down the algorithm. For this reason, We derive another two bounding strategies for Q^m and Q^n as follows. We first split MSE into the sum of two terms:

$$MSE(x, y) = \frac{\sum_g I_g s_g \sigma_g^2}{\sum_g I_g s_g} + \frac{\sum_g I_g s_g(\mu_g - y)^2}{\sum_g I_g s_g}. \tag{4.10}$$

Next, we apply Theorems 4.1.5 and 4.1.6 to find the lower bound of each term in Equation 4.10 by setting $e_{g,v} \leftarrow s_g\sigma_g^2$, $e_{g,h} \leftarrow s_g$ and $e_{g,v} \leftarrow s_g(\mu_g - y)^2$, $e_{g,h} \leftarrow s_g$ respectively. Finally, the MSE lower bound is the sum of lower bounds computed for these two terms. We call these two approaches $\mathcal{B}_c^{add}$ and $\mathcal{B}_r^{add}$, which applies Theorems 4.1.5 and 4.1.6, respectively. These two approaches are computationally more efficient due to that we only need to sort σ_g^2 and μ_g once while applying the Theorems 4.1.5 and 4.1.6. On the other hand, the *mse* approaches are more likely to generate better bounds, since they take into account the interaction between the two terms in Equation 4.10.

To find the upper bound for the quality $Q^c(x, y)$ of the classification problem, we can apply theorems 4.1.5 and 4.1.6 directly to find the upper bound by setting $v_g \leftarrow s_g^y$ and $h_g \leftarrow s_g$. We call these two approaches $\mathcal{B}_c^{conf}$ and $\mathcal{B}_r^{conf}$, which applies Theorems 4.1.5 and 4.1.6, respectively. Besides these two upper bounds for confidence, we derive another simpler upper bound used by Harmony [24] to prune the search space:

$$Q^c(x, y) = \text{conf}(x \to y) = \frac{s_x^y}{s_x} \le \frac{s_x^y}{s_0} \le \frac{s_\emptyset^y}{s_0}.$$

Combined with conf$(x \to y) \leq 1$, we get

$$Q^c(x, y) \leq \min\,(\frac{s_\emptyset^y}{s_0}, 1). \tag{4.11}$$

Here $s_\emptyset^y$ is effectively the number of instances with $y_i = y$ in the database. We call the bounding approach using the RHS of Equation 4.11 as $\mathcal{B}_h^{conf}$.

To show how the upper bound $s_\emptyset^y/s_0$ is related to the solutions presented in Theorems 4.1.5 and 4.1.6, we rewrite $s_\emptyset^y$ as

$$s_\emptyset^y = \sum_g s_g^y = \sum_g v_g = \sum_g e_{g,v},$$

where we have applied the substitution $e_{g,v} = v_g = s_g^y$ for the quality function $Q^c(x, y)$. So, in the geometric space, $s_\emptyset^y$ is the summation of all $\mathbf{e}_g$s' vertical components. Based on this analysis, the upper bound $s_\emptyset^y/s_0$ is shown to be the slope of the vector OH in Figures 4.1 and 4.2. One can see that this upper bound is worse than the solutions provided by Theorems 4.1.5 and 4.1.6.

4.1.4 Implementations

Our implementation of the IBIMiner algorithm is based on the FP-growth approach and utilizes a modified version of FP-Tree [71] data structure.

FP-Tree is a compact representation of the transaction database so that it can fit into the main memory. The FP-growth algorithm performs just two passes over the database. First, it scans the database to count the support of all items. These items are then sorted according to their support from large to small. Items in each instance are also sorted according to this order. Next, given an empty FP-Tree, the FP-growth algorithm scans the database for the second time and insert instances one by one to the FP-Tree. Each instance's itemset maps into a path of the FP-Tree. Each node on the path represents an item and is associated with a count of how many times that node has been visited. The nodes with the same item are linked, and the summation of their counts is the support of the item.

During the rule discovery process, FP-Tree facilitates the items pruning and conditional database construction. To prune an item, one simply find the head of the linked list for that item, and walk down the list to remove each node at a time. It may be necessary to combine nodes and update node counts. To construct the database conditional on item m, one creates a new FP-Tree by inserting all paths containing m from the original FP-Tree to the new tree.

FP-Tree was originally designed for frequent itemset mining. In this work, we extend it to deal with the regression and classification data. We observe that each instance group (Definition 4.1.2) is mapped to a unique path of the FP-Tree. We append a leaf node below each instance group and store additional data within it. In particular, each leaf node for group g stores s_g, μ_g, σ_g for the regression data and s_g, s_g^y $\forall y$ for the classification data. It also stores the set of instance indices within that group. All the leaf nodes are also linked just like other items. Notice that leaf nodes may be merged during items pruning and conditional database construction, in which case, the data stored within the leaf node needs to be updated accordingly. The union of all instance indices stored in leaf nodes is what we denoted as $\mathcal{I}_x$ in Algorithm 1. Because of the itemset quality upper bound approach developed above, the number of indices stored in the leaf node may be smaller than the number of actual instances for that group. Also notice that how appending the leaf nodes helps us to compute rule quality bounds developed in Theorem 4.1.5 and 4.1.6.

4.2 Experimental Evaluation of IBIMiner Algorithm

We evaluate the effectiveness of the IBIMiner algorithm, by focusing on the effects of different quality upper bounding strategies on reducing running times and itemset search spaces. We also study how IBIMiner can scale to large databases.

4.2.1 Datasets

In order to evaluate IBIMiner with the quality functions (Q^σ, Q^m and Q^n) for the regression problem, we select four datasets (BestBuy.wdep, CitySearch.wdep, Yelp.wdep and MovieLens) from Table 3.1. The rest of the datasets are *easy* in that the running times are very short and there is no need for the advanced strategies. We add two more datasets (*SmileMTP* [72] and *census* [73]) to see how IBIMiner algorithm behaves on very *difficult* datasets. This will help us to gain more in-depth understanding of the nature of our algorithm, especially, the different bounding strategies.

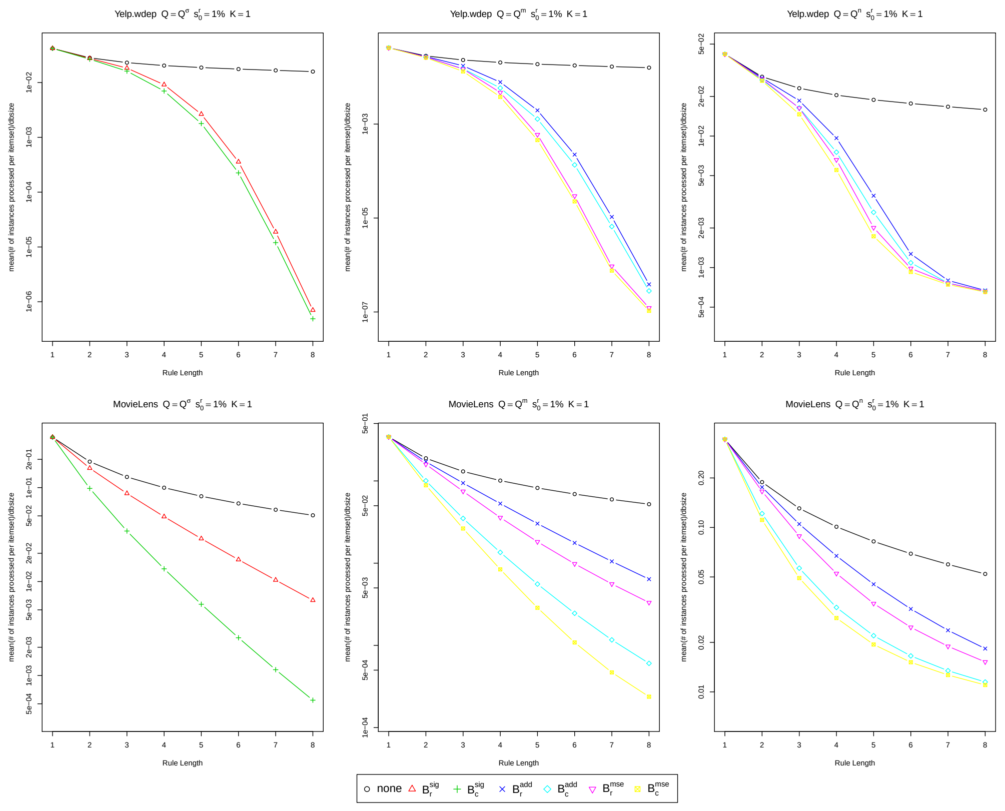

Figure 4.3: pruning effects: regression

SmileMTP The *SmileMTP* dataset comes from the domain of chemical informatics. The target variable is the melting point, which has been scaled to have mean zero and variance one. The features are extracted from the molecular structures. We use the OpenBabel [74] to convert molecules from the SMILES format to the SDF format, which are then parsed by AFGen [75] to generate a set of items. The *SmileMTP* dataset has 3559 instances, 9575 items, with density 2.4%.

Census The *census* dataset is a discretized version of 1990 US Census Data. The target variable is the *dHour89* which takes value from zero to five. We randomly sampled 8000 instances from the original datasets. The number of items and density are 385 and 17.4% respectively.

We evaluate the rule mining algorithm for the classification problem with quality

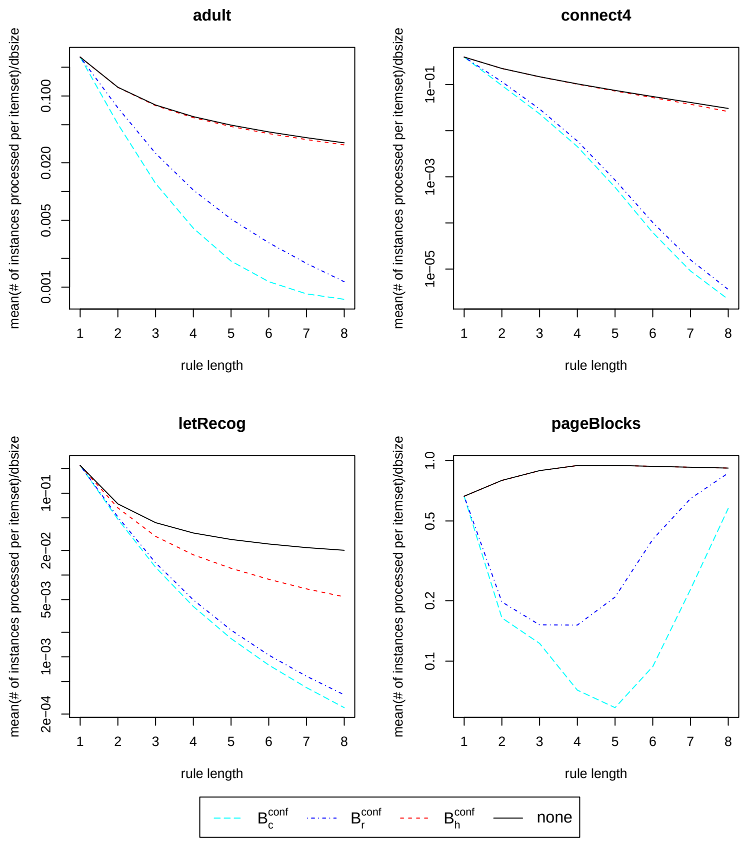

Figure 4.4: Pruning Effects ($s_0^r = 1.0\%$, $K = 1$): classification

function Q^c on four datasets: adult, connect4, letRecog and pageBlocks. The rest of the datasets in table 3.12 are easy from the rule mining's perspective: the running times are very small even when no bounding strategy is applied.

4.2.2 Running Time Comparison

Tables 4.1, 4.2, 4.3 and 4.4, report the running times for different quality bounding strategies on the different datasets for the four qualities Q^σ, Q^m, Q^n and Q^c, respectively (boldfaced entries correspond to the best performing results). Looking at these results, we can make a number of observations in the following.

For most quality functions under study, our best algorithms are more than one order of magnitude faster than the baseline approach on many datasets. For Q^σ, the

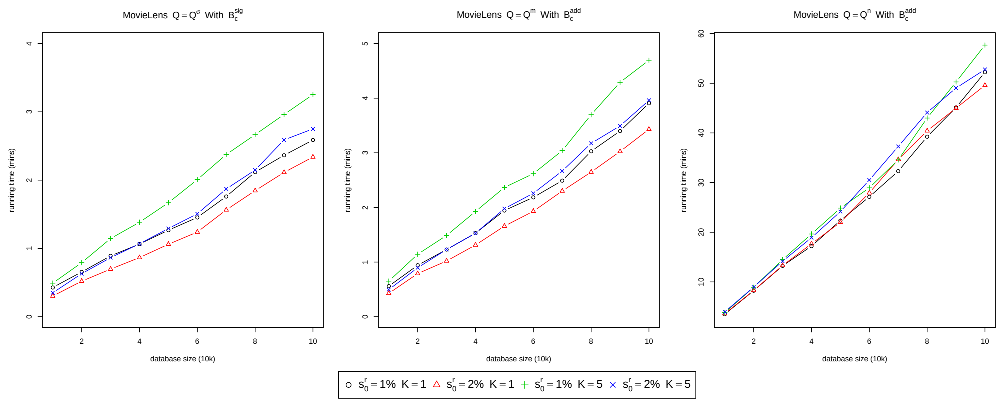

Figure 4.5: Scalability Study: regression

pruning strategy $\mathcal{B}_c^{sig}$ is more than an order of magnitude faster than the baseline on BestBuy.wdep, Yelp.wdep, MovieLens and census. For Q^m, the pruning strategies $\mathcal{B}_c^{mse}$ and $\mathcal{B}_c^{add}$ are more than an order of magnitude faster than the baseline on MovieLens, SmileMTP and census. For Q^c, the pruning strategy $\mathcal{B}_c^{conf}$ is more than an order of magnitude faster than the baseline on adult and connect4.

For all datasets and all parameters, our algorithm runs significantly faster when Q^m is used as the quality function instead of Q^n. This is due to that Q^m itself is an upper bound of Q^n, so our bounding strategies are tighter for Q^m than for Q^n. Note that we compare Q^m against Q^n because the same pruning strategies are applied for them and the predictive performances of these two qualities are very close.

For most cases, the bounding strategy *add* performs similarly to the bounding strategy *mse*. Despite that the *add* strategy is expected to have less pruning power, its bound is easier to compute, which helps the method to compete well with the *mse* strategy.

For the *MovieLens* dataset, the complete version of the bounding strategies $\mathcal{B}_c^{sig}$, $\mathcal{B}_c^{mse}$ and $\mathcal{B}_c^{add}$ are significantly better than the corresponding relaxed version strategies $\mathcal{B}_r^{sig}$, $\mathcal{B}_r^{mse}$ and $\mathcal{B}_r^{add}$. For other regression datasets, the complete version yields only slightly better results than the relaxed version. Our analysis shows that this is related to the fact that *MovieLens* is a dense dataset. So the performance of the complete version of the bounding strategy is improved due to that instance group g_0 contains many instances.

The dataset *census* represents the kind of datasets which are very *difficult* from the

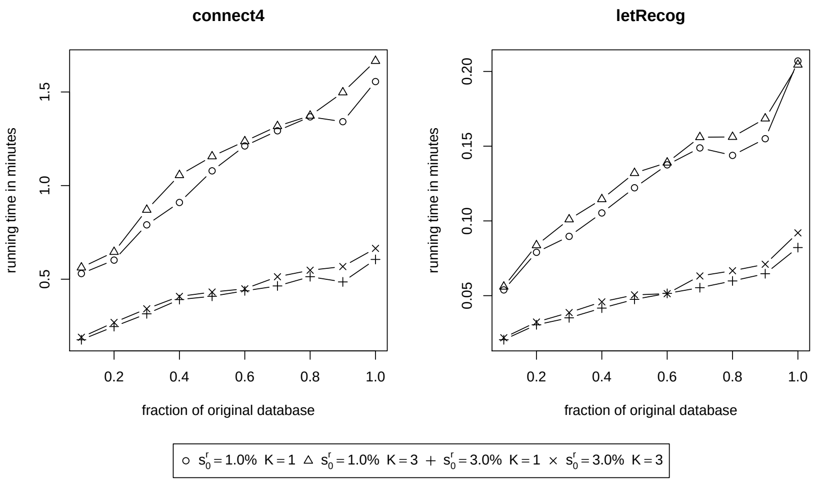

Figure 4.6: Scalability Study ($\mathcal{B}_c^{conf}$): classification

itemset mining's perspective. In fact, we have to set the minimum support to be very high (i.e., $s_0^r = 0.5$) in order to reasonably estimate the running time of the baseline approach. For this kind of datasets, in the case when the baseline approach takes more than four hours to finish, our bounding strategy can reduce the running time to a few seconds.

