PRIMER – A Regression-Rule Learning System for Intervention Optimization

Greg Harris[1], Anand Panangadan[2], and Viktor K. Prasanna[3]

[1] Department of Computer Science
University of Southern California, Los Angeles, California
gfharris@usc.edu
[2] Department of Computer Science
California State University, Fullerton, California
apanangadan@fullerton.edu
[3] Ming-Hsieh Department of Electrical Engineering
University of Southern California, Los Angeles, California
prasanna@usc.edu

Abstract. We introduce intervention optimization as a new area of exploration for data mining research. Interventions are events designed to impact a corresponding time series. The task is to maximize the impact of such events by training a model on historical data. We propose PRIMER as a new regression-rule learning system for identifying sets of event features that maximize impact. PRIMER is for use when domain experts with knowledge of the intervention can specify a transfer function, or the form of the expected response in the time series. PRIMER's objective function includes the goodness-of-fit of the average response of covered events to the transfer function. Incorporating domain knowledge in this way makes PRIMER robust to over-fitting on noise or spurious responses. PRIMER is designed to produce interpretable results, improving on the interpretability of even competing regression-rule systems for this task. It also has fewer and more intuitive parameters than competing rule-based systems. Empirically, we show that PRIMER is competitive with state-of-the-art regression techniques in a large-scale event study modeling the impact of insider trading on intra-day stock returns.

Keywords: regression rules, intervention analysis, rule induction, event response, time series, intervention optimization, rule learning

1 Introduction

Intervention analysis was introduced in 1975 by Box and Tiao [1] as a means of assessing the impact of a special event on a time series. In one example, they evaluate whether gasoline regulation in 1960 impacted smog levels in Los Angeles. The effect is not obvious in the noisy graph of monthly smog levels. However, their method is able to quantify even weak effects in such noisy time series. Their method has three steps:

1. Identification – frame a model for change which describes what is expected to occur given knowledge of the known intervention;
2. Fitting – work out the appropriate data analysis based on that model;
3. Diagnostic Checking – if the model proves inadequate for inference, make necessary adjustments and repeat the analysis.

1.1 Intervention Optimization

Our research extends intervention analysis from the case of one event to the case of many events. The goal is to reliably predict which events will have the highest *impact* (defined later in this section) on their corresponding time series. We call this *intervention optimization* and have not found it previously discussed in data-mining literature. An example use-case is optimizing the impact of advertising campaigns on same-store sales for a retail business. In this case, the events include various kinds of advertising campaigns, such as locally airing a television ad or mailing fliers. Each event is expected to affect a corresponding time series, in this case, sales at the targeted store location. Another example use-case is optimizing cyclic steam injection for enhanced oil recovery in highly viscous oil fields. Pausing production periodically to send steam down an oil well warms the surrounding oil, making it easier to extract once pumping resumes. The increase in production depends on the well, the reservoir, and the characteristics of the steam job. Intervention optimization is useful in these use-cases, because it helps maximize the impact of limited resources (ad budget or steam supply).

Intervention optimization requires training a model that predicts the impact of an event based on a set of descriptive features. Modeling the highest-impact events is challenging due to the propensity to over-fit. Models that make inferences based on too-few historical high-impact examples can have low out-of-sample accuracy. Additionally, over-fitting can be caused by noise in the time series, which affects the estimation of impact. PRIMER, the intervention optimization system we propose, adapts to noisy time series by requiring more samples for inference. We evaluate PRIMER and other modeling techniques by the average impact of their top-predicted $x\%$ of intervention events, using held-out data and given only the features describing each event. The same techniques can be applied to general event studies where one has no direct control over the events, but would still benefit from a predictive or explanatory impact model.

We define the *impact* of an event as the cumulative subsequent "boost" in time series values caused by the event. Concretely, we assume a given event E occurs at time t_0 and is expected to affect time series R of realized values. In our notation, R_t refers to the value of R at time index t. The first value of R subject to the influence of E is R_{t_0}, and the value one time step after the event is R_{t_0+1}. We first calculate a baseline time series B of expected values had the event not occurred. In the simplest case, $B_t = R_{t_0-1}, \forall t : t \geq t_0$, which assumes the time series would have simply remained at the last known pre-event level. In more complex cases, B must be determined from a domain-specific model, including considerations such as autocorrelation and seasonality. We define S such that:

$$S_t = R_t - B_t \tag{1}$$

making it the unexplained residual after all known effects unrelated to E have been removed from R. If E were to have no impact, then the expectation $\mathbf{E}[S] = 0$. Finally, we define the impact of E on S over a finite period (n time steps) as:

$$impact_E(S) = \sum_{t=t_0}^{t_0+n-1} S_t \tag{2}$$

We assume the effect of E begins at t_0, meaning the event is unanticipated in the time series: $B_t = R_t, \forall t : t < t_0$.

