AREM: A Novel Associative Regression Model Based On EM Algorithm

Zhonghua Jiang and George Karypis

Department of Computer Science & Engineering,
Digital Technology Center, University of Minnesota,
Minneapolis, MN, 55455, U.S.A.
`{zjiang,karypis}@cs.umn.edu`

Abstract. In recent years, there have been increasing efforts in applying 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 starts with finding a set of regression rules by applying the instance based pruning strategy, in which the best rules for each instance are discovered and combined. Then a probabilistic model is trained by applying the EM algorithm, in which the right hand side of the rules and their importance weights are updated. The extensive experimental evaluation shows that our model can perform better than both the previously proposed AR model and some of the state of the art regression models, including Boosted Regression Trees, SVR, CART and Cubist, with the Mean Squared Error (MSE) being used as the performance metric.

Keywords: association rule; regression rule; associative regression; probabilistic model; EM algorithm; instance based pruning;

1 Introduction

In recent years, there have been increasing efforts in applying association rule mining to build classification models [1] [2] [3] [4] [5], which have resulted in the area of Associative Classification (AC) modeling. Several studies [1] [2] [3] have provided empirical evidence that AC classifiers can outperform tree-based [6] and rule-induction based models [7] [8]. 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.* [9] that 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 the 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 [10] [11] and rule-induction based [12] regression models.

We present an associative regression model utilizing expectation maximization [13], called AREM. An AR model consists of three major components: (i) the method used to identify the sets of itemsets that form the left hand sides of the rules, (ii) the method used to estimate the right hand sides of the rules, and (iii) the method used to compute a prediction. Drawing upon approaches used for developing AC models, AREM uses an instance-based approach to select a subset of frequent itemsets that are used to form the left hand side of the rules. However, unlike existing AC and AR models, it develops and utilizes a probabilistic model coupled with an EM-based optimization approach to determine the right hand side of the rules and also assign a weight to each rule that is used during prediction. The advantage of this probabilistic model is that it allows AREM to capture the interactions of the various rules and to learn the parameters that lead to more accurate predictions. Our experimental evaluation shows that AREM outperforms several state of the art regression models including RBA [9], Boosted Regression Trees [10], SVR [14], CART [11] and Cubist [12] 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. AREM is formally presented in Section 4. In Section 5, we explain the experimental design and results for model evaluation. And finally Section 6 concludes.

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 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 [15].

Let the data set $\mathcal{D} = \{(\tau_i, y_i) | i = 1, 2, ..., N\}$ 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 the real-valued 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 support of itemset x, is defined as the number of instances in $\mathcal{D}$ that contain x. The itemset x is frequent if its support is not less than s_0, where s_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 $\mathcal{D}$ that contain x.

A regression rule is of the form $r_x : x \rightarrow \alpha_x$. The rule's left hand side (LHS) x is an itemset.The rule's right hand side (RHS) α_x is the target value predicted by this rule. Each rule is also associated with a positive value w_x which is used as the weight when combining multiple rules together for making predictions. The rule r_x is frequent if its itemset x is frequent.

3 Related Work

To our best knowledge, the RBA [9] model is the only previous work on associative regression. It starts with mining the set of frequent itemsets which form the set of rules' LHS. For each frequent itemset x, RBA computes the rule's RHS as the mean of x. It also computes the standard deviation σ_x of x. These rules are then ranked by variance (i.e., σ_x^2) from small to large. The database sequential coverage is applied to prune rules which are ranked low. For making predictions, three weighting schemes for w_x are developed: (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 inverse proportional to the variance σ_x^2.

Associative Classification (AC) [16] is an area that applies similar techniques, but the focus is on the Classification task. Among the many methods developed for AC modeling [1] [2] [3] [5], Harmony [4] is the model that employs a similar rule pruning strategy to AREM: it mines the highest confidence rules for each instance and combines them to the final rule set.

