

Predicting Equity Returns from Securities Data

C. Apte and S.J. Hong

**Advances in Knowledge Discovery and Data Mining
AAAI Press, 1995**



PREDICTING EQUITY RETURNS FROM SECURITIES DATA WITH MINIMAL RULE GENERATION

Chidanand Apte and Se June Hong
T.J. Watson Research Center
IBM Research Division

Abstract

Our experiments with capital markets data suggest that the domain can be effectively modeled by classification rules induced from available historical data for the purpose of making gainful predictions for equity investments. New classification techniques developed at IBM Research, including minimal rule generation (R-MINI) and contextual feature analysis, seem robust enough for consistently extracting useful information from noisy domains such as financial markets. We will briefly introduce the rationale for our minimal rule generation technique, and the motivation for the use of contextual information in analyzing features. We will then describe our experience from several experiments with the S&P 500 data, illustrating the general methodology, and the results of correlations and simulated managed investment based on classification rules generated by R-MINI. We will sketch how the rules for classifications can be effectively used for numerical prediction, and eventually to an investment policy. Both the development of robust “minimal” classification rule generation, as well as its application to the financial markets, are part of an on-going study.

1.1 Introduction

There is currently a surge of interest in financial markets data mining. Large amounts of historical data is available for this domain in machine readable form. Analyses of this data for the purpose of abstracting and understanding market behavior, and using the abstractions for making predictions about future market movements, is being seriously

explored (AI on Wall Street 91; AI on Wall Street 93). Some firms have also deployed analytical data mining methods for actual investment portfolio management (Barr and Mani 93). We report here on our recent experiments with applying classification rule generation to S&P 500 data.

The R-MINI rule generation system (Hong 94a) can be used for generating “minimal” classification rules from tabular data sets where one of the columns is a “class” variable and the remaining columns are “independent” features. The data set is completely discretized (i.e., continuous valued features are discretized into a finite set of discrete values, categorical features are left untouched) by a feature discretization sub-system prior to rule generation. The feature discretization performs feature ranking (of both continuous valued as well as categorical features) as well as the conversion of continuous valued features into discretized features using an optimal cutting algorithm (Hong 94b). Once rule generation is completed, the R-MINI system can be used for classifying unseen data sets and measuring the performance using various error metrics.

The R-MINI rules are in Disjunctive Normal Form (DNF) (Indurkha and Weiss 91). There have been many approaches to generating DNF rules from data. These include (Michalski et al. 86; Clark and Niblett 89; Weiss and Indurkha 93b) which work in principle by iteratively forming one rule at a time to cover some examples from the training data which are removed from consideration before repeating the iteration. The other primary approach (Pagallo 89; Quinlan93) is decision tree based, i.e., a decision tree is generated from a given set of training examples, using some combination of covering and pruning strategies, where each leaf of the tree represents a classification rule.

While the R-MINI approach to generating classification rules is similar to the former, it differs from both approaches in its primary goal, which is to strive for a “minimal” rule set that is complete and consistent with the training data. Completeness implies that the rules cover all of the examples in the training data while consistency implies that the rules cover no counter-examples for their respective intended classes. Others too have argued for generating complete and consistent classification models before applying error minimizing pruning processes (Breiman et al. 84; Weiss and Kulikowski 91). The R-MINI system utilizes a logic minimization methodology to generate “minimal” complete and consistent rules. This technique was first developed for programmable logic array circuit minimization (Hong et al. 74) and is considered to be one of the best known logic circuit minimization techniques. The merits of striving for minimality have been well discussed (Blumer et al. 89; Rissanen 89). The principal hypothesis here is that a model that describes data is inversely proportional in its complexity to its accuracy. Thus, if two different models (in the same representation) both describe a particular data set, the less complex of the two will be more accurate in its description. Complexity is measured differently for different modeling techniques. For DNF rules, it would be total

number of rules and total number of tests in all the rules. A smaller description will tend to be better in its predictive accuracy, and this has been borne out in our extensive evaluations.

In the rest of this chapter, we will first describe highlights of the minimal rule generation and contextual feature analysis methodologies, followed by a detailed description of how these techniques were utilized in a specific application study, that of extracting DNF rules from capital markets data, and using these rules in a simulated investment portfolio management scheme.

1.2 Minimal Rule Generation

A data set with N features may be thought as a collection of discrete points (one per example) in an N -dimensional space. A classification rule is a hypercube in this space that contains one or more of these points. When there is more than one cube for a given class, all the cubes are Or-ed to provide a complete classification for the class. Within a cube the conditions for each part are And-ed, thereby giving the DNF representation for the overall classification model. The size of a cube indicates its generality, i.e., the larger the cube, the more vertices it contains, and potentially cover more example-points. R-MINI's minimality objective is first driven by the minimal number of cubes, and then the most general cubes. The most general cubes are prime cubes that cannot be further generalized without violating the consistency of that cube.

The minimality objective translates to finding a minimal number of prime cubes that cover all the example-points of a class and not cover any example-points of any counter-class. This objective is similar to many switching function minimization algorithms.