1.2 Interpretability

Modeling impact as a function of input variables is a regression problem, and we include state-of-the-art regression algorithms in the experiments in Section 4. However, our interest is in interpretable models which more easily provide insight to the user. For this reason, we also evaluate regression-rule learning algorithms, which are designed to emphasize interpretability over accuracy. Regression-rules are simple piece-wise constant models, taking the form:

$$(F_1 \wedge F_2 \wedge F_3) \longrightarrow C$$

This is understood to mean that an event containing features F_1, F_2, and F_3 would have a predicted impact of C. Some systems generate ordered rule lists where the predicted value of a sample comes from the first rule that *covers* it, meaning the first rule where the sample contains all the rule features [2, 14, 15]. Other systems generate unordered ensembles of rules, where *all* rules that cover a sample provide an additive contribution to the overall prediction [5].

Currently-proposed regression-rule learning systems are not specifically designed to produce rules easily interpretable for the task of intervention optimization. The ideal set of rules for this task should be:

- short, covering only the highest-impact events
- sorted in descending order of predicted impact
- free of exceptions or caveats to the predictions

Current systems do not produce rule sets with these attributes. Ensemble-based systems produce rule sets which cannot be shortened without affecting the predictions. Ensemble rule sets can be sorted in descending order of impact contribution to improve interpretability, but caveats remain in the form of rules with negative impact contribution. For example, a rule with features F_1 and F_2 may have a high impact contribution, but another rule with only feature F_1 may have a negative impact contribution. Therefore, a user cannot reliably identify high-impact events by simply remembering the high-impact rules. Likewise, ordered rule lists generated by current systems are complicated by exceptions. The rules are not ordered according to impact, but are ordered according to how well they reduce a loss function such as mean squared error. The rule lists cannot be re-sorted by impact, because the order of precedence must be maintained for

accurate predictions. For example, a high-impact rule in the middle of the list with feature F_1 may be preempted by an earlier low-impact rule with feature F_2. So, a user cannot simply say that events with feature F_1 will have high impact, without also mentioning the exception for events that also contain feature F_2.

PRIMER is a rule-based system designed specifically to generate interpretable rules for intervention optimization. The rule lists generated by PRIMER are ordered according to expected impact. The user can review and retain only as as many rules as needed to cover a sufficient number of events. Because of the rule ordering, predictions can be interpreted as minimum predictions. There is no concern about exceptions, because earlier rules in the list have predictions at least as high.

The requirement of using interpretable models sometimes means accepting lower accuracy. PRIMER, however, is able to maintain high accuracy by taking advantage of domain knowledge specific to the task of intervention optimization. The domain knowledge provided by the user is the functional form of the expected response pattern. By knowing the expected pattern to find in the time series, PRIMER is better able to disregard spurious fluctuations and noise. In this paper, we limit our scope to response patterns that exhibit a strong initial response that decays following the event, until eventually the effect has dissipated. The two example use-cases mentioned both have this form of response pattern. In the retail business use-case, the effect of an ad is likely to be strongest initially, before it slowly gets forgotten. In the oil recovery use-case, production is highest immediately after steam injection. As the oil cools, production gradually decays back to its original level.

PRIMER has a unique objective function designed specifically for intervention optimization. It combines three heuristics to improve out-of-sample performance: impact, coverage, and goodness-of-fit to the expected response pattern. We show its effectiveness in a large-scale event study modeling the impact of insider trading filings on intra-day stock returns. The study of market reactions to news, and filings in particular, is an active area of research in Behavioral Finance [16, 20]. PRIMER has the capability of providing insight into which filing characteristics most influence the market. In our tests, we show that that PRIMER is competitive with state-of-the-art regression algorithms at identifying high-impact filings, while the model output has improved interpretability.