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) [11] partitions the input space into smaller, rectangular regions, and assigns the average of the target variables as the predicted value to each region. Cubist [12] is a rule based algorithm and fits a linear regression model to each of the regions. Boosting [10] 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 [17].

4 The AREM Model

The AREM model training consists of two major components. First, it discovers a set of frequent regression rules $r_x : x \rightarrow \mu_x$, where μ_x is the mean value of x in $\mathcal{D}$. We denote this set of rules by $\mathcal{R}$. Second, for each $r_x \in \mathcal{R}$, AREM updates its RHS to a new value α_x by learning a probabilistic model. The EM algorithm is applied for model learning where α_x is learned together with the rule's weight w_x.

For the rule discovery component (i.e., the first component above), AREM follows a two-step approach to find the rule set $\mathcal{R}$. First, it uses the FP Growth algorithm [18] to find all frequent itemsets x in $\mathcal{D}$. For each frequent itemset x, AREM generates the rule $r_x : x \to \mu_x$, where μ_x is the mean value of x in $\mathcal{D}$. Let $\mathcal{F}$ be this set of frequent rules. Second, for each training instance i, let $\mathcal{F}_i$ be the set of rules r_x from $\mathcal{F}$ such that $x \subseteq \tau_i$. AREM selects K rules from $\mathcal{F}_i$ to form the set $\mathcal{R}_i$. Finally, $\mathcal{R}$ is the union of these rules $\mathcal{R}_i$ over all training instances i in $\mathcal{D}$. Since $\mathcal{R}$ will in general contain fewer rules than $\mathcal{F}$, this step applies instance based approach to prune the initial set of frequent rules.

Using the set of updated rules $\mathcal{R}$ with the associated weights, AREM predicts the target variable of a test itemset τ 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., $(x_i \to \alpha_{x_i}) \in \mathcal{R}_\tau$ 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

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

which is nothing more than the average of the RHS of the k rules weighted by their corresponding w_{x_i} values. In the case when the test itemset τ is not covered by rules in $\mathcal{R}$, i.e., $|\mathcal{R}_\tau| = 0$, we simply predict $\hat{y}$ as the global mean of target variables in database $\mathcal{D}$.

AREM model requires the specification of four parameters: (i) the minimum support s_0, (ii) the number of rules K that are selected for each training instance, (iii) the number of EM steps M for rule parameter learning, and (iv) the number of rules k from $\mathcal{R}$ that are used for predicting the target variable. Even though the optimal values of these parameters need to be determined using a cross-validation framework, our experience has been that the performance of AREM remains consistently good for a wide range of these values.

In the rest of this section we describe the probabilistic model that we developed for estimating from $\mathcal{D}$ the α_x and w_x parameters of the rules in $\mathcal{R}$ and the method used to select for each training instance i the K rules from $\mathcal{F}_i$.

4.1 The Probabilistic Model

Let $\mathcal{X}$ be the set of itemsets of rules in $\mathcal{R}$ (i.e., $\mathcal{X} = \{x | r_x \in \mathcal{R}\}$). 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 independence $P[y|\tau, x] = P[y|x]$. That 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 β_x. That is,

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

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 1 now becomes

$$\hat{y} = \frac{\sum_{r_{x_i} \in \mathcal{R}_\tau} w_{x_i} \alpha_{x_i}}{\sum_{r_{x_i} \in \mathcal{R}_\tau} w_{x_i}} = \sum_x \alpha_x \frac{I_{x \subseteq \tau} w_x}{\sum_{x' \subseteq \tau} w_{x'}}, \tag{3}$$

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,

$$\hat{y} = E[y|\tau] = \sum_x E[y|\tau, x] P[x|\tau]. \tag{4}$$

From Equation 2, we get $E[y|\tau, x] = \alpha_x$. To specify $P[x|\tau]$, we compare Equation 3 with 4, and get

$$P[x|\tau] = \frac{I_{x \subseteq \tau} w_x}{\sum_{x' \subseteq \tau} w_{x'}}. \tag{5}$$

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 5. In the second step, the target variable y is generated by x with probability $P[y|x]$ given by Equation 2.

