Compression-based Induction and Genome Data

Rattikorn Hewett[1], John Leuchner[1], Choh Man Teng[1], Sean D. Mooney[2] and Teri E. Klein[2]
1 Institute for Human and Machine Cognition, University of West Florida
40 South Alcaniz, Pensacola, FL 32501, USA
2 Stanford Medical Informatics, School of Medicine, Stanford University
Medical School Office Building, X215, 251 Campus Drive, Stanford, CA 94305-5479
rhewett@ai.uwf.edu, jleuchner@ai.uwf.edu, cmteng@ai.uwf.edu, mooney@smi.stanford.edu, klein@smi.stanford.edu

Abstract

Our previous work developed SORCER, a learning system that induces a set of rules from a data set represented as a *second-order decision table*. Second-order decision tables are database relations in which rows have *sets* of atomic values as components. Using sets of values, which are interpreted as disjunctions, provides compact representations that facilitate efficient management and enhance comprehensibility. SORCER generates classifiers with a near minimum number of rows. The induction algorithm can be viewed as a table compression technique in which a table of training data is transformed into a second-order table with fewer rows by merging rows in ways that preserve consistency with the training data. In this paper we propose three new mechanisms in SORCER: (1) compression by removal of table columns, (2) inclusion of simple rules based on statistics, and (3) a method for partitioning continuous data into discrete clusters. We apply our approach to classify clinical phenotypes of a genetic collagenous disorder, Osteogenesis imperfecta, using a data set of point mutations in COLIA1 gene. Preliminary results show that on the average, over ten 10-fold cross validations, SORCER obtained an error estimate of 16.7 %, compared to 35.1 % obtained from the decision tree learner, C4.5.

Keywords
Machine learning, induction, data mining, molecular biology

1 Introduction

In machine learning, choice of representation models has impacts on both accuracy and users' ability to gain additional understanding from results. In many applications (e.g., scientific data understanding, medical diagnosis and treatment, financial analysis for loan approval) mere prediction or recommendation is not adequate; abstracted models must be comprehensible, and thus explainable, as well as accurate. Significant advances have been made on constructing models to achieve high accuracy [Fayad et al., 1996; Mitchell, 1997]. However, because of the well known tradeoff between accuracy and complexity of learning models, very "simple" models (e.g., decision tables) are rarely used in machine learning since they are expected to be less accurate than more complex models (e.g., decision trees, Bayes nets, neural nets).

Our research into data mining approaches using comprehensible models to abstract useful information from data has produced SORCER (Second-Order Relation Compression for Extraction of Rules) [Hewett and Leucher, 2001], a learning system that induces classification rules from a data set represented as *second-order decision tables*. Based on the framework developed in [Leuchner and Hewett, 1997], second-order decision tables are database relations in which tuples (rows) have *sets* of atomic values as components. Using sets of values, interpreted as disjunctions, provides compact representations that facilitate efficient management and enhance comprehensibility.

SORCER's induction algorithm can be viewed as decision table compression in which a table representing training data is transformed into a shorter table of more general rules by merging rows in ways that preserve consistency with original data. Unlike other systems, SORCER generates classifiers with a near minimum number of rows. This bias toward fewer rows, based on Occam's razor [Domingos, 1999], thus further facilitates comprehensibility.

This paper describes the extension of SORCER's capabilities by: (1) attribute selection, which compresses tables by removing columns, (2) inclusion of simple rules based on training set statistics, and (3) a method for partitioning continuous data into discrete clusters. We describe each of these approaches and their application to the problem of classifying lethal and non-lethal clinical phenotypes of Osteogenesis imperfecta, also known as brittle bone disease, from a data set of point mutations in COLIA1 genes. The application of SORCER to this problem illustrates our extended approach to rule induction and does not represent a complete analysis of the problem domain.

Since second-order table compression is a recent development, we give a brief overview in Section 2. Section 3 presents the extensions to SORCER. Section 4 describes the gene mutation data and the compression algorithms used in our experiments. Results are given in Sections 5. Section 6 discusses related work and presents our conclusions.

2 An Induction Algorithm

We first describe the major concepts used in our approach and then give an overview of the algorithm. More details on the algorithm and its theoretical framework are given in [Hewett and Leuchner, 2001; Leuchner and Hewett, 1997]. We use the terms table and row to refer to the second-order structures.

