

Learning from data using RIPPER rules

Le Minh Nghia, G0600731E,
School of Computer Engineering

November 29, 2006

Abstract

This report provides an brief introduction to RIPPER rule learning algorithm and also illustrates how we can learn from data using machine learning. Rule-based learning is one of the nonmetric methods, such as decision tree, in classification; from the data, the concept or structure underlying is extracted as the set of rules which are very similar to human knowledge. RIPPER rule learning was studied in the report due to its advances over IREP algorithm and its comparability to the C4.5rules [4]. Rules generated from RIPPER on diabetes and US income data, using Weka software, were analyzed in details in order to obtain some direct knowledge about the topics.

1 Introduction

The presence ability of many devices (i.e. computers, sensor network) in collecting and storing information provides us huge amount of data in many areas. The need for advance machine learning techniques is not only for enabling automation processes but also due to our inability to understand these large data. To extract knowledge about a domain from its collected data, especially for classification purposes, many techniques have been proposed in various approaches (like parametric, non-parametric or nonmetric methods). However, the approach that is most convenient and similar to our human understanding is rule-based learning approach. Rule-based learning extracts the underlying concept or structure of data in the form of a set of rules, which find their analogy very much to decision trees. Nevertheless, rule sets are relatively easy for people to understand, and according to [4], rule learning systems outperform decision tree learners on many problems.

In the report, an experiment of learning from data of diabetes female patients was conducted using rule-based learning techniques (RIPPER algorithm). Besides, the technique of feature selection was also performed on the data to compare with the rules and point out some most informative features in forecasting the onset of diabetes cases. In addition, a US Census data on if an income of a person is larger than 50K USD was also examined in order to illustrate on how the rules obtained reflected our knowledge of a domain.

2 Rule-based learning methods

Rule learner and decision tree learning belong to the supervised learning scheme, in which a teacher provides a class label for each instance in a training set and seeks to construct a classifier model of the data, estimates its unknown parameters that can help reduce the error on the training set. The predicted description of a data can come in various forms, depend on which classifier is employed, such as Bayesian probability, decision border like hyperplane or some more intuitive ways like by a sequence of questions (i.e. decision tree) or a set of rules (i.e. if an instance satisfies these conditions, then it is classified as W1). It is apparent that decision tree and rules are quite connected since decision tree can be interpreted as a set of rules by traveling the tree in the depth-first-search manner and vice versa.

In fact, many of the techniques used in modern rule learners have been adapted from decision tree learning. Many decision tree learning systems use an *overfit-and-simplify* [4], or pruning technique rather than specifying some stop conditions for constructing the tree. Here the hypothesis is formed by first growing a complex tree which may overfit the data, and then simplifying or pruning the tree. One effective technique in pruning trees is *reduced error pruning (REP)*, which has been adapted for rules.

2.1 Reduced error pruning (REP)

In REP for rules, the training data is split into a *growing set* and a *pruning set*. First, an initial rule set is formed that overfits the growing set, using some heuristic method (i.e. information gain). The rule set is then simplified by some *pruning operators*. The operators would typically delete any *single* condition or *single* rule which yields the greatest reduction of error on the *pruning set*. Simplification ends when applying any pruning operator would increase error on the pruning set [4]. However, REP is computationally expensive for large datasets (i.e. $O(n^4)$, given sufficiently noisy data). In response to the inefficiency of REP, a novel learning algorithm called *incremental reduced error pruning (IREP)* was proposed by Furnkranz and Widmer.

2.2 Incremental reduced error pruning (IREP)

IREP is shown to be competitive to REP with respect to error rates, but much faster due to its improvement to REP in the pruning phase. In IREP, the criteria in the pruning phase is localized to a single rule rather than the complete rule set. The procedure of IREP for 2-class version, taken from [4] is shown in Algorithm 1. As shown in the pseudo code, IREP integrated tightly pruning process with a separate-and-conquer rule learning algorithm (i.e. `GrowRule()`); each rule is simplified immediately after learnt and instances covered by the rule will be deleted from the training set.