The R-MINI rule generation technique works with training data in which all features are completely pre-discretized. All numeric features are therefore discretized by a feature analysis and discretization sub-system prior to rule generation. Categorical features remain as they are. The rule generation technique is based upon a highly successful heuristic minimization technique that was used for minimizing large switching functions (MINI [Hong et al. 74]). Similar heuristic minimization techniques have been developed for a publicly available switching function minimization package, ESPRESSO (Brayton et al. 84).

We now summarize the salient steps of the R-MINI process, starting from a given set of data points. Let F be the set of initial cubes (initially, each cube corresponds to an example in the data set) for which the rules are being generated, FB be the set of initial cubes representing examples of all counter-classes, and DC is the (implicit) set of cubes representing space not covered by either F or FB .

The core heuristics used in the R-MINI system for achieving minimality consists of iterating (for a reasonable number of rounds) over two key sub-steps:

1. Generalization step, EXPAND, which takes each rule in the current set (initially each example is a rule) and opportunistically generalizes it to remove other rules that are subsumed. The sequence of operations in EXPAND are:
 - 2.1) Cube ordering: orders the cubes by a heuristic merit so that the cubes that are "hard" to combine with other cubes are to be processed first.
 - 2.2) Current cube expansion:
 - 2.2.1) Part ordering: decides the order of parts in which the single part expansion will proceed. Single part expansion takes a part of the given cube and turns 0s to 1s in the part, if doing so does not make the new cube contain any FB vertices. The purpose of this heuristic ordering is to effect a maximal coverage of other cubes when the cube is fully expanded.
 - 2.2.2) Expand each part of the cube in order. Here the cube is generalized recursively one part at a time until no more generalization is possible without violating the consistency.
 - 2.3) Shrinking other cubes: With the newly expanded cube, all other cubes are individually shrunk to the smallest size needed considering some of the vertices of the cube may be covered by the newly expanded cube. If a cube shrinks to a null size, it is removed from the list, thereby decreasing the number of cubes in the current F.
 - 2.4) Shrinking the current expanded cube: The new cube is shrunk to the smallest necessary size in view of the fact that some of the vertices it covers may also be covered by other cubes.
 - 2.5) Repeat steps 2.2)-2.4) for all surviving unexpanded cubes of F in the list in the order given by step 2.1).
2. Specialization/Reformulation, REDUCE, which takes each rule in the current set and specializes it to the most specific rule necessary to continue covering only the unique examples it covers. Cube specialization takes each cube in the order of another cube ordering heuristic designed to maximize the cube reductions, and shrinks it to the smallest size to contain the unique vertices it must cover outside the already covered vertices (by previous cubes) and the DC cover. The cube shrinking within the EXPAND shrinks cubes against the cube just expanded, whereas here the cube is shrunk against all the other cubes taken together. Redundant cubes disappear during this step.

The heuristics used for selecting what precise generalization or specialization to execute are chosen on a randomized basis. Therefore, each successive iteration may generalize and specialize in different directions. A iteration step may result either in an unchanged rule set (since completeness and consistency are enforced) or a smaller rule set. An indefinite iteration through this loop will result in monotonically decreasing classification rules that are consistent and complete at every stage in their minimization.

R-MINI's strength is in its heuristics that control the generalization and specialization. These heuristics essentially enable the generation of minimal number of rules that completely and consistently partition these example-points into unique regions. The minimization process may be viewed as a generalization process, in which regions that are adjacent to example-points but not themselves populated are available for expanding (generalizing) the rules. R-MINI's heuristics, based upon MINI's proven logic minimization heuristics, have also been empirically seen to be highly effective in generating accurate classification rule models.

This annealing style approach to rule generation (via iterative improvements) may be potentially indefinite. A limit is used that controls how long the system should keep iterating without observing a reduction. If no reduction takes place within this limit, we can stop the minimization process. In practice, we have observed that R-MINI satisfactorily converges the rule set as long as it is allowed to go through at least 5 iterations without a reduction, on most well known test data sets as well as on some of the specific real applications in which we have been using the system.

R-MINI has been applied to several real data sets, up to those with a few hundred features and tens of thousands of examples. Preliminary evaluations suggest that complete and consistent full cover rule sets that result from applying R-MINI are much smaller than similar rule sets generated by other known techniques. Initial benchmarking studies also suggest that the predictive accuracy of R-MINI's rule sets is competitive with the DNF rule sets generated by other well known methods. An in-depth detailed discussion of the rule generation component of R-MINI appears in (Hong 94a).

1.3 Contextual Feature Analysis

As mentioned in the previous section, R-MINI rule generation requires all features to be in categorical form, and hence the reason for discretizing all numerical features employing a feature analysis and discretization sub-system. There is also another important reason for applying this step prior to rule generation. Classification model generators will typically work only as well as the quality of the features from which they are trying to generate a model. Poor features will almost always result in weakly performing classification models.