Our proposed pruning methods for instance based classification rule mining are significantly better than the pruning strategy applied by Harmony [24]. In particular, when $s_0^r = 1.0\%$ and $K = 1$, the running times of the complete bounding strategy (i.e., $\mathcal{B}_c^{conf}$) are 7.9%, 2.2%, 55.8%, and 14.3% of the running times of Harmony's bounding strategy (i.e., $\mathcal{B}_h^{conf}$), for datasets adult, connect4, letRecog, and pageBlocks, respectively.

The connect4 dataset contains some frequent items which slow down the rule mining process dramatically. In [24], Wang *et al.* removed these items to make the mining problem easier. In contrast, we keep them and are able to show that our bounding strategies can improve the running time from around an hour to only a minute or so.

Finally, we observe that for the classification datasets, the complete bounding strategy ($\mathcal{B}_c^{conf}$) performs slightly worse than the relaxed strategy ($\mathcal{B}_r^{conf}$) on dataset letRecog, but it is better than $\mathcal{B}_r^{conf}$ on the other three datasets. As we will see, $\mathcal{B}_c^{conf}$ prunes away more search space than $\mathcal{B}_r^{conf}$ does. But due to its more complex computation, whether it will produce a smaller running time is determined by the balance of these two factors and is dataset dependent.

4.2.3 Pruning Effects

To better illustrate how the different quality upper bound pruning strategies reduce the size of the search space, Figures 4.3 and 4.4 show the average number of training instances processed by each itemset and how these numbers evolve as the itemsets grow. Recall from Algorithm 1, that the number of instances processed by the itemset x is equal to the size of set $\mathcal{I}_x$ (lines 4-15), and this number will depend on the effectiveness of the pruning strategy. Let n_l be the number of itemsets of length l that are processed by the baseline approach when no quality bounding strategy is applied. Now, consider the case when the bounding strategy $\mathcal{B}$ (including the baseline: *none*) is applied. Let c_l be the sum of $|\mathcal{I}_x|$ for all itemsets x with length l. Then the average number of instances processed by an itemset with length l is $s_l = c_l/n_l$. In Figures 4.3 and 4.4, we first normalize s_l by the database size (*dbsize*), and then plot it against the itemset length l. The plot is on log-scale so that we can distinguish very small values. Note that only two datasets (Yelp.wdep and MovieLens) are presented in Figure 4.3 because other datasets we experimented show very similar trends as Yelp.wdep.

From Figures 4.3 and 4.4, we see that the number of instances processed per itemset is much smaller when the quality bounding strategy is applied. We can also see the pruning effects of the quality function Q^n, which could not be observed by simply counting the number of itemsets being processed. One can see that much more pruning is done for Q^m than for Q^n by looking at the range of Figure 4.3's vertical axis. For the *MovieLens* dataset, the pruning effects of $\mathcal{B}_c^{sig}$, $\mathcal{B}_c^{mse}$ and $\mathcal{B}_c^{add}$ are much stronger than those of $\mathcal{B}_r^{sig}$, $\mathcal{B}_r^{mse}$ and $\mathcal{B}_r^{add}$. This is consistent with the running time results we observed in Tables 4.1, 4.2 and 4.3. Additionally, when the quality function Q^m is applied, we see that the pruning effects of the *mse* approach are much stronger than those of the corresponding *add* approach. This observation is consistent with the Theorem 4.1.7. However, this is not reflected in the running time experiments. We think this is due to that the bound of the *mse* approach is more time consuming to compute than the bound of the *add* approach. From Figure 4.4, we can clearly rank the bounding strategies by pruning effects as $\mathcal{B}_c^{conf} \geq \mathcal{B}_r^{conf} \geq \mathcal{B}_h^{conf}$.

4.2.4 Scalability Study

To study how our algorithm scales to the size of regression datasets, we select *MovieLens*, since this is the dataset we have more than 100k instances. We randomly sampled ten datasets, with sizes ranging from $10k$ to $100k$. Then we run our IBIMiner algorithm with one of the best pruning strategies $\mathcal{B}_c^{add}$ for Q^m (Q^n) and $\mathcal{B}_c^{sig}$ for Q^σ on these datasets, and the results are summarized in Figure 4.5. It can be seen that for all parameters, our algorithm scales linearly with the size of the dataset.

To study how our algorithm scales to the size of classification datasets, we randomly sample subsets of 10%, 20%, up to 100% of instances for datasets connect4 and letRecog, and run our rule mining algorithm with the strategy $\mathcal{B}_c^{conf}$ for $s_0^r = 1.0\%, 3.0\%$ and $K = 1, 3$. The results in Figure 4.6 show the linear relationship between the running times and the dataset sizes.

Table 4.1: Running Time Reduction For $Q = Q^{\sigma}$

dataset	K	s_0^r	time[a]	$\mathcal{B}_r^{sig}$	$\mathcal{B}_c^{sig}$
BestBuy.wdep	1	1.0%	9.02	0.10	**0.09**
		2.0%	2.41	0.11	**0.10**
	5	1.0%	8.94	0.11	**0.10**
		2.0%	2.51	0.12	**0.11**
CitySearch.wdep	1	1.0%	2.51	0.18	**0.17**
		2.0%	0.95	0.18	0.18
	5	1.0%	2.53	0.19	**0.18**
		2.0%	0.99	0.19	**0.18**
Yelp.wdep	1	1.0%	33.0	0.09	0.09
		2.0%	10.7	0.09	**0.08**
	5	1.0%	32.5	0.10	**0.09**
		2.0%	11.5	0.09	**0.08**
MovieLens	1	1.0%	16.2	0.13	**0.03**
		2.0%	13.9	0.09	**0.02**
	5	1.0%	16.5	0.14	**0.03**
		2.0%	14.5	0.10	**0.03**
SmileMTP	1	4.0%	240	0.13	**0.11**
		5.0%	16.1	0.13	**0.11**
	5	4.0%	240	0.15	**0.12**
		5.0%	16.2	0.13	**0.12**
census	1	50%	240	2e-4	2e-4
		60%	74.5	8e-3	8e-3
	5	50%	240	1e-3	1e-3
		60%	73.1	1e-2	1e-2

[a] Running time in minutes for the baseline approach when no quality bounding strategy is applied. [b] For each quality bounding strategy, we report the running time reduction as the ratio of its running time to the baseline approach's running time. [c] If the baseline approach did not finish within 240 minutes, we report its running time as 240. [d] In case when the base line approach did not finish, we report the running time reduction as actual-time/240.

Table 4.2: Running Time Reduction For $Q = Q^m$

dataset	K	s_0^r	time[a]	$\mathcal{B}_r^{add}$	$\mathcal{B}_r^{mse}$	$\mathcal{B}_c^{add}$	$\mathcal{B}_c^{mse}$
BestBuy.wdep	1	1.0%	6.41	0.16	0.16	**0.14**	**0.14**
		2.0%	1.69	0.17	0.26	**0.15**	0.23
	5	1.0%	6.14	0.18	0.18	**0.16**	**0.16**
		2.0%	1.68	0.19	0.28	**0.17**	0.25
CitySearch.wdep	1	1.0%	1.84	0.24	0.37	**0.22**	0.34
		2.0%	0.73	**0.22**	0.48	**0.22**	0.45
	5	1.0%	1.83	0.26	0.40	**0.23**	0.35
		2.0%	0.75	0.24	0.49	**0.23**	0.45
Yelp.wdep	1	1.0%	21.4	**0.13**	0.18	**0.13**	0.16
		2.0%	7.79	**0.11**	0.20	**0.11**	0.19
	5	1.0%	21.7	0.14	0.18	**0.13**	0.17
		2.0%	7.78	0.12	0.21	**0.11**	0.19
MovieLens	1	1.0%	6.18	0.29	0.31	**0.08**	**0.08**
		2.0%	5.27	0.23	0.24	**0.07**	**0.07**
	5	1.0%	6.56	0.31	0.32	**0.08**	**0.08**
		2.0%	5.80	0.23	0.25	**0.07**	0.08
SmileMTP	1	4.0%	240	0.07	0.04	0.06	**0.03**
		5.0%	10.5	0.09	0.07	0.08	**0.06**
	5	4.0%	240	0.07	0.04	0.06	**0.03**
		5.0%	10.2	0.10	0.08	0.09	**0.07**
census	1	50%	240	7e-4	7e-4	6e-4	**6**e-4
		60%	56.7	3e-3	4e-3	**3**e-3	4e-3
	5	50%	240	8e-4	7e-4	7e-4	**6**e-4
		60%	57.1	4e-3	8e-3	**4**e-3	7e-3

[a] Running time in minutes for the baseline approach when no quality bounding strategy is applied. [b] For each quality bounding strategy, we report the running time reduction as the ratio of its running time to the baseline approach's running time.
[c] If the baseline approach did not finish within 240 minutes, we report its running time as 240. [d] In case when the base line approach did not finish, we report the running time reduction as actual-time/240.

Table 4.3: Running Time Reduction For $Q = Q^n$

dataset	K	s_0^r	time[a]	$\mathcal{B}_r^{add}$	$\mathcal{B}_r^{mse}$	$\mathcal{B}_c^{add}$	$\mathcal{B}_c^{mse}$
BestBuy.wdep	1	1.0%	6.96	0.37	0.40	**0.35**	0.36
		2.0%	1.93	0.32	0.45	**0.29**	0.42
	5	1.0%	6.96	0.38	0.41	**0.37**	**0.37**
		2.0%	1.89	0.35	0.50	**0.33**	0.43
CitySearch.wdep	1	1.0%	2.07	0.35	0.48	**0.33**	0.43
		2.0%	0.82	**0.37**	0.65	**0.37**	0.60
	5	1.0%	2.06	0.38	0.53	**0.35**	0.48
		2.0%	0.83	0.39	0.66	**0.36**	0.60
Yelp.wdep	1	1.0%	22.8	0.34	0.40	**0.33**	0.35
		2.0%	8.14	0.32	0.46	**0.30**	0.41
	5	1.0%	24.7	0.32	0.38	**0.31**	0.33
		2.0%	8.36	0.33	0.48	**0.32**	0.43
MovieLens	1	1.0%	6.70	0.59	0.67	**0.37**	0.40
		2.0%	6.19	0.53	0.70	**0.39**	0.50
	5	1.0%	7.43	0.59	0.65	**0.36**	0.39
		2.0%	6.54	0.55	0.70	**0.41**	0.46
SmileMTP	1	4.0%	240	0.63	0.67	0.63	**0.61**
		5.0%	12.0	0.40	0.43	0.41	**0.38**
	5	4.0%	240	0.68	0.73	0.68	**0.65**
		5.0%	11.6	0.45	0.49	0.49	**0.44**
census	1	50%	240	**0.21**	0.28	0.22	0.25
		60%	65.8	**0.11**	0.15	**0.11**	0.13
	5	50%	240	**0.21**	0.28	0.22	0.25
		60%	66.2	**0.20**	0.26	**0.20**	0.23

[a] Running time in minutes for the baseline approach when no quality bounding strategy is applied. [b] For each quality bounding strategy, we report the running time reduction as the ratio of its running time to the baseline approach's running time.
[c] If the baseline approach did not finish within 240 minutes, we report its running time as 240. [d] In case when the base line approach did not finish, we report the running time reduction as actual-time/240.

Table 4.4: Running Time Reduction For $Q = Q^c$

dataset	K	s_0^r	time[a]	$\mathcal{B}_h^{conf}$	$\mathcal{B}_r^{conf}$	$\mathcal{B}_c^{conf}$
adult	1	1.0%	0.512	1.07	0.15	**0.08**
		3.0%	0.461	0.92	0.12	**0.08**
	3	1.0%	0.631	1.00	0.17	**0.10**
		3.0%	0.535	0.98	0.13	**0.10**
connect4	1	1.0%	81.427	0.86	**0.02**	**0.02**
		3.0%	58.178	0.82	**0.01**	**0.01**
	3	1.0%	86.494	0.88	**0.02**	**0.02**
		3.0%	58.032	0.87	**0.01**	**0.01**
letRecog	1	1.0%	0.739	0.50	**0.25**	0.28
		3.0%	0.314	0.46	**0.25**	0.26
	3	1.0%	0.695	0.56	**0.29**	**0.29**
		3.0%	0.324	0.49	**0.28**	**0.28**
pageBlocks	1	1.0%	0.007	1.00	0.29	**0.14**
		3.0%	0.007	1.14	**0.14**	**0.14**
	3	1.0%	0.007	1.14	0.29	**0.14**
		3.0%	0.008	0.88	**0.12**	**0.12**

[a] Running time in minutes for the baseline approach when no quality bounding strategy is applied. [b] For each quality bounding strategy, we report the running time reduction as the ratio of its running time to the baseline approach's running time.

Chapter 5

Mining Association Rules With Dyadic Interaction

5.1 Notations And Definitions

A dyadic transactional database $\mathcal{D}^0 = \{R_1, R_2, \ldots, R_N\}$ is a set of instances of the form $R_k = \langle X_k, Y_k, Z_k \rangle$, where X_k and Y_k represent two sets of distinct types of features and Z_k contains a set of target values. We study the type of data in which features and target values are fully discretized [1] into items, and refer to X_k, Y_k and Z_k as itemsets. Denote $X_0 = \cup_k X_k$, $Y_0 = \cup_k Y_k$ and $Z_0 = \cup_k Z_k$. Given an itemset $T \subseteq X_0 \cup Y_0 \cup Z_0$, we say that T is contained in R_k (or R_k contains T) if $T \subseteq X_k \cup Y_k \cup Z_k$. The conditional database (or subspace) $\mathcal{D}^0_T$ is the set of instances from $\mathcal{D}^0$ that contain the itemset T. The support of T w.r.t. database $\mathcal{D}$ (e.g., $\mathcal{D}^0$ or any of its conditional databases), denoted as $\text{supp}(T|\mathcal{D})$, is the number of instances from $\mathcal{D}$ which contain T.

An association rule is of the form $T \to Z$, where its left hand side (LHS) $T \subseteq X_0 \cup Y_0$ and right hand side (RHS) $Z \subseteq Z_0$. We define its confidence w.r.t. $\mathcal{D}$ as

$$\text{conf}(T \to Z|\mathcal{D}) = \frac{\text{supp}(T \cup Z|\mathcal{D})}{\text{supp}(T|\mathcal{D})}, \tag{5.1}$$

and its lift w.r.t. $\mathcal{D}$ as

$$\text{lift}(T \to Z|\mathcal{D}) = \frac{\text{conf}(T \to Z|\mathcal{D})}{\text{conf}(\emptyset \to Z|\mathcal{D})}. \tag{5.2}$$

A dyadic (association) rule is of the form $\langle X, Y \rangle \rightarrow Z$, where $X \subseteq X_0$, $Y \subseteq Y_0$, $Z \subseteq Z_0$, and the difference between itemsets X and Y is emphasized by requiring the rule to satisfy properly designed metrics. When the database $\mathcal{D}$ is omitted, it is assumed to be $\mathcal{D}^0$. If an item is found in the place where a set is expected, it is assumed to be the set containing that item.

5.2 Dual-Lift Measure

Our goal is to mine the set of dyadic association rules $\langle X, Y \rangle \rightarrow Z$, in which the interaction between X and Y leads to the prevalence of the target set Z. The standard association rule mining problems typically mine a set of high confidence rules. However, confidence can not capture the interaction between two groups of LHS itemsets. To illustrate this, consider the extreme case when the items in Y_0 are all redundant features, i.e., they are randomly sampled and do not correlate with the other items. In this case, we can mine the set of high confidence rules by first ignoring Y_0. Assume we get $X \rightarrow Z$ whose confidence is high. Then we have for any $Y \subseteq Y_0$, the confidence of $X \cup Y \rightarrow Z$ is also high due to conf$(X \cup Y \rightarrow Z) \approx$ conf$(X \rightarrow Z)$ [1] . But we know Y does not interact with X from our assumption. Using lift from Equation 5.2 does not help either, since it is simply the normalized version of confidence and the normalization term is independent on X and Y.