4.2 EM Algorithm: Learning $\boldsymbol{\alpha_x}$, $\boldsymbol{\beta_x}$ and $\boldsymbol{w_x}$

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

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_i \log\left(P[y_i|\tau_i, \boldsymbol{\theta}]\right) = \sum_i \log\left(\sum_{x_i} P[y_i, x_i|\tau_i, \boldsymbol{\theta}]\right), \tag{6}$$

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

$$\mathcal{Q}(\boldsymbol{\theta}_{t+1}, \boldsymbol{\theta}_t) = \sum_i \sum_{x_i} P[x_i|\tau_i, y_i, \boldsymbol{\theta}_t] \log(P[y_i, x_i|\tau_i, \boldsymbol{\theta}_{t+1}]). \tag{7}$$

This optimization problem is much easier than the original one for Equation 6, 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 $\mathcal{D}$. For the E-step, we first apply Bayes' Theorem so that

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

According to Equations 5 and 2, we have

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

Combining these two Equations, we get

$$\pi_{i,x_i,t} = \frac{\mathcal{N}(y_i|\alpha_{x_i,t}, \beta^2_{x_i,t}) w_{x_i,t} I_{x_i \subseteq \tau_i}}{\sum_{x' \subseteq \tau_i} \mathcal{N}(y_i|\alpha_{x',t}, \beta^2_{x',t}) w_{x',t}}. \tag{8}$$

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:

$$\alpha_{x,t+1} = \frac{\sum_{i:x\subseteq\tau_i} \pi_{i,x,t} y_i}{\sum_{i:x\subseteq\tau_i} \pi_{i,x,t}}, \tag{9}$$

and,

$$\beta^2_{x,t+1} = \frac{\sum_{i:x\subseteq\tau_i} \pi_{i,x,t}(y_i - \alpha_{x,t+1})^2}{\sum_{i:x\subseteq\tau_i} \pi_{i,x,t}}. \tag{10}$$

In Equations 9 and 10, the parameters α_x and β_x 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 5 into $\mathcal{Q}_2$, and taking the derivative, we get

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

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

$$\frac{w_{x,t+1}}{w_{x,t}} = \frac{\sum_{i:x\subseteq\tau_i} \pi_{i,x,t}}{\sum_{i:x\subseteq\tau_i} \frac{w_{x,t}}{\sum_{x'\subseteq\tau_i} w_{x',t}}}. \tag{11}$$

From Equations 9, 10 and 11, 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 $w_{x,M}$.

4.3 Instance Based Rule Mining

The instance based rule mining is applied in the rule discovery component of AREM discussed at the beginning of Section 4, which selects K rules from $\mathcal{F}_i$ to form $\mathcal{R}_i$ for each training instance i. For this, AREM first ranks rules in $\mathcal{F}_i$ by some "quality" metric, and then select the top K rules. The "quality" metric captures the quality of a rule from an instance's perspective. From our probabilistic model, $P[x|\tau_i, y_i]$ is the natural choice for the "quality" metric: a rule is

Table 1. Data Set Summary

Data Set	BestBuy		CitySearch		Yelp		Airline	Socmob	Pollen	Spacega
	dep	wf	dep	wf	dep	wf				
# of instances	10k	10k	10k	10k	10k	10k	10k	1156	3848	3107
# of items	1347	1010	1530	1080	2273	1662	676	44	17	24
density (%)[a]	1.29	1.52	1.36	1.95	1.35	1.94	1.63	11.36	23.53	25.00
# of trials[b]	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 if the data is converted into the matrix format.