2 Related Work

Intervention optimization is a regression problem, and we include common regression models in our experiments. Due to our emphasis on model interpretability, however, we describe only other regression-rule learning systems in this related work section:

RegENDER [5] is a system for learning an ensemble of regression rules using forward stage-wise additive modeling. Each rule is added greedily, one by one. The rules are chosen to minimize a loss function, which is either the sum of squared error or the sum of absolute error, calculated over all samples. The

number of rules to learn is an input to the algorithm which acts as the stopping criterion. The minimization technique can be specified as either gradient boosting or a least angle approach. To reduce correlation between rules as well as computational complexity, the training of each rule is done using a random subset of the training data. The fraction of samples to use for training is an input to the algorithm. The final parameter is a shrinkage factor which reduces the degree to which previously generated rules affect the generation of the successive one in the sequence. The algorithm outputs an unordered list of rules. The prediction for a given sample is calculated by summing the contributions of all rules that cover the sample.

SeCoReg [14] is a regression rule system based on the separate-and-conquer strategy [8]. The algorithm uses hill-climbing to find each rule. The objective function maximized by hill-climbing is a weighted combination of the relative root mean squared error and the relative coverage:

$$h_{cm} = \alpha \cdot (1 - L_{RRMSE}) + (1 - \alpha) \cdot relCov \tag{3}$$

Here, the parameter α controls the trade-off between error and coverage. The stopping criterion for the algorithm is set as the fraction of samples that can be left uncovered by the rules learned. A third user-specified parameter controls the number of splitpoints found by a supervised clustering algorithm for discretizing numeric features.

Ant-Miner-Reg [2] is a version of SeCoReg with hill-climbing replaced by Ant Colony Optimization. It requires three additional user-specified paramaters to control the optimization.

Dynamic Reduction to Classification [15] is a method of converting a regression problem into a multi-class classification problem. This enables the use of well-studied classification rule induction techniques. For each rule, the predicted value is the median of the values covered. The rule quality is measured by how well it identifies samples valued within one standard deviation of the predicted value. Samples valued within one standard deviation of the predicted value are set as the positive class, and all other values outside this range are considered negative. In this way, traditional classification rule heuristics can be used as an objective function. The heuristics tested by its authors include: correlation, relative cost measure, Laplace measure, and weighted relative accuracy. The algorithm uses separate-and-conquer combined with hill-climbing. The stopping criterion for the algorithm is the fraction of samples that can be left uncovered by the rules learned.

M5'Rules [12] learns rules that have a linear model in the head instead of a constant prediction. The algorithm uses separate-and-conquer to learn the set of rules, stopping when all samples are covered. Each rule is learned by first generating a decision tree and then extracting the rule corresponding to the best leaf. The best leaf is determined according to a heuristic. Its authors tested three heuristics: percent root mean squared error, mean absolute error divided by coverage, and the correlation between predicted and actual values for samples covered by a leaf multiplied by the number of samples in the leaf.

3 PRIMER

We propose a new method for intervention optimization: Pattern-specific Rule-based Intervention analysis Maximizing Event Response (PRIMER). In this section, we refer to Algorithm 1 as we describe each part of the method.

The inputs to PRIMER, as shown in Algorithm 1, include a set of events. Each event contains a descriptive set of binary features. Each also contains the corresponding response, which is a short time series segment beginning with the event at time t_0 and lasting until the impact of the event has substantially decayed. The event response is a sub-sequence of S, defined in Eq. 1.

3.1 Separate-and-Conquer with Beam Search

PRIMER uses the separate-and-conquer strategy [8] common to most rule learning systems. This strategy iteratively identifies the events not yet covered by any rule (separate), and then learns a new rule using only the uncovered events (conquer). The new rule covers additional events, which are then removed from consideration in the next iteration. This guarantees subsequent rules have diversity in coverage. Lines 1 – 7 in Algorithm 1 describe our implementation of this strategy. This loop is repeated until rules are discovered that cover all events, or until the empty rule is returned because no better rule could be found.

To find each new rule, PRIMER uses top-down beam search. Beam search is a greedy heuristic search method which is used by the vast majority of rule learning algorithms [9]. It finds good rules quickly, while covering only a small fraction of the search space. Beam search starts with an empty rule that covers all samples and greedily refines it by adding features as conditions. It maintains a beam of the best b rules of each rule length. To find rules of length l, each feature is successively added as a refinement to each rule in the beam for length $l-1$. The refined candidate rules are then evaluated, and the top b rules are stored in the beam for length l. After reaching some stopping criterion, the best rule from all lengths is selected.