Our approach is based on the comparison between the following three quantities: conf$(X \cup Y \rightarrow Z)$, conf$(X \rightarrow Z)$, and conf$(Y \rightarrow Z)$. If conf$(X \cup Y \rightarrow Z)$ is higher than both conf$(X \rightarrow Z)$ and conf$(Y \rightarrow Z)$, this indicates that X and Y are positively correlated w.r.t. the target Z, and the increase of confidence can be seen as the result of the positive interaction between X and Y. On the other hand, if conf$(X \cup Y \rightarrow Z)$ is lower than both conf$(X \rightarrow Z)$ and conf$(Y \rightarrow Z)$, this indicates that X and Y are negatively correlated w.r.t. the target Z, and the decrease of confidence can be seen as the result of the negative interaction between X and Y. If conf$(X \cup Y \rightarrow Z)$ is in between of conf$(X \rightarrow Z)$ and conf$(Y \rightarrow Z)$, it becomes difficult to interpret the interaction between X and Y. From this analysis, we propose the dual-lift measure

[1] To derive this equation, notice that the confidence of conf$(X \cup Y \rightarrow Z)$ can be seen as the conditional probability $P[Z|X \cup Y]$. From our assumption, this probability should be conditionally independent on Y. So we have $P[Z|X \cup Y] \approx P[Z|X]$.

defined as

$$\begin{aligned}&\text{dlift}(\langle X, Y\rangle \to Z)\\ = \ &\min\{\frac{\text{conf}(X \cup Y \to Z)}{\text{conf}(X \to Z)}, \frac{\text{conf}(X \cup Y \to Z)}{\text{conf}(Y \to Z)}\}.\end{aligned} \tag{5.3}$$

We are interested in dyadic rules whose dual-lift is high, which indicates that X and Y are positively interacting with each other so that combining them together leads to the increase of the confidence of the target Z.

5.2.1 Subspace Interpretation

We reformulate the dual-lift measure defined in Equation 5.3, so that we can gain more insights about the association rules we will be discovering. First, we show that the support can be reformulated in the conditional database, that is, for any $X_p \subseteq X_0$, $X \subseteq X_0$, $Y_p \subseteq Y_0$ and $Y \subseteq Y_0$, we have

$$\begin{aligned}&\text{supp}(X_p \cup Y_p \cup X \cup Y \cup Z|\mathcal{D}^0)\\ = \ &\text{supp}(X \cup Y \cup Z\rangle|\mathcal{D}^0_{X_p \cup Y_p}).\end{aligned} \tag{5.4}$$

This follows directly from the observation: any instances R_k in $\mathcal{D}^0$ that contain $X_p \cup Y_p \cup X \cup Y \cup Z$ are in $\mathcal{D}^0_{X_p \cup Y_p}$ and also contain $X \cup Y \cup Z$ and vice versa. By combining Equations 5.4 with 5.1, we can get

$$\begin{aligned}&\text{conf}(X_p \cup Y_p \cup X \cup Y \to Z|\mathcal{D}^0)\\ = \ &\text{conf}(X \cup Y \to Z|\mathcal{D}^0_{X_p \cup Y_p}).\end{aligned} \tag{5.5}$$

From Equation 5.5, we can rewrite the three confidences involved in dual-lift as:

$$\text{conf}(X \cup Y \to Z) = \text{conf}(\emptyset \to Z|\mathcal{D}^0_{X \cup Y}), \tag{5.6}$$

$$\text{conf}(X \to Z) = \text{conf}(\emptyset \to Z|\mathcal{D}^0_{X}), \tag{5.7}$$

and,

$$\text{conf}(Y \to Z) = \text{conf}(\emptyset \to Z|\mathcal{D}^0_{Y}). \tag{5.8}$$

The RHS of the above three Equations, i.e., $\text{conf}(\emptyset \to Z|\mathcal{D})$ with $\mathcal{D} = \mathcal{D}^0_{X \cup Y}, \mathcal{D}^0_X, \mathcal{D}^0_Y$, is the ratio of the number of instances in $\mathcal{D}$ that contain Z to the size of the database

$\mathcal{D}$. This can be viewed as the density of target Z in $\mathcal{D}$. From this, *an alternative interpretation of rules with high dual-lift is that the density of target Z is higher in the subspace determined by $X \cup Y$ than its density in the subspace determined by either X or Y.*

We can reformulate the ratios of two confidences on the RHS of Equation 5.3 as

$$\begin{aligned}
&\frac{\text{conf}(X \cup Y \to Z)}{\text{conf}(X \to Z)} \\
&= \frac{\text{conf}(Y \to Z|\mathcal{D}^0_X)}{\text{conf}(\emptyset \to Z|\mathcal{D}^0_X)} \\
&= \text{lift}(Y \to Z|\mathcal{D}^0_X), \qquad (5.9)
\end{aligned}$$

and similarly,

$$\frac{\text{conf}(X \cup Y \to Z)}{\text{conf}(Y \to Z)} = \text{lift}(X \to Z|\mathcal{D}^0_Y). \qquad (5.10)$$

From Equations 5.9 and 5.10, *we can further interpret the dyadic rule $\langle X, Y\rangle \to Z$ with high dual-lift as the combination of two rules $Y \to Z$ and $X \to Z$ with high lift which are discovered in subspaces $\mathcal{D}^0_X$ and $\mathcal{D}^0_Y$, respectively.* Note that these two rules need to be combined to ensure that X and Y are positively interacted. Based on this analysis, *our dual-lift formulation can be used to discover rules hidden in the subspace of the data which are otherwise difficult to discover by looking at the global database.*

5.2.2 Connection To Biclustering

There is a subtle but interesting connection between our rule mining problem and biclustering. To start, we reconstruct the database $\mathcal{D}^0$ into the matrix format A^0 with n rows and m columns, so that each instance R_k corresponds to a pair of matrix indices (i, j). The matrix element $A^0_{i,j}$ takes the value Z_k. Row i and column j of matrix A^0 are associated with additional features $X_i = X_k$ and $Y_j = Y_k$, respectively. We allow the matrix A^0 to have missing values which do not correspond to any data instances.

Let $I_0 = \{1, 2, \ldots, n\}$ and $J_0 = \{1, 2, \ldots, m\}$ be two sets of indices. A bicluster associated with matrix A^0 is a submatrix $A^0_{I,J}$ where $I \subseteq I_0$, $J \subseteq J_0$ and $a_{i,j} \in A^0_{I,J}$ if $i \in I$ and $j \in J$. Biclustering algorithms tackle the problem of finding a set of biclusters (or submatrices) which satisfy a given quality criterion.

What makes our formulation different from the standard biclustering is that rows and columns of matrix A^0 are also associated with additional features. We can use these features to convert the itemset pair $\langle X, Y\rangle$ to the indices pair (I_X, J_Y). Specifically, let $I_X = \{i|X \subseteq X_k\}$ and $J_Y = \{j|Y \subseteq Y_k\}$. In this way, $\langle X, Y\rangle$ determines a bicluster or submatrix $A^0_{I_X,J_Y}$. This submatrix contains all the data instances in the conditional database $\mathcal{D}^0_{X\cup Y}$.

Given a dyadic association rule $\langle X, Y\rangle \to Z$, its LHS can be used to determine a bicluster $A^0_{I_X,J_Y}$. To evaluate the bicluster, we can convert A^0 to a binary matrix whose (i, j) element is one if $Z \subseteq A^0_{i,j}$ and zero otherwise. In this way, our dual-lift measure becomes a quality criterion that can be used to evaluate the submatrix of the binary matrix. So our rule mining problem becomes a special type of biclustering problem. However, due to that the bicluster is determined by two itemsets X, Y and the quality criterion is dependent on the itemset Z, traditional biclustering algorithms can not be applied to solve this problem.

5.3 Problem Formulation

Our formulation of the dyadic association rule mining problem consists of three components that define the interesting rules: (i) minimum dual-lift constraint, which utilizes the dual-lift measure developed in Section 5.2, (ii) non-redundant constraint, which aims to remove redundant rules, and (iii) minimum support constraint, which defines the set of frequent rules. The formal definitions of these components are as follows:

Definition 5.3.1 (Minimum dual-lift constraint). *The dyadic association rule $\langle X, Y\rangle \to Z$ satisfies the minimum dual-lift constraint of $l_0 > 1$ in database $\mathcal{D}^0$ if*

$$\textit{dlift}(\langle X, Y\rangle \to Z|\mathcal{D}^0) \geq l_0.$$

Definition 5.3.2 (Non-redundant constraint). *The dyadic association rule $\langle X, Y\rangle \to Z$ satisfies the non-redundant constraint (or is not redundant) if there is no $\langle X', Y'\rangle \to Z'$ such that $X' \subseteq X$, $Y' \subseteq Y$, $Z' \subseteq Z$ and*

$$\textit{dlift}(\langle X', Y'\rangle \to Z'|\mathcal{D}^0) > \textit{dlift}(\langle X, Y\rangle \to Z|\mathcal{D}^0).$$

Definition 5.3.3 (Minimum support constraint). *The dyadic association rule $\langle X, Y\rangle \to Z$ satisfies minimum support constraint of $s_0 > 0$ if*

$$\textit{supp}(X \cup Y \cup Z|\mathcal{D}^0) \geq s_0.$$

A rule satisfying minimum support constraint is called a frequent rule.

The idea of redundant rule comes from the following: the sub-rule $\langle X', Y' \rangle \rightarrow Z'$ is generally preferred if it has a better quality (measured by dual-lift) since it can be used to replace its super-rule $\langle X, Y \rangle \rightarrow Z$ in all scenarios (it is contained in all instances that can contain its super-rule). The minimum support constraint is used to ensure the prevalence of the rule. This is important because dual-lift of an infrequent rule can have very large variability and thus may not be significant.

In this paper, we only consider the case when Z contains a single item z. With the aid of above definitions, our problem can be stated as:

Problem 5.3.4. *Given database $\mathcal{D}^0$ and user specified parameters $l_0 > 1$, $s_0 > 0$, identify all possible dyadic association rules of the form $\langle X, Y \rangle \rightarrow z$ satisfying the minimum dual-lift constraint of l_0, non-redundant constraint and minimum support constraint of s_0, where $X \subseteq X_0$, $Y \subseteq Y_0$ and $z \in Z_0$.*

5.4 DLiftMiner Algorithm

The naive approach for solving Problem 5.3.4 is to first apply a traditional itemset mining algorithm ([7] or [71]) to discover a set of frequent itemsets, and then post-process these itemsets to construct the set of dyadic association rules satisfying all three constraints. However, the itemset mining step of this approach may generate a huge number of itemsets (especially for dense datasets). And the majority of these itemsets will not contribute to the final rule set. Thus the large amount of computation has been wasted in mining them. Instead, we develop an efficient algorithm called DLiftMiner to discover the final set of rules directly. It applies several strategies to push the three constraints deeply into the rule mining framework so that the running time can be reduced dramatically.

The DLiftMiner (see Algorithm 2) algorithm follows the depth-first approach for rule mining and grows the rule's LHS by adding one item at a time [70]. It reduces the size of the dataset successively by constructing the conditional database which is the subset of instances that contain the current rule's LHS that is being grown.

Before we get into the details of the Algorithm 2, we first explain its input parameters. The current dual-lift threshold for each target item is stored in $l[z]$ which is initialized to be l_0. The prefix itemsets $\langle X_p, Y_p\rangle$ contain the items that have been searched, and are initialized to be $\langle\emptyset,\emptyset\rangle$. There are three input databases $\mathcal{D}^{xy}$, $\mathcal{D}^x$ and $\mathcal{D}^y$, which represents the three conditional databases ($\mathcal{D}^0_{X_p\cup Y_p}$, $\mathcal{D}^0_{X_p}$, and $\mathcal{D}^0_{Y_p}$ respectively) with some items being removed. In particular, $\mathcal{D}^x$ ($\mathcal{D}^y$) only contains items in $X_0\cup Z_0$ ($Y_0\cup Z_0$). The initialization of the input databases are determined by the initialization of $\langle X_p, Y_p\rangle$. In particular, $\mathcal{D}^{xy}$, $\mathcal{D}^x$, and $\mathcal{D}^y$ are initialized to be $\mathcal{D}^0$, $\mathcal{D}^0$ with only items in $X_0\cup Z_0$, and $\mathcal{D}^0$ with only items in $Y_0\cup Z_0$, respectively. The output of DLiftMiner is the set of rules that contain the prefix itemsets. With the initial prefix itemsets to be empty sets, DLiftMiner will print all dyadic association rules we want to discover.

The DLiftMiner algorithm starts with removing infrequent items from $\mathcal{D}^{xy}$ (line 1), which will be explained in detail later. Next it removes all items in $\mathcal{D}^x$ and $\mathcal{D}^y$ that are not in $\mathcal{D}^{xy}$ (line 2). This is due to that we are only concerned about items in $\mathcal{D}^{xy}$ and this step helps to make the databases as dense as possible, which will help in getting more pruning. After this, the DLiftMiner algorithm calculates for each target item z of $\mathcal{D}^{xy}$ the dual-lift of the rule $\langle X_p, Y_p\rangle\to z$ (lines 4-7). The reason why this calculation is correct can be seen from the Equations 5.6, 5.7, 5.8 and the definition of the dual-lift (Equation 5.3). If the calculated dual-lift is larger or equal to the threshold of the corresponding target item, rule $\langle X_p, Y_p\rangle\to z$ is printed and $l[z]$ is updated (line 10). We explain why the update of $l[z]$ in line 10 is correct in details later.

The next part of DLiftMiner algorithm is to get the next item i from $\mathcal{D}^{xy}$ (line 13) and decide whether we need to add this item into the current prefix items to form the new prefix itemsets $\langle X'_p, Y'_p\rangle$, create new conditional databases $\mathcal{D}'^{xy}$, $\mathcal{D}'^{x}$, $\mathcal{D}'^{y}$ (lines 14, 16 and 18) and call DLiftMiner algorithm recursively on them (lines 30 and 31). For this, DLiftMiner calculates, for each frequent target item z from $\mathcal{D}'^{xy}$, the dual-lift upper bound of any frequent rules with RHS being z that may be discovered from database $\mathcal{D}'^{xy}$ (lines 22-24). Note that in our algorithm, there is no need to actually construct $\mathcal{D}'^{xy}$, $\mathcal{D}'^{x}$ and $\mathcal{D}'^{y}$ in lines 14, 16 and 18; we can operate on the original databases $\mathcal{D}^{xy}$, $\mathcal{D}^x$ and $\mathcal{D}^y$ for evaluating the dual lift upper bound. If this upper bound is smaller than the corresponding dual-lift threshold (line 25), the target item z is ignored (line 26). If the resulting set of frequent target items is not empty (line 29), new databases $\mathcal{D}'^{xy}$, $\mathcal{D}'^{x}$, $\mathcal{D}'^{y}$ are constructed with necessary items removed (line 30), and the DLiftMiner algorithm is called with updated parameters (line 31). Finally, item i is removed (lines 33-38) and the next item is considered.

DLiftMiner utilizes various approaches to prune the search space. The support based pruning is used to remove infrequent items. The effect of the redundancy based pruning is to set the dual-lift thresholds higher and higher. Finally the dual-lift based pruning finds the upper bound of the dual-lift measure for target item z. And if the upper bound is less than current dual-lift threshold, the rest of search space is pruned away for item z. Additional details of these pruning methods are provided in the rest of this section.

5.4.1 Support Based Pruning

Support based pruning can be derived from the minimum support constraint in Definition 5.3.3. We are interested in frequent rules of the form $\langle X_p \cup X, Y_p \cup Y \rangle \rightarrow z$, such that $\text{supp}(X_p \cup Y_p \cup X \cup Y \cup z | \mathcal{D}^0) \geq s_0$. From Equation 5.4, we can compute support in the conditional database $\mathcal{D}^0_{X_p \cup Y_p}$, which is the $\mathcal{D}^{xy}$ in Algorithm 2. So we get $\text{supp}(X \cup Y \cup z | \mathcal{D}^{xy}) \geq s_0$. According to the anti-monotone property of support [7], we have (i) $\text{supp}(z|\mathcal{D}^{xy}) \geq s_0$, and (ii) for $\forall x \in X$ ($\forall y \in Y$), $\text{supp}(x \cup z|\mathcal{D}^{xy}) \geq s_0$ ($\text{supp}(y \cup z|\mathcal{D}^{xy}) \geq s_0$). From (i), we remove all the infrequent target items z. From (ii), we remove all items x (y) such that for $\forall z$ $\text{supp}(x \cup z|\mathcal{D}^{xy}) < s_0$ ($\text{supp}(y \cup z|\mathcal{D}^{xy}) < s_0$). Our algorithm applies support based pruning in lines 1, 2 and 20.