Various approaches have been used to alleviate this problem. The decision tree based methods have relied on information theoretic measures (such as the “entropy” and “gini” functions) to determine the best feature to use at each node while expanding the decision tree (Breiman 84). This basic principle may be thought of as a 1-level lookahead algorithm that determines the best feature to use at a node based on how well the feature partitions the training examples into their respective classes. Variants of this method include 2-level and more lookahead methods as well as employing simple conjuncts of features (instead of single features) as decision tests for nodes.

One well known method used in a DNF rule generator is backtracking based local optimization (Weiss and Indurkha 93b). This method works in principle by attempting to constantly improve the performance of a rule while being constructed by swapping member tests (features) with new tests. Although this method appears more powerful than the decision tree methods, it may not perform well in the presence of extremely large number of features.

In most dynamic feature selection methods the discretization of numerical features is done in-process during model generation. This has an inherent weakness due to the serial nature of selecting the landmark values, or the cut points. Consider a problem where a numeric feature inherently has two or more optimum cut points, i.e. the intervals partitioned by them are meaningful in the problem domain. It is not likely that one of these points will be chosen in a serial process where the one “best” point is sought one at a time based on its ability alone to distinguish the classes. Our approach seeks multiple “optimal” cut points based on contextual demands for the separation of values. Even when it is not done serially, if the discretization considers only the given variable’s correlation to the class, the result becomes a weaker approximation to the context based consideration we employ in our approach.

The R-MINI system therefore employs a contextual feature analyzer that simultaneously ranks the features in terms of their classificatory power as well as determining the “optimal” number of cuts for each numerical feature for discretization so as to maximize that feature’s ability to discriminate. Features are ranked based upon merits that are computed for each of them. Merits are computed by taking for each example in a class a set of “best” counter examples, and accumulating a figure for each feature that is a function of the example-pair feature values. Dynamic programming is then used for producing optimum cuts (Aggarwal et al. 93) for the numeric variables by simultaneously looking at all numerical features and their value spans. The contextual merit computation algorithm works in principle as follows:

Contextual Merit and Span Generation Algorithm, CMSG

CM0) Initialize M, and let SPAN be an empty list of triplets.

```

CM1) Compute the numeric threshold vector T for numeric features.
CM2) For each  $e_i$  of class  $C_1$ , do
    CM2.0) Initialize  $M_i$ 
    CM2.1) Compute distances to all examples of  $C_2$  and let  $\log_2 N_2$ 
           (at least 1) of the nearest distance counter examples
           be the BESTSET.
    CM2.2) For each counter example,  $e_j$ , in the BESTSET, do
        CMSG2.2.0) Append the triplet,  $(i, j, 1/D_{ij}^2)$  to SPAN
        CM2.2.1) Compute  $M_{ij}$ :  $m_{k_{ij}} = d_{k_{ij}}/D_{ij}^2 \quad \forall k$ 
        CM2.2.2) Update  $M_i$ :  $M_i = M_i + M_{ij}$ 
    CM2.3) Update  $M$ :  $M = M + M_i$ 
CM3) Return M
CMSG4) Return (generic) SPAN

```

M is the contextual merit list for the entire feature set. The SPAN list obtained as a by product, is used to discretize all the numeric features. It contains $N_1 \times \log_2 N_2$ entries (order $N \log N$). For a numeric feature, X_k , we develop a specialization, $SPAN_k$, by substituting the actual values of X_k of the pair of examples given by the i and j indices, and removing those entries if the two values are same. Each entry in the $SPAN_k$ represents two points of a span in the X_k value line, along with a “merit” value to be scored when a cut point separates the end points of the span. It is now a simple matter of invoking an optimal interval covering algorithm to find the optimum c cut points for a given c . One has a choice of dynamic programming algorithms to achieve this with varying complexity.

This process is iteratively repeated until a reasonable level of convergence emerges in the merits and the cut values. In comparison to the tree based methods, the R-MINI contextual feature analyzer may be thought of as a full-level lookahead feature analyzer. It will not suffer from falling into false local minima because of its ability to analyze merits of features in a global context. An in-depth discussion of contextual feature analysis appears in (Hong 94b).

1.4 Experiments with S&P 500 Data

We undertook a study to determine the feasibility of applying DNF rule generation technology to managing equity investment portfolios. Initial results appear promising.

All our experiments have been conducted with S&P 500 data, for a contiguous period of 78 months. The data spans 774 securities (S&P deletes and adds new securities to its

500 index over time, so that the index continues to reflect the true market capitalization of large cap. firms). The data comprises of 40 variables for each month for each security. The type of information conveyed through these variables is both fundamental (company performance data) as well as technical (stock performance data). Some of the variables provide trend information (ranging from month-to-month trends to 5-yearly trends). With the exception of one variable, the industry sector identifier, which is categorical, all the rest are numerical.

