Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 Keller Hall
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 13-011

A Novel Regression Model Combining Instance Based Rule Mining With EM Algorithm

Zhonghua Jiang and George Karypis

April 01, 2013

A Novel Regression Model Combining Instance Based Rule Mining With EM Algorithm

Zhonghua Jiang and George Karypis

March 30, 2013

Abstract

In recent years, there have been increasing efforts to apply association rule mining to build Associative Classification (AC) models. However, the similar area that applies association rule mining to build Associative Regression (AR) models has not been well explored. In this work, we fill this gap by presenting a novel regression model based on association rules called AREM. AREM derives a set of regression rules by: (i) applying an instance based approach to mine itemsets which form the regression rules' left hand side, and (ii) developing a probabilistic model which determines, for each mined itemset, the corresponding rule's right hand side and the importance weight. To address the computational bottleneck of the traditional two-step approach for itemset mining, AREM utilizes an Instance-Based Itemset Miner (IBIMiner) algorithm that directly discovers the final set of itemsets. IBIMiner incorporates various methods to bound the quality of any future extensions of the itemset under consideration. These bounds are then used to prune the search space. In addition, AREM treats the regression rules' right hand side and importance weights as parameters of a probabilistic model, which are then learned in the expectation and maximization (EM) framework. The extensive experimental evaluation shows that our bounding strategies allow IBIMiner to considerably reduce the runtime and the EM optimization can improve the predictive performance dramatically. We also show that our model can perform better than some of the state of the art regression models.

Keywords: association rule, regression rule, associative regression, EM algorithm, instance based approach, quality upper bound pruning

1 Introduction

In recent years, there have been increasing efforts to apply association rule mining to build classification models [1–7], which have resulted in the area of Associative Classification (AC) modeling. Several studies [2, 4, 5] have provided empirical evidence that AC classifiers can outperform tree-based [8] and rule-induction based models [9, 10]. The good performance of the AC 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.

Regression is a data mining task that is applicable to a wide-range of application domains. However, despite the success of association rule mining for classification, it has not been extensively applied to develop models for regression. We are only aware of the Regression Based on Association (RBA) method developed by Ozgur *et al.* [11], which uses association rule mining to derive a set of regression rules. Since regression models need to predict a continuous value, whereas the classification models need to predict a categorical value, the methods developed for AC modeling are in general not applicable for solving regression problems.

Motivated by the success of AC modeling, we study the problem of applying association rule mining to build an Associative Regression (AR) model. We believe this is an important problem for the following two reasons. First, an AR model is built upon a set of regression rules, which, in many cases, can be easily interpreted by domain experts and thus provide valuable insights. Second, the good performance of the well studied AC classifiers leads us to believe that the AR model may potentially perform better than the tree-based [12, 13] and rule-induction based regression models.

In general, an AR model consists of a set of regression rules. A regression rule has a left hand side (LHS), which is a set of features (or an itemset), and a right hand side (RHS), which is a numeric value. A 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 the RHSs of multiple rules during prediction. In such cases, each rule is associated with an importance weight, and the prediction is the average of the RHSs of a set of rules, weighted by their importance weights. In this paper, we present an AR modeling framework called AREM [1], which determines a set of regression rules by first mining a set of itemsets that form the LHS of the rules, and then assigning to each mined itemset, its corresponding RHS value and importance weight.

Many AC/AR models [1–3, 11] employ a two-step approach for itemset mining, where the first step applies a standard algorithm (e.g., Apriori [15] or FP-growth [16]) to discover the complete set of frequent itemsets, and the second step applies various selection strategies (e.g., database sequential coverage or instance based approach) to select a subset of high quality itemsets. However, because the final set of selected itemsets is often orders of magnitude smaller than the initial set, this two-step approach ends up performing a large amount of unnecessary computations. AREM employs an instance-based approach to select the itemsets, and it reduces the number of itemsets that need to be mined and then discarded by using an efficient Instance-Based Itemset Miner (IBIMiner) to directly discover the final set of itemsets [6, 7, 17, 18]. Developing efficient itemset mining algorithms for AREM is challenging because the quality metric that is used to select the final set of itemsets is not downward closed. We present novel methods for finding the upper bound of the quality of any future extensions of the current itemset that are used to prune the search space. Our experimental evaluation shows that IBIMiner can be several orders of magnitude faster than the baseline approach and that it scales linearly with the size of the database.

AREM develops a probabilistic model that captures the interactions of the various itemsets. In this model, the rules' RHS and importance weights are treated as the parameters, which are learned in the EM framework. This is a major deviation after existing approaches that do not take into account complex interactions among itemsets [11]. Our experimental evaluation shows that the EM optimization approach AREM takes can greatly improve the predictive performance on almost all the datasets. In addition, we show that AREM outperforms several state of the art regression models including RBA [11], Boosted Regression Trees [13], SVR [19], CART [12] and Cubist on many data sets, with the Mean Square Error (MSE) being used as the performance metric.

The remainder of this paper is organized as follows. Section 2 introduces some notations and definitions. Section 3 presents the related work in this area. Section 4 presents the two modeling components of AREM: instance based itemset mining and the probabilistic model for assigning rule parameters. In Section 5, we present the IBIMiner algorithm and focus on the pruning methods to reduce the search space. In Section 6, we derive the EM framework that is used to learn model parameters. Section 7 describes the evaluation strategy we employ to test the AREM framework. The experimental results are presented in Section 8. Finally, Section 9 concludes the paper.

2 Notations And Definitions

The methods developed in this work apply to datasets whose instances are described by a set of features that are present. Such datasets occur naturally in mar-

---

[1] A short version of this work has been accepted in [14] for publication.

ket 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 [20].

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 numeric target variable. 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 $s^r_x = s_x/|\mathcal{D}_0|$. Itemset x is frequent if $s_x \geq s_0$ ($s^r_x \geq s^r_0$), where $s_0 > 0$ ($s^r_0 > 0$) is the user specified parameter. For itemset x, we define its mean (μ_x) and standard deviation (σ_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 $r_x : x \xrightarrow{w_x} \alpha_x$, where the rule's LHS x is an itemset,the rule's RHS α_x is the target value predicted by this rule, and w_x is a positive value which is used as the importance weight when combining multiple rules together for making predictions.

3 Related Work

To the best of our knowledge, the RBA [11] model is the only previous work on associative regression. RBA discovers the set of itemsets by following the two-step approach. It first applies the Apriori algorithm [15] to mine the set of frequent itemsets. Then it sorts all frequent itemsets in increasing variance (i.e., σ^2_x) order and applies the database sequential covering approach to select a subset of itemsets. For each itemset under consideration, RBA checks whether the itemset is contained in some instances in the current database, and whether it is able to reduce the training error. If both conditions are satisfied, the itemset is selected and instances that have been covered K times are removed from the database. For each mined itemset x, RBA computes the rule's RHS as the mean of x. Three weighting schemes for w_x are developed in RBA: (1) *equal*, where rules are equally weighted, (2) *supp*, where the rule r_x is weighted by the support of x, and (3) *inv-var*, where the rule's weight is inversely proportional to the variance σ^2_x.

Associative Classification (AC) [21] is an area that applies similar techniques, but focus on the classification task. Among the many methods developed for AC modeling [2, 4, 5, 7], Harmony [6] employs a similar rule selection strategy to AREM: it mines the highest confidence rules for each instance and combines them to the final rule set. To address the limitation of the two-step approach for itemset mining, several works [6, 7, 17, 18] focus on developing methods for directly mining the itemsets used for building AC models. DDPMine [7] discovers the pattern with the maximum information gain iteratively. FARMER [17] finds interesting rule groups for the microarray databases. In [18], the authors propose to discover top-k covering rule groups for each row of a gene expression dataset. All these approaches incorporate various pruning methods into the itemset mining framework to reduce the unnecessary computations. However, their methods focus on classification rules and cannot be applied for the regression rule discovery problem.