5.4.2 Redundancy Based Pruning

From the non-redundant constraint (Definition 5.3.2), any rules ($\langle X_p \cup X, Y_p \cup Y \rangle \rightarrow z$) to be discovered by DLiftMiner should satisfy the inequality: $\text{dlift}(\langle X_p \cup X, Y_p \cup Y \rangle \rightarrow z) \geq \text{dlift}(\langle X_p, Y_p \rangle \rightarrow z)$. A closer investigation suggests that we need to consider this inequality only when the rule $\langle X_p, Y_p \rangle \rightarrow z$ satisfies the both the support and the dual-lift constraints. If the support constraint is violated, the super rule $\langle X_p \cup X, Y_p \cup Y \rangle \rightarrow z$ will also be infrequent and need not be considered. If the dual-lift constraint is violated, the above inequality will be automatically satisfied when the super rule satisfies the dual-lift constraint. When both constraints are satisfied by $\langle X_p, Y_p \rangle \rightarrow z$, the effect of the above inequality is shown in line 10, where we update the dual-lift threshold for the target item z to the dual-lift of the current rule. By doing this, the dual-lift thresholds will get higher and higher, which will help the dual-lift pruning strategy (as explained later) to achieve better performances.

However, it should be noted that the current implementation of our algorithm does not take into account all the redundancy that can be exploited by Definition 5.3.2 due to the nature of depth first search. This is because that not all sub-rules are processed before the current rule $\langle X_p, Y_p \rangle \rightarrow z$. To solve this issue, we post-process the set of rules discovered by DLiftMiner to ensure that all redundant rules are removed.

5.4.3 Dual-lift Based Pruning

The dual-lift based pruning strategy calculates the upper bound of the dual-lift measure for all the frequent rules $\langle X'_p \cup X, Y'_p \cup Y \rangle \rightarrow z$ to be discovered from database $\mathcal{D}'^{xy}$ (lines 22-24). If this upper bound is smaller than the current dual-lift threshold $l[z]$, all the rules from $\mathcal{D}'^{xy}$ with RHS being z can be pruned. This is accomplished by removing z from $\mathcal{D}'^{xy}$ (lines 26 and 30). If all the target items are removed (line 29), the conditional database $\mathcal{D}'^{xy}$ needs not be constructed so that the remaining search space is pruned.

To calculate the dual-lift upper bound of the frequent rule $\langle X'_p \cup X, Y'_p \cup Y \rangle \rightarrow z$, we compute the confidence upper bound of rule $X'_p \cup Y'_p \cup X \cup Y \rightarrow z$ and the confidence lower bound of two rules $X'_p \cup X \rightarrow z$ and $Y'_p \cup Y \rightarrow z$. According to Equation 5.5, these computations can be done within the conditional databases $\mathcal{D}'^{xy}$ (or $\mathcal{D}^0_{X'_p \cup Y'_p}$), $\mathcal{D}'^{x}$ (or $\mathcal{D}^0_{X'_p}$), and $\mathcal{D}'^{y}$ (or $\mathcal{D}^0_{Y'_p}$), so that we can drop the prefix items X'_p and Y'_p from the LHS of rules under consideration (lines 22-23). After that, we combine these three confidence bounds together to get the dual-lift upper bound (line 24).

From the above analysis, we have transformed the dual-lift upper bounding problem into the confidence upper and lower bounding problems. We develop effective strategies $\mathcal{B}^u$ and $\mathcal{B}^l$ for the confidence upper and lower bound, respectively. For comparison's purpose, we also develop two naive strategies $\mathcal{B}^u_n$ and $\mathcal{B}^l_n$ for the confidence upper and lower bound, respectively. These methods are presented in the rest of this section.

Bound the Confidence

The problem of bounding the confidence is defined as:

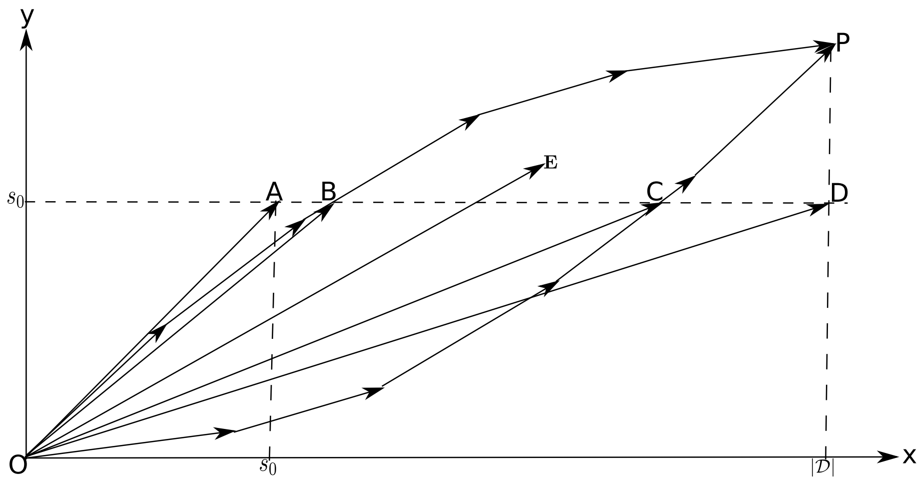

Figure 5.1: Bound the Confidence

Problem 5.4.1 (Bounding the Confidence)**.** *Given database $\mathcal{D}$ and target item z, the problem of bounding the confidence is to find the upper and lower bound for quantity conf($T \to z|\mathcal{D}$) for any itemset T such that supp($T \cup z|\mathcal{D}$) $\geq s_0$.*

We start with the naive solutions for Problem 5.4.1. Since the confidence cannot be greater than one, we set the upper bound of conf($T \to z|\mathcal{D}$) to be one. The lower bound can be derived as

$$\mathrm{conf}(T \to z|\mathcal{D}) = \frac{\mathrm{supp}(T \cup z|\mathcal{D})}{\mathrm{supp}(T|\mathcal{D})} \geq \frac{s_0}{|\mathcal{D}|}.$$

These two approaches are called $\mathcal{B}_n^u$ and $\mathcal{B}_n^l$ for the upper and lower bounds, respectively.

The more effective methods that we develop are based on the concept of instance groups:

Definition 5.4.2. *We partition all the instances of database $\mathcal{D}$ into a set of instance groups $(X_g, Y_g, \mathcal{Z}_g)$, where instance group g contains all instances whose feature sets are X_g and Y_g. And $\mathcal{Z}_g$ is a set of target itemsets for these instances.*

We introduce the indicator variable I_G to be one if G is true and zero otherwise.

We rewrite:

$$\begin{aligned}\text{supp}(T \cup z|\mathcal{D}) &= \sum_k I_{T\subseteq X_k \cup Y_k} I_{z\in Z_k} \\ &= \sum_g \sum_{k:Z_k\in\mathcal{Z}_g} I_{T\subseteq X_k \cup Y_k} I_{z\in Z_k} \\ &= \sum_g I_{T\subseteq X_g \cup Y_g} \sum_{k:Z_k\in\mathcal{Z}_g} I_{z\in Z_k},\end{aligned}$$

and,

$$\begin{aligned}\text{supp}(T|\mathcal{D}) &= \sum_k I_{T\subseteq X_k \cup Y_k} \\ &= \sum_g \sum_{k:Z_k\in\mathcal{Z}_g} I_{T\subseteq X_k \cup Y_k} \\ &= \sum_g I_{T\subseteq X_g \cup Y_g} |\mathcal{Z}_g|.\end{aligned}$$

To simply the above results, let $I_g = I_{T\subseteq X_g\cup Y_g}$, $s_g = |\mathcal{Z}_g|$, and $s_g^z = \sum_{k:Z_k\in\mathcal{Z}_g} I_{z\in Z_k}$. We have, $\text{supp}(T\cup z|\mathcal{D}) = \sum_g I_g s_g^z$ and $\text{supp}(T|\mathcal{D}) = \sum_g I_g s_g$. The confidence can be written as

$$\text{conf}(T \to z|\mathcal{D}) = \frac{\sum_g I_g s_g^z}{\sum_g I_g s_g}. \tag{5.11}$$

The minimum support constraint now becomes

$$\sum_g I_g s_g^z \geq s_0. \tag{5.12}$$

With the help of instance groups, we have transformed Problem 5.4.1 into the problem of finding the upper and lower of the quantity in the RHS of Equation 5.11, where the binary variables I_g need to satisfy the constraint in Equation 5.12.

Next, we formulate this problem into the geometric notation. For each instance group g in $\mathcal{D}$, define a 2-D vector $\mathbf{e}_g = (e_{g,h}, e_{g,v})$ whose horizontal component $e_{g,h} = s_g$ and vertical component $e_{g,v} = s_g^z$. Let $\mathbf{E} = \sum_g I_g \mathbf{e}_g$. The RHS of Equation 5.11 is the slope of $\mathbf{E}$ denoted as $||\mathbf{E}||$, and Equation 5.12 becomes a constraint on the vertical component of $\mathbf{E}$: $E_v \geq s_0$. So the Problem 5.4.1 is equivalent to the following:

Problem 5.4.3. *Given a sequence of 2-D vectors $\mathbf{e}_g$ ($g = 1, 2, \ldots, n$) satisfying $e_{g,h} > 0$, $e_{g,v} \geq 0$ and $\sum_g e_{g,v} \geq s_0$. The problem is to find the upper and lower bound of $||\mathbf{E}||$ where $\mathbf{E} = \sum_g I_g \mathbf{e}_g$ for any $I_g \in \{0, 1\}$ satisfying $E_v \geq s_0$.*

Table 5.1: Parameters for synthetic dataset.

parameter	**description**	**value**
ntrans	number of transactions	$10k$
nitems	number of items	500
npats	number of maximal potentially large itemsets	100
patlen	average size of the maximal potentially large itemsets	30
tlen	average size of transactions	30

Table 5.2: Running times for the first synthetic dataset.

$\mathbf{l_0}$**\bounding Methods**	none	$(B^u_n + B^l_n)$	$(B^u + B^l_n)$	$(B^u_n + B^l)$	$(B^u + B^l)$
2	87.50	30.93	31.28	2.67	2.63
3	87.50	18.22	18.08	1.15	1.10
4	87.50	9.97	10.10	0.88	0.83
5	87.50	8.73	8.50	0.63	0.63

[a] All times are in minutes

In Figure 5.1, we sort vectors $\mathbf{e}_g$ in increasing (decreasing) slope order and connect them sequentially to form the curve OCP (OBP). The minimum support horizontal line s_0 cuts the convex region OCPBO into two parts. The top part BCPB is where the vector $\mathbf{E}$ can fall into. It can be seen that the slope of vector OB (OC) is the upper (lower) bound that we are looking for. The naive upper (lower) bound is also shown as the slope of vector OA (OD). We summarize our intuitions from Figure 5.1 into the following theorem:

Theorem 5.4.4. *Assume the same notations in Problem 5.4.3. Without loss of generality, we assume vectors* $\mathbf{e}_g$ *are sorted as* $||\mathbf{e}_1|| \leq ||\mathbf{e}_2|| \leq \ldots \leq ||\mathbf{e}_n||$ *(*$||\mathbf{e}_1|| \geq ||\mathbf{e}_2|| \geq \ldots \geq ||\mathbf{e}_n||$*). Find index* $1 \leq g_0 \leq n$ *and parameter* $0 < \alpha \leq 1$ *such that* $\mathbf{E}_b = \sum_{1 \leq g < g_0} \mathbf{e}_g + \alpha \mathbf{e}_{g_0}$ *satisfies* $E_{b,v} = s_0$*. Then* $||\mathbf{E}_b||$ *is the lower (upper) bound solution to Problem 5.4.3.*

Proof. See Section B.5. □

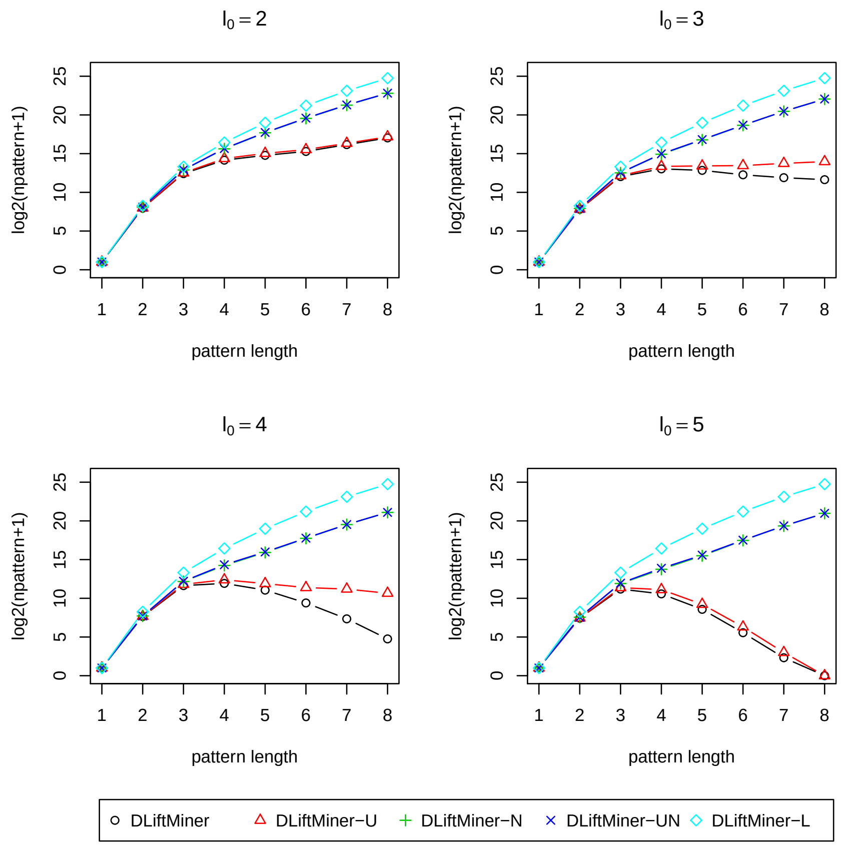

Figure 5.2: Pruning effects on synthetic dataset

5.5 Experimental Study

5.5.1 DLiftMiner Algorithm Evaluation

DataSets

To evaluate the performance of our algorithm, we use the IBM Quest market-basket synthetic data generator [76] to generate a series of datasets. The generator takes the parameters described in Table 5.1. We split the generated items into three types of items so that X_0 covers 57%, Y_0 covers 33% and Z_0 covers the rest 10%. We keep only the transactions containing all three types of items in the dataset. The parameters for the first synthetic dataset are shown in the third column of Table 5.1 and the actual

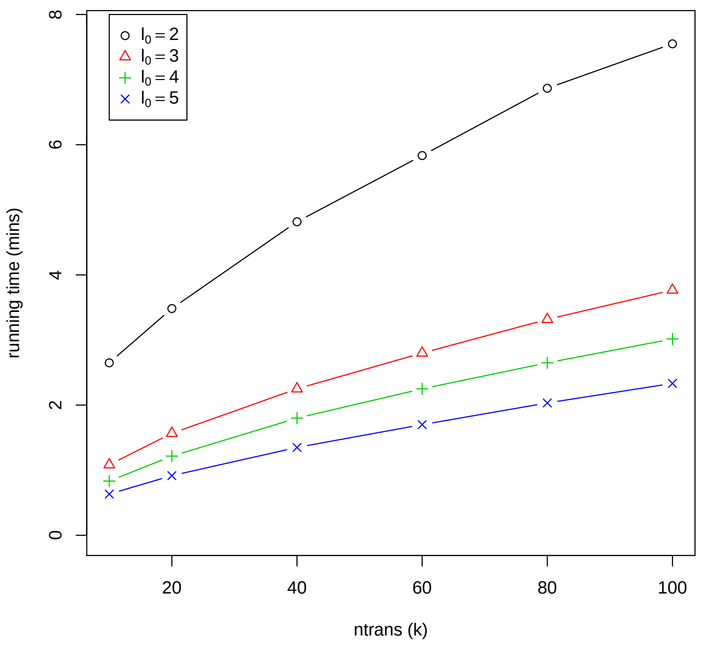

Figure 5.3: scalability of *DLiftMiner*

size of the dataset is 7401 records.