- **GrowRule** - in reference to [4], the implementation of `GrowRule` is a propositional version of FOIL [2]. It begins with an empty conjunctions of conditions and repeatedly adds the conditions that maximized FOIL's information gain criterion until the rule covers no negative examples from the growing dataset (`GrowPos`)

- **PruneRule** - after learning a rule, the rule is immediately pruned. A deletion of the sequence of conditions from the rule is considered and the one that maximizes the function (1) is chosen. This process is repeated until no deletion improves the value of v .

$$v(\text{Rule}, \text{PrunePos}, \text{PruneNeg}) = \frac{p + (N - n)}{P + N} \quad (1)$$

where P (respectively N) is the total number of example in PrunePos (PruneNeg) and p (n) is the number of example in PrunePos (PruneNeg) covered by *Rule*.

- **Separate-and-conquer** - a learning strategy concentrates on creating one rule at a time, each of them covering a part of the training examples. The examples covered by the last learned rule are removed from the training set (*separated*) before subsequent rules are learnt (before the remaining training examples are *conquered*). It is different from the *divide-and-conquer* strategy, employed by decision tree algorithms in tree construction phase, such as ID3, CART, C4.5 [3]. In such *divide-and-conquer* strategy, all rules are only discovered once the algorithm finishes constructing a decision tree.

Also, IREP supports missing attributes, numerical features and multiple classes. This makes it applicable to a wider range of benchmark problems.

Algorithm 1 The IREP algorithm

procedure IREP(Pos, Neg)

begin

Ruleset := \emptyset

while Pos $\neq \emptyset$ **do**

 grow and prune a new rule

 split(Pos, Neg) into (GrowPos, GrowNeg) and (PrunePos, PruneNeg)

 GrowRule - propositional version of FOIL

 Rule = GrowRule(GrowPos, GrowNeg)

 prune the rule immediately

 Rule = PruneRule(Rule, PrunePos, PruneNeg)

if the error rate of Rule on (PrunePos, PruneNeg) exceeds 50% **then**
 return Ruleset

else

 add Rule to Ruleset

 remove examples covered by Rule from (Pos, Neg)

endif

endwhile

return Ruleset

end

2.3 Repeated Incremental Pruning to Produce Error Reduction (RIPPER)

RIPPER was proposed in [4] as an improved version of IREP (called IREP*) based on Minimum Description Length heuristic. It also introduces additional steps after generating initial rules by IREP*, called Rule Optimization, of re-pruning each rule in the rule set so as to minimized the error of the *complete* rule set. Two modifications to IREP are

- Since IREP seemed to be unduly sensitive to the "small disjunct problem" due to its stop condition in growing phase, the new stop condition was proposed in [4] based on Minimum Description Length (MDL) heuristic. After each rule is added, the total *description length* D of the current rule set and the examples is computed. Growing phase will continues when there is still uncovered positive examples and description length $D_{new} < d(bits) + \min D_i$ found so far.
- An example in [4] illustrates for the problem with IREP's metric in pruning phase: the metric $\frac{p+N-n}{P+N}$ prefers a rule R_1 that covers $p_1 = 2000$ positive examples and $n_1 = 1000$ negative examples to a rule R_2 that covers $p_1 = 1000$ positive examples and $n_1 = 1$ negative examples; even R_2 is more predictive than R_1 . Hence, IREP's metric $v(Rule, PrunePos, PruneNeg)$ is replaced with

$$v^*(Rule, PrunePos, PruneNeg) = \frac{p - n}{p + n} \quad (2)$$

which seems to have more intuitively satisfying behavior.

Rule set R_1, R_2, \dots, R_k generated by this modified version of IREP (IREP*) still can be optimized so as to minimize error of the entire rule set on the pruning data. The optimization process considers each rule in the order in which they were learnt. For each rule, two alternative rules are constructed: the *replacement* rule R'_i and the *revision* rule. The decision of whether the final hypothesis should include the revised rules or the original rule is made using the MDL heuristic. Optimization can also be iterated by optimizing the rule set output by RIPPER. Hence, RIPPER k is used for the algorithm that repeatedly optimizes k times.

Algorithm 2 The RIPPER algorithm

```

procedure RIPPER(Pos, Neg)
begin
  Ruleset :=  $\emptyset$ 
  Ruleset := IREP*(Pos, Neg)
  Repeat k times
    Ruleset := MDLOptimize(RuleSet)
  Pos* := positive examples not covered by RuleSet
  RuleSet := RuleSet + IREP*(Pos*, Neg)
return Ruleset
end

```

3 Experiments

3.1 Data mining software - WEKA

Weka [5] is a collection of machine learning algorithms for data mining tasks, written in Java. The algorithms can either be applied directly to a dataset or called as APIs by another Java program. Interestingly, Weka provides a wide range of tools from data pre-processing, classification (including JRip- a Java implementation of RIPPER rule learner), regression, clustering, association rules, to visualization to meet various needs of users, as illustrated in the experiment later. The GUI comes in 4 modes, suited for simple as well as complex analysis tasks

- Console
- Explorer
- Experimenter
- KnowledgeFlow

Distributed as open source software under GNU License, Weka is also well-suited for developing new machine learning schemes.