AR and AC models are descriptive in that they can be easily interpreted by end users. Tree based and rule induction based models are another two groups of descriptive models. The classification and regression tree (CART) [12] partitions the input space into smaller, rectangular regions, and assigns the average of the target variables as the predicted value to each region. Cubist is a rule based algorithm that fits a linear regression model to each of the regions. Boosting [13] 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 [22].

4 AREM: An Associative Regression Model

Our proposed AREM model consists of a set of regression rules $\mathcal{R} = \{r_x : x \xrightarrow{w_x} \alpha_x\}$. 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., $(x_i \xrightarrow{w_{x_i}} \alpha_{x_i}) \in \mathcal{R}_\tau$ 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 $\mathcal{R}^k_\tau$, is then used to predict the target variable using the formula

$$\hat{y} = \frac{\sum_{r_{x_i} \in \mathcal{R}^k_\tau} w_{x_i} \alpha_{x_i}}{\sum_{r_{x_i} \in \mathcal{R}^k_\tau} w_{x_i}}, \tag{1}$$

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 of the LHS of the rules in $\mathcal{R}$, we insert the empty rule $\emptyset \xrightarrow{w_\emptyset} \alpha_\emptyset$ into $\mathcal{R}$. The RHS and weight of this empty rule 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) the discovery of 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) the assignment to each itemset $x \in \mathcal{X}$ the two numeric values α_x and w_x to form the rule $x \xrightarrow{w_x} \alpha_x$. We explain these two components in the remaining of this section.

4.1 Instance Based Approach For Itemset Discovery

Our approach for itemset discovery is to require the itemsets in $\mathcal{X}$ to satisfy the following four properties. First, to control the model complexity and prevent overfitting, we want to impose the minimum frequency requirement so that the number of instances containing the itemset $x \in \mathcal{X}$ is larger than some threshold value 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 at least one and preferably more itemsets from $\mathcal{X}$ that are contained in that instance. Third, we prefer itemsets that are likely to achieve better predictive performances when used as the LHS of the set of rules in $\mathcal{R}$. Fourth, we prefer to have a set of non-redundant itemsets, where an itemset x is considered to be redundant if one of its proper subset $x' \subset x$ has the same quality as x. The rationale for preferring x' over x is that x' tends to generalize better on the test dataset since any instances that contain x will also contain x'.

We utilize the instance based approach for itemset discovery to address these requirements. 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 by $\mathcal{F}$ 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 that are contained in that instance, i.e., $x \in \mathcal{F}_i$ if $x \subseteq \tau_i$. Our approach identifies for each instance i, K itemsets $\mathcal{X}_i$ from $\mathcal{F}_i$ (i.e., $\mathcal{X}_i \subseteq \mathcal{F}_i$) and then derives the final set of itemsets by taking the union over all training instances, i.e., $\mathcal{X} = \cup_i \mathcal{X}_i$. In this way, the minimum frequency requirement is satisfied and there are at least K itemsets covering each instance (assuming that each instance supports at least K frequent itemsets). To satisfy the third requirement, we need to make sure that the itemsets in $\mathcal{X}_i$ are better than the rest of the itemsets in $\mathcal{F}_i$. For this, we need a way to evaluate the quality of an itemset. We do this by using an instance based quality metric, denoted by $Q(x, y_i)$, which measures the quality of an itemset $x \in \mathcal{F}_i$ from instance i's perspective. We assume the quality metric is of the form $Q(x, y_i)$ so that its dependency on instance i is captured by the target variable y_i. To satisfy the non-redundant requirement, we remove itemsets x such that there exists $x' \subset x$ satisfying $Q(x, y_i) = Q(x', y_i)$ for $\forall y_i$.

4.2 Itemset Quality Function

We derive three quality functions for the itemset x and numeric 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

$$Q^\sigma(x, y_i) = \frac{1}{\sigma_x}. \tag{2}$$

Note that we use the inverse of the standard deviation σ_x, thus preferring itemsets with higher quality. However, the function Q^σ has the limitation that it does not capture the instance's target variable y_i, which can lead to a sub-optimal set of itemsets. We propose two additional quality functions to address this limitation.

We derive the second quality function by considering an itemset $x \in \mathcal{F}_i$ to be of high quality if the target variables of the instances containing x are close to that of instance i. We measure the closeness between the target variables of 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):

$$MSE(x, y_i) = \frac{1}{s_x} \sum_{x \subseteq \tau_j} (y_j - y_i)^2 = (\mu_x - y_i)^2 + \sigma_x^2. \tag{3}$$

Thus, we propose the following quality function:

$$Q^m(x, y_i) = \frac{1}{\sqrt{MSE(x, y_i)}} = \frac{1}{\sqrt{(\mu_x - y_i)^2 + \sigma_x^2}}. \tag{4}$$

We derive the third quality function by following a 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 $\mathcal{N}(y_i|\mu_x, \sigma_x^2)$. We propose the quality function based on $\mathcal{N}(y_i|\mu_x, \sigma_x^2)$ as

$$Q^n(x, y_i) = \frac{1}{\sigma_x} e^{-\frac{(\mu_x - y_i)^2}{2\sigma_x^2}}, \tag{5}$$

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 to Section 5.

4.3 The Probabilistic Model For Determining α_x And w_x

In the rest of this section, 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 $(x \xrightarrow{w_x} \alpha_x) \in \mathcal{R}$. To determine the rule's RHS, a straightforward approach is to use the itemset's mean value μ_x. However, a closer investigation suggests that this approach has the drawback of ignoring the interactions among the different itemsets. 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 determined 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 [11] 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 $x \xrightarrow{w_x} \alpha_x$ and standard deviation is β_x. That is,

$$P[y|x] = \mathcal{N}(y|\alpha_x, \beta_x^2). \tag{6}$$

Next, we specify $P[x|\tau]$ by considering how AREM makes predictions. In order to simplify the model, 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 1 now becomes

$$\hat{y} = \frac{\sum_{r_{x_i} \in \mathcal{R}_\tau} w_{x_i} \alpha_{x_i}}{\sum_{r_{x_i} \in \mathcal{R}_\tau} w_{x_i}} = \sum_x \alpha_x \frac{I_{x \subseteq \tau} w_x}{\sum_{x' \subseteq \tau} w_{x'}}, \tag{7}$$

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,

$$\hat{y} = E[y|\tau] = \sum_x E[y|\tau, x] P[x|\tau]. \tag{8}$$

From Equation 6, we get $E[y|\tau, x] = \alpha_x$. To specify $P[x|\tau]$, we compare Equation 7 with 8, and get

$$P[x|\tau] = \frac{I_{x \subseteq \tau} w_x}{\sum_{x' \subseteq \tau} w_{x'}}. \tag{9}$$

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 9. In the second step, the target variable y is generated by x with probability $P[y|x]$ given by Equation 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 6.