In addition, for scalability analysis, we generated another five datasets by setting *ntrans* to be $20k$, $40k$, $60k$, $80k$ and $100k$ (the other parameters are the same as in Table 5.1). Due to the fact that only transactions with all three types of items are kept, the actual sizes of these datasets are 14824, 29536, 44301, 59036 and 73729.

Performance of Pruning Methods

To evaluate our DLiftMiner algorithm, we compare the running times and pruning effects of different dual-lift bounding options. In our experiment, we set $s_0 = 100$, and we limit the maximum rule length to 8 (the length of rule $\langle X, Y \rangle \to z$ is defined as $|X| + |Y|$).

The experimental results are summarized in Table 5.2 and Figure 5.2. From Table 5.2, one can see that the pruning methods incorporated into *DLiftMiner* lead to dramatic

Table 5.3: Fields Extracted from VAERS dataset

VAERS field	**description**	**item type**
OTHER_MEDS	narrative about prescription or non-prescription drugs the vaccine recipient was taking at the time of vaccination.	X
CUR_ILL	narrative about any illness at the time of vaccination.	
HISTORY	narrative about any pre-existing physician-diagnosed allergies, birth defects, medical conditions that existed at the time of vaccination.	
PRIOR_VAX	prior vaccination event information.	
VAX_TYPE	the set of coded vaccines	Y
SYMPTOM	the set of MedDRA (Medical Dictionary for Regulatory Activities) coded symptoms	Z
DIED	died or not	
DISABLE	disability	
ER_VISIT	had ER visit or not	
HOSPITAL	hospitalized or not	
L_THREAT	life threatening or not	
RECOVD	recovered or not	
X_STAY	prolonged hospitalization or not	

improvements. Our best bounding strategy $(B^u + B^l)$ takes only 3.0%, 1.3%, 0.99%, and 0.72% of the running time of the baseline approach (where no dual-lift based pruning is applied), when l_0 takes values 2, 3, 4 and 5, respectively. Even compared with $(B_n^u + B_n^l)$, where only naive pruning is applied, $(B^u + B^l)$ takes only about 7% of its running time.

From Table 5.2, we can also see that there are dramatic performance improvements when the confidence lower bound is improved. However the confidence upper bound does not seem to play much role in reducing the running time, so that $(B^u + B^l)$ and $(B_n^u + B^l)$ ($(B^u + B_n^l)$ and $(B_n^u + B_n^l)$) have almost the same results.

Figure 5.2 plots the number of rules processed (i.e., *nrules*) versus different *rule length* for different bounding strategies of the algorithm. The vertical axis has been scaled to log scale ($\log_2(nrules + 1)$). It can be seen that most of the pruning has been produced by the confidence lower bound.

Scalability Study

Figure 5.3 illustrates how the *DLiftMiner* algorithm (with $(B^u + B^l)$) scales with respect to the size of input databases for four different l_0 values. We set the corresponding minimum support s_0 to be 1% of the *ntrans* parameter used for generating the dataset. It can be seen that the *DLiftMiner* algorithm scales linearly with the size of input dataset.

Effect of the Redundancy Based Pruning

Our experiments show that the redundancy based pruning can reduce the number of rules generated greatly. For instance, with $s_0 = 100$ and $l_0 = 2$, the number of rules generated from the first synthetic dataset is 79355 without redundancy pruning and 40517 with the redundancy pruning. However, we are not able to observe significant running time reduction by incorporating the redundancy based pruning. This may partially due to the fact that our DLiftMiner algorithm does not take into account all the redundancy defined in Definition 5.3.2.

5.5.2 Vaccine Adverse Reaction Detection

To evaluate the ability of the dual-lift measure to identify interesting rules, we applied the DLiftMiner algorithm to the Vaccine Adverse Event Reporting System (VAERS) [6] dataset. VAERS is a postmarketing survillance of spontaneous reporting system, co-managed by the Food and Drug Administration (FDA) and the Centers for Disease Control and Prevention (CDC), which is used for identifying evidences of vaccine adverse reactions. It collects information about adverse reactions (possible side effects) that occur after the administration of vaccines licensed for use in the United States. In additional to the vaccine and adverse reaction information, each report of VAERS also contains historical medical conditions under which the vaccine adverse reactions are developed, such as, medications the vaccine recipient was taking, pre-existing physician diagnosed allergies and birth defects, and any illness at the time of vaccination. Existing approaches for adverse reaction detection ([77], [78]) focus on vaccine-adverse reaction pairs that are statistically overrepresented. Potentially rich background information (e.g., historical medical conditions) associated with the reports are ignored. In contrast to their approaches, we treat the historical medical conditions and vaccines as two types of features and adverse reactions as the target variables, so that the data can be viewed as a dyadic dataset.

We downloaded the VAERS datasets that contain 21 years (from 1990 to 2010) of vaccination reports plus an additional non-domestic dataset. Fields listed in Table 5.3 were extracted and transformed to the $\langle X, Y, Z \rangle$ representation. For example, fields *OTHER_MEDS*, *CUR_ILL*, *HISTORY*, and *PRIOR_VAX* were converted into a set of historical medical conditions (items of set X) for each record. The items for sets Y and Z are constructed similarly.

One challenge we faced was that the fields we used to extract historical medical conditions were free text and thus not coded. To address this issue, we followed the following process to get a clean set of features. First, we split the free text into a bag of words (unigrams). Second, in order to capture some important terms beyond unigrams, we split the text by different delimiters: space, comma, semicolon, period and combinations of them. Third, among the set of all terms generated in the above two steps, we kept only those with frequency greater than or equal to 100. Next, we removed those terms that are obviously non-medical related. Finally, when we observed several terms having the same meanings, we mapped them to a normalized term. For example, "allergy codeine", "allergy to codeine" and "allergic to codeine" were mapped to "codeine allergy". After cleaning up, the total number of terms we got was 1187. For each record and field, the identified terms that were contained in the textual description were included into the set X. For the target items of Z, we removed those symptoms which do not appear very informative, for example, "Unevaluable event", "Accidental overdose", "Computerised tomogram normal", "Drug exposure during pregnancy" and so on. Also we removed all "injection site" symptoms and all symptoms containing the word "negative".

For our experiment, we set $s_0 = 15$ and $l_0 = 3$. And we used the *DLiftMiner* algorithm to generate four sets of rules as explained in the following. First, let S_1 contain all non-redundant frequent rules $Y \rightarrow z$ such that $\text{lift}(Y \rightarrow z|\mathcal{D}^0) \geq l_0$[2] . S_1 is the solution to the traditional adverse reaction detection problem, where historical medical conditions are not taken into account. We found 3617 rules in S_1. Second, let S_2 contain all non-redundant frequent rules $\langle X, Y \rangle \rightarrow z$ satisfying minimum dual-lift constraint. S_2 is the solution to Problem 5.3.4. Experiments show that our idea of combining dual-lift with redundancy is quite selective. We found 169 rules in S_2. Third, let S_3 contain the

[2] Since only one *lift* measure is involved, the concept of *redundancy* here is re-defined accordingly.

set of rules in S_2 whose vaccine reaction pairs also appear in S_1. The vaccine reactions in set S_3 are not new as they are also in S_1. However, we are able to find the set of historical medical conditions that interact with the vaccines which further increase the prevalence of the reactions. These additional historical medical conditions are the new information provided by the dual-lift measure. We found 91 rules S_3. Fourth, S_4 contains the rest of rules in S_2. S_4 is the set of new vaccine adverse reactions that cannot be discovered without incorporation of historical medical conditions. We found 78 rules in S_4. For each vaccine set Y from S_3 and S_4, we present the rule with the highest dual-lift in Tables 5.4 and 5.5, respectively. Rules listed in these two tables can be easily interpreted. For example, the rule $\langle$*OTHER_MEDS*-"estradiol", HPV4, *HOSPITAL*-"Y"$\rangle$ can be interpreted as that individuals taking the medication estradiol are more likely to be hospitalized when vaccinated with HPV4. However, the actual medical significance of these rules has to be determined by domain experts.

To evaluate the generated rules of our approach, we need a list of vaccine adverse reactions (ideally with associated medical conditions) that have been determined significant by domain experts. The Vaccine Injury Table is the only source of this kind that we are aware of [79]-[80] (see Table 5.6). The Vaccine Injury Table we are using is from [80] with coding system for symptoms changing from Coding Symbols for a Thesaurus of Adverse Reaction Terms (COSTART) to MedDRA. Only four of eight symptoms are listed in Table 5.6 because no rules were detected for another four (Arthritis, Encephalitis, Brachial plexopathy and Poliomyelitis).

Next, we present the set of rules in the Vaccine Injury Table that can be discovered by our method. However, be noted that this evaluation is limited due the small number of cases available. Table 5.7 contains rules in set S_1 whose symptoms are in Table 5.6 and vaccine sets have at least one of the reported vaccines for the corresponding symptoms. Our results in Table 5.7 suggests that vaccine-vaccine interactions play important role when developing these symptoms. Rotavirus vaccine (RV) was licensed on August 31 1998 for routine use in infants in a three-dose series given at two, four and six months of age. In mid-May 1999, nine reports were submitted. On October 14 1999, the vaccine manufacturer voluntarily withdrew its product from the market [81]. Our result in Table 5.7 shows RV-intussusception can be detected, and it also shows RV can interact with other vaccines and develop intussusception together. Table 5.8 contains rules in set S_2

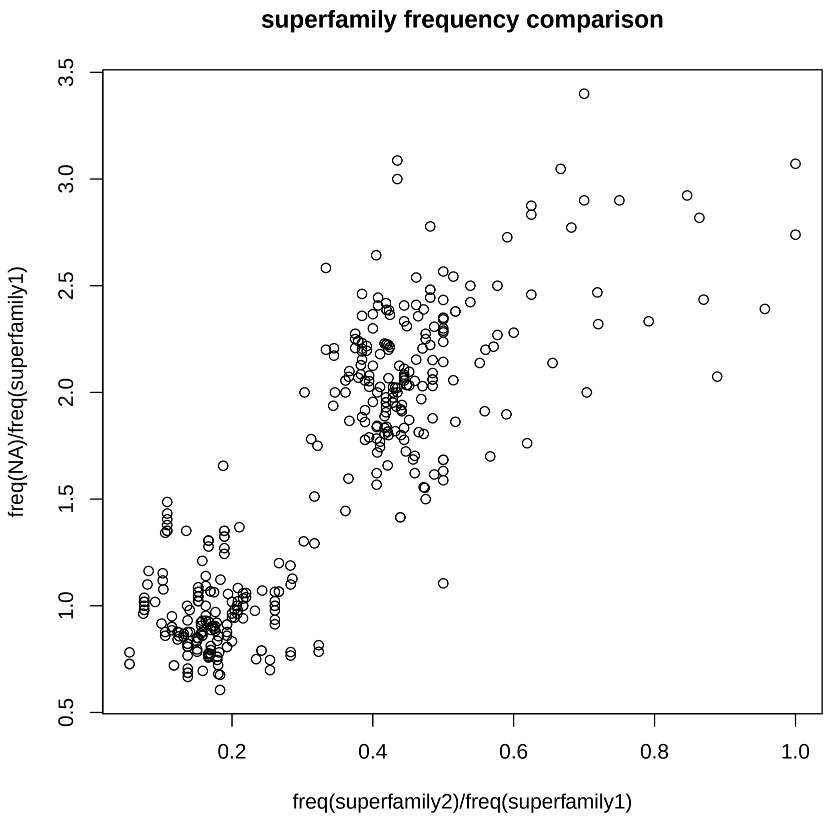

Figure 5.4: Protein Superfamily Frequency Comparison

whose symptoms and at least one of the vaccines appear in the Vaccine Injury Table. Only two rules are detected, both of which also appear in set S_4. This suggests that by incorporating historical medical conditions, we are likely to identify new rules other than existing well known ones.

5.5.3 Protein-Pocket Association Study

We apply the DLiftMiner algorithm to our in house protein-pocket association dataset from the chemical informatics application domain. The protein-ligand binding site prediction is an important problem for the structure based drug designs [82]. Studies [83] have shown that the binding site is commonly found in the largest pocket. As a result, the understanding of protein pocket associations is also of critical importance.

In this dataset, each instance consists of a set of protein features (i.e., items in X_0), a set of pocket features (i.e., items in Y_0) and the target variable (i.e. items in Z_0) which takes the value one (zero), indicating that the pocket is (not) associated with the protein. The protein features are subgraphs extracted from the protein's three dimensional structural representation. The pocket features contain the spacial information about different amino-acid residues. For the positive instances ($z = 1$), the pocket is identified for its associated protein by an in house pocket search algorithm. We construct the same number of negative instances ($z = 0$) by randomly matching the set of proteins and pockets from positive instances. We remove protein and pocket features that are either too infrequent (relative support less than 1%) or too frequent (relative support larger than 50%). The resulting dataset has a total of 10520 instances, 3825 protein features, and 16778 pocket features.

Our DLiftMiner algorithm can identify the set of protein features and pocket features that interact with each other so that the likelihood of the protein and pocket being associated is increased. These underlying features that are identified to be important for the protein-pocket associations may potentially be useful in the structure based drug designs. However, to avoid the extremely long running time (which is caused by the excessive number of pocket features), we limit the rule length to be 2 (i.e., $|X| = |Y| = 1$) [3] . We set the minimum support to be 105 (1%), the minimum dual-lift to be 1.5, and discovered 5715 dyadic association rules. As expected from the approach used to construct the negative instances, these rules' target values are all ones. In addition, they only contain 711 protein features and 229 pocket features.

To evaluate the discovered rules, we consider the set of instances that contain the rule $\langle X, Y \rangle \rightarrow z$. These set of proteins are associated with their pockets due to the same underlying feature interactions (i.e., the interactions between X and Y). It is reasonable to hypothesize that these proteins have similar functions and thus belong to the same superfamily. We test this hypothesis by utilizing both the SCOP and CATH protein classification systems. In the following, we only present our results for SCOP by simply noting that we get similar results for CATH. To reduce the number of rules under consideration, we first select a small set of more important rules by an instance

[3] An alternative approach is the dimension reduction for pocket features, which we leave as the future research.

based approach. That is, for each instance, we select the best rule (with the highest dual-lift) among all rules contained in that instance. This gives us a total of 397 rules. For each of these rules, we retrieve the set of instances containing the rule and roll up the corresponding proteins to the superfamily level of SCOP. During this step, if a protein does not have SCOP classification, we call its superfamily *NA*. Note that it may also be the case that a protein is mapped to multiple superfamilies due to the multiple residues associated with each pocket. After this, we get a superfamily distribution for each rule. For ease of discussion, we call the most frequent superfamily *superfamily1* and the second most frequent superfamily *superfamily2* excluding *NA*.

If the above hypothesis (i.e., proteins for each mined rule should, to a large extent, belong to the same superfamily) is correct, we would expect the distribution of superfamilies to be concentrated on *superfamily1*. So we compute the ratio of the frequency of *superfamily2* to the frequency of *superfamily1* and expect this value to be small. However, our results are affected by the fact that many proteins do not have SCOP superfamilies. To incorporate this effect, we plot *freq(NA)/freq(superfamily1)* against the *freq(superfamily2)/freq(superfamily1)* for all the mined rules in Figure 5.4. From Figure 5.4, we can see two clusters, where the first cluster is less affected by *NA*s and is also more consistent with our hypothesis, due to that *freq(superfamily2)* is much smaller than *freq(superfamily1)*. Next, we show that we have identified a set of rules with different superfamily distributions. For this, we count the number of rules for each (*superfamily1*, *superfamily2*) pair and present the results in Table 5.9. In order to get the feeling of how the superfamily distributions are different from the global distribution, which is computed by mapping all proteins to their superfamilies. We convert each superfamily to an id which is its ranking order in the global distribution. In this way, the global distribution can be represented as the pair $(1, 2)$. Table 5.9 shows that we have identified a set of non-trivial rules whose superfamily distributions are not only different from each other but also different from the global distribution. It can also be observed that each *superfamily1* tends to be followed by a different set of *superfamily2*s.