3.2 Data

Diabetes data was taken from UCI Machine Learning database [1]. Instances were collected on female patients at least 21 years old of Pima Indian heritage. Each instance has 8 attributes and a class label indicating if a patient shows signs of diabetes according to World Health Organization criteria. The data contains 768 instances in which 268 instances of Positive class (i.e. tested positive for diabetes) and 500 instances of Negative class. 8 continuous features are listed in the table below. The data was origi-

Features	Explanation
pregnant	Number of times pregnant
glucose	Plasma glucose concentration a 2 hours in an glucose tolerance test
bloodPressure	Diastolic blood pressure (mm Hg)
skinThick	Triceps skin fold thickness (mm)
insulin	2-hour serum insulin (μ U/ml)
massIndex	Body mass index
pedigree	Diabetes pedigree function
age	Age (years)
class	Positive or Negative

Table 1: Features in Diabetes data

nally used in the research on ADAP learning algorithm to forecast the onset of diabetes mellitus. As mentioned in the description of the data, found in [1], the sensitivity and

specificity of the algorithm was 76% on 192 test instances after being trained using 576 training instances.

Data about individual's annual income, owned by US Census Bureau, was obtained from the dataset provided in Delve website. It contains 48842 instances, including those with missing attributes. Each instance comes with 14 attributes specifying from an individual's age, gender, race to his/her occupation, education, marital-status...etc. A class label indicates if the annual income (salary) is higher than 50K (" $>50K$ ") or less than 50K (" $\leq 50K$ ").

3.3 Results on Diabetes data

Rules learning by JRip from the diabetes dataset with no missing values and continuous features is shown in Result 2. The 10-cross validation was performed on the model and the error report is shown in Result 1. The model yields an accuracy $A(M)$ of 74.74% and error $L(M)$ of 25.26% which is quite acceptable, in comparison with the results obtained by ADAP algorithm. Due to the sensitivity $S_+(M)$ is 61.2% which is not very high, the model may not be reliable enough in detecting true positive cases. However, the model can still be useful in recognizing true negative cases as its specificity $S_-(M)$ is relatively high (82%). In this way, the rules can be still useful as a quick diagnostic, yet more accurate tests should be conducted if positive result is obtained on a patient. From the rules, we can also identify some important features which

Result 1 The result of JRip on diabetes dataset

Stratified cross-validation

Correctly Classified Instances	574	74.7396 %
Incorrectly Classified Instances	194	25.2604 %
Total Number of Instances	768	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	F-Measure	ROC Area	Class
0.612	0.18	0.646	0.628	0.721	Pos
0.82	0.388	0.798	0.809	0.721	Neg

=== Confusion Matrix ===

a	b	<-- classified as
164	104	a = Pos
90	410	b = Neg

help in determining if the case is positive or negative, based on their occurrences in the rules. These features are *glucose*, *massIndex*, *age*, *insulin*. The features reflect our knowledge of diabetes which is "a exceeding amount of glucose in bloodstream due to the lack of insulin". The rules also suggest a hypothesis that diabetes may likely occur

Result 2 The result of JRip on diabetes dataset

```
Rule 1: ( glucose >= 128) and ( glucose >= 155)
and ( massIndex >= 31.6) and ( insulin <= 249)
=> class=Pos (63.0/6.0)
Rule 2: ( glucose >= 128) and ( massIndex >= 30)
=> class=Pos (144.0/51.0)
Rule 3: ( age >= 30) and ( glucose >= 108) and
( massIndex >= 27.9) and ( skinThick <= 24)
and ( bloodPress <= 82)
=> class=Pos (28.0/6.0)
Rule 4: ( age >= 28) and ( insulin >= 144)
=> class=Pos (39.0/17.0)
Rule 5: otherwise => class=Neg (494.0/74.0)
```

for middle-age (i.e. 30 years old) and heavy person, relatively to a person's height (a person with body mass index ≥ 30 , indicated in the rules, is categorized as obesity). Notes that pregnancy shows no effect on the decision of the predictor.