Algorithm 1 Instance Based Itemset Miner

```
Input: quality function $Q$, prefix itemset $x_p$, conditional database
    $\mathcal{D}_{x_p}$, instance index set $\mathcal{T}_{x_p}$, instance $i$'s current set of itemsets
    $\mathcal{X}_i$ for $\forall i$
Output: updated set $\mathcal{X}_i$ for $\forall i$
prune infrequent items in $\mathcal{D}_{x_p}$
prune support equivalence items in $\mathcal{D}_{x_p}$
compute $\mu_{x_p}$ and $\sigma_{x_p}$
for each instance $i$ in $\mathcal{T}_{x_p}$ do
    $Q(x_p, y_i) \leftarrow$ quality of itemset $x_p$ for instance $i$
    $Q_i^{thr} \leftarrow$ quality threshold for instance $i$
    if $Q(x_p, y_i) \geq Q_i^{thr}$ then
        update $\mathcal{X}_i$ by adding the new itemset $x_p$
    else
        $B_i^{x_p} \leftarrow$ 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 discov-
        ered from $\mathcal{D}_{x_p}$ such that $x_p \cup x \subseteq \tau_i$.
        if $B_i^{x_p} < Q_i^{thr}$ then
            remove $i$ from $\mathcal{T}_{x_p}$
        end if
    end if
end for
if $|\mathcal{T}_{x_p}| > 0$ then
    while there exists more items in $\mathcal{D}_{x_p}$ do
        $m \leftarrow$ next item in $\mathcal{D}_{x_p}$, next prefix itemset $x'_p \leftarrow x_p \cup$
        $\{m\}$
        $\mathcal{T}_{x'_p} \leftarrow$ indices in $\mathcal{T}_{x_p}$ whose instances in $\mathcal{D}_{x_p}$ contain $m$
        if $|\mathcal{T}_{x'_p}| > 0$ then
            $\mathcal{D}_{x'_p} \leftarrow$ instances in $\mathcal{D}_{x_p}$ which contain $m$
            prune item $m$ from $\mathcal{D}_{x'_p}$
            run Instance Based Itemset Miner with $Q$, $x'_p$, $\mathcal{D}_{x'_p}$,
            $\mathcal{T}_{x'_p}$ and $\mathcal{X}_i \forall i$
        end if
        prune item $m$ from $\mathcal{D}_{x_p}$
    end while
end if
```

5 Instance Based Itemset Miner

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 [15] or FP-growth [16]) 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.

The IBIMiner algorithm follows the depth-first approach and grows the current itemset by adding to it one item at a time [23]. 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 itemset generation early. This can dramatically reduce the total number of itemsets that need to be considered and lead to considerable reductions in runtime.

The IBIMiner algorithm (Algorithm 1) operates as follows. Let x_p be the current prefix itemset, which is the set of items that have been searched so far. Let $\mathcal{D}_{x_p}$ be the conditional database, which consists of the set of instances from $\mathcal{D}_0$ that contain x_p. Let $\mathcal{T}_{x_p}$ be the set of indices of instances in $\mathcal{D}_{x_p}$ whose top K itemsets $\mathcal{X}_i$ need to be updated. 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 for all instances i in $\mathcal{T}_{x_p}$ their corresponding $\mathcal{X}_i$ sets. We initialize x_p to be the empty set, $\mathcal{D}_{x_p}$ to be $\mathcal{D}_0$, $\mathcal{T}_{x_p}$ to be the set of indices of all training instances, and $\forall i$, $\mathcal{X}_i$ to be the empty set. Given this initialization, when IBIMiner completes, it will have identified the required $\mathcal{X}_i$ sets for all the instances.

For the current itemset x_p, whose μ_{x_p} and σ_{x_p} are computed in line 3, the algorithm iterates through each instance i in $\mathcal{T}_{x_p}$ (line 4) and attempts to update $\mathcal{X}_i$ by adding x_p to it (lines 5-8). For instance i, we define the quality threshold, denoted by Q_i^{thr}, as the lowest quality of the itemsets in $\mathcal{X}_i$ if $|\mathcal{X}_i| \geq K$ and negative infinity otherwise. IBIMiner computes the quality $Q(x_p, y_i)$ of x_p (line 5) and then compares it against the quality threshold Q_i^{thr}. 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 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 $\mathcal{T}_{x'_p}$ (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 $\mathcal{T}_{x'_p}$ 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.

5.1 Infrequent Items Pruning

Infrequent items pruning is based on the anti-monotone property of the support measure [15]. 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 $\mathcal{D}_{x_p}$.

5.2 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 $x_p \cup x \cup \{m\}$. According to the non-redundant requirement, we remove all these itemsets $x_p \cup x \cup \{m\}$, which is equivalent to removing the item m from the database.

5.3 Quality Upper Bound Pruning

In the instance based itemset mining approach, the itemset $x_p \cup x$ to be mined from $\mathcal{D}_{x_p}$ has the potential of being included into $\mathcal{X}_i$ only if it satisfies the following three constraints: (i) $s_{x_p \cup x} \geq s_0$ (i.e., itemset $x_p \cup x$ is frequent), (ii) $x_p \cup x \subseteq \tau_i$ (i.e., $x_p \cup x \in \mathcal{F}_i$), and, (iii) $Q(x_p \cup x, y_i) \geq Q_i^{thr}$, where Q_i^{thr} is instance i's current quality threshold. Our strategy is to find the upper bound of $Q(x_p \cup x, y_i)$ for all $x_p \cup x$ satisfying the first two constraints. Denote this upper bound as $B_i^{x_p}$. Then if we have $B_i^{x_p} < Q_i^{thr}$, we can be sure that $Q(x_p \cup x, y_i) \leq B_i^{x_p} < Q_i^{thr}$ so that the third constraint will be violated for all itemsets that we need to consider. If this is true for instance i, it means that we do not need to search for instance i's best itemsets any more, and the conditional database $\mathcal{D}_{x_p}$ can be pruned for instance i. In that case, we remove i from $\mathcal{T}_{x_p}$ (line 12) so that the instance i will not be considered in any subsequent calls to the IBIMiner (line 23). Our algorithm terminates early (line 16) if all the instances in $\mathcal{T}_{x_p}$ are removed due to the quality upper bound pruning strategy.

Finding the upper bound of $Q(x_p \cup x, y_i)$ is a challenging problem because the three quality functions we developed are not downward closed. However, we are able to develop bounding approaches. Due to the rather lengthy mathematical derivations of the bounds, the complete details are provided in the appendix, and in the rest of this section, we briefly describe some of these results. We develop three bounding functions as follows: (i) $\mathcal{B}^{sig}$ for Q^{σ}, (ii) $\mathcal{B}^{mse}$ for both Q^m and Q^n, and (iii) $\mathcal{B}^{add}$ for both Q^m and Q^n. The bounding function $\mathcal{B}^{mse}$ is tighter for Q^m than for Q^n since Q^m is itself an upper bound of Q^n. The bounding function $\mathcal{B}^{add}$ is an approximation to $\mathcal{B}^{mse}$ because the computational cost of $\mathcal{B}^{mse}$ is high.

6 An EM Framework For Learning Rule Parameters

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 4.3. The maximum likelihood estimation of $\boldsymbol{\theta}$ given the training data set is to maximize

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \sum_i \log\left(P[y_i|\tau_i, \boldsymbol{\theta}]\right) \\ &= \sum_i \log\left(\sum_{x_i} P[y_i, x_i|\tau_i, \boldsymbol{\theta}]\right), \end{aligned} \tag{10}$$

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. EM algorithm is the standard approach 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 $\boldsymbol{\theta}_t$. This is accomplished by maximizing the function

$$\begin{aligned} &\mathcal{Q}(\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t) \\ (11)= \quad &\sum_i \sum_{x_i} P[x_i|\tau_i, y_i, \boldsymbol{\theta}_t] \log(P[y_i, x_i|\tau_i, \boldsymbol{\theta}_{t+1}]). \end{aligned}$$

This optimization problem is much easier than the original one for Equation 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 $\mathcal{Q}(\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t)$ 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

$$\begin{aligned} \pi_{i,x_i,t} &= P[x_i|\tau_i, y_i, \boldsymbol{\theta}_t] \\ &= \frac{P[y_i|\tau_i, x_i, \boldsymbol{\theta}_t] P[x_i|\tau_i, \boldsymbol{\theta}_t]}{P[y_i|\tau_i, \boldsymbol{\theta}_t]} \\ &\propto P[y_i|\tau_i, x_i, \boldsymbol{\theta}_t] P[x_i|\tau_i, \boldsymbol{\theta}_t]. \end{aligned}$$

According to Equations 9 and 6, we have

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

Combining these two Equations, we get

$$\pi_{i,x_i,t} = \frac{\mathcal{N}(y_i|\alpha_{x_i,t}, \beta^2_{x_i,t})w_{x_i,t}I_{x_i\subseteq\tau_i}}{\sum_{x'\subseteq\tau_i}\mathcal{N}(y_i|\alpha_{x',t}, \beta^2_{x',t})w_{x',t}}. \tag{12}$$

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

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

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 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:

$$\alpha_{x,t+1} = \frac{\sum_{i:x\subseteq\tau_i}\pi_{i,x,t}y_i}{\sum_{i:x\subseteq\tau_i}\pi_{i,x,t}}, \tag{13}$$

and,

$$\beta^2_{x,t+1} = \frac{\sum_{i:x\subseteq\tau_i}\pi_{i,x,t}(y_i - \alpha_{x,t+1})^2}{\sum_{i:x\subseteq\tau_i}\pi_{i,x,t}}. \tag{14}$$

In Equations 13 and 14, 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

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

By plugging Equation 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 analytic 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

$$\frac{w_{x,t+1}}{w_{x,t}} = \frac{\sum_{i:x\subseteq\tau_i}\pi_{i,x,t}}{\sum_{i:x\subseteq\tau_i}\frac{w_{x,t}}{\sum_{x'\subseteq\tau_i}w_{x',t}}}. \tag{15}$$

From Equations 13, 14 and 15, 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 $w_{x,M}$.

7 Experimental Design

7.1 Data Sets

Table 1 summarizes the set of datasets that we use to evaluate various models.

Text Reviews Data The first six data sets are randomly sampled from user reviews downloaded from three websites: BestBuy [2], CitySearch [3], and Yelp [4]. 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 [24] between words in each sentence are extracted. Each

---

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

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

[4] `http://www.yelp.com/academic_dataset`

Table 1: Data Set Summary

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

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 [5] 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 [6]. 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 [7]. 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.

Among the set of datasets in Table 1, we select four datasets (BestBuy.wdep, CitySearch.wdep, Yelp.wdep and MovieLens) to evaluate the IBIMiner algorithm. The rest of the datasets are *easy* in that the running times are very short and there is no need for the advanced strategies.

7.2 AREM Model

For the experimental study, we denote the AREM model as $AREM_k$, where k is the number of rules used for prediction. Given a quality function Q, which can be one of Q^σ, Q^m and Q^n, the IBIMiner takes two input parameters: (i) K, which is the number of top itemsets mined for each instance, and (ii) s_0^r, which is the minimum relative support of the 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. During the EM optimization,

---

[5] `http://www.grouplens.org/node/12`

[6] `http://stat-computing.org/dataexpo/2009`

[7] `http://lib.stat.cmu.edu/datasets`

AREM 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.

7.3 Comparison Models

We compare the performance of $AREM_k$ against the following models:

SVR We use libsvm [25] for *SVR* [19], 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) [12] and the Boosted Regression Tree [13] where CART of fixed size is acting as the weak learners. So, $CART_k$ stands for CART being boosted k times [26]. 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 [8] 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 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 [11]. 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*.

7.4 Evaluation

Performance Metrics & Model Selection 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 1. Finally, we reported the average MSE on all testing trials.

Model Comparison 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 $\mu_{m_1-m_2} = \mu_2 - \mu_1$, $\sigma_{m_1-m_2} = \sqrt{\sigma_1^2/n_1 + \sigma_2^2/n_2}$, and the Normalized Mean Difference (NMDIFF) between two models as $\mu_{m_1-m_2}/\sigma_{m_1-m_2}$. The NMDIFF is the quantity used in statistical testing [27] for the comparison of two population means. Under the null hypothesis that two population means are the same, NMDIFF 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.

8 Experimental Results

To evaluate our AREM model, we conduct three sets of experiments. First, we evaluate the effectiveness of the IBIMiner algorithm. In particular, we focus on the effects of different quality upper bounding strategies on reducing running times. We also study how IBIMiner can scale to large databases. Second, we evaluate how the predictive performances are affected by using different quality functions. We also study the effectiveness of the EM optimization by comparing it to the straightforward approach for determining rule's parameters. Third, we compare the AREM model against the other state of the art regression models.

---

[8] `http://www.rulequest.com`

Table 2: Running Time Reduction For Different Quality Bounds

dataset	K	s_0^r	$Q = Q^\sigma$		$Q = Q^m$			$Q = Q^n$		
			time[a]	$\mathcal{B}^{sig}$	time[a]	$\mathcal{B}^{add}$	$\mathcal{B}^{mse}$	time[a]	$\mathcal{B}^{add}$	$\mathcal{B}^{mse}$
BestBuy .wdep	1	2.0%	2.41	0.10	1.69	**0.15**	0.23	1.93	**0.29**	0.42
		1.0%	9.02	0.09	6.41	0.14	0.14	6.96	**0.35**	0.36
	5	2.0%	2.51	0.11	1.68	**0.17**	0.25	1.89	**0.33**	0.43
		1.0%	8.94	0.10	6.14	0.16	0.16	6.96	0.37	0.37
CitySearch .wdep	1	2.0%	0.95	0.18	0.73	**0.22**	0.45	0.82	**0.37**	0.60
		1.0%	2.51	0.17	1.84	**0.22**	0.34	2.07	**0.33**	0.43
	5	2.0%	0.99	0.18	0.75	**0.23**	0.45	0.83	**0.36**	0.60
		1.0%	2.53	0.18	1.83	**0.23**	0.35	2.06	**0.35**	0.48
Yelp.wdep	1	2.0%	10.7	0.08	7.79	**0.11**	0.19	8.14	**0.30**	0.41
		1.0%	33.0	0.09	21.4	**0.13**	0.16	22.8	**0.33**	0.35
	5	2.0%	11.5	0.08	7.78	**0.11**	0.19	8.36	**0.32**	0.43
		1.0%	32.5	0.09	21.7	**0.13**	0.17	24.7	**0.31**	0.33
MovieLens	1	2.0%	13.9	0.02	5.27	0.07	0.07	6.19	**0.39**	0.50
		1.0%	16.2	0.03	6.18	0.08	0.08	6.70	**0.37**	0.40
	5	2.0%	14.5	0.03	5.80	**0.07**	0.08	6.54	**0.41**	0.46
		1.0%	16.5	0.03	6.56	0.08	0.08	7.43	**0.36**	0.39

[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] Boldfaced values correspond to the cases when $\mathcal{B}^{add}$ is performing better than $\mathcal{B}^{mse}$.

8.1 Instance Based Itemset Mining Algorithm Evaluation

8.1.1 Running Time Comparison

Table 2 reports the running times for different quality bounding strategies for the different datasets. Looking at these results, we can make a number of observations. First, for qualities Q^σ and Q^m, our algorithms are more than one order of magnitude faster than the baseline approach on many datasets. Second, for all datasets and all parameters, our algorithm runs significantly faster when Q^σ and Q^m are used as the quality functions instead of Q^n. The fact that Q^m is faster is expected due to that Q^m is an upper bound of Q^n, so our bounding strategies are tighter for Q^m than for Q^n. Third, for most cases, the bounding strategy $\mathcal{B}^{add}$ performs better than the bounding strategy $\mathcal{B}^{mse}$ (see boldfaced entries). This is due to the fact that even though $\mathcal{B}^{add}$ is less effective in pruning the search space than $\mathcal{B}^{mse}$, its bounds can be computed faster than $\mathcal{B}^{mse}$'s, leading to an overall lower runtime.

8.1.2 Scalability Study

To study how our algorithm scales to large 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 the pruning strategy $\mathcal{B}^{add}$ for Q^m (Q^n) and $\mathcal{B}^{sig}$ for Q^σ on these datasets, and the results are summarized in Figure 1. It can be seen that for all parameters, our algorithm scales linearly with the size of the dataset.

8.2 Predictive Performances Of Different Quality Functions And EM Optimization

To evaluate the effectiveness of the EM optimization (or, in short, EMOpt) of AREM, we compare it against a Direct approach for assigning rule parameters. In the Direct approach, the rule's RHS and weight are simply μ_x and $1/\sigma_x$ (i.e., Q^σ). Similar approaches have been applied by RBA [11] and achieves reasonably good results.

Table 3 presents the mean values of MSE for different quality functions with two approaches (Direct and EMOpt) for assigning rule parameters. From this

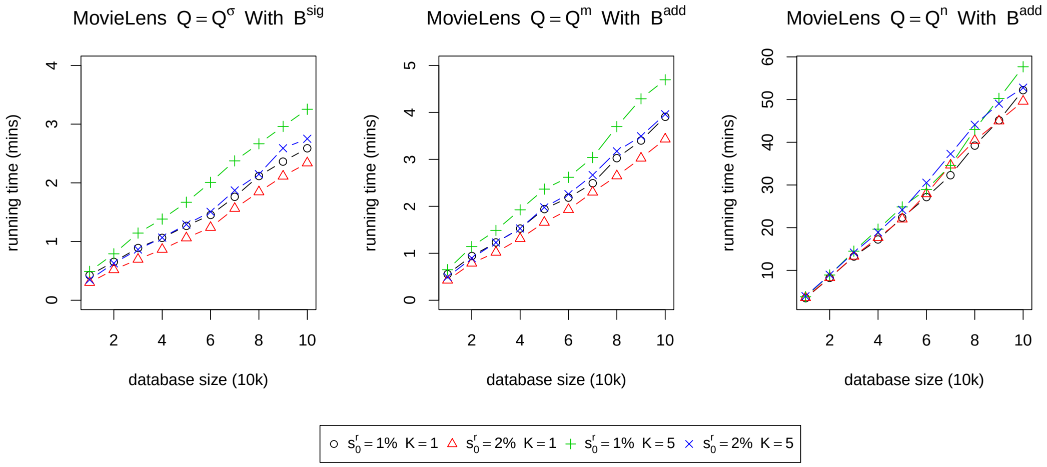

Figure 1: Scalability Study

table, we can see that the EM algorithm improves the performance dramatically for all three quality functions on almost all the datasets. In addition, both methods for assigning rule parameters (i.e., Direct and EMOpt) suggest that Q^m is performing slightly better than Q^n and Q^σ performs the worst among these three quality functions. This is reasonable because Q^m and Q^n take into account the instance's target variable when selecting the best instance based itemsets.

8.3 Comparing AREM To The State Of the Art Regression Models

For the comparison against the state of the art regression models discussed in Section 7.3, we run $AREM_k$ with the quality function Q^m, as it is the best choice among the three quality functions we developed (see Section 8.2). The average MSE for these models on the various data sets are shown in the Table 4, where the best results have been highlighted. For ease of comparisons, we present the NMDIFF values for these model pairs in Table 5 and the corresponding win-tie-loss values in Table 6. Note that $CART_1$ is the standard CART model, in contrast to $CART_k$ which stands for the boosted regression tree.

Tables 5 and 6 show that AREM is performing better than all competing methods. In particular, it is much better (with very high NMDIFF) 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 ($|\text{NMDIFF}| \geq 3$). 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 1), we think this is an indication that AREM is more suitable for high-dimensional and sparse data sets. Finally, from Table 4, we can see how different k values affect the AREM's performance. When $k = 1$, the performance is not satisfactory. This is not surprising because our probabilistic model is optimized for large number of itemsets. However, as k becomes sufficiently large (e.g., 20), the performance improves considerably and remains quite stable.

Table 3: *mean*(MSE) for Different Methods for Assigning Rule Parameters

Method	Q	k	BestBuy		CitySearch		Yelp		MovieLens	Airline	Socmob	Pollen	Spacega
			wdep	swf	wdep	swf	wdep	swf					
Direct	Q^{σ}	1	1.195	1.182	1.442	1.381	1.143	1.123	1.280	0.693	0.876	0.514	0.519
		10	1.169	1.133	1.301	1.265	1.108	1.100	1.244	0.689	0.704	0.503	0.516
		20	1.163	1.116	1.271	1.243	1.103	1.095	1.240	0.690	0.696	0.503	0.523
	Q^{m}	1	1.193	1.182	1.391	1.379	1.133	1.121	1.281	0.693	0.796	0.514	0.516
		10	1.123	1.077	1.143	1.130	1.053	1.046	1.230	0.709	0.521	0.560	0.509
		20	1.087	1.043	1.079	1.068	1.045	1.022	1.214	0.735	0.511	0.593	0.511
	Q^{n}	1	1.195	1.182	1.403	1.374	1.136	1.121	1.281	0.693	0.804	0.514	0.518
		10	1.117	1.087	1.150	1.154	1.051	1.045	1.229	0.707	0.537	0.559	0.509
		20	1.078	1.051	1.089	1.094	1.040	1.021	1.212	0.743	0.530	0.607	0.511
EMOpt	Q^{σ}	1	1.150	1.180	1.327	1.205	1.147	1.132	1.275	0.681	0.505	0.519	0.510
		10	1.010	0.781	1.017	0.895	1.038	0.897	1.229	0.672	0.398	0.482	0.479
		20	1.020	0.791	0.981	0.865	1.014	0.879	1.230	0.673	0.397	0.481	0.476
	Q^{m}	1	1.209	1.235	1.453	1.258	1.186	1.195	1.281	0.692	0.431	0.508	0.527
		10	0.868	**0.729**	0.886	0.790	0.934	0.844	1.238	0.659	0.293	0.482	0.491
		20	0.844	0.746	0.851	0.775	**0.907**	0.827	1.220	0.650	**0.292**	**0.480**	**0.474**
	Q^{n}	1	1.212	1.267	1.488	1.243	1.186	1.194	1.282	0.689	0.475	0.509	0.531
		10	0.868	0.741	0.895	0.790	0.946	0.838	1.235	0.659	0.303	0.483	0.497
		20	**0.842**	0.746	**0.846**	**0.773**	0.920	**0.821**	**1.215**	**0.647**	0.305	**0.480**	0.477

[a] For each dataset, the boldfaced (underlined) value corresponds to the best performing model for the EMOpt (Direct) method.

Table 4: Comparing $AREM_k$ To Other Models: *mean*(MSE)

model	BestBuy		CitySearch		Yelp		MovieLens	Airline	Socmob	Pollen	Spacega
	wdep	swf	wdep	swf	wdep	swf					
SVR	0.890	0.826	0.859	0.845	**0.881**	0.837	1.234	0.643	0.535	**0.469**	0.480
$CART_1$	1.116	0.904	1.336	1.020	1.149	1.137	1.228	0.649	0.440	0.488	0.488
$CART_{10}$	0.918	0.793	0.984	0.845	0.947	0.952	1.199	0.642	0.349	0.482	0.481
$CART_{20}$	0.857	0.777	**0.850**	0.825	0.901	0.864	**1.195**	**0.640**	0.341	0.483	0.484
$CUBIST_1$	0.989	0.930	1.135	1.048	1.080	1.024	1.237	0.658	0.363	0.501	0.491
$CUBIST_{10}$	1.034	0.987	1.131	0.993	1.070	1.014	1.260	0.664	0.370	0.500	0.492
$CUBIST_{20}$	1.042	0.988	1.134	0.997	1.081	1.012	1.260	0.665	0.369	0.500	0.493
RBA_1	1.139	1.031	1.291	1.145	1.123	1.102	1.291	0.730	0.522	0.508	0.522
RBA_{10}	0.982	0.903	1.033	0.946	1.027	0.972	1.238	0.691	0.595	0.497	0.496
RBA_{20}	0.968	0.897	1.000	0.937	1.015	0.963	1.235	0.691	0.605	0.497	0.496
$AREM_1$	1.209	1.235	1.453	1.258	1.186	1.195	1.281	0.692	0.431	0.508	0.527
$AREM_{10}$	0.868	**0.729**	0.886	0.790	0.934	0.844	1.238	0.659	0.293	0.482	0.491
$AREM_{20}$	**0.844**	0.746	0.851	**0.775**	0.907	**0.827**	1.220	0.650	**0.292**	0.480	**0.474**

[a] For each dataset, the boldfaced value corresponds to the best performing model.

9 Conclusions

We have proposed a novel regression model based on association rules called AREM. AREM consists of two major components. First, it applies an efficient algorithm for mining the instance based itemsets. The efficiency of our algorithm comes from several pruning strategies (infrequent items pruning, support equivalence items pruning and quality upper bound pruning) that are applied to aggressively reduce the search space. Extensive evaluation shows that the proposed algorithm is more than an order of magnitude faster than the baseline approach. Second, AREM learns the RHS of the rules together with their importance weights in a probabilistic framework, in which the rule's parameters are learned by the EM algorithm. Experiments based on many in house and public datasets show that the EM optimization can improve performance dramatically over the Direct approach. In addition, our model can perform better than RBA [11], Boosted Regression Trees [13], SVR [19], CART [12] and Cubist.

Table 5: Comparing $AREM_k$ To Other Models: NMDIFF

Model	BestBuy		CitySearch		Yelp		MovieLens	Airline	Socmob	Pollen	Spacega
	wdep	swf	wdep	swf	wdep	swf					
$CART_k$	0.64	3.01	−0.07	4.01	−0.50	3.25	−1.52	−0.94	2.56	0.35	0.27
SVR	2.27	5.21	0.59	5.81	−2.30	1.08	0.82	−0.66	10.56	−1.94	0.24
RBA_k	5.47	9.03	8.95	13.45	9.55	13.74	0.85	3.40	9.31	3.01	0.86
$CART_1$	12.35	9.83	30.67	17.24	18.90	29.16	0.47	−0.09	6.92	1.24	0.57
$CUBIST_k$	7.16	9.88	16.82	15.34	11.30	15.16	1.00	0.71	3.72	3.36	0.63

[a] For each of the four k-dependent models (i.e., $AREM_k$, $CART_k$, RBA_k and $CUBIST_k$), the calculation of NMDIFF is based on the k value that achieves the smallest mean MSE.

Table 6: Comparing $AREM_k$ To Other Models: win-tie-loss

comparing criteria[a]\model	$CART_k$	SVR_k	RBA_k	$CART_1$	$CUBIST_k$
$\lvert$NMDIFF$\rvert \geq 1$	4-6-1	5-4-2	9-2-0	8-3-0	9-2-0
$\lvert$NMDIFF$\rvert \geq 2$	4-7-0	4-6-1	9-2-0	7-4-0	8-3-0
$\lvert$NMDIFF$\rvert \geq 3$	3-8-0	3-8-0	9-2-0	7-4-0	8-3-0

[a] Comparing criteria |NMDIFF| $\geq n$ means that while comparing two models on a specific dataset, we count it as a tie of |NMDIFF| $< n$, and otherwise a win or loss depending on the sign of NMDIFF.

[b] The reported win-tie-loss are the counts accumulated over all datasets.

APPENDIX: Details About The Quality Upper Bound Pruning

In line 10 of the IBIMiner algorithm, we try 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 three quality functions Q^σ, Q^m and Q^n. 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 $\mathcal{T}_{x_p}$) 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 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 $\mathcal{B}(\tau, y) \geq Q(x, y)$.*

A.1 Instance Group's Formulation

To solve Problem 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 based on the concept of instance group which is defined in the following:

Definition A.1 *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. Define s_g to be the number of instances in group g, and (μ_g, σ_g) to be the (mean, standard deviation) of target variables for instances in group g.*

For any instance group g, we define $I_g = 1$ if $x \subseteq \tau_g$ and $I_g = 0$ otherwise, so that the original itemset x now mapped to a number of binary variables I_gs. Next, we focus on reformulating different parts of Problem 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 A.1, the support s_x of itemset x

can be rewritten as

$$s_x = \sum_i I_i = \sum_g \sum_{i:\tau_i=\tau_g} I_i = \sum_g \sum_{i:\tau_i=\tau_g} I_g = \sum_g I_g s_g, \tag{16}$$

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

$$\sum_g I_g s_g \geq s_0. \tag{17}$$

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,

$$I_{g_0} = 1. \tag{18}$$

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 A.1, we have

$$\begin{aligned}
\sigma_x^2 &= \frac{1}{s_x} \sum_i I_i (y_i - \mu_x)^2 \\
&= \frac{1}{s_x} \sum_g I_g \sum_{i:\tau_i=\tau_g} (y_i - \mu_x)^2 \\
&= \frac{1}{s_x} \sum_g I_g \sum_{i:\tau_i=\tau_g} [(y_i - \mu_g)^2 + (\mu_g - \mu_x)^2] \\
&\geq \frac{1}{s_x} \sum_g I_g \sum_{i:\tau_i=\tau_g} (y_i - \mu_g)^2 = \frac{1}{s_x} \sum_g I_g s_g \sigma_g^2.
\end{aligned} \tag{19}$$

By plugging Equation 16 into Equation 20, we get

$$\sigma_x^2 \geq \frac{\sum_g I_g s_g \sigma_g^2}{\sum_g I_g s_g}. \tag{20}$$

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 20.

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 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 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:

$$\begin{aligned}
MSE(x, y) &= \frac{1}{s_x} \sum_i I_i (y_i - y)^2 \\
&= \frac{1}{s_x} \sum_g I_g \sum_{i:\tau_i=\tau_g} (y_i - y)^2 \\
&= \frac{1}{s_x} \sum_g I_g \sum_{i:\tau_i=\tau_g} [(y_i - \mu_g)^2 + (\mu_g - y)^2] \\
&= \frac{\sum_g I_g s_g [\sigma_g^2 + (\mu_g - y)^2]}{\sum_g I_g s_g}.
\end{aligned} \tag{21}$$

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 21.

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 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 4 can be rewritten as

$$Q^m(x, y) = \frac{1}{\sigma_x} \frac{1}{\sqrt{1 + (\mu_x - y)^2/\sigma_x^2}}. \tag{22}$$

By comparing Q^m in Equation 22 to Q^n in Equation 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 $Q^m(x, y) \leq Q^\sigma(x, y)$. The fact that Q^m is an upper bound of Q^n implies that the same instance group's representation in Equation 21 can be used for finding the upper bound of Q^n.

In summary, two bounding constraints in Problem 1 are now reformulated as Equations 17 and 18. The upper bounding problem of the three quality functions are also transformed into the lower bounding problem of the RHS of Equations 20 and 21. So Problem 1 can now be fully represented in the instance group's notation. In addition, notice that the RHS of Equations 20 and 21 are of the same form:

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

and,

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

So, we only need to focus on finding the lower 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 17 and 18. 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 2 *Given $h_g > 0$ and v_g for each instance group g, our goal is to find the lower bound of the quantity*

$$\frac{v_{g_0} + \sum_{g \neq g_0} I_g v_g}{h_{g_0} + \sum_{g \neq g_0} I_g h_g}, \tag{23}$$

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 18.

A.2 The Geometric Formulation And Solutions

We find that it is much easier to present the solution and proof to the Problem 2 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 23 is the slope of $\mathbf{E}$, which is denoted as $||\mathbf{E}|| = E_v / E_h$. In addition, the constraint in Equation 17 becomes a constraint on the horizontal component of $\mathbf{E}$: $E_h \geq s_0$. So the Problem 2 can translated into a problem of finding the lower bound of the slope of vector $\mathbf{E}$ satisfying $E_h \geq s_0$. This problem is formally defined as:

Problem 3 *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 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 3 is described in the following:

Theorem A.1 *Assume the notations introduced for Problem 3. Without loss of generality, we assume the set of vectors are ordered as*

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

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 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

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

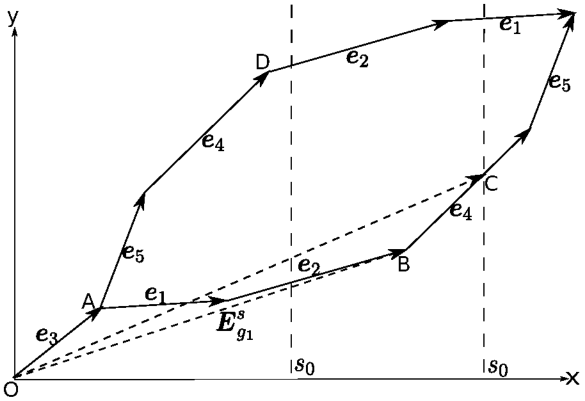

Figure 2: Geometric Solution To Lower Bounds

satisfies $E_{c,h} = s_0$*. We have* $||\mathbf{E}_c||$ *is the solution to problem 3. That is, given* $\mathbf{E} = \mathbf{e}_{g_0} + \sum_{g \neq g_0} I_g \mathbf{e}_g$ *for any* $I_g \in \{0, 1\}$ *satisfying* $E_h \geq s_0$*, we have*

$$||\mathbf{E}_c|| \leq ||\mathbf{E}||.$$

Proof A.2 *See Appendix A.4.1.*

Theorem A.1 is graphically shown in Figure 2, which contains five instance groups and $g_0 = 3$. The convex region ABCD is formed by connecting vectors $\mathbf{e}_g$ ($g \neq g_0$) sequentially in the order (and the reverse order) by which they are sorted. The minimum support vertical line s_0 splits this region into two parts. The right part of ABCD is the region which $\mathbf{E}$ can fall into. The vector OB is $\mathbf{E}^s_{g_1}$ defined in Theorem A.1. Intuitively, the slope of OB represents the lower bound for slopes of all possible vectors in ABCD. So, when $E^s_{g_1,h} \geq s_0$, $||\mathbf{E}^s_{g_1}||$ is the solution we are looking for. When $E^s_{g_1,h} \leq s_0$, we can improve this bound by adding more vectors to $\mathbf{E}^s_{g_1}$, until the horizontal component is s_0, which gives the vector OC as in Figure 2.

A.3 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 three 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^2_x 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 $v_g \leftarrow s_g \sigma^2_g$, $h_g \leftarrow s_g$ and $v_g \leftarrow s_g[\sigma^2_g + (\mu_g - y)^2]$, $h_g \leftarrow s_g$, and then apply Theorem A.1 to the RHS of Equations 20 and 21, respectively. We denote the methods for computing the upper bound of Q^σ (Q^m and Q^n) be $\mathcal{B}^{sig}$ ($\mathcal{B}^{mse}$).

However, there is a computational limitation of the methods $\mathcal{B}^{mse}$. In this approach, we need to sort instance groups $\mathbf{e}_g$ by $||\mathbf{e}_g|| = \sigma^2_g + (\mu_g - y)^2$. 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 bounding strategy for Q^m and Q^n as follows. We first split MSE into the sum of two terms:

$$(24) \quad MSE(x, y) = \frac{\sum_g I_g s_g \sigma^2_g}{\sum_g I_g s_g} + \frac{\sum_g I_g s_g (\mu_g - y)^2}{\sum_g I_g s_g}.$$

Next, we apply Theorem A.1 to each term in Equation 24 by setting $e_{g,v} \leftarrow s_g \sigma^2_g$, $e_{g,h} \leftarrow s_g$ 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 this approach $\mathcal{B}^{add}$. This approach is computationally more efficient due to that we only need to sort σ^2_g and μ_g once while applying the Theorem A.1. 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 24.

A.4 Theorems Proof

In the remaining of this section, we prove Theorem A.1. For this, the following lemmas are applied:

Lemma A.3 *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* $||\mathbf{e}_1|| \le ||\mathbf{e}_2||$, *then* $||\mathbf{e}_1|| \le ||\mathbf{e}_1 + \mathbf{e}_2|| \le ||\mathbf{e}_2||$.
2. *If* $||\mathbf{e}_1 + \mathbf{e}_2|| \le ||\mathbf{e}_2||$, *then* $||\mathbf{e}_1|| \le ||\mathbf{e}_2||$.
3. *If* $||\mathbf{e}_1 + \mathbf{e}_2|| \ge ||\mathbf{e}_1||$, *then* $||\mathbf{e}_1|| \le ||\mathbf{e}_2||$.

Proof A.4 *(1):* $||\mathbf{e}_1 + \mathbf{e}_2|| \le ||\mathbf{e}_2|| \Leftrightarrow (e_{1,v} + e_{2,v})/(e_{1,h} + e_{2,h}) \le e_{2,v}/e_{2,h} \Leftrightarrow e_{1,v} * e_{2,h} \le e_{2,v} * e_{1,h} \Leftrightarrow e_{1,v}/e_{1,h} \le e_{2,v}/e_{2,h} \Leftrightarrow ||\mathbf{e}_1|| \le ||\mathbf{e}_2||$. *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 A.5 *For* $n \ge 1$, *let* $\{\mathbf{e}_g | g = 1, 2, \ldots, n\}$ *be a set of 2-D vectors satisfying* $e_{g,h} > 0$. *Assume* $||\mathbf{e}_1|| \le ||\mathbf{e}_2|| \le \cdots \le ||\mathbf{e}_n||$. *We have*

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

Proof A.6 *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,* $||\mathbf{e}_1|| \le ||\mathbf{E}_{n-1}|| \le ||\mathbf{e}_{n-1}||$. *Combined with* $||\mathbf{e}_{n-1}|| \le ||\mathbf{e}_n||$, *we have* $||\mathbf{E}_{n-1}|| \le ||\mathbf{e}_n||$. *From Lemma A.3, we have* $||\mathbf{E}_{n-1}|| \le ||\mathbf{E}_{n-1} + \mathbf{e}_n|| = ||\mathbf{E}_n|| \le ||\mathbf{e}_n||$. *So,* $||\mathbf{e}_1|| \le ||\mathbf{E}_n|| \le ||\mathbf{e}_n||$, *i.e., the conclusion is also correct for* n.

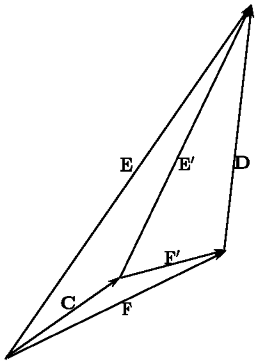

Figure 3: Lemmas A.7 and A.9

Lemma A.7 *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)* $E_h \ge F_h > 0$, *(d)* $||\mathbf{E}'|| \ge ||\mathbf{F}'||$ *if* $F'_h > 0$, *and (e)* $||\mathbf{E}'|| \ge ||\mathbf{F}||$ *if* $E'_h > 0$. *Then we have* $||\mathbf{E}|| \ge ||\mathbf{F}||$.

Proof A.8 *This lemma is graphically shown in Figure 3. We define* $\mathbf{D} = \mathbf{E} - \mathbf{F}$. *From (c), we have* $D_h \ge 0$.

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* $E'_v \ge F'_v \Rightarrow E_v \ge F_v$. *Combined with* $E_h = F_h$, *we get* $||\mathbf{E}|| \ge ||\mathbf{F}||$.

Next, consider the case $D_h > 0 \Rightarrow E'_h > 0$. *First, we want to prove* $||\mathbf{D}|| \ge ||\mathbf{E}'||$. *If* $F'_h = 0$, *from (b), we have* $\mathbf{F}' = 0 \Rightarrow \mathbf{D} = \mathbf{E}'$. *If* $F'_h > 0$, *from (d),* $||\mathbf{E}'|| \ge ||\mathbf{F}'||$. *We apply Lemma A.3 repeatedly to get* $||\mathbf{E}'|| = ||\mathbf{D} + \mathbf{F}'|| \ge ||\mathbf{F}'|| \Rightarrow ||\mathbf{D}|| \ge ||\mathbf{F}'|| \Rightarrow ||\mathbf{D}|| \ge ||\mathbf{D} + \mathbf{F}'|| = ||\mathbf{E}'||$. *Second, combining* $||\mathbf{D}|| \ge ||\mathbf{E}'||$ *with (e) gives* $||\mathbf{D}|| \ge ||\mathbf{F}||$. *Then from Lemma A.3,* $||\mathbf{D}|| \ge ||\mathbf{F}|| \Rightarrow ||\mathbf{E}|| = ||\mathbf{F} + \mathbf{D}|| \ge ||\mathbf{F}||$.

Lemma A.9 *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)* $||\mathbf{F}|| \ge ||\mathbf{F}'||$ *if* $F'_h > 0$, *and (e)* $||\mathbf{E}'|| \ge ||\mathbf{F}||$ *if* $E'_h > 0$. *Then we have* $||\mathbf{E}|| \ge ||\mathbf{F}||$.

Proof A.10 *This lemma is graphically shown in Figure 3.*

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 A.3 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 A.3 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* $||\mathbf{E}|| = ||\mathbf{C}|| \geq ||\mathbf{F}||$*.*

A.4.1 Proof Of Theorem A.1

(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 A.9 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 A.9 are satisfied.

To prove condition (d), we assume $F'_h > 0 \Rightarrow g_1 > 0$. From the definition of $\mathbf{F}'$ and Lemma A.5, 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 A.3, 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 A.3, we get $||\mathbf{F}|| \leq ||\mathbf{e}_{g_2}||$. From the definition of $\mathbf{E}'$ and Lemma A.5, 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 A.7 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 A.7 are satisfied.

To prove condition (d), assume $E'_h \geq F'_h > 0$. From the definition of $\mathbf{E}'$ and Lemma A.5, we have $||\mathbf{E}'|| \geq ||\mathbf{e}_{g_c}||$. From the definition of $\mathbf{F}'$ and Lemma A.5, 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 A.5, we get $||\mathbf{F}|| \leq ||\mathbf{e}_{g_c}||$. We have proved $||\mathbf{E}'|| \geq ||\mathbf{e}_{g_c}||$ above. So we get $||\mathbf{E}'|| \geq ||\mathbf{F}||$.

References

[1] Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. (1998) 80–86

[2] Li, W., Han, J., Pei, J.: Cmar: Accurate and efficient classification based on multiple class-association rules. In: ICDM. (2001) 369–376

[3] Baralis, E., Garza, P.: A lazy approach to pruning classification rules. In: ICDM. (2002) 35–42

[4] Yin, X., Han, J.: Cpar: Classification based on predictive association rules. In: SDM. (2003)

[5] Thabtah, F.A., Cowling, P., Peng, Y.: Mmac: A new multi-class, multi-label associative classification approach. Data Mining, IEEE International Conference on **0** (2004) 217–224

[6] Wang, J., Karypis, G.: Harmony: Efficiently mining the best rules for classification. In: In Proc. of SDM. (2005) 205–216

[7] Cheng, H., Yan, X., Han, J., Yu, P.S.: Direct discriminative pattern mining for effective classification. In: ICDE. (2008) 169–178

[8] Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)

[9] Quinlan, J.R., Cameron-Jones, R.M.: Foil: A midterm report. In: ECML. (1993) 3–20

[10] Cohen, W.W.: Fast effective rule induction. In: ICML. (1995) 115–123

[11] Ozgur, A., Tan, P.N., Kumar, V.: Rba: An integrated framework for regression based on association rules. In: SDM. (2004)

[12] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984)

[13] Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics **29** (2000) 1189–1232

[14] Jiang, Z., Karypis, G.: 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)

[15] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. of 20th Intl. Conf. on VLDB. (1994) 487–499

[16] Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD Conference. (2000) 1–12

[17] Cong, G., Tung, A.K.H., Xu, X., Pan, F., Yang, J.: Farmer: Finding interesting rule groups in microarray datasets. In: SIGMOD Conference. (2004) 143–154

[18] Cong, G., Tan, K.L., Tung, A.K.H., Xu, X.: Mining top-k covering rule groups for gene expression data. In: SIGMOD Conference. (2005) 670–681

[19] Smola, A.J., Schlkopf, B.: A tutorial on support vector regression. Technical report, STATISTICS AND COMPUTING (2003)

[20] Kotsiantis, S., Kanellopoulos, D.: Discretization techniques: A recent survey. (2006)

[21] Thabtah, F.A.: A review of associative classification mining. Knowledge Eng. Review **22**(1) (2007) 37–65

[22] Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. Journal of Machine Learning Research - Proceedings Track **14** (2011) 1–24

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

[24] de Marneffe, M.C., Manning, C.D.: The stanford typed dependencies representation. CrossParser '08, Stroudsburg, PA, USA, Association for Computational Linguistics (2008) 1–8

[25] Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2** (2011) 27:1–27:27 Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

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

[27] Triola, M.: Elementary statistics. Pearson/Addison-Wesley (2004)