Also available for each monthly stream of data for a security is the monthly total return for that security, where the variables' values are all at the beginning of a month while the monthly total return (stock change + dividends) is at the end of the month. From this 1-month return variable, one can compute 3-month, 6-month, as well as 12-month returns, for each security for each month. One can also compute the difference between these returns and the capitalization weighted mean as well as simple mean for each of the returns. Thus, if one can envision the basic 40 variable set as the "features", then we have available several ways to assign classes to each of the examples (a monthly stream of feature values for a security) by picking from one of the computed returns.

We have conducted a series of compute-intensive classification experiments with this data, using different ways to assign class labels as well as different ways to partition the data. We will focus in the rest of this chapter on one particular study, which attempts to generate rules for classifying examples based upon the differential between monthly return and simple mean of monthly returns for a given stream of data. The idea here is to use these rules to predict the differential for unseen data for the following year(s) and utilize the predictions in a portfolio management scheme for maximizing the investment returns. The portfolio management strategy strives to remain constantly above the average market return, and therefore the use of the differential as a class label. A positive differential merely implies a return that is higher than the market average. The actual return could be positive or negative.

1.4.1 Generating Classification Rules for Equity Returns

There are several issues at hand for determining how much data to choose for generating DNF classification rules for this domain. A routinely used approach would be to hide a portion of the data, ranging from 10-30%, and generate rules from the remaining "training" data, and evaluate their performance on the hidden "test" data. However, that approach is not adequate for the financial markets domain. There is a strong time-dependent behavior in the securities market which needs to be accounted for. An accepted practice is to use the "sliding window" approach, in which the data is laid out in temporal sequence, and the classification generation and performance evaluation experiments are repeatedly performed on successive sets of training and test data. This

Table 1.1
Number of S&P 500 data examples per class for years 1-4

| Class | Returns | Year 1 | Year 2 | Year 3 | Year 4 |
|-------|-----------------------|--------|--------|--------|--------|
| C0 | < -6 | 880 | 857 | 559 | 674 |
| C1 | $\geq -6 \ \& \ < -2$ | 1101 | 997 | 1188 | 936 |
| C2 | $\geq -2 \ \& \ < 2$ | 1347 | 1180 | 1533 | 1295 |
| C3 | $\geq 2 \ \& \ < 6$ | 874 | 883 | 977 | 1015 |
| C4 | ≥ 6 | 699 | 808 | 560 | 847 |

method can be used for determining whether the performance of a particular approach withstands the time-dependent variations that are encountered as one moves from set to set.

Adopting this latter methodology in one of our experiments, we chose to generate classification rules from a consecutive 12 months of data, and tested the performance of those rules on the following sets of 12 month streams. The idea here was to evaluate the rate of decline in the predictive power of classification rules as one moved forward in time. Once this rate is known, one can establish a policy of re-generating the rules once every “n” years from the immediate past data so as to continue holding up the predictive performance. Our data provided us with over 6 consecutive streams of 12 month data. We are conducting our experiments from the earliest point onwards, i.e., generate classification rules from the earliest available 12 month data (year 1), apply those rules to year 2, year 3, etc. until the performance becomes unacceptable, say at year “n”. Then re-generate classification rules from the 12-month data for year “n-1”, and repeat the process.

For the class label, we chose the differential between the next month’s 1-month total return and the S&P 500 average 1-month return. This label is essentially a numerical valued assignment. We further discretized this assignment by attempting to emulate typical security analysts’ categorization method for stocks, which would include the range “strongly performing”, “moderately performing”, “neutral”, “moderately under-performing”, and “strongly under-performing”. Based upon preliminary analysis of the data distribution, we chose to assign the cut-point -6, -2, +2, and +6. That is, all examples who had a class label value of 6% or more were put in one class (the “strongly performing” class), all examples with class label values of 2% or more and less than 6% were put in another class (the “moderately performing” class) and so on. Using this

class partitioning scheme, Table 1.1 illustrates how the first 4 years of data break up by class. Note that although 12 months worth of data for 500 securities should translate to about 6000 examples, the actuals vary for each time period because we chose to discard examples which had one or more missing values for the 40 features. The actual examples that we worked with are 4901 for year 1, 4725 for year 2, 4817 for year 3, and 4767 for year 4.

Table 1.2
Feature Merits and Cut Points for Year 1 Data

| Feature | Merit | Cut Points | Feature | Merit | Cut Points |
|------------|-------|------------|---------|-------|------------|
| ret1by1m | 322 | 5 | das | 58 | xx |
| pr2 | 230 | 4 | derat | 52 | xx |
| ret1mret1 | 229 | 4 | gprat | 48 | xx |
| ret1 | 228 | 4 | quality | 42 | xx |
| pr1 | 223 | 4 | ccr | 40 | xx |
| pr3 | 216 | 4 | ass | 40 | xx |
| pr6 | 201 | 4 | growth | 29 | xx |
| valueprice | 197 | 3 | cne | 21 | xx |
| veer | 152 | 1 | | | |
| cflowprice | 131 | 1 | | | |
| earntr | 122 | 2 | | | |
| epsprice | 121 | 2 | | | |
| bookprice | 115 | 1 | | | |
| eps5 | 112 | 2 | | | |
| aprice | 111 | 1 | | | |
| hsg | 110 | 2 | | | |
| per | 105 | 1 | | | |
| cap | 105 | 1 | | | |
| beta | 105 | 1 | | | |
| eps12price | 100 | 2 | | | |
| roe | 94 | 1 | | | |
| rinveq | 91 | 2 | | | |
| sects | 90 | 0 | | | |
| fund | 78 | 2 | | | |
| peg | 77 | 1 | | | |
| odl | 67 | 2 | | | |
| yld | 66 | 1 | | | |
| salshr | 61 | 4 | | | |
| epsvar | 61 | 1 | | | |