2.1 Preliminaries

We define the partial ordering *covers* on the set of all rows (over a fixed set of attributes) as component-wise set inclusion. That is, row s is covered by row r if $s(A) \subseteq r(A)$ for each attribute A. *Flat rows* are defined to be those whose components are either singletons or empty. (An empty component is an unknown value representing missing information.) The *flat extension of table R* is the set of all flat tuples that are covered by at least one row in table R. A table S is said to *subsume* relation R if the flat extension of R is a subset of the flat extension of S. Two relations are *equivalent* if each subsumes the other. The transformation that transforms a table R into table S is *equivalence-preserving* if R is equivalent to S.

A decision table, which represents or approximates a target function, which assigns classifications to conditions, is *consistent* if it associates at most one classification to any *total* condition (i.e., a condition with no empty components). A decision table is *complete* if it can classify all possible conditions. The transformation of a table R into table S is *consistency-preserving* if (1) every total condition classified by R is given the same classification(s) by S, and (2) any total condition which is covered by S but not by R is uniquely classified by S. Clearly, equivalence-preserving operations are consistency-preserving.

2.2 The Algorithm - Compression by Rows

Figure 1 gives a basic induction algorithm that starts with a flat table representing a training data set. The table is repeatedly transformed to produce a shorter table which subsumes the original table but is more general (covers more conditions). At any point, the table can be viewed as a (partial) approximation to an unknown target function that assigns classifications to conditions. The transformations correspond to a search through the hypothesis space of second-order tables, for a suitable approximating target function.

```
Input:  a (second-order) decision table $T$
output: a (second-order) decision table $R$ such that $R$
        is consistent with $T$ and the size of $R$ is mini-
        mal or near minimal within cost constraints.
  (1) Apply equivalence-preserving transformations,
      guided by heuristics, subject to cost constraints.
  (2) Infer additional rules or attribute values to com-
      ponents of individual rules.
  (3) Repeat Steps (1) and (2) until neither changes
      the relation.
  (4) Apply consistency-preserving transformations,
      guided by heuristics, subject to cost constraints.
  (5) Go to Step (1). Stop when no further transfor-
      mation has occurred within the cost constraints.
```

Fig. 1 Basic induction algorithm.

Examples of *equivalence-preserving* transformations include: *delete redundant rule* (remove a row from a table if it is subsumed by the other rows of the table), and *merge joinable* (replace a pair of rows, which agree on all attributes except one, by their *join* (component-wise union)).

An example of a *consistency-preserving* transformations is *merge consistent*: merge a pair of rules whose join does not introduce inconsistency. Such a pair is said to be *consistently joinable*. Merging such rows may add new conditions to the decision table but does not assign additional classifications to any previously covered condition.

Since many decision problems involving second-order tables (e.g., determining whether a table covers a row) are NP-hard [Hewett and Leuchner, 2001], resource constraints (e.g., number of iterations) may be applied for operations that are likely to be prohibitively expensive. Heuristics based on domain knowledge, such as ranking of attributes in order of discriminatory power, may also be applied to help select appropriate operation or rows.

The rule set produced by the algorithm may not be complete. For conditions not covered by the model, a rule in the classifier model is heuristically selected to provide a classification. The heuristics include a preference for rules that (1) cover the query on more attributes, (2) cover fewer conditions, and (3) give the most common classification appearing in the table.

3 Proposed Approaches

We now describe how the proposed mechanisms can be implemented in SORCER.

3.1 Compression by Columns

The bias to smaller tables, which are easier to understand and can be managed and applied more efficiently, assumes that a small number of rules can (though not always) accurately describe classification of conditions in the problem domain. Also, the possibility of overfitting tends to grow as the number of attributes increases. The problem can be severe when training data are small. For these reasons, we propose a method to reduce the number of table columns prior to induction.

SORCER can first perform a greedy search, using the training data set, to find a subset of the condition attributes to be used for classification. The resulting subset of attributes does not necessarily include all relevant attributes, and this step can be omitted when the number of attributes is small or when all attributes are known to have high predictive power.

In terms of a state space search [Mitchell, 1997], each subset of attributes constitutes a state. SORCER provides a *projection* operation that removes all but a specified subset of attributes from a table. The heuristic evaluation function for the search uses a *k*-fold cross validation, in which a data set is randomly partitioned into *k* approximately equally sized subsets (or folds or tests). The induction algorithm is executed *k* times; each time it is trained on the data outside one of the subsets, and the resulting classifier is tested on that subset. The estimated error is computed as the average error over the *k* test sets. The initial node of the greedy search can be either the set of all attributes or the empty set, and there is no goal state. Our search algorithm (not yet fully automated) terminates after a fixed number of node expansions does not yield a node with a lower estimated error than the current best estimate. This type of search process is sometimes referred to as "feature selection" and can be viewed as compression along the width of the table.