Algorithm 2 DLiftMiner

```
Input: current dual-lift threshold for each target item $l[z]$, the prefix itemsets $\langle X_p, Y_p\rangle$,
    and input databases $\mathcal{D}^{xy}$, $\mathcal{D}^{x}$, $\mathcal{D}^{y}$.
Output: a set of rules of the form $\langle X_p \cup X, Y_p \cup Y\rangle \to z$.
 1: remove infrequent items in $\mathcal{D}^{xy}$
 2: remove items from $\mathcal{D}^{x}$ and $\mathcal{D}^{y}$ that are not in $\mathcal{D}^{xy}$
 3: for $z$ in all target items of $\mathcal{D}^{xy}$ do
 4:     $C_z^{xy} \leftarrow \text{conf}(\emptyset \to z|\mathcal{D}^{xy})$
 5:     $C_z^{x} \leftarrow \text{conf}(\emptyset \to z|\mathcal{D}^{x})$
 6:     $C_z^{y} \leftarrow \text{conf}(\emptyset \to z|\mathcal{D}^{y})$
 7:     $\text{dlift}_z \leftarrow \min(C_z^{xy}/C_z^{x}, C_z^{xy}/C_z^{y})$
 8:     if $\text{dlift}_z \geq l[z]$ then
 9:         print $\langle X_p, Y_p\rangle \to z$.
10:         $l[z] \leftarrow \text{dlift}_z$.
11:     end if
12: end for
13: while get next item $i \in X_0 \cup Y_0$ from $\mathcal{D}^{xy}$ do
14:     $\mathcal{D}'^{xy} \leftarrow \mathcal{D}_i^{xy}$
15:     if $i \in X_0$ then
16:         $X_p' \leftarrow X_p \cup i$, $Y_p' \leftarrow Y_p$, $\mathcal{D}'^{x} \leftarrow \mathcal{D}_i^{x}$, $\mathcal{D}'^{y} \leftarrow \mathcal{D}^{y}$
17:     else
18:         $Y_p' \leftarrow Y_p \cup i$, $X_p' \leftarrow X_p$, $\mathcal{D}'^{x} \leftarrow \mathcal{D}^{x}$, $\mathcal{D}'^{y} \leftarrow \mathcal{D}_i^{y}$
19:     end if
20:     let $Z$ be the set of frequent target items in $\mathcal{D}'^{xy}$
21:     for $z$ in $Z$ do
22:         let $b^u_{z|\mathcal{D}'^{xy}}$ be the confidence upper bound of frequent rules $X \cup Y \to z$ in $\mathcal{D}'^{xy}$
23:         let $b^l_{z|\mathcal{D}'^{x}}$ ($b^l_{z|\mathcal{D}'^{y}}$) be the confidence lower bound of frequent rules $X \to z$ ($Y \to$
            $z$) in $\mathcal{D}'^{x}$ ($\mathcal{D}'^{y}$)
24:         $\textit{dlift-bnd}_z = \min(b^u_{z|\mathcal{D}'^{xy}}/b^l_{z|\mathcal{D}'^{x}}, b^u_{z|\mathcal{D}'^{xy}}/b^l_{z|\mathcal{D}'^{y}})$
25:         if $\textit{dlift-bnd}_z < l[z]$ then
26:             remove $z$ from $Z$
27:         end if
28:     end for
29:     if $|Z| > 0$ then
30:         construct $\mathcal{D}'^{xy}$; $i$ and target items not in $Z$ are not copied; construct $\mathcal{D}'^{x}$ and
            $\mathcal{D}'^{y}$
31:         call DLiftMiner with $l[z]$, $X_p'$, $Y_p'$, $\mathcal{D}'^{xy}$, $\mathcal{D}'^{x}$, $\mathcal{D}'^{y}$
32:     end if
33:     remove $i$ from $\mathcal{D}^{xy}$
34:     if $i \in X_0$ then
35:         remove $i$ from $\mathcal{D}^{x}$
36:     else
37:         remove $i$ from $\mathcal{D}^{y}$
38:     end if
39: end while
```

Table 5.4: Subset of vaccine-symptom associations with medical conditions in S_3.

Symptom	**vaccine code(s)**	**medical condition(s)**[a]	**dual-lift**
Autism	MMR	*HISTORY*-“fever”	4.7
C-reactive protein increased	PNC	*HISTORY*-“Pregnancy”	9.3
Drug toxicity	HEP	*HISTORY*-“Pregnancy”	7.8
Febrile convulsion	DTPHIB	*HISTORY*-“seizure”	3.3
Influenza like illness	LYME	*HISTORY*-“Anxiety”	3.4
Injected limb mobility decreased	PPV	*CUR_ILL*-“allergy”	3.6
Irritability	HIBV	*OTHER_MEDS*-“prevacid”	5.2
	(MMR,OPV)		3.0
	(DTP,MMR)		3.4
Otitis media	(HIBV,MMR,OPV)	*CUR_ILL*-“Ear infection”	3.5
	(DTP,HIBV,MMR)		3.6
	(DTP,MMR,OPV)		3.9
Rash vesicular	VARZOS	*CUR_ILL*-“Hypothyroidism”	3.3
Swelling	(IPV,MMR)	*PRIOR_SYM*-“Swelling” [b]	3.8
Throat tightness	FLU	(*HISTORY*-“Asthma”,*HISTORY*-“allergy”)	3.7
Urine human chorionic gonadotropin positive	HPV4	*HISTORY*-“Pregnancy”	8.8
Varicella	VARCEL	*CUR_ILL*-“Asthma”	4.2

[a] Medical conditions are expressed in the form of *FIELD_NAME*-“term”, where *FIELD_NAME* is from Table 5.3
[b] *PRIOR_SYM* is symptoms developed from prior vaccination, extracted from *PRIOR_VAX* field.

Table 5.5: Subset of vaccine-symptom associations with medical conditions in S_4.

Symptom	**vaccine code(s)**	**medical condition(s)**[a]	**dual-lift**
HOSPITAL-"Y"[b]	HPV4	*OTHER_MEDS*-"estradiol"	3.6
	PNC	*HISTORY*-"diabetes"	3.2
Drug ineffective	RAB	*OTHER_MEDS*-"Hepatitis"	7.5
Erythema	VARCEL	(*CUR_ILL*-"allergy",*HISTORY*-"allergy")	3.6
Herpes zoster	(MMR,VARCEL)	*HISTORY*-"Otitis media"	4.6
Hypotension	TD	*OTHER_MEDS*-"purified protein derivative"	3.3
Joint range of motion decreased	FLU	*HISTORY*-"deficit"	3.8
Lymphadenopathy	MMR	*OTHER_MEDS*-"premarin"	5.5
Nervous system disorder	HEP	*HISTORY*-"Pregnancy"	6.3
Otitis media	(DTP,HIBV,OPV)	*PRIOR_SYM*-"fever"[c]	3.2
Rash	(HIBV,OPV)	*HISTORY*-"penicillin"	3.2
Swelling	IPV	*PRIOR_SYM*-"Swelling"[c]	3.6
	(DTAP,MMR)		3.2
Urticaria	(DTAP,IPV,MMR)	*HISTORY*-"premature"	3.4
Vasodilatation	DTP	(*HISTORY*-"Asthma",*HISTORY*-"allergy")	4.5
	OPV	(*HISTORY*-"allergy",*HISTORY*-"ceclor")	4.2
Wheezing	DTAP	(*HISTORY*-"allergy",*OTHER_MEDS*-"albuterol")	3.0
White blood cell count increased	PPV	*HISTORY*-"Anxiety"	3.8

[a] Medical conditions are expressed in the form of *FIELD_NAME*-"term", where *FIELD_NAME* is from Table 5.3
[b] *HOSPITAL*-"Y" is extracted from *HOSPITAL* field, meaning the patient is hospitalized. [c] *PRIOR_SYM* is symptoms developed from prior vaccination, extracted from *PRIOR_VAX* field.

Table 5.6: Vaccine-Symptom associations from vaccine injury table.

Symptom	**Reported vaccine code(s)**
Anaphylactic reaction	DT,DTAP,DTAPH,DTP HEP,IPV,MMR,TD
Encephalopathy	DTAP,DTAPH DTP,MMR
Intussusception	RV
Thrombocytopenic purpura	M[a], MM[a], MMR, MR[a]

[a] Vaccines excluded from analysis due to low or zero frequencies

Table 5.7: Vaccine-Symptom associations from Vaccine Injury Table found in S_1.

Symptom	**Vaccine code(s)**	**lift**
Encephalopathy	(DTAP, HIBV)	3.2
	(HIBV, MMR)	3.1
Intussusception	RV	85.2
	(IPV, RV)	95.5
	(HIBV, RV)	86.4
	(DTAP, RV)	88.2
	(HIBV, IPV, RV)	105.0
	(DTAP, IPV, RV)	101.8
	(DTAP, HIBV, RV)	95.5
	(DTAP, HIBV, IPV, RV)	110.1
Thrombocytopenic purpura	(HIBV , MMR)	4.5

Table 5.8: Vaccine-Symptom associations from Vaccine Injury Table found in S_2 and S_4

Symptom	**Vaccine code(s)**	**Medical Condition(s)**	**dual-lift**
Anaphylactic reaction	MMR	cur_ill-“allergy”	3.9
	(MMR, VARCEL)	history-“allergy”	3.0

Table 5.9: Protein (superfamily1, superfamily2) distribution

id1\id2	1	2	6	9	10	13	17	22	25	27	30	33	36	44	55	111	194
1			5												46	9	
2			34	13			21			13							
3					74	1		1			5	11					1
6	2	7							70				83	1			

[a] Superfamily id (i.e., id1, id2 for superfamily1, superfamily2, respectively) is defined to be the ranking order from the superfamily global distribution, which is computed by mapping all proteins to their superfamilies.

Chapter 6

Conclusion and Future Research

In summary, we presented a novel predictive modeling framework that integrates the association rule mining with the EM optimization. This framework is applied to both the regression and the classification problems, and the corresponding models are called AREM and ACEM. Model training of AREM/ACEM consists of two major components, where the first component is the instance based itemset mining and the second component is a probabilistic model for learning rule's parameters. For the instance based itemset mining component, we develop an efficient algorithm IBIMiner that applies various pruning strategies to reduce the search space so that the running time is reduced. Experimental evaluation shows that IBIMiner can be of an order of magnitude faster than the baseline approach. For the rule's parameters learning component, we develop the probabilistic model that captures interactions among various itemsets. Based on the probabilistic model, an EM algorithm is derived to learn rule's parameters. Experimental evaluation shows that the EM steps improve the predictive performance greatly on almost all the datasets. We also show that AREM/ACEM can perform better than many of the state of art regression/classification models. The last part of this thesis studies the association rule mining problem on the dyadic dataset. We proposed a new metric called dual-lift to capture the dyadic interactions between two distinct types of feature itemsets. An efficient algorithm is developed to push various constraints into the rule mining process and achieves good performance. We apply the dual-lift mining formulation to some real world applications and present some interesting results.

We identify some future directions to extend the research in this thesis. First, our predictive modeling framework can potentially to be improved. For example, there is an inconsistency between the approach we used to train the model and the approach we used to make predictions. That is, during training, all itemsets from $\mathcal{X}$ that are contained in the training instance are used, while during prediction, only the top k itemsets are used. One approach for solving this problem is that after some EM steps, we can pick k itemsets with the highest weights for each instance and continue the EM training using only these itemsets. It is reasonable to believe that this approach may improve the performance since the function that is used for prediction is directly optimized. Second, our modeling framework may be able to extend to more problems than regression and (multi-class) classification. One example is the multi-label classification, in which each instance may be associated with multiple class labels. One approach is that instead of predicting the class label with the highest score, we may predict multiple labels by applying some selection criterion. The second approach is to incorporate multi-labels into the probabilistic model directly, in which we will need a probabilistic model for $P[y|x]$, where y is a set of multiple class labels. Third, better quality functions may be developed to further improve the model. Especially for the classification model, where we only used the confidence as the quality function. However, in developing new quality functions, it should be kept in mind that the itemset mining algorithm needs to be adapted to them as well. In particular, new pruning strategies will be needed in order to reduce the running times. Fourth, our itemset mining algorithm primarily focused on finding the exact solution to the instance based mining problem. It could be the case that finding an approximate solution can reduce the running time further without necessarily downgrading the predictive performances. Finally, the idea of dyadic interaction for association rule mining may be extended to more complex applications (e.g., the sequential dyadic datasets).

References

[1] Sotiris Kotsiantis and Dimitris Kanellopoulos. Discretization techniques: A recent survey. 2006.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[3] Zhonghua Jiang and George Karypis. Arem: A novel associative regression model based on em algorithm. In *Proceedings of the 17th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '13. Springer-Verlag, 2013.

[4] Zhonghua Jiang and George Karypis. A novel regression model combining instance based rule mining with em algorithm. *TKDD*, submitted.

[5] Zhonghua Jiang and George Karypis. Automatic detection of vaccine adverse reactions by incorporating historical medical conditions. In *Proceedings of the 2nd ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, pages 547–549, 2011.

[6] RT Chen, SC Rastogi, JR Mullen, and *et al.* The vaccine adverse event reporting system (vaers). *Vaccine*, 12:542–550, 1994.

[7] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. of 20th Intl. Conf. on VLDB*, pages 487–499, 1994.

[8] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD Conference*, pages 1–12, 2000.

[9] Mohammed J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12:372–390, 2000.

[10] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. pages 80–86, 1998.

[11] Maria luiza Antonie, Osmar R. Zaiane, Alexandru Coman, and Ru Coman. Associative classifiers for medical images. In *Lecture Notes in A.I. 2797, Mining Multimedia and Complex Data*, pages 68–83. Springer-Verlag, 2003.

[12] Osmar R. Zaïane and Maria-Luiza Antonie. Classifying text documents by associating terms with text categories. In *Australasian Database Conference*, 2002.

[13] Wenmin Li, Jiawei Han, and Jian Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *ICDM*, pages 369–376, 2001.

[14] Elena Baralis and Paolo Garza. A lazy approach to pruning classification rules. In *ICDM*, pages 35–42, 2002.

[15] Elena Baralis, Silvia Chiusano, and Paolo Garza. On support thresholds in associative classification. In *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pages 553–558, New York, NY, USA, 2004. ACM.

[16] Fadi A. Thabtah, Peter I. Cowling, and Yonghong Peng. Mcar: multi-class classification based on association rule. In *AICCSA*, page 33, 2005.

[17] Fadi A. Thabtah, Peter Cowling, and Yonghong Peng. Mmac: A new multi-class, multi-label associative classification approach. *Data Mining, IEEE International Conference on*, 0:217–224, 2004.

[18] Xiaoxin Yin and Jiawei Han. Cpar: Classification based on predictive association rules. In *SDM*, 2003.

[19] J. Ross Quinlan and R. Mike Cameron-Jones. Foil: A midterm report. In *ECML*, pages 3–20, 1993.

[20] Ke Wang, Senqiang Zhou, and Yu He. Growing decision trees on support-less association rules. In *KDD*, pages 265–269, 2000.

[21] Bing Liu, Yiming Ma, and Ching kian Wong. Classification using association rules: Weaknesses and enhancements, 2001.

[22] Aysel Ozgur, Pang-Ning Tan, and Vipin Kumar. Rba: An integrated framework for regression based on association rules. In *SDM*, 2004.

[23] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[24] Jianyong Wang and George Karypis. Harmony: Efficiently mining the best rules for classification. In *In Proc. of SDM*, pages 205–216, 2005.

[25] Hong Cheng, Xifeng Yan, Jiawei Han, and Philip S. Yu. Direct discriminative pattern mining for effective classification. In *ICDE*, pages 169–178, 2008.

[26] Gao Cong, Anthony K. H. Tung, Xin Xu, Feng Pan, and Jiong Yang. Farmer: Finding interesting rule groups in microarray datasets. In *SIGMOD Conference*, pages 143–154, 2004.

[27] Gao Cong, Kian-Lee Tan, Anthony K. H. Tung, and Xin Xu. Mining top-k covering rule groups for gene expression data. In *SIGMOD Conference*, pages 670–681, 2005.

[28] Adriano Veloso, Wagner Meira Jr., and Mohammed J. Zaki. Lazy associative classification. In *ICDM*, pages 645–654, 2006.

[29] S. P. Ibrahim, Syed, K. R. Chandran, and C. J. Kanthasamy, Kabila. Laci: Lazy associative classification using information gain. *IJET, International Journal Of Engineering and Technology*, 4:1–6, 2012.

[30] Johannes Frnkranz and Gerhard Widmer. Incremental reduced error pruning, 1994.

[31] William W. Cohen. Fast effective rule induction. In *ICML*, pages 115–123, 1995.

[32] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[33] J. Ross Quinlan. Cubist. `http://www.rulequest.com`.

[34] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[35] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. *Journal of Machine Learning Research - Proceedings Track*, 14:1–24, 2011.