Before proceeding to apply R-MINI's feature analysis and discretization step, we tried to carefully adjust the features based upon discussions with the domain experts. As we pointed out in the previous section, the quality of features is of extreme importance in ensuring the quality of the generated classification model. Some of our adjustments to the raw data included the normalization of some features and the inclusion of additional trend indicating features. This preprocessing usually results in the transformation of input raw features into a new set of features, and cannot be done in the absence of domain experts. However, if this expertise is available, then utilizing it to refine the features is always very desirable. Once these transformations were made, we applied

R-MINI's feature discretization step to the data for Year 1, which corresponds to 4901 examples. The result of this step is the assignment of merits to all the features and the assignment of cut points to the numerical features. Table 1.2 illustrates the merit and cut assignment for this experiment. Note that features with a 0 value for cut-points indicates that the feature is categorical. Also, we chose the merit values to discard certain features from the rule generation step. These features appear the lower end of the table. Their cut points are not important, since they do not play a subsequent role in the classification experiments, and are therefore not indicated.

Rule 1:
 monthr12: NOT ($5.50 \leq X < 9.50$;)
 aprice: ($X < 43.19$;)
 beta: ($X < 1.10$;)
 epsprice: NOT ($0.05 \leq X < 0.06$;)
 eps5: ($3.98 \leq X$)
 peg: ($1.54 \leq X$)
 pr3: NOT ($0.93 \leq X < 1.01$; $1.07 \leq X < 1.17$;)
 valueprice: ($0.86 \leq X$)
 veer: ($-2.37 \leq X$)
 ret1mret1: ($-10.85 \leq X$)
 ret1bylm: ($1.03 \leq X$)
 Then \implies C0

Figure 1.1
 Examples of R-MINI Classification Rules Generated from Year 1 Data

Rule 481:
 beta: ($1.10 \leq X$)
 cap: ($X < 2743.50$;)
 epsvar: ($6.25 \leq X$)
 hsg: NOT ($4.89 \leq X < 9.45$;)
 peg: ($X < 1.54$;)
 pr1: ($X < 1.10$;)
 pr2: NOT ($1.06 \leq X < 1.14$;)
 pr3: NOT ($0.93 \leq X < 1.01$;)
 pr6: NOT ($0.91 \leq X < 1.02$;)
 rinveq: NOT ($5.63 \leq X < 10.64$;)
 valueprice: NOT ($0.70 \leq X < 0.86$;)
 veer: NOT ($-2.37 \leq X < 0.84$;)
 ret1mret1: NOT ($-10.85 \leq X < -4.05$;)
 ret1bylm: ($X < 1.03$; $1.06 \leq X < 1.06$;)
 Then \implies C4

Figure 1.2
 Examples of R-MINI Classification Rules Generated from Year 1 Data

Using the selected features and fully discretized data values, we then apply R-MINI's

rule generation step to the training data, which is now essentially a 5-class problem with 4901 examples and 30 features. Since R-MINI uses a randomization process in its minimization phase, we run R-MINI several times (typically 5-6) on the same data set, and go with the combination of the smallest rule sets that was generated from the multiple runs. As an example, the smallest rule set size from one particular run is 569. That is, 569 rules completely and consistently classified the 4901 training examples. Figures 1.2 and 1.1 illustrate just 2 of these rules, where the first rule, Rule 1, is for Class 0, which corresponds to “strongly under-performing” and the second rule, Rule 481, is for Class 4, which corresponds to “strongly performing”. Since R-MINI employs a randomized process in its generalization and minimization phases, different runs may potentially generate different minimal rule sets. Combining different minimal solutions allows us to simultaneously exploit the unique generalization nuances of each solution.

1.4.2 Rule-based Regression

To be able to precisely quantify the predictive performance, especially from an investment management point of view, it is necessary that the classification rules predict the actual return, and not the discretized class segments. We have developed metrics for assigning numeric predictions for the R-MINI classification rules. While primarily motivated by the current set of experiments, it is conceivable that this approach could be used in any domain where it is required to predict numerical values. In a sense, this metric extends our R-MINI classification system for applications in non-linear multi-variate regression.