3.2 An Additional Type of Transformations

We propose another type of transformation, inclusion of *statistically determined* rules. Because the rules induced by SORCER are likely to be an imperfect approximation of the unknown target function, SORCER can add rules based on statistics or information theoretic measures (e.g., entropy [Mitchell, 1997]) computed from a training data set. This operation is called *add high probability rows (p)*, where *p* is a parameter specifying the minimum accuracy required. Currently, SORCER only considers simple rules with one condition attribute. For example, if $p = 0.90$ the rule ($A = a$) => ($Class = 0$) is added to the table if ($Class = 0$) for at least 90% of the training data set examples in which ($A = a$). Note that statistically determined rules may fail to preserve consistency, and although theoretically they can be applied at any point of the transformation process, SORCER currently only applies them to tables of flat rows.

Conceptually, equivalence-preserving transformations can be used for data compacting and to identify meaningful clusters of values, both of which aid comprehensibility. Consistency-preserving transformations can be applied to "generalize" a table to cover more conditions, which in turn can also simplify classification rules. Inclusion of statistically determined rules allows creation of simple rules (i.e., rules with small number of condition attributes) with a specified level of accuracy.

3.3 Discretization

The induction algorithm in SORCER requires discrete data. We have implemented an algorithm for discretizing data with continuous attribute values. The idea is to recursively merge adjacent clusters (initially each cluster is a single data point) until the number of clusters (or ranges) specified by the user is obtained. To determine which clusters are to be merged first, we use a heuristic function that is based on the width, distance, and uniformity of the cluster determined. The uniformity of a cluster is the proportion of the dominant class in the cluster, i.e., *n*/*s* where *n* is the number of points having the class that occurs most frequently in the cluster and *s* is the size of the cluster. Due to space limitation, we omit further details of our discretization method in this paper.

4 Application to Genome Data

We apply SORCER to the problem of classifying lethal and non-lethal clinical phenotypes of Osteogenesis Imperfecta (OI), also known as brittle bone disease, from a data set of point mutations in the COLIA1 genes. OI is a genetic disorder whose causes and severity are associated with mutations in the COLIA1 and COLIA2 genes. These genes encode for the peptides of type I collagen. The study of collagen genes to understand the structural features that determine a lethal phenotype is an ongoing research problem [Mooney et al., 2001; Hunter and Klein, 1993; Klein and Wong, 1992]. Our results are preliminary in the sense that they do not reflect the full scope of either the problem domain or the induction technique.

4.1 Data Set and Descriptions

Pos: position on the amino acid mutation in the collagen gene
HB-W: solute hydrogen bond wild type
HB-M: solute hydrogen bond mutant
S-HB-W: solute-solvent hydrogen bond wild type
S-HB-M: solute-solvent hydrogen bond mutant
STD-M: average standard deviation of RMS-M.
STD-W: average standard deviation of RMS-W.
RMS-M: average root mean squared deviation of structure between the simulated protein (with present mutation) and an idealized collagen structure
RMS-W: similar to RMS-M but without presence of mutation
SEQ: a sequence of the mutation and neighboring amino acids
Class: lethal (OI type II) or non-lethal (OI types I, III, IV)

Fig. 2. Attributes and descriptions of the data.

The raw data set reflects a series of amino acid mutations as described in Figure 2. S-HB-W (HB-W) are the number of hydrogen bonds that are collagen to solvent (collagen) in the unmutated protein and are present more than 80% of the time. S-HB-M (HB-M) are the same as S-HB-W (HB-W) except that the mutation is present. All attributes have numeric values except SEQ which is a string of 29 letters representing the mutation and neighboring amino acids. The amino acids considered in this analysis were the standard twenty plus hydroxyproline (O), a modified proline amino acid found in collagen, thus, a total of 21 possible amino acids.

4.2 Data Preprocessing

The sequence of the mutation and neighboring amino acids is likely to be important for disease classification but too noisy to be applied as a whole. We therefore partitioned the sequence (SEQ in Fig. 2) into 29 positions, one for each individual amino acid in the sequence. The 15th (middle) position is the mutated residue with 14 neighboring residues on its left and another 14 on its right. The data set contains a total of 39 attributes. We eliminated instances from the data set that contain no class information (disease type). We applied the discretization technique described in Section 3.3 to attributes with numerical values (the first nine in Fig. 2).