[36] Meghana Deodhar and Joydeep Ghosh. A framework for simultaneous co-clustering and learning from complex data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 250–259, New York, NY, USA, 2007. ACM.

[37] Meghana Deodhar and Joydeep Ghosh. Mining for the most certain predictions from dyadic data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 249–258, New York, NY, USA, 2009. ACM.

[38] David Lo, Hong Cheng, and Lucia. Mining closed discriminative dyadic sequential patterns. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 21–32, New York, NY, USA, 2011. ACM.

[39] Luc De Raedt and Albrecht Zimmermann. Constraint-based pattern set mining. In *SDM*, 2007.

[40] Jian Pei and Jiawei Han. Can we push more constraints into frequent pattern mining? In *KDD*, pages 350–354, 2000.

[41] Raymond T. Ng, Laks V.S. Lakshmanan, Alex Pang, and Jiawei Hah. Exploratory mining and pruning optimizations of constrained associations rules. pages 13–24. ACM Press, 1998.

[42] S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. *In Proc. of SIGMOD'97*, pages 265–276, 1997.

[43] Carson Kai-Sang Leung, Laks V. S. Lakshmanan, and Raymond T. Ng. Exploiting succinct constraints using fp-trees. *SIGKDD Explorations*, 4(1):40–49, 2002.

[44] J. Pei and J. Han. Can we push more constraints into frequent pattern mining? *In Proceedings of ACM SIGKDD'00*, 2000.

[45] Daniel Kifer, Johannes Gehrke, Cristian Bucila, and Walker M. White. How to quickly find a witness. In *Constraint-Based Mining and Inductive Databases*, pages 216–242, 2004.

[46] Francesco Bonchi and Claudio Lucchese. Pushing tougher constraints in frequent pattern mining. *In Proc. of PAKDD*, 2005.

[47] Masakazu Seno and George Karypis. Lpminer: An algorithm for finding frequent itemsets using length-decreasing support constraint. *ICDM'01, Nov*, 2001.

[48] Shinichi Morishita and Jun Sese. Traversing itemset lattices with statistical metric pruning. *POD, Dallas, T X USA. ACM*, 2000.

[49] Hui Xiong, Shashi Shekhar, Pang-Ning Tan, and Vipin Kumar. Exploiting a support-based upper bound of pearson's correlation coefficient for efficiently identifying strongly correlated pairs. *KDD'04, August, Seattle, Washington, USA. ACM*, pages 22–25, 2004.

[50] Roberto J. Bayardo Jr., Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. *In Proc. of the 15th Int'l Conf. on Data Engineering*, pages 188–197, 1999.

[51] Franciso Herrera, Cristbal Carmona, Pedro Gonzlez, and Mara del Jesus. An overview on subgroup discovery: foundations and applications. *Knowledge and Information Systems*, pages 1–31, 2010.

[52] Martin Atzmueller and Florian Lemmerich. Fast and effective subgroup mining for continuous target variables. In *Proc. LWA 2009 (Knowledge Discovery and Machine Learning Track)*, Darmstadt, Germany, 2009. University of Darmstadt.

[53] Henrik Grosskreutz, Stefan Rping, and Stefan Wrobel. Tight optimistic estimates for fast subgroup discovery. In *In ECML/PKDD*, pages 440–456, 2008.

[54] Bestbuy. `http://www.bestbuy.com`.

[55] Citysearch. `http://www.citysearch.com`.

[56] Yelpacademic. `http://www.yelp.com/academic_dataset`.

[57] Marie-Catherine de Marneffe and Christopher D. Manning. The stanford typed dependencies representation. CrossParser '08, pages 1–8, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[58] Movielens. http://www.grouplens.org/node/12.

[59] Airline. http://stat-computing.org/dataexpo/2009.

[60] Statlib. http://lib.stat.cmu.edu/datasets.

[61] Alex J. Smola and Bernhard Schlkopf. A tutorial on support vector regression. Technical report, STATISTICS AND COMPUTING, 2003.

[62] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

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

[64] M.F. Triola. *Elementary statistics*. Pearson/Addison-Wesley, 2004.

[65] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18, 1990.

[66] Yelp. http://www.yelp.com.

[67] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[68] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[69] Yongwook Yoon and Gary Geunbae Lee. Text categorization based on boosting association rules. In *ICSC*, pages 136–143, 2008.

[70] Mohammed J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12:372–390, 2000.

[71] J Han, J Pei, and Y Yin. *Mining frequent patterns without candidate generation*, volume 8, pages 1–12. ACM Press, 2000.

[72] Muthukumarasamy Karthikeyan, Robert C. Glen, and Andreas Bender. General melting point prediction based on a diverse compound data set and artificial neural networks. *Journal of Chemical Information and Computer Sciences*, 45(3):581–590, 2005.

[73] Census. `http://archive.ics.uci.edu/ml/databases/census1990/`.

[74] Noel O'Boyle, Michael Banck, Craig James, Chris Morley, Tim Vandermeersch, and Geoffrey Hutchison. Open Babel: An open chemical toolbox. *Journal of Cheminformatics*, 3(1):33+, October 2011.

[75] Nikil Wale and George Karypis. Acyclic subgraph-based descriptor spaces for chemical compound retrieval and classification. Technical Report #06-008, UMN CSE, 2006.

[76] Rakesh Agrawal and Ramarishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, 1994.

[77] E Van Puijenbroek, W Diemont, and K Van Groothest. Application of quantitative signal detection in the dutch spontaneous reporting system for adverse drug reactions. *Drug Saf*, 26 (5):293–301, 2003.

[78] A Bate, M Lindquist, I Edwards, and *et al.* A bayesian neural network method for adverse drug reaction signal generation. *Eur J Clin Pharmacol*, 54:315–21, 1998.

[79] Stratton KR, Howe CJ, and Johnston RB (eds). *Adverse Events Associated with Childhood Vaccines: Evidence Bearing on Casuality.* National Academy Press: Washington, DC, Vaccine Safety Committee, Institute of Medicine, 1994.

[80] David Banks, Emily Jane Woo, and *et al.* Comparing data mining methods on the vaers database. *Pharmacoepidemiology and Drug Safety*, 14:601–609, 2005.

[81] Manette T. Niu, Diane E. Erwin, and M. Miles Braun. Data mining in the us vaccine adverse event reporting system (vaers): early detection of intussusception and other events after rotavirus vaccination. *Vaccine*, 19:4627–4634, 2001.

[82] Alasdair T. Laurie and Richard M. Jackson. Methods for the prediction of protein-ligand binding sites for structure-based drug design and virtual ligand screening. *Current protein & peptide science*, 7(5):395–406, October 2006.

[83] R. Laskowski. SURFNET: A program for visualizing molecular surfaces, cavities, and intermolecular interactions. *Journal of Molecular Graphics*, 13(5):323–330, October 1995.

[84] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

Appendix A

The Generic EM Algorithm

Assume the training database is given by $\boldsymbol{y}$. Let $\boldsymbol{x}$ be a list of hidden variables that are not directly observable from the training database. Let $\boldsymbol{\theta}$ is a list of model parameters that we want to learn by the maximum likelihood estimation.

The log-likelihood function for the training database is

$$\log(P[\boldsymbol{y}|\boldsymbol{\theta}]) = \log(\sum_{\boldsymbol{x}} P[\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta}]).$$

The direct optimization of $\log(P[\boldsymbol{y}|\boldsymbol{\theta}])$ is difficult, since it is the logarithm of a summation. But the optimization of the complete data log-likelihood $\log(P[\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta}])$ is significantly easier. The whole idea of the EM algorithm is to move the logarithm function inside the summation so that the problem gets much easier. The generic EM algorithm described in this Section is based on [84].

Introduce an arbitrary distribution function $q(\boldsymbol{x})$ over the hidden variable $\boldsymbol{x}$. The log-likelihood function can be written as

$$\log(P[\boldsymbol{y}|\boldsymbol{\theta}]) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q||p),$$

where

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log(\frac{P[\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta}]}{q(\boldsymbol{x})}),$$

$$\mathrm{KL}(q||p) = -\sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log(\frac{P[\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}]}{q(\boldsymbol{x})}),$$

where $\mathrm{KL}(q||p)$ is the Kullback-Leibler divergence between $q(\boldsymbol{x})$ and $p = P[\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}]$. The trick of EM algorithm lies in the fact that $\mathrm{KL}(q||p) \geq 0$, so $\mathcal{L}(q, \boldsymbol{\theta})$ is a lower bound of $\log(P[\boldsymbol{y}|\boldsymbol{\theta}])$.

EM algorithm is a two-stage iterative optimization technique. Suppose at step t, the current set of parameters is $\boldsymbol{\theta}_t$. In the E-step, the lower bound $\mathcal{L}(q, \boldsymbol{\theta}_t)$ is maximized with respect to $q(\boldsymbol{x})$. This happens when $\mathrm{KL}(q||p) = 0$, or, $q_t(\boldsymbol{x}) = P[\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}_t]$. So we have $\log(P[\boldsymbol{y}|\boldsymbol{\theta}_t]) = \mathcal{L}(q_t, \boldsymbol{\theta}_t)$. In the M-step, the goal is to maximize function $\mathcal{L}(q_t, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. Let the optimal parameters be $\boldsymbol{\theta}_{t+1}$. We have

$$\log(P[\boldsymbol{y}|\boldsymbol{\theta}_{t+1}]) \geq \mathcal{L}(q_t, \boldsymbol{\theta}_{t+1}) \geq \mathcal{L}(q_t, \boldsymbol{\theta}_t) = \log(P[\boldsymbol{y}|\boldsymbol{\theta}_t]).$$

So, after the E-step and M-step, the log-likelihood function is guaranteed to increase (unless the optimal parameters have been reached).

If we substitute $q_t(\boldsymbol{x}) = P[\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}_t]$ into $\mathcal{L}(q_t, \boldsymbol{\theta})$, we get

$$\mathcal{L}(q_t, \boldsymbol{\theta}) = \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}_t) + \text{const},$$

where,

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}_t) = \sum_{\boldsymbol{x}} P[\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}_t] \log(P[\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta}]).$$

As we mentioned before, the logarithm function has been moved inside the summation in $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}_t)$. And notice that $P[\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}_t]$ is independent of $\boldsymbol{\theta}$ so the original optimization problem gets simplified dramatically in the M-step. In the case when training instances are independent of each other, we have

$$
\begin{aligned}
\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}_t) &= \sum_{\boldsymbol{x}} P[\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}_t] \log(P[\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta}]) \\
&= \sum_{\boldsymbol{x}} P[\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}_t] \sum_i \log(P[y_i, x_i|\boldsymbol{\theta}]) \\
&= \sum_i \sum_{\boldsymbol{x}} P[\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}_t] \log(P[y_i, x_i|\boldsymbol{\theta}]) \\
&= \sum_i \sum_{\boldsymbol{x}} (\prod_{j\neq i} P[x_j|y_j, \boldsymbol{\theta}_t]) P[x_i|y_i, \boldsymbol{\theta}_t] \log(P[y_i, x_i|\boldsymbol{\theta}]) \\
&= \sum_i (\prod_{j\neq i} \sum_{x_j} P[x_j|y_j, \boldsymbol{\theta}_t]) \sum_{x_i} P[x_i|y_i, \boldsymbol{\theta}_t] \log(P[y_i, x_i|\boldsymbol{\theta}]) \\
&= \sum_i \sum_{x_i} P[x_i|y_i, \boldsymbol{\theta}_t] \log(P[y_i, x_i|\boldsymbol{\theta}]) \\
&= \sum_i \sum_{x_i} \pi_{i,x_i,t} \log(P[y_i, x_i|\boldsymbol{\theta}]), \qquad \text{(A.1)}
\end{aligned}
$$

where

$$\pi_{i,x_i,t} = P[x_i|y_i, \boldsymbol{\theta}_t].$$

In summary, in order to maximize $\log(P[\boldsymbol{y}|\boldsymbol{\theta}])$, we can iteratively update parameters $\boldsymbol{\theta}$. At iteration t, the E-step is to calculate $\pi_{i,x_i,t}$ defined as $P[x_i|y_i, \boldsymbol{\theta}_t]$, and the M-step is to find $\boldsymbol{\theta}_{t+1}$ to maximize $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}_t)$.

Appendix B

Theorems Proof

In this appendix, we prove Theorems 4.1.5, 4.1.6, 4.1.7 and 5.4.4. Note that we only provide the proofs for the lower bounding problems, since the proofs for the upper bounding problems are similar.

B.1 Lemmas

We will apply the following lemmas repeated:

Lemma B.1.1. *Let* $\mathbf{e}_1$*,* $\mathbf{e}_2$ *be 2-D vectors, satisfying* $e_{1,h} > 0$ *and* $e_{2,h} > 0$*. We have*

1. *If* $||\mathbf{e}_1|| \leq ||\mathbf{e}_2||$*, then* $||\mathbf{e}_1|| \leq ||\mathbf{e}_1 + \mathbf{e}_2|| \leq ||\mathbf{e}_2||$*.*

2. *If* $||\mathbf{e}_1 + \mathbf{e}_2|| \leq ||\mathbf{e}_2||$*, then* $||\mathbf{e}_1|| \leq ||\mathbf{e}_2||$*.*

3. *If* $||\mathbf{e}_1 + \mathbf{e}_2|| \geq ||\mathbf{e}_1||$*, then* $||\mathbf{e}_1|| \leq ||\mathbf{e}_2||$*.*

Proof. (1): $||\mathbf{e}_1 + \mathbf{e}_2|| \leq ||\mathbf{e}_2|| \Leftrightarrow (e_{1,v} + e_{2,v})/(e_{1,h} + e_{2,h}) \leq e_{2,v}/e_{2,h} \Leftrightarrow e_{1,v} * e_{2,h} \leq e_{2,v} * e_{1,h} \Leftrightarrow e_{1,v}/e_{1,h} \leq e_{2,v}/e_{2,h} \Leftrightarrow ||\mathbf{e}_1|| \leq ||\mathbf{e}_2||$. It can also be seen that the equality holds only when $||\mathbf{e}_1|| = ||\mathbf{e}_2||$. The other direction is similar and thus omitted.

(2), (3): Assume the opposite is true: $||\mathbf{e}_1|| > ||\mathbf{e}_2||$. From the proof for (1), we have $||\mathbf{e}_1|| > ||\mathbf{e}_1 + \mathbf{e}_2|| > ||\mathbf{e}_2||$, which is the contradiction. □

Lemma B.1.2. *For $n \geq 1$, let $\{\mathbf{e}_g | g = 1, 2, \ldots, n\}$ be a set of 2-D vectors satisfying $e_{g,h} > 0$. Assume $||\mathbf{e}_1|| \leq ||\mathbf{e}_2|| \leq \cdots \leq ||\mathbf{e}_n||$. We have*

$$||\mathbf{e}_1|| \leq ||\sum_{g=1}^{n} \mathbf{e}_g|| \leq ||\mathbf{e}_n||.$$

Proof. For $n = 1$, there is only one term in the summation, so it is correct. When $n > 1$, let $\mathbf{E}_i = \sum_{g=1}^{i} \mathbf{e}_g$. Assume the conclusion is correct for $n - 1$. So, $||\mathbf{e}_1|| \leq ||\mathbf{E}_{n-1}|| \leq ||\mathbf{e}_{n-1}||$. Combined with $||\mathbf{e}_{n-1}|| \leq ||\mathbf{e}_n||$, we have $||\mathbf{E}_{n-1}|| \leq ||\mathbf{e}_n||$. From Lemma B.1.1, we have $||\mathbf{E}_{n-1}|| \leq ||\mathbf{E}_{n-1} + \mathbf{e}_n|| = ||\mathbf{E}_n|| \leq ||\mathbf{e}_n||$. So, $||\mathbf{e}_1|| \leq ||\mathbf{E}_n|| \leq ||\mathbf{e}_n||$, i.e., the conclusion is also correct for n. $\square$

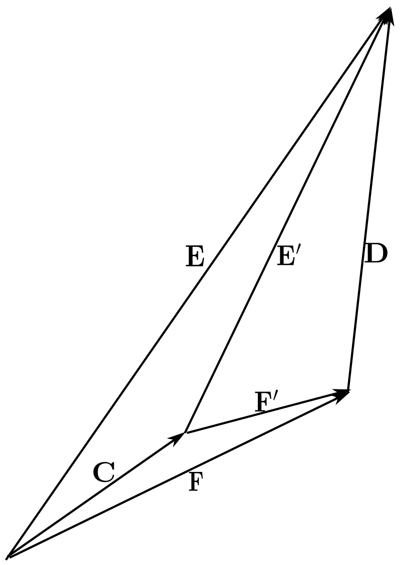

Figure B.1: Lemmas B.1.3 and B.1.4

Lemma B.1.3. *Given 2-D vectors $\mathbf{C}$, $\mathbf{E}'$, $\mathbf{F}'$, $\mathbf{E} = \mathbf{C} + \mathbf{E}'$ and $\mathbf{F} = \mathbf{C} + \mathbf{F}'$ satisfying (a) $E'_h > 0$ unless $\mathbf{E}' = 0$, (b) $F'_h > 0$ unless $\mathbf{F}' = 0$, (c) $E_h \geq F_h > 0$, (d) $||\mathbf{E}'|| \geq ||\mathbf{F}'||$ if $F'_h > 0$, and (e) $||\mathbf{E}'|| \geq ||\mathbf{F}||$ if $E'_h > 0$. Then we have $||\mathbf{E}|| \geq ||\mathbf{F}||$.*

Proof. This lemma is graphically shown in Figure B.1. We define $\mathbf{D} = \mathbf{E} - \mathbf{F}$. From (c), we have $D_h \geq 0$.