Limiting the beam size limits the extent of the search. Setting $b = 1$ makes beam search equivalent to hill-climbing. Setting $b = \infty$ makes it equivalent to exhaustive search. PRIMER implements top-down beam search in the FindBestRule function (lines 8 – 21). The function returns the single highest-scoring rule discovered during the search.

3.2 Objective Function

During beam search, each candidate rule is evaluated and scored by an objective function. PRIMER's unique objective function combines three heuristics to improve out-of-sample performance: impact, goodness-of-fit, and coverage.

Impact. The goal of intervention optimization is to maximize the impact of future interventions. We evaluate rules according to the average impact of the

out-of-sample events they cover, so historical impact is naturally an important heuristic. For rule evaluation, the impact for an event is defined in Eq. 2. During model training, however, PRIMER optimizes only on impact that fits the expected response pattern given by a user with domain expertise. This helps avoid over-fitting to noise or spurious fluctuations in the time series.

The key to the intervention analysis method proposed by Box and Tiao [1] is specifying the expected response pattern. Their method is able to quantify weak effects in noisy time series by relying on the use of a *transfer function*, a tentative specification of the stochastic model form. The transfer function is based on prior knowledge of the intervention, and how the time series is expected to react. Some example transfer functions they listed include linear, pulse, and step functions. PRIMER inherits the use of a transfer function from their work on intervention analysis.

With PRIMER, we have tested response patterns that exhibit an abrupt initial response at time t_0 which decays back down to the pre-intervention level within n time steps. Specifically, we have tested the exponential decay function:

$$f(t) = Ae^{-k(t-1)}, \quad t \geq 1 \tag{4}$$

and the power law function:

$$f(t) = At^{-k}, \quad t \geq 1 \tag{5}$$

In both cases, A is the scaling parameter, and k determines the rate of decay. Restricting the impact to only include the fitted area under the transfer function curve makes the algorithm more robust by down-weighting, for example, spurious spikes that occur well after t_0.

The first step in calculating the fitted impact of a rule is to average the responses of all events covered by the rule (*avgResponse* in line 23 of Algorithm 1). The next step is to fit the the transfer function to the average response by minimizing the sum of squared differences:

$$\begin{aligned} \underset{A,k}{\text{minimize}} \quad & \sum_{t=1}^{n} (f(t) - avgResponse_t)^2 \\ \text{subject to} \quad & A, k \geq 0 \end{aligned} \tag{6}$$

We use the trust-region-reflective optimization algorithm [4], which allows us to specify lower bound constraints of zero on the fit parameters.

Goodness-of-Fit. In PRIMER, we score each rule conservatively, based on the goodness-of-fit of the average response to the transfer function. In line 24 of Algorithm 1, we calculate confidence intervals for the fit parameters optimized in Eq. 6. We calculate the confidence intervals based on the asymptotic normal distribution for the parameter estimates [19]. The level of confidence used in the interval calculation is a user-specified parameter, α, which produces $100 \cdot (1 - \alpha)$ percent confidence intervals.

In line 28 of Algorithm 1, we choose the more conservative values from the confidence intervals for each fit parameter. For the scale parameter A in the exponential decay transfer function (Eq. 4) and the power law transfer function (Eq. 5), we use the lower bound confidence interval. For the rate of decay parameter k, we use the upper bound confidence interval. We calculate the rule score as the area under the transfer function using the conservative confidence interval values for parameters (line 29). This score is lower than the fitted impact due to the reduced scale and increased rate of decay. This penalizes rules with poorly-fitting average response curves which would otherwise have had high impact according to the fitted parameters.

We also use the confidence intervals as a stopping criterion. If both parameters are not significantly greater than zero according to the confidence intervals, the rule is given a score of zero. If no rule can be found with a score greater than zero, the default empty rule is chosen by the beam search, since it has been assigned a small positive score (line 10). The empty rule has no conditions and covers all remaining samples, causing the separate-and-conquer loop to terminate.

Coverage. PRIMER includes a bias toward rules with high coverage. This trade-off of impact for coverage in the objective function has the potential to improve out-of-sample performance, because high-coverage rules are less prone to over-fitting.