To extend the classification model to a rule-based regression model, we compute additional metrics for the rules. Based upon the training examples and the rules generated from this training data, we compute for each rule three parameters; μ , the mean of all actual class values (in this case, the differential between 1-month total return and mean S&P 500 1-month total return) of training examples covered by that rule; σ , the standard deviation of these values; and N , the total number of training examples covered by that rule.

When a rule set of this nature is applied to hidden “test” or unseen data, each example in that set will potentially have zero or more rules that apply to that example. In the case that no rules apply to an unseen example, we can assign or predict a numerical value that suggests a default for the domain, based upon priors such as the normal expected mean for the class. In the case that one or more rules apply to an unseen example, we compute an average from the rule coverage metrics, and assign that value as the class label for that example. Two straightforward averaging approaches are the simple and weighted approach. In the simple averaging approach, we compute for each example the simple average of μ of all rules that cover it as its predicted value (assigning a prediction value that is the mean of the class feature values in the training set if no rules cover it).

A similar rule-based regression technique was employed in (Weiss and Indurkha 93). In the weighted average approach, we compute and assign a prediction of the weighted average of $\frac{\sqrt{N}}{\sigma}\mu$ of all rules that cover it. In general, weighted averaging usually leads to smoother correlations between predicted and actual values.

More sophisticated statistical averaging methods are also available within our system in addition to these. Typically, we try and determine the best averaging method to use based upon which does the best on the training example, and utilize that for making the predictions from the corresponding set of rules on unseen data.

1.4.3 Investment Portfolio Management with R-MINI Rules

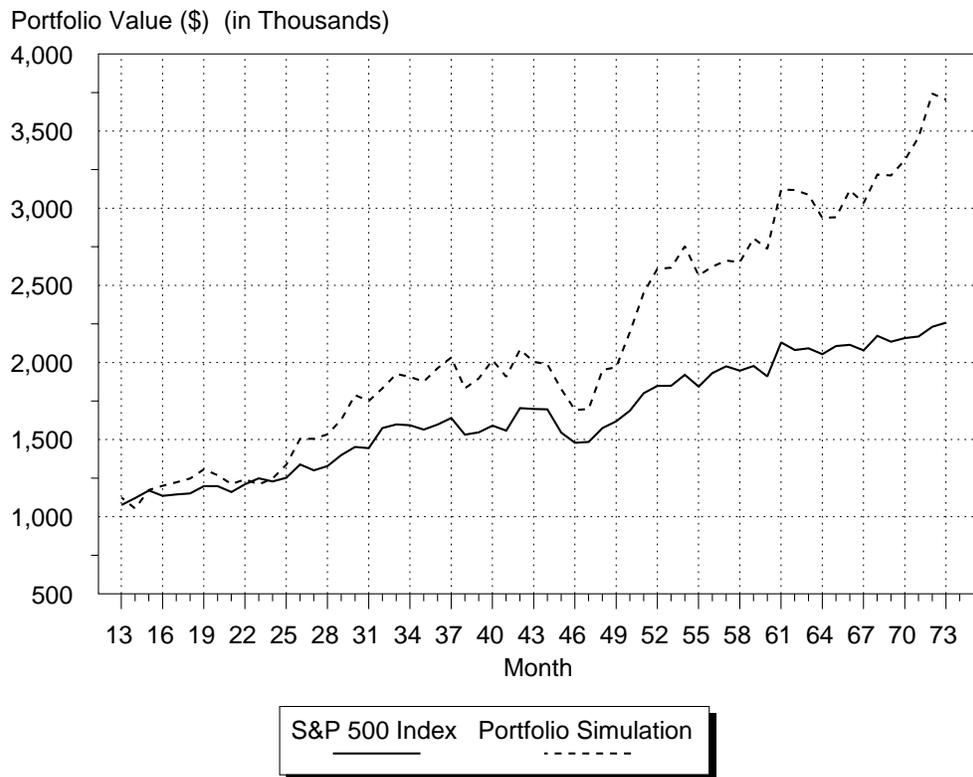


Figure 1.3
Comparing total returns of simulated S&P 500 index portfolio and a simulated rule-managed portfolio