To verify our knowledge derived from the rules, we conduct Feature Selection test using CfsSubsetEval evaluator and BestFirst search, provided in Weka on the data of 9 features (including class label). The result suggests four selected features: *glucose*, *massIndex*, *pedigree*, *age*, which is similar to the result given by the rules in 3 features *glucose*, *massIndex*, *age*. Nevertheless, the feature *pedigree* can verify our knowledge of diabetes as a possibly hereditary disease (i.e. Type 2 diabetes is due to the inheritance of susceptibility genes plus environmental factors such as obesity).

Figure 1 shows the 3D visualization of the diabetes data using the 3 key features found (*glucose*, *massIndex*, *age*) whose values were normalized. In fact, 2 classes shown in the visualization is already quite separable using these 3 features.

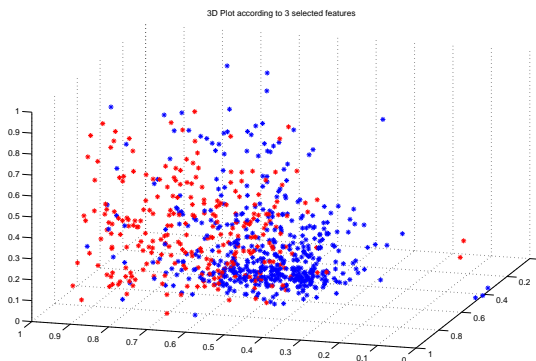


Figure 1: Visualization using 3 selected features

3.4 Results on US Income data

Rules learning by JRip from the US Income dataset with some missing values and all nominal features is shown in Result 3. As the number of instances in the original dataset is too large to be processed by Weka (due to JVM's memory limitation), the small dataset of 4884 instances randomly sampled from the dataset was used for learning. The cross validation on the model yields the accuracy of 84.01% and error of

Result 3 The result of JRip on US Income dataset

```
Rule 1: ( marital-status = Married-civ-spouse) and
( educational-num >= 12) and
( capital-gain >= 5178) => income =>50K (97.0/0.0)
Rule 2: ( marital-status = Married-civ-spouse) and
( educational-num >= 13) => income =>50K (540.0/184.0)
Rule 3: ( marital-status = Married-civ-spouse) and
( capital-gain >= 4386) => income =>50K (82.0/8.0)
Rule 4: ( marital-status = Married-civ-spouse) and
( educational-num >= 9) and (age >= 37) and
( hours-per-week >= 36) and
( occupation = Exec-managerial) and ( workclass = Private)
=> income =>50K (47.0/13.0)
Rule 5: ( marital-status = Married-civ-spouse) and
( educational-num >= 9) and (age >= 37) and
( capital-loss >= 1848) => income =>50K (37.0/5.0)
Rule 6: => income <=50K (4081.0/538.0)
```

15.99%. Some quite interesting knowledge of how to get high income are drawn from the rules obtained

- Get marriage
- Have relatively high education
- Or work at managerial or executive levels
- Wait until you get older (middle age)

4 Conclusion

In this report, we introduce the methodology of rule-based learning through several algorithms and mainly focus on RIPPER rule learner. By pruning rules on both locally (i.e. each rule) and globally (i.e. on the whole rule set), employing MDL heuristic and new metric to guide rule pruning, the results produced by RIPPER is very competitive with C4.5-rules. In addition, RIPPERk is much more efficient than C4.5-rules on noisy dataset as discussed in [4].

RIPPER implementation in Weka software (i.e. JRip) is then used to analyze the data of female patients regarding diabetes symptom and the data of US Census regarding if an individual's annual incomes is higher than 50K. The prediction model is presented as a set of rules. The experiment illustrates that rules is an effective way to learn about a domain based on collected data, due to its similarity to human knowledge. Also the rule set is relatively easy to be interpreted by human, especially when the number of rules is small.

References

- [1] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.
- [2] Michael Pazzani and Dennis Kibler. The utility of knowledge in inductive learning. *Machine Learning*, Volume 9, Number 1:57–94, June 1992.
- [3] Peter E. Hart Richard O. Duda and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.
- [4] William W.Cohen. Fast effective rule induction. *Machine Learning: Proceedings of the Twelfth International Conference (ML95)*, 1995.
- [5] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd edition, 2005.