Due to random noise in the time series, an individual event response is unlikely to closely resemble the transfer function. A rule that covers only a small number of events will have a noisy average response. As discussed previously, a poor fit of the average response to the transfer function reduces the rule score, possibly to zero. A rule that covers many events will have a well-behaved average response where the random noise averages out. With less noise, the fit becomes better, and the rule score increases. In this way, high-coverage rules are favored. The trade-off between coverage and impact is controlled by the parameter α, which was introduced in the previous section. High α equates to low-confidence bounds, which are close to the least squares fit parameters. Inversely, low α increases confidence that the true parameters are within the intervals by widening the intervals. Effectively, lowering α reduces tolerance for noisy average responses, which then increases the bias toward rules with less noise; and rules with less noise tend to be rules with higher coverage.

Fig. 1 illustrates how increasing coverage increases the score for a rule on synthetic data. This plot involves samples with the same power law decay response added to noisy time series. Adding samples smooths the average response, which raises the lower bound on the fitted curve.

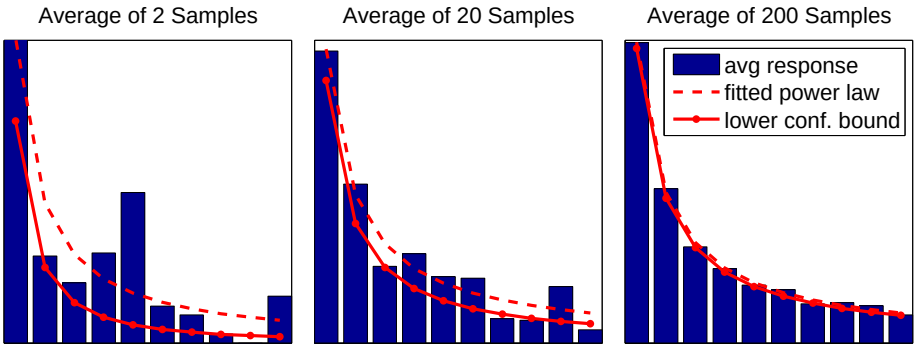

Fig. 1. Illustration using synthetic data, where each response sample is generated using the power law function with white noise added. Averaging many samples improves the goodness-of-fit, raising the lower confidence bound and increasing the score of a rule.

4 Experiments

We evaluate PRIMER on readily available data from the U.S. financial markets. Our objective is to predict which insider trading reports have the largest positive effect on intra-day stock prices. We cannot influence such financial news, so this this use-case is not true intervention optimization. However, as an event study, we can evaluate PRIMER's ability to identify high-impact events.

4.1 Data

Events. For events, we use Form-4 regulatory filings disseminated by the U.S. Securities and Exchange Commission.[4] Form-4 filings are submitted by insiders (officers, directors, etc.) in publicly traded companies to disclose changes in personal ownership of company shares. We filter these down to include only purchases of common stock, which are considered more informative by analysts and market participants than sales. Insiders may sell for a variety of uninformative reasons, including diversification and raising cash for personal reasons, whereas they buy primarily because they believe company shares will rise in value. We further filter the filings down to just those that become public during business hours when the markets are open. This allows us to measure the intra-day response to each filing. Each filing has such features as:

- the insider type: director, officer, 10% owner, or other
- the total dollar value of direct purchases, discretized
- the total dollar value of indirect purchases, discretized
- various transaction codes

[4] `ftp://ftp.sec.gov/edgar/Feed/`

Algorithm 1 PRIMER

Input:

- *events* ▷ set of events or interventions
  - *event.features* ▷ each event has a set of descriptive features
  - *event.response* ▷ time series segment starting at the time of the event
- b ▷ beam size
- l ▷ max rule length
- T ▷ transfer function
- α ▷ alpha for calculating fit parameter confidence intervals

Output:

- *ruleList* ▷ ordered list of discovered rules

---

ruleList ← [ ] ▷ initialize empty rule list
repeat
  rule ← FindBestRule(*events*, b, l, T, α)
  ruleList.append(*rule*)
  coveredEvents ← events in *events* covered by *rule*
  events ← *events* \ *coveredEvents*
until *events* = ∅

function FindBestRule(*events*, b, l, T, α)
  emptyRule.conditions ← {} ▷ empty rule has no conditions, covers all events
  emptyRule.score ← ϵ ▷ a tiny score makes it rank higher than 0-score rules
  beam ← [ ] ▷ *beam* is an array of rule lists
    ▷ *beam*[2] holds a list of length-2 rules, etc.
  beam[0] ← [*emptyRule*] ▷ initialize length-0 rule list to hold the empty rule
  for $i = 1$ to l **do**
    rules ← all refinements to rules in *beam*[$i - 1$]
    for each *rule* ∈ *rules* **do**
      rule.score ← EvaluateRule(*rule*, *events*, T, α)
    end for
    beam[i] ← Best(*rules*, b) ▷ keep b best rules of length i
  end for
  return best rule in all of *beam*
end function