5 Experiments and Results

5.1 The Compression Algorithms Applied

The transformations described above can be applied in various combinations to create different compression algorithms. Algorithms used in our experiments are summarized in Figure 3. These algorithms can be specified easily in SORCER by generating script files of SORCER commands.

Alg.	Transformations and Operations
S1	Merge joinable
S2	Merge joinable Merge consistent
S3	Merge all close consistently joinable *k*
S4	Add high probability rows *p* Merge joinable, Merge consistent
S5	Add high probability rows *p* Merge all close consistently joinable *k*

Fig. 3 Various Compression Algorithms.

By applying only equivalence-preserving transformation, S1 gives us a classifier that simply remembers all seen cases. S1 gives SORCER's baseline accuracy when there is no generalization involved. S2 first merges locally joinable pairs, until no more such joins are possible, and then merges pairs of consistently joinable rows until no more consistent joining is possible. By applying merge joinable before merge consistent, S2 attempts to give priority to generalization according to the structure of the knowledge partially formed by equivalence-preserving transformation of a training data set. S3 merges "close" consistently joinable rows using *merge all close consistently joinable k* (as described below). S4 and S5 apply statistically determined rules to add simple rules that have probability of occurrence exceeding a specified threshold.

SORCER provides a method to merge consistently joinable rows in an order determined by a "distance" function. Given a limit *k*, rows whose distance is no more than *k* can be merged repeatedly until each row in the table is more than distance *l* from every other row. This operation is invoked by the command *merge close consistently joinable l*, where *l* specifies a distance limit. The operation invoked by *merge all close consistently joinable k*, where $0 < k < 1$, applies the preceding operation repeatedly with distance limits mk for $m = 1, 2, 3, \ldots$, until $mk \geq 1$. The definition and details of the distance function applied are described in [Hewett and Leuchner, 2002]. Basically, the distance function favors merging rows with large components that share many values. For the experiments in this paper, we applied S3 and S5 with $k = 0.5$, and S4 and S5 with the probability $p = 0.9$. In each fold in the cross validation, in order to obtain a consistent classifier, SORCER resolves inconsistent training data by retaining instances that occur more frequently. In addition, since the order of training data may affect the result of compression, SORCER shuffles the training data before applying the compression algorithm. Note that inconsistency is likely to occur as a result of attribute selection.

5.2 Experiments and Results

Because the string of amino acids are likely to be an important factor for classification and discretization can effect the resulting analysis, we divided our experiments into two independent sets, using only 29 attributes of amino acids in the first and all 39 attributes in the second. Thus, the first set of experiments uses data that do not require discretization.

SORCER selects attributes by taking the initial table representing the training data set, in this case 48 instances, and performing a breadth-first, greedy search to find a subset of attributes that gives the lowest estimated future error obtained from a 10-fold cross validation using the basic compression algorithm S2. The search starts from an empty set of attributes, repeatedly moves to a state containing a larger set of attributes, and stops when it reaches a state whose expansion does not give a better error estimate than the current best one. The number of columns of training data, excluding the class attribute, is reduced from 29 to 2 for classifiers in the first set of the experiments, and from 39 to 3 for the second. Next SORCER applies algorithms S1-S5 to the projected table to induce classifiers with a (near) minimal number of rows. For each algorithm, as suggested in [Kohavi, 1995], we ran 10-fold cross-validation 10 times since the estimated error for each cross-validation test is a random variable that depends on the random partitioning of the data.

Figure 4 shows results obtained by SORCER compared to C4.5 [Quinlan, 1993] (release 8 with and without pruning). For each error rate entry, the numbers before and after "±", respectively, represent the average and the standard deviation of the error rate (ratio of the number of misclassifications to the number of test examples expressed in a percentage) computed over 10 ten-fold cross validations. Note that for C4.5, size indicates the number of nodes in a tree, whereas for SORCER size is the number of rows in the classifier (compressed table).

Algorithms	Train Error	Test Error	Size
C4.5	5.4 ± 0.28	44.3 ± 6.76	104.2
C4.5 (pr.)	35.0 ± 3.03	37.8 ± 1.96	4.9
SORCER S1	0	26.9 ± 2.49	7.1
SORCER S2 *	0	21.2 ± 1.32	3.9
SORCER S3	0	21.0 ± 3.02	4.1
SORCER S4	0	26.0 ± 1.47	4.4
SORCER S5	0	26.5 ± 1.41	4.2

(a) Analysis on attributes representing amino acid sequence.