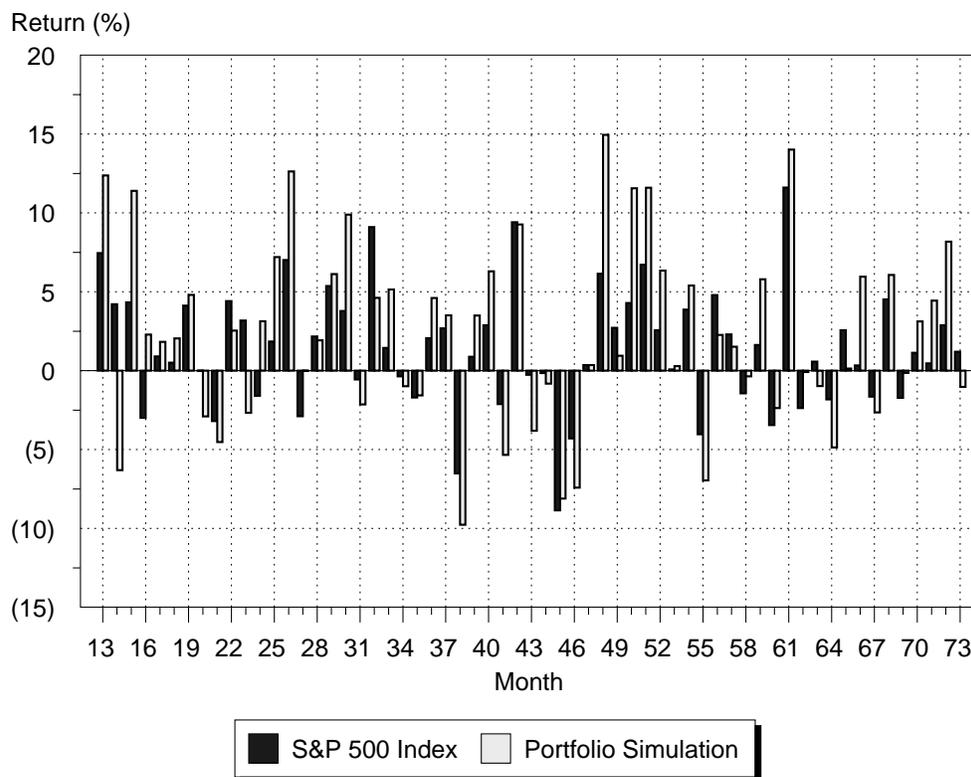


Figure 1.4
Comparing monthly returns of simulated S&P 500 index portfolio and a simulated rule-managed portfolio

To evaluate the performance of the generated rules, we applied them to subsequent year data. For example, rules generated from data for months 1-12, after some minimal pruning, are applied to the the following two years of data. We pruned out rules that cover 3 or less examples, since they are assumed to be covering the noise component in the data. For the remaining set of rules, we computed the μ , σ , and N for each rule, and then applied them to data for months 13-48.

For effective comparison of how these rules would perform if realistically used, we constructed a portfolio management scheme based upon these rules, and compared it to a simulated S&P 500 index fund performance. An index fund is passive in nature, and all that a S&P 500 index fund does is to constantly try and reflect the make up of the

S&P 500 index, by investing in those companies in proportion of their capitalization¹. In contrast to this passively managed approach, a portfolio management scheme based upon R-MINI rules will need to be highly active, since every month the rules will be making predictions which will need to be acted upon. What this active management policy does in principle is to start out with a investment which reflects the S&P 500 index fund, but then make trades every month based upon the rule predictions. One strategy that we have shown to be successful is as follows:

1. Generate rules (and use only those that cover > 3 examples).
2. Start with \$1 million S&P 500 index portfolio.
3. Execute monthly action at the end of month as follows:
 - (a) Update portfolio value for each equity position based upon the month's actual total return for that equity.
 - (b) Apply rules to month-end data for making predictions.
 - (c) Sort predictions in descending order.
 - (d) Sell all securities from sorted list whose predicted excess return is less than -6%, applying 0.5% transaction fee to every trade.
 - (e) Buy all securities from sorted list whose predicted excess return is greater than 6%, applying 0.5% transaction fee to every trade, in equal amounts.

The buy and sell cutoff points and thresholds are parameters that can be adjusted for controlling the behavior of the portfolio. For example, they can be adjusted to make the portfolio “aggressive” or “conservative”. Aggressive portfolios are characterized by high turnover and large positions in limited equities. Conservative portfolios hold relatively larger number of equities and trade less. The buy/sell cutoff points and thresholds for our investment portfolio simulator can be varied to achieve different behaviors on this spectrum.

For the above settings, Figures 1.3 and 1.4 illustrate how our active portfolio performs against a passive portfolio, on a monthly basis as well as a cumulative basis. Over a six year period, three sets of rules were generated from data for years 1, 3 and 5. The year 1 rules were used for predicting returns for years 2 and 3. The year 3 rules were used for predicting returns for years 4 and 5. The year 5 rules were used for predicting returns for the remaining period, year 6. Thus there is a 5 year period over which the combined rule predictions can be evaluated against a benchmark, such as the S&P 500 index.

¹The simulation of the passive index fund was performed on only that subset of the available S&P 500 that did not have missing information, i.e., the data in Table 1.1. This simulation may not correspond exactly to the real S&P 500 performance, although it is very close.

It can be observed that the simulated predictions-based portfolio returns a total of 270% over this period, compared to the S&P 500 index return of 110%. Compared to the 500 securities comprising the S&P 500 index, our simulated portfolio held an average of 30 securities over the 5 year period, and traded on average 5 securities per month. These characteristics can be adjusted by adjusting the buy/sell cutoff and threshold parameters. Many experiments with these parameters suggest that the simulated portfolio can be made more aggressive by carrying out much more trading and holding a lot fewer securities and consequently generating higher returns.