function EvaluateRule(*rule*, *events*, T, α)
  avgResponse ← average response of events covered by *rule*
  ci ← calculate parameter confidence intervals of T fitted to *avgResponse*
  if min(*ci*) ≤ 0 **then**
    score = 0 ▷ because a parameter is not significantly different from zero
  else
    p_{min} ← MinAUC(*ci*) ▷ for each parameter, choose the value from
      ▷ *ci* that minimizes the area under the curve
    score ← AUC(T, p_{min}) ▷ smallest confident area under the curve
  end if
  return *score*
end function

We also include the market capitalization of the company prior to the filing, discretized. Our final set of events has 136 binary features covering 158,983 events from years 2004 – 2014.

Time Series. For time series, we use stock returns of the company associated with each filing. We use tick data provided by Wharton Research Data Services.[5] We pre-process the data by converting it to 5-minute bars, creating a time series P of the last traded price within each 5-minute time period. The return time series is calculated as:

$$S_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{7}$$

The event response listed as an input in Algorithm 1 is the length-10 subsequence of S beginning with the first return affected by the event:

$$event.response = [S_{t_0}, S_{t_0+1}, \ldots, S_{t_0+8}, S_{t_0+9}] \tag{8}$$

In cases with insufficient trades to calculate each bar of the response, the event is removed from the dataset. The total response is the target variable to be maximized by PRIMER.

4.2 PRIMER Settings

We choose a transfer function based on prior knowledge that the financial markets respond positively to insider purchases, and that the effect of the news decays once the information is fully disseminated and acted upon by market participants. We choose the power law decay transfer function over the exponential decay function, because it more closely fits the average response of all events in the dataset.

The parameter b for beam size varies the extent of search. For the entire experiment, we use $b = 5$. This value has been commonly used in rule learning [3, 17]. Similarly low values of b have been shown to often out-perform large values when tested out-of-sample [18, 13].

The parameter l constrains the maximum rule length. We set $l = 10$ for the entire experiment, a value we believe is high enough to impose minimal constraint.

The parameter α determines the confidence intervals. A low value means more emphasis on the goodness-of-fit heuristic. If the value is too low, few rules will be discovered. We experiment with three values: $\alpha = \{0.1, 0.2, 0.3\}$.

4.3 Evaluation Method

Evaluation is based on the average impact (average sum of return bars) as a function of the percent of test data covered. First, test events are sorted in descending order according to predicted impact. Then the average actual impact

[5] `https://wrds-web.wharton.upenn.edu/wrds/`

is calculated using the top $x\%$ of events, for $x = 1 \ldots 100$. Models are preferred which best predict the highest impact events for each given coverage percentile.

All experiments are evaluated using 10-fold cross-validation, where each model is trained on 90% of the data and tested on the remaining 10%. This is performed ten times, once for each test partition, and the results are averaged. All models are evaluated using the same ten randomly partitioned folds. For some models, we optimize a hyper-parameter by further use of 10-fold cross validation in an inner-loop. For each fold of the outer-loop, once the hyper-parameter value is chosen, it is used to train a model on the full set of training data in the fold. In total, for a model with one hyper-parameter, we run the training 110 times on subsets of data.

4.4 Baselines for Comparison

We compare PRIMER with common regression models as well as state-of-the-art regression rule learning algorithms:

- **Ridge Regression.** For linear regression with L_2-norm regularization, we use the MATLAB function `ridge`, in conjunction with cross-validation to find the optimal value for the regularization parameter, out of values: $\lambda = \{10^{-4}, 10^{-3}, \ldots, 10^{5}, 10^{6}\}$.
- **LASSO.** For linear regression with L_1-norm regularization, we use the MATLAB function `lasso`, which has built-in cross-validation [7]. For λ, we set the function to use a geometric sequence of ten values, the largest just sufficient to produce a model with all zeros.
- **Support Vector Regression.** For linear L_2-regularized Support Vector Regression [11], we use `liblinear` [6]. We use cross-validation to find the optimal value for the cost parameter, out of values: $c = \{10^{-4}, 10^{-3}, \ldots, 10^{3}, 10^{4}\}$.
- **RegENDER.** Regression Ensemble of Decision Rules (REGENDER) is available as a Java library which integrates with the Weka data-mining environment [10]. We run it with the parameters recommended by the authors (gradient boosting, squared-error loss function, 200 rules, $\nu = 0.5$, with resampling set to 50% drawn without replacement) [5]. We also run it with the default parameters in Weka, which have three differences (simultaneous minimization, 100 rules, and $\nu = 1.0$).
- **M5'Rules.** We use the implementation of M5'Rules [12] included with Weka. We use the default parameters (minimum number of instances = 4, build regression tree = false, unpruned = false, use unsmoothed = false).
- **Dynamic Reduction to Classification.** Dynamic Reduction to Classification [15] also integrates with Weka. We tried numerous heuristics and found Laplace to be the best. We tried using a minimum coverage of 90% as recommended by the authors, but found that using 100% worked better for our dataset. In this case, we report results only for the best settings.
- **SeCoReg.** Separate-and-Conquer Regression (SeCoReg) [14] testing was inconclusive. We confirmed that the recommended parameters ($\alpha = 0.591$, minimum coverage of 90%) worked well on small datasets. However, our dataset is too large, and we did not wait for completion.

5 Results

Table 1 and Fig. 2 both show the test results. No model performed significantly better than all others over the entire set of coverage levels. RegENDER, using its authors' recommended parameters, has the best overall performance. However, several models have similar performance, including PRIMER.

	Average Response of x% of Top Predictions				
Algorithm	**1%**	**5%**	**10%**	**25%**	**50%**
PRIMER, $\alpha = 0.1$	0.0207	0.0142	0.0099	0.0045	0.0023
PRIMER, $\alpha = 0.2$	0.0203	0.0142	0.0100	0.0046	0.0023
PRIMER, $\alpha = 0.3$	0.0204	0.0141	**0.0101**	0.0046	0.0024
Ridge Regression	0.0202	0.0136	0.0097	0.0047	0.0024
LASSO	0.0202	0.0129	0.0098	0.0047	0.0024
Support Vector Regression	0.0161	0.0126	0.0087	0.0042	0.0023
RegENDER (authors)	**0.0214**	**0.0145**	0.0100	**0.0049**	0.0025
RegENDER (Weka)	0.0208	0.0143	0.0100	0.0049	**0.0025**
M5'Rules	0.0203	0.0141	0.0098	0.0049	0.0025
Dynamic Reduction	0.0198	0.0128	0.0084	0.0040	0.0022

Table 1. Average total response of the events covered by the top x% of predictions of each model (top 1%, top 5%, etc.)

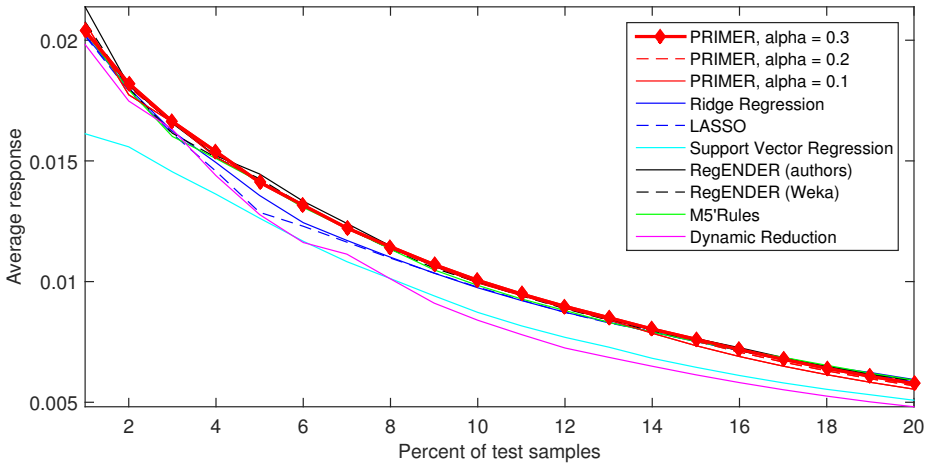

Fig. 2. Average total response of the events covered by the top x% of predictions. PRIMER is highlighted to show it is competitive with state-of-the-art models.

