Rule discovery

Włodzisław Duch,
Department of Informatics, Nicolaus Copernicus University, Poland,
School of Computer Engineering, Nanyang Technological University, Singapore,
wduch@is.umk.pl

Synonyms

Logical rule extraction; understanding data

Definition

Rule discovery or rule extraction from data are data mining techniques aimed at understanding data structures, providing comprehensible description instead of only black-box prediction. Rule-based systems should expose in a comprehensible way knowledge hidden in data, providing logical justification for drawing conclusions, showing possible inconsistencies, and avoiding unpredictable conclusions that black box predictors may generate in untypical situations. Sets of rules are useful if rules are not too numerous, comprehensible, and have sufficiently high accuracy. Rules are used to support decision making in classification (Classification, Machine Learning), regression(Regression, Statistics) and association tasks. Various forms of rules that allow expression of different types of knowledge are used: classical prepositional logic (C-rules), association rules (A-rules), fuzzy logic (F-rules), M-of-N or threshold rules (T-rules), similarity or prototype-based rules (P-rules). Algorithms for extraction of rules from data have been advanced in Statistics, Machine Learning, Computational Intelligence and Artificial Intelligence fields.

Characteristics

Types of Rules

Different types of rules are used to express different types of knowledge.

Classical logic rules (**C-Rules)** that have the form of *logical propositions* IF ... THEN provide the simplest and most comprehensible way of expressing knowledge. Arguments (conditions) and conclusion are logical (binary) functions that may take two values, true or false. Decision rules in classification or regression problems have this form, with the consequent part of the rule representing conclusion in situations characterized by some object X (usually a feature vector) for which certain conditions are satisfied:

$$\text{IF condition}_1(X) \text{ AND condition}_2(X) \ldots \text{ THEN conclusion}(X)$$

For example, a diagnostic rule derived from a benchmark dataset on breast cancer (Duch et al., 2004) shows in which conditions recurrence of breast cancer is expected:

IF number of involved nodes >1 AND degree of malignancy = 3 (highest) THEN recurrence ELSE no-recurrence

The accuracy of this rule is on par with any other classification system, each of the two conditions involves a threshold and a single feature (out of nine features that describe each case in the database), and contains rather trivial (for medical doctors') knowledge: recurrence of the breast cancer is expected if there are many involved nodes, and the cancer is highly malignant.The ELSE conclusion handles the default class, covering the space that has not been taken by explicit rules.

This particular dataset does not contain more information related to the recurrence/non-recurrence question. However, it may contain interesting correlations between features, correlations that characterize a cluster of interrelated attribute values. **Association rules** (A-rules; Piatetsky-Shapiro, 1991) represent such correlations as rules with implications:

$$\text{IF attribute}_1 \in S_1 \text{ AND attribute}_2 \in S_2 \ldots \text{ THEN attribute}_k \text{ in } S_k$$

where S_i is a subset of attribute values.

Advantages: prepositional C-rules give the simplest and most comprehensible description of the data; predicate functions that define conditions may use any type of attributes; it is easy to control complexity of data description and thus to avoid overfitting of the data, thereby ensuring good generalization.

Disadvantages: C-rules always partition the feature space into hyperboxes, therefore for continuous features, they provide a step-like approximation of decision borders, limiting accuracy in many cases. A small number of accurate C-rules is rather rare.

Decision trees are equivalent to hierarchical sets of C-rules, splitting first the whole feature space and then successively each subspace, reducing flexibility of rule-based knowledge representation. In general, the expressive power of C-rules is limited. Three other types of rules – *fuzzy*, *threshold* and *prototype-based* rules – offer more flexible decision borders while still retaining interpretability.

Fuzzy rules (F-rules)

Statements like "Old age and overweight and high blood pressure are risk factors for heart problems" are commonly used but knowledge contained in them cannot be easily converted into crisp logical rule. *Fuzzy logic rules* have the same form as propositional rules, except that their conditions are not binary predicate functions, but rather real valued functions estimating the degree to which a given condition is fulfilled or true. Predicate functions condition(X) are simply replaced by membership functions that characterize in a numerical way the membership of X in sets described by linguistic terms. For example "old_age(age(X))" estimates to which degree a variable age defined for X fits the definition of old age. This is expressed by conditions in form of "age of X is old", or simply by defining membership function *old_age