[b] Number of trials the data set is randomized and then splitted into 80% training set, 10% validation set and 10% testing set.

better if it has a higher probability of being generated by the instance. We use the initialized rule parameters $w_{x,0}$, $\alpha_{x,0}$ and $\beta_{x,0}$ for computing $P[x|\tau_i, y_i]$. From $P[x|\tau_i, y_i] \propto P[x, y_i|\tau_i] \propto \mathcal{N}(y_i|\alpha_{x,0}, \beta_{x,0}^2)w_{x,0}$, We have that for the ranking's purpose $P[x|\tau_i, y_i]$ is equivalent to $\mathcal{N}(y_i|\alpha_{x,0}, \beta_{x,0}^2)$, where $w_{x,0} = 1$ is dropped. Thus, AREM uses $\mathcal{N}(y_i|\alpha_{x,0}, \beta_{x,0}^2)$ for rule ranking for each instance.

4.4 Comparing AREM With RBA

We summarize the main differences between AREM and RBA as follows. First, in determining a small set of itemsets to form the final rules' LHS, AREM applies an instance based approach, while RBA applies the database sequential coverage technique. Second, in determining the final rules' RHS, AREM learns them in the EM framework, while RBA simply uses the mean of the rules' itemsets. It turns out that, in AREM, the rule's RHS is the weighted mean, which is likely to be a better estimation than the unweighted mean used by RBA. Third, in determining the rule weights used for predictions, AREM learns them together with rules' RHS, while RBA pre-specifies methods for computing them. These pre-specified methods may be reasonable but they are not optimized. Finally, in determining top k rules used for making predictions, AREM selects rules with the highest weights, while RBA selects rules with the smallest variance. Our choice is consistent with our probabilistic model in that rules with higher chance of being generated (see Equation 5) are more important and should be selected.

5 Experimental Study

5.1 Data Sets

We evaluate the performance of AREM on 10 data sets summarized in Table 1. The first six data sets are randomly sampled from user reviews downloaded from three websites: "BestBuy" [19], "CitySearch" [20], and "Yelp" [21]. 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: "dep" and "wf". For "dep", the Stanford dependencies [22] between words in each sentence are extracted. Each dependency is a triplet containing the name of the relation, the governor and the dependent. For "wf", words in the review text are extracted. We remove the infrequent items whose relative supports (that is, the support divided by $|\mathcal{D}|$) are less than 0.5%. The "Airline" data set is downloaded from DataExpo09 competition [23]. The last three data sets are downloaded from CMU StatLib [24].

5.2 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) [14] is an exception. It is included because it is one of the best known and standard models for regression.
SVR We use "libsvm" [25] 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.
$\boldsymbol{CART}_k$ This group of models contain the Classification And Regression Tree (CART) [11] and the Boosted Regression Tree [10] 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.
$\boldsymbol{CUBIST}_k$ Cubist [12] 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.
$\boldsymbol{RBA}_k$ We implemented the RBA model following [9]. 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*.
$\boldsymbol{AREM}_k$ Here, k is the number of top ranked rules used for prediction. We tuned three parameters for $AREM_k$: s_0, K and M, where s_0 is the minimum support threshold, K is the number of high quality rules for each training instance during pruning, and M is the number of EM steps during model training.

The parameter k in the above models (except *SVR*) can be uniformly interpreted as the number of rules used for making predictions. For our experimental study, we choose k to be 1, 5, 10, 15 and 20 for all four models. The rationale of choosing these values comes from the following: 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. We choose the maximum k value to be 20 as a compromise from these two extreme case considerations.