6 Discussion

PRIMER may not work with every dataset. Ideally, the average response of the full set of events closely fits the given transfer function, despite the diluted average impact. Because PRIMER uses greedy search, an average response sufficiently fitting the transfer function must be located at least within the first search iteration (length-1 rule). Otherwise, the program will exit with only the default rule. For best results, there must be a greedy path through the search tree to find high-impact branches. The decay patterns we have studied work well in this regard. The average of two exponential decay functions is another exponential decay function if the decay parameters are the same and only the scale parameters differ. The same is true for power law functions. Even when the decay parameters are not the same, the average of two decay functions often resembles another decay function closely enough for greedy search to work.

7 Conclusion

The results of our experiments show that, with respect to impact prediction ranking, PRIMER is competitive with state-of-the-art regression techniques in a large financial event study. One advantage of using PRIMER is in the interpretability of the model. PRIMER outputs a list of rules ordered by predicted impact. This allows the top rules to stand alone without exceptions or caveats. It also allows the algorithm to be terminated early, once a sufficient number of the highest-impact rules are found. This is useful in domains where resource constraints limit the potential number of interventions.

Rule learning systems require the user to specify multiple critical operational parameters. The optimal values are domain-dependent, yet domain experts have no intuitive way of selecting values, except through trial-and-error or by using default values. PRIMER has an advantage in this regard. The most critical user input to PRIMER is the transfer function, which is likely well-known to a domain expert. The confidence interval parameter, α, has some effect on the number of rules learned. Experimental results, however, show relative insensitivity to α.

Acknowledgments. This work is supported by Chevron USA, Inc. under the joint project Center for Interactive Smart Oilfield Technologies (CiSoft), at the University of Southern California.

We would also like to thank Dr. Frederik Janssen for providing support with the SeCoReg and Dynamic Reduction to Regression algorithms.

References

1. George EP Box and George C Tiao. Intervention analysis with applications to economic and environmental problems. *Journal of the American Statistical Association*, 70(349):70–79, 1975.

2. James Brookhouse and Fernando E.B. Otero. Discovering regression rules with ant colony optimization. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1005–1012, New York, NY, USA, 2015. ACM.
3. Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.
4. Thomas F Coleman and Yuying Li. A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM Journal on Optimization*, 6(4):1040–1058, 1996.
5. Krzysztof Dembczyński, Wojciech Kotłowski, and Roman Słowiński. Solving regression by learning an ensemble of decision rules. In *International Conference on Artificial Intelligence and Soft Computing, 2008*, volume 5097 of *Lecture Notes in Artificial Intelligence*, pages 533–544. Springer-Verlag, 2008.
6. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: a library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.
7. Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.
8. Johannes Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.
9. Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrač. *Foundations of Rule Learning*. Springer Science & Business Media, 2012.
10. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.
11. Chia-Hua Ho and Chih-Jen Lin. Large-scale linear support vector regression. *The Journal of Machine Learning Research*, 13(1):3323–3348, 2012.
12. Geoffrey Holmes, Mark Hall, and Eibe Frank. Generating rule sets from model trees. In *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence (AI-99)*, pages 1–12. Springer, 1999.
13. Frederik Janssen and Johannes Fürnkranz. A re-evaluation of the over-searching phenomenon in inductive rule learning. In *SDM*, pages 329–340. SIAM, 2009.
14. Frederik Janssen and Johannes Fürnkranz. Separate-and-conquer regression. *Proceedings of LWA 2010: Lernen, Wissen, Adaptivität, Kassel, Germany*, pages 81–89, 2010.
15. Frederik Janssen and Johannes Fürnkranz. Heuristic rule-based regression via dynamic reduction to classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1330, 2011.
16. Edward Xuejun Li and K Ramesh. Market reaction surrounding the filing of periodic SEC reports. *The Accounting Review*, 84(4):1171–1208, 2009.
17. Martin Možina, Janez Demšar, Jure Žabkar, and Ivan Bratko. *Why is rule learning optimistic and how to correct it*. Springer, 2006.
18. J Quinlan and R Cameron-Jones. Oversearching and layered search in empirical learning. *breast cancer*, 286:2–7, 1995.
19. G.A.F. Seber and C.J. Wild. *Nonlinear Regression*. John Wiley & Sons, New York, 1989.
20. Haifeng You and Xiao-jun Zhang. Financial reporting complexity and investor underreaction to 10-K information. *Review of Accounting Studies*, 14(4):559–586, 2009.