Algorithms	Train Error	Test Error	Size
C4.5	6.8 ± 0.32	40.4 ± 3.38	46.0
C4.5 (pr.)	33.7 ± 0.93	37.3 ± 1.87	1.7
C4.5 disc	5.0 ± 0.60	39.9 ± 3.40	77.6
C4.5 (pr.) disc	23.0 ± 1.65	35.1 ± 2.89	10.0
SORCER S1	0	23.5 ± 4.40	10.3
SORCER S2	0	23.5 ± 2.95	5.8
SORCER S3	0	22.1 ± 3.83	6.0
SORCER S4	0	18.1 ± 2.95	6.3
SORCER S5 *	0	16.7 ± 2.60	6.1

(b) Analysis on all attributes.

Fig. 4 Averages over ten 10-fold cross validations.

Figure 4 (a) gives results in the first set of experiments using only data on amino acid sequence. S2's performance is best for SORCER with error estimate close to the lowest, smallest variance and average size for the classifiers. SORCER produces no errors on the training data sets. Since the classifiers generated by Algorithms S1-S3 preserve consistency with the training data set, classifications have a zero error rate on consistent training data sets. The additional statistically determined rules in Algorithms S4-S5 turned out (though not necessary) to be consistent to the training data as well.

Figure 4 (b) gives results in the second set where all attributes are used. S5 obtained the best performance for SORCER. Results on train errors are the same as in part (a). Both parts agree on the same two positions of amino acid sequence as attributes that have greatest predictive power. Results from C4.5 were obtained from both the original (the first two lines of Fig. 4 (b)) and discretized, using our discretization algorithm, (the third and forth lines of Fig. 4 (b)) data sets.

6 Related Work and Conclusion

Our approach for reducing the number of columns (or attribute selection) is adapted from a wrapper technique [Kohavi, 1995]. SORCER's induction technique is most similar to that of R-MINI [Hong, 1994]. Unlike SORCER, R-MINI does not explicitly represent missing data. Details of SORCER and its related work are discussed in [Hewett and Leuchner, 2002]. In recent years, machine learning has been applied to molecular biology, including the classification of severity of OI disease (See a more in depth discussion of the problem domain in [Moony et al, 2001]). Our preliminary results show that attribute selection is important in analysis of data with high dimensionality such as genome data. The significantly lower error estimate obtained by SORCER compared to C4.5 may be due to the fact that the classifiers obtained by C4.5 were induced over all attributes producing accuracy that were not much different from random guessing with estimated error 35.4%. Future work could include experimentation with different discretization techniques, inclusion of prior knowledge in the compression process, and extending the scope of the problem domain.

Acknowledgement

T. Klein and S. Mooney gratefully acknowledge the National Institute of Arthritis and Musculoskeletal and Skin Diseases for financial support (AR47720-01: T. Klein).

References

Domingos, P., The role of Occam's Razor in Knowledge Discovery, *Data Mining and Knowledge Discovery*, 3: 409-425, 1999.

Fayad, U., G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, Eds., *Advances in Knowledge Discovery and Data Mining*: AAAI Press/The MIT Press, Menlo Park, California, 1996.

Hong, S., R-MINI: A Heuristic Algorithm for Generating Minimal Rules from Examples, in *Proceedings of the Third Pacific Rim International Conference on Artificial Intelligence*, PRICAI'94, 331-337, 1994.

Hewett, R. and J. Leuchner, Knowledge Discovery with Second-Order Relations to appear in *Knowledge and Information Systems*, 2002.

Hunter, L. and T. Klein, Finding Relevant Biomolecular Features, *Proc. ISMB*, AAAI Press, 190-197, 1993.

Klein, T. and E. Wong, Neural Networks Applied to the Collagenous Disease Osteogenesis Imperfecta, *Proc. Int. Conf. on System Science,* IEEE Press, Vol.1, 1992.

Kohavi, R., The Power of Decision Tables, in *Proc. of ECML*, 1995.

Leuchner, J. and R. Hewett, A Formal Framework for Large Decision Tables, *Proc.KRUSE*, 165-179, 1997.

Mitchell, T., *Machine Learning*, New York, NY: McGraw-Hill Companies, 1997.

Mooney, S.D., C.C. Huang, P.A. Kollman and T.E. Klein, Computed Free Energy Difference Between Point Mutations in a Collagen-like Peptide, *Biopolymers* 58:347-353, 2001.

Quinlan, J., *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1993.