First, consider the case $D_h = 0$. We have $E_h = F_h$ and $E'_h = F'_h$. If $E'_h = F'_h = 0$, from (a), (b), we have $\mathbf{E}' = \mathbf{F}' = 0$. So $\mathbf{E} = \mathbf{F} = \mathbf{C}$, and the conclusion follows. If

$E'_h = F'_h > 0$, from (d), we get $E'_v \geq F'_v \Rightarrow E_v \geq F_v$. Combined with $E_h = F_h$, we get $||\mathbf{E}|| \geq ||\mathbf{F}||$.

Next, consider the case $D_h > 0 \Rightarrow E'_h > 0$. First, we want to prove $||\mathbf{D}|| \geq ||\mathbf{E}'||$. If $F'_h = 0$, from (b), we have $\mathbf{F}' = 0 \Rightarrow \mathbf{D} = \mathbf{E}'$. If $F'_h > 0$, from (d), $||\mathbf{E}'|| \geq ||\mathbf{F}'||$. We apply Lemma B.1.1 repeatedly to get $||\mathbf{E}'|| = ||\mathbf{D} + \mathbf{F}'|| \geq ||\mathbf{F}'|| \Rightarrow ||\mathbf{D}|| \geq ||\mathbf{F}'|| \Rightarrow ||\mathbf{D}|| \geq ||\mathbf{D} + \mathbf{F}'|| = ||\mathbf{E}'||$. Second, combining $||\mathbf{D}|| \geq ||\mathbf{E}'||$ with (e) gives $||\mathbf{D}|| \geq ||\mathbf{F}||$. Then from Lemma B.1.1, $||\mathbf{D}|| \geq ||\mathbf{F}|| \Rightarrow ||\mathbf{E}|| = ||\mathbf{F} + \mathbf{D}|| \geq ||\mathbf{F}||$.

□

Lemma B.1.4. *Given 2-D vectors* $\mathbf{C}$, $\mathbf{E}'$, $\mathbf{F}'$, $\mathbf{E} = \mathbf{C} + \mathbf{E}'$ *and* $\mathbf{F} = \mathbf{C} + \mathbf{F}'$ *satisfying (a)* $E'_h > 0$ *unless* $\mathbf{E}' = 0$, *(b)* $F'_h > 0$ *unless* $\mathbf{F}' = 0$, *(c)* $C_h > 0$, *(d)* $||\mathbf{F}|| \geq ||\mathbf{F}'||$ *if* $F'_h > 0$, *and (e)* $||\mathbf{E}'|| \geq ||\mathbf{F}||$ *if* $E'_h > 0$. *Then we have* $||\mathbf{E}|| \geq ||\mathbf{F}||$.

Proof. This lemma is graphically shown in Figure B.1.

Consider the case $F'_h = 0$. From (b), we have $\mathbf{F}' = 0$, and $\mathbf{E} = \mathbf{F} + \mathbf{E}'$. If $E'_h > 0$, from (e), we have $||\mathbf{E}'|| \geq ||\mathbf{F}||$. Then we apply Lemma B.1.1 and get $||\mathbf{E}|| = ||\mathbf{E}' + \mathbf{F}|| \geq ||\mathbf{F}||$. If $E'_h = 0$, from (a), we have $\mathbf{E}' = 0$, so $\mathbf{E} = \mathbf{F}$.

Consider the case $F'_h > 0$, and we know $C_h > 0$, $||\mathbf{F}|| \geq ||\mathbf{F}'||$ from (c), (d). Applying Lemma B.1.1 gives $||\mathbf{F}'|| \leq ||\mathbf{F}|| = ||\mathbf{F}' + \mathbf{C}|| \Rightarrow ||\mathbf{F}'|| \leq ||\mathbf{C}|| \Rightarrow ||\mathbf{F}|| = ||\mathbf{F}' + \mathbf{C}|| \leq ||\mathbf{C}||$. If $E'_h > 0$, combining with $||\mathbf{E}'|| \geq ||\mathbf{F}||$ from (e), we get $||\mathbf{E}|| = ||\mathbf{C} + \mathbf{E}'|| \geq \min\{||\mathbf{C}||, ||\mathbf{E}'||\} \geq ||\mathbf{F}||$. If $E'_h = 0 \Rightarrow \mathbf{E}' = 0$ from (a), we have $||\mathbf{E}|| = ||\mathbf{C}|| \geq ||\mathbf{F}||$.

□

B.2 Proof Of Theorem 4.1.6

Let $\mathbf{F} = \mathbf{E}_r$. Our goal is to construct $\mathbf{C}$, $\mathbf{E}'$ and $\mathbf{F}'$ so that Lemma B.1.3 can be applied. In fact, we can define $\mathbf{C} = \sum_{g<g_r \wedge I_g=1} \mathbf{e}_g + \delta_r I_{g_r} \mathbf{e}_{g_r}$, $\mathbf{E}' = \sum_{g>g_r \wedge I_g=1} \mathbf{e}_g + (1-\delta_r) I_{g_r} \mathbf{e}_{g_r}$, and, $\mathbf{F}' = \sum_{g<g_r \wedge I_g=0} \mathbf{e}_g + \delta_r(1 - I_{g_r})\mathbf{e}_{g_r}$. Then we have $\mathbf{E} = \mathbf{C} + \mathbf{E}'$, and $\mathbf{F} = \mathbf{C} + \mathbf{F}'$.

Given that $\mathbf{E}'$ and $\mathbf{F}'$ are summations of a sequence of vectors, where each vector's horizontal component is positive and the coefficient in front of each vector is non-negative, we have conditions (a) and (b) from Lemma B.1.3 are satisfied. The condition (c) follows from the support constraint: $E_h \geq s_0$ and $F_h = E_{r,h} = s_0$. To prove conditions (d) and (e), we can assume $E'_h > 0$ and $F'_h > 0$. From how $\mathbf{E}'$ and $\mathbf{F}'$

are defined, we can apply Lemma B.1.2 and get $||\mathbf{E}'|| \geq ||\mathbf{e}_{g_r}||$, $||\mathbf{F}'|| \leq ||\mathbf{e}_{g_r}||$, and $||\mathbf{F}|| \leq ||\mathbf{e}_{g_r}||$. Combining them together, we get (d) and (e).

B.3 Proof Of Theorem 4.1.5

(1) Consider the case $E^s_{g_1,h} \geq s_0$, so that $\mathbf{E}_c = \mathbf{E}^s_{g_1}$. Let $\mathbf{F} = \mathbf{E}^s_{g_1}$. Our goal is to construct $\mathbf{C}$, $\mathbf{E}'$ and $\mathbf{F}'$ so that Lemma B.1.4 can be applied. Define $\mathbf{C} = \mathbf{e}_{g_0} + \sum_{g \neq g_0 \wedge g \leq g_1 \wedge I_g = 1} \mathbf{e}_g$, $\mathbf{E}' = \sum_{g \neq g_0 \wedge g > g_1 \wedge I_g = 1} \mathbf{e}_g$, and, $\mathbf{F}' = \sum_{g \neq g_0 \wedge g \leq g_1 \wedge I_g = 0} \mathbf{e}_g$. We have $\mathbf{E} = \mathbf{C} + \mathbf{E}'$, and $\mathbf{F} = \mathbf{C} + \mathbf{F}'$. It is clear that conditions (a), (b), and (c) from Lemma B.1.4 are satisfied.

To prove condition (d), we assume $F'_h > 0 \Rightarrow g_1 > 0$. From the definition of $\mathbf{F}'$ and Lemma B.1.2, we have $||\mathbf{F}'|| \leq ||\mathbf{e}_{g_1}||$. From the definition of g_1, we have $||\mathbf{F}|| \leq ||\mathbf{F} - \mathbf{e}_{g_1}||$. Let $\mathbf{F}'' = \mathbf{F} - \mathbf{e}_{g_1}$, then $||\mathbf{F}|| = ||\mathbf{F}'' + \mathbf{e}_{g_1}|| \leq ||\mathbf{F}''||$. From Lemma B.1.1, we get $||\mathbf{e}_{g_1}|| \leq ||\mathbf{F}''|| \Rightarrow ||\mathbf{e}_{g_1}|| \leq ||\mathbf{F}'' + \mathbf{e}_{g_1}|| = ||\mathbf{F}||$. Combined with $||\mathbf{F}'|| \leq ||\mathbf{e}_{g_1}||$, we get $||\mathbf{F}'|| \leq ||\mathbf{F}||$.

To prove condition (e), we assume $E'_h > 0$. In this case, g_1 cannot be the last index g' of sequence $\mathbf{E}^s_{g'}$. Let $g_2 \neq g_0$ be its next index. From the definition of g_1, we have $||\mathbf{F}|| \leq ||\mathbf{F} + \mathbf{e}_{g_2}||$. From Lemma B.1.1, we get $||\mathbf{F}|| \leq ||\mathbf{e}_{g_2}||$. From the definition of $\mathbf{E}'$ and Lemma B.1.2, we get $||\mathbf{e}_{g_2}|| \leq ||\mathbf{E}'||$. Combine them together, we get $||\mathbf{E}'|| \geq ||\mathbf{F}||$.

(2) Consider the case $E^s_{g_1,h} < s_0$. Let $\mathbf{F} = \mathbf{E}_c$. Our goal is to construct $\mathbf{C}$, $\mathbf{E}'$ and $\mathbf{F}'$ so that Lemma B.1.3 can be applied. Define $\mathbf{C} = \mathbf{e}_{g_0} + \sum_{g \neq g_0 \wedge g < g_c \wedge I_g = 1} \mathbf{e}_g + I_{g_c} \delta_c \mathbf{e}_{g_c}$, $\mathbf{E}' = \sum_{g \neq g_0 \wedge g > g_c \wedge I_g = 1} \mathbf{e}_g + I_{g_c}(1 - \delta_c)\mathbf{e}_{g_c}$, and, $\mathbf{F}' = \sum_{g \neq g_0 \wedge g < g_c \wedge I_g = 0} \mathbf{e}_g + (1 - I_{g_c})\delta_c \mathbf{e}_{g_c}$. We have $\mathbf{E} = \mathbf{C} + \mathbf{E}'$, and $\mathbf{F} = \mathbf{C} + \mathbf{F}'$. It is clear that conditions (a), (b), and (c) from Lemma B.1.3 are satisfied.

To prove condition (d), assume $E'_h \geq F'_h > 0$. From the definition of $\mathbf{E}'$ and Lemma B.1.2, we have $||\mathbf{E}'|| \geq ||\mathbf{e}_{g_c}||$. From the definition of $\mathbf{F}'$ and Lemma B.1.2, we have $||\mathbf{F}'|| \leq ||\mathbf{e}_{g_c}||$. Combining them together gives $||\mathbf{E}'|| \geq ||\mathbf{F}'||$.

To prove condition (e), assume $E'_h > 0$. From the definition of $\mathbf{F}$, we can write $\mathbf{F} = \mathbf{E}^s_{g_1} + \mathbf{e}_{g_2} + \cdots + \delta_c \mathbf{e}_{g_c}$, where $g_2 \neq g_0$ is g_1's next index. In part (1), we have proved $||\mathbf{E}^s_{g_1}|| \leq ||\mathbf{e}_{g_2}||$ (Note that in part (1), $\mathbf{E}^s_{g_1}$ is denoted as $\mathbf{F}$ and we proved $||\mathbf{F}|| \leq ||\mathbf{e}_{g_2}||$). Combining this with Lemma B.1.2, we get $||\mathbf{F}|| \leq ||\mathbf{e}_{g_c}||$. We have proved $||\mathbf{E}'|| \geq ||\mathbf{e}_{g_c}||$ above. So we get $||\mathbf{E}'|| \geq ||\mathbf{F}||$.

B.4 Proof Of Theorem 4.1.7

If δ_c is not involved in $\mathbf{E}_c$ (i.e., when $E^s_{g_1} \geq s_0$), or if $\delta_c = 1$, we can simply apply Theorem 4.1.6 by setting $\mathbf{E} \leftarrow \mathbf{E}_c$ and get what we need to prove: $||\mathbf{E}_r|| \leq ||\mathbf{E}_c||$. So we only need to worry about the case when $E^s_{g_1} < s_0$ and $\delta_c < 1$ in the following. And, in this case, $E_{r,h} = E_{c,h} = s_0$.

We only need to consider the case when $\mathbf{E}_r$ and $\mathbf{E}_c$ are different. They both sum up a sequence of vectors until the support constraint is satisfied. The difference comes from that $\mathbf{e}_{g_0}$ is guaranteed to be in $\mathbf{E}_c$. If $g_0 < g_r$, $\mathbf{e}_{g_0}$ is also in $\mathbf{E}_r$. So in this case $\mathbf{E}_r$ and $\mathbf{E}_c$ are the same.

Now, we consider $g_0 \geq g_r$ only. In this case $g_c \leq g_r$, and when $g_c = g_r$, we have $\delta_c < \delta_r$. Note that due to $g_c \neq g_0$, g_c and g_0 can not be both g_r. We introduce $\mathbf{C} = \sum_{g<g_c} \mathbf{f}_g + \delta_c \mathbf{f}_{g_c} + I_{g_0=g_r}\delta_r \mathbf{f}_{g_r}$, where $I_{g_0=g_r}$ ($I_{g_c=g_r}$) takes one if $g_0 = g_r$ ($g_c = g_r$) and zero otherwise. Then let $\mathbf{F}' = \sum_{g_c<g<g_r} \mathbf{f}_g + (1-\delta_c)(1-I_{g_c=g_r})\mathbf{f}_{g_c} + (\delta_r - \delta_c I_{g_c=g_r})(1-I_{g_0=g_r})\mathbf{f}_{g_r}$, and $\mathbf{E}' = (1 - I_{g_0=g_r}\delta_r)\mathbf{f}_{g_0}$. We have $\mathbf{E}_r = \mathbf{C} + \mathbf{F}'$ and $\mathbf{E}_c = \mathbf{C} + \mathbf{E}'$.

From $E_{r,h} = E_{c,h}$, we get $E'_h = F'_h$. If $E'_h = F'_h = 0$, we get $\mathbf{E}' = \mathbf{F}' = 0$, and $\mathbf{E}_c = \mathbf{E}_r = \mathbf{C}$. If $E'_h = F'_h > 0$, we have $||\mathbf{E}'|| = ||\mathbf{f}_{g_0}|| \geq ||\mathbf{F}'|| \Rightarrow E'_v \geq F'_v \Rightarrow E_{c,v} \geq E_{r,v} \Rightarrow ||\mathbf{E}_c|| \geq ||\mathbf{E}_r||$.

B.5 Proof Of Theorem 5.4.4

Theorem 5.4.4 is very similar to the Theorem 4.1.6. The only difference is that now the minimum support constraint is applied to the vertical component, i.e., $E_v \geq s_0$ and $E_{b,v} = s_0$, so that $E_v \geq E_{b,v}$. Note that if $E_h < E_{b,h}$, we will have $E_v/E_h \geq E_{b,v}/E_{b,h} \Rightarrow ||\mathbf{E}|| \geq ||\mathbf{E}_b||$. So we only need to consider the case $E_h \geq E_{b,h}$. With this inequality applied to the horizontal component, the proof of Theorem 4.1.6 can be applied directly here for the rest of the proof.