The monthly return plot provides insight at a more detailed level, highlighting the months when the rules actually generated predictions that were accurate enough to beat the S&P 500 index, and the months when the predictions were either weak or completely wrong. In addition to the confirmation that cumulatively the rule managed portfolio beats the S&P 500 index return, one can utilize the monthly returns to evaluate metrics such as the volatility and risk-vs-reward characteristics of the simulated rule-based portfolio. These evaluations suggest that the predictions do a reasonable job of beating the S&P 500 index without too much additional volatility or risk.

We make a few simplifying assumptions when constructing the portfolio management simulations. Issues such as the real-time granularity of the data availability, and the ability to transact stable trades, could potentially make our simulation strategy not practical. To be realistic and take all such factors into consideration, we hope to develop a more powerful portfolio management strategy that will compensate for additional real-market constraining factors such as these.

1.5 Discussion

We can draw a few key conclusions based upon our experiments. First, the S&P 500 data, as characterized by the features illustrated in Table 1.2, seem to provide adequate information for useful classification rule generation. Second, our techniques and methodology have the ability to extract this information from what is well known to be noise prone data.

The application of DNF classification rules in non-linear multi-variate regression applications is in itself another interesting direction to explore. The advantages of using DNF rules for these applications is clear; they provide a superior level of representation and interpretability in contrast to black-box style mathematical functions. Expert analysts can examine and understand these rules, and potentially even hand-edit them for improved performance.

We have demonstrated the predictive power of R-MINI's minimal rule generation phi-

losophy in conjunction with its contextual feature analysis. We have observed that the R-MINI generated rules, when embedded in an appropriate portfolio management scheme, can outstrip passive index funds in performance.

Although automatic classification methods such as ours provide an additional level of sophistication for culling out useful information from vast amounts of data, they are certainly not to be viewed as black box entities that can be applied in isolation. Every domain and its data has its own peculiarities, and careful introspective analysis of the data is an important adjunct to leveraging these methods maximally. We have attempted to do that in these experiments, trying to understand the data entities, determining whether new and derived features need to play a role, determining which features to discard, and finally, carefully evaluating the performance of the generated rules in conjunct with experts. The nature and longevity of the predictive power of the rules suggest the style in which we need to re-generate the rules, and the specific applications that can be built around the rule models. Our approach so far has been quite promising in the results it has delivered. We have begun to explore options for embedding our methodology into an actual deployment.

Bibliography

- Aggarwal A., Schieber B., Tokuyama T. 1993. Finding a Minimum Weight K-link Path in Graphs with Monge Property and Applications. Technical report, IBM Research Division, Yorktown Heights, New York.
- Artificial Intelligence Applications on Wall Street Proceedings. IEEE Computer Society, 1991.
- Artificial Intelligence Applications on Wall Street Proceedings. Software Engineering Press, 1993.
- Barr D. and Mani G. 1993. Neural Nets in Investment Management: Multiple Uses. In Artificial Intelligence Applications on Wall Street Proceedings, pages 81-87, 1993.
- Blumer A., Ehrenfeucht A., Haussler D., and Warmuth M. 1989. Learnability and the Vapnik-Chervonenkis Dimension. *JACM*, 36:929-965.
- Brayton R., Hachtel G., McMullen C., and Sangiovanni-Vincentelli A. 1984. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers.
- Breiman L., Friedman J., Olshen R., and Stone C. 1984. *Classification and Regression Trees*. Wadsworth, Monterrey, CA.
- Clark P. and Niblett T. The CN2 Induction Algorithm. *Machine Learning*, 3:261-283.

- Hong S.J., Cain R., and Ostapko D. 1974. MINI: A Heuristic Algorithm for Two-Level Logic Minimization. *IBM Journal of Research and Development*, 18(5):443-458.
- Hong S.J. 1994a. R-MINI: A Heuristic Algorithm for Generating Minimal Rules from Examples. *Proceedings of the 3rd Pacific Rim International Conference on Artificial Intelligence - PRICAI' 94*, pages 331-337.
- Hong 1994b. Use of Contextual Information for Feature Ranking and Discretization. Technical Report RC 19664, IBM Research Division, 1994.
- Indurkha N. and Weiss S. 1991. Iterative Rule Induction Methods. *Journal of Applied Intelligence*, 1:43-54.
- Michalski J, Mozetic I., Hong J. 1986. and Lavrac N. The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. *Proceedings of the AAAI-86*, pages 1041-1045.
- Pagallo G. 1989. Learning DNF by Decision Trees. *Proceedings of the Eleventh IJCAI*, pages 639-644.
- Quinlan J.R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.
- Rissanen J. 1989. Stochastic Complexity in Statistical Inquiry. *World Scientific Series in Computer Science*, 15.
- Weiss S. and Indurkha N. 1993a. Optimized Rule Induction. *IEEE EXPERT*, 8(6):61-69.
- Weiss S. and Indurkha N. 1993b. Rule-Based Regression. *Proceedings of IJCAI-93*.
- Weiss S. and Kulikowski C.A. 1991. *Computer Systems That Learn*. Morgan Kaufmann.