5.3 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,

Table 2. Model Comparison: Average MSE

model\data	BestBuy		CitySearch		Yelp		Airline	Socmob	Pollen	Spacega
	dep	wf	dep	wf	dep	wf				
SVR	0.945	0.810	0.961	0.814	0.935	0.770	**0.643**	0.535	**0.469**	**0.480**
$CART_1$	1.014	0.875	1.131	0.974	1.118	0.924	**0.649**	0.440	0.487	**0.488**
$CART_5$	0.937	0.815	0.997	0.847	0.994	0.804	**0.640**	0.349	0.481	**0.480**
$CART_{10}$	0.921	0.799	0.962	0.827	0.962	0.782	**0.642**	0.349	0.482	**0.481**
$CART_{15}$	0.913	0.790	0.956	0.809	0.946	0.765	**0.640**	0.349	0.483	**0.482**
$CART_{20}$	0.909	0.787	0.949	0.814	0.939	**0.755**	**0.640**	0.341	0.483	**0.484**
$CUBIST_1$	1.043	0.880	1.210	0.990	1.130	0.959	0.658	0.363	0.501	0.490
$CUBIST_5$	1.070	0.937	1.213	0.966	1.129	0.949	0.663	0.367	0.500	0.494
$CUBIST_{10}$	1.074	0.943	1.216	0.973	1.138	0.946	0.664	0.370	0.499	0.492
$CUBIST_{15}$	1.080	0.947	1.218	0.976	1.138	0.944	0.664	0.369	0.499	0.493
$CUBIST_{20}$	1.081	0.951	1.221	0.985	1.137	0.944	0.664	0.369	0.499	0.493
RBA_1	1.111	1.004	1.200	1.141	1.156	1.023	0.730	0.533	0.507	0.530
RBA_5	0.969	0.898	1.044	0.928	1.026	0.930	0.682	0.562	0.496	0.496
RBA_{10}	0.964	0.878	1.041	0.894	1.019	0.915	0.685	0.594	0.497	0.496
RBA_{15}	0.962	0.872	1.040	0.893	1.015	0.904	0.685	0.603	0.497	0.497
RBA_{20}	0.964	0.872	1.038	0.890	1.013	0.903	0.685	0.603	0.497	0.497
$AREM_1$	1.248	1.235	1.354	1.248	1.311	1.241	0.754	0.421	0.581	0.628
$AREM_5$	0.875	0.763	0.908	0.844	0.953	0.799	0.670	**0.307**	0.499	0.529
$AREM_{10}$	**0.862**	**0.751**	**0.896**	0.784	**0.920**	**0.753**	0.657	**0.299**	0.483	0.507
$AREM_{15}$	**0.864**	**0.753**	**0.894**	**0.773**	**0.921**	**0.748**	0.652	**0.299**	0.481	0.490
$AREM_{20}$	**0.865**	**0.758**	**0.899**	**0.770**	**0.926**	**0.749**	**0.646**	**0.300**	0.481	**0.483**

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.

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 [27] 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.

5.4 Experimental Results

The average MSE for the discussed set of models on the various data sets are shown in the Table 2, where the best results have been highlighted. Table 3 shows the quantity $\mu_{m_1-m_2}/\sigma_{m_1-m_2}$ for comparing $AREM_k$ to the rest of the models. Note that $CART_1$ is the standard CART model, in contrast to $CART_k$ which stands for the boosted regression tree. For easy comparison, we derive the win-tie-loss from Table 3 and present them in Table 4.

Table 3. Compare $AREM_k$ To Other Models: $\mu_{m_1-m_2}/\sigma_{m_1-m_2}$

Model\Data	BestBuy		CitySearch		Yelp		Airline	Socmob	Pollen	Spacega
	dep	wf	dep	wf	dep	wf				
$CART_k$	2.86	2.71	4.26	2.65	1.73	0.56	-0.61	2.29	0.09	-0.12
SVR	4.65	4.16	4.97	2.95	1.26	1.80	-0.35	10.51	-2.14	-0.11
RBA_k	4.98	8.15	10.18	8.11	8.64	12.17	3.15	9.68	2.74	0.52
$CART_1$	7.89	8.36	16.78	11.48	16.50	13.84	0.23	6.77	1.15	0.23
$CUBIST_k$	8.04	7.50	20.25	11.49	16.43	13.76	1.03	3.49	3.15	0.31

Table 4. Compare $AREM_k$ To Other Models: win-tie-loss

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

[a] It is a tie if $|\mu_{m_1-m_2}| < n\sigma_{m_1-m_2}$. Otherwise, it is a win or loss depending on the sign of $\mu_{m_1-m_2}$.

Tables 3 and 4 show that AREM is performing better than all competing methods on most of the data sets. For almost all cases, AREM is either better or at least as good as the competing method (with the only exception on “Pollen” when compared to SVR). 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 2, 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 rules. However, as k becomes sufficiently large (15 or 20), the performance improves considerably and remains quite stable.

6 Conclusions

We have proposed a novel regression model based on association rules called AREM. AREM applies the instance based rule mining approach to discover a set of high quality rules. Then the rules' RHS and importance weights are learned consistently within the EM framework. Experiments based on 10 in house and public datasets show our model can perform better than RBA [9], Boosted Regression Trees [10], SVR [14], CART [11] and Cubist [12].

Acknowledgment

This work was supported in part by NSF (IOS-0820730, IIS-0905220, OCI-1048018, CNS-1162405, and IIS-1247632) and the Digital Technology Center at the University of Minnesota. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

References

1. Li, W., Han, J., Pei, J.: Cmar: Accurate and efficient classification based on multiple class-association rules. In: ICDM. (2001) 369–376
2. Yin, X., Han, J.: Cpar: Classification based on predictive association rules. In: SDM. (2003)
3. Thabtah, F.A., Cowling, P., Peng, Y.: MMAC: A New Multi-class, Multi-label Associative Classification Approach. In: Proceedings of the 4th IEEE International Conference on Data Mining, ICDM '04. (2004) 217–224
4. Wang, J., Karypis, G.: Harmony: Efficiently mining the best rules for classification. In: In Proc. of SDM. (2005) 205–216
5. Cheng, H., Yan, X., Han, J., Yu, P.S.: Direct discriminative pattern mining for effective classification. In: ICDE. (2008) 169–178
6. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
7. Quinlan, J.R., Cameron-Jones, R.M.: Foil: A midterm report. In: ECML. (1993) 3–20
8. Cohen, W.W.: Fast effective rule induction. In: ICML. (1995) 115–123
9. Ozgur, A., Tan, P.N., Kumar, V.: Rba: An integrated framework for regression based on association rules. In: SDM. (2004)
10. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics **29** (2000) 1189–1232
11. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984)
12. Quinlan, J.R.: Cubist. `http://www.rulequest.com`
13. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B **39**(1) (1977) 1–38
14. Smola, A.J., Schlkopf, B.: A tutorial on support vector regression. Technical report, STATISTICS AND COMPUTING (2003)
15. Kotsiantis, S., Kanellopoulos, D.: Discretization techniques: A recent survey. (2006)
16. Thabtah, F.A.: A review of associative classification mining. Knowledge Eng. Review **22**(1) (2007) 37–65
17. Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. Journal of Machine Learning Research - Proceedings Track **14** (2011) 1–24
18. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD Conference. (2000) 1–12
19. BestBuy. `http://www.bestbuy.com`
20. CitySearch. `http://www.citysearch.com`
21. Yelp. `http://www.yelp.com`
22. Marneffe de, M.C., Manning, C.D.: The stanford typed dependencies representation. In: Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation. CrossParser '08 (2008) 1–8
23. Airline. `http://stat-computing.org/dataexpo/2009`
24. StatLib. `http://lib.stat.cmu.edu/datasets`
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
26. Pedregosa, F., Varoquaux, G., Gramfort, A., et al.: Scikit-learn: Machine Learning in Python . Journal of Machine Learning Research **12** (2011) 2825–2830
27. NIST-handbook: Two-sample t-test for equal means. `http://www.itl.nist.gov/div898/handbook/eda/section3/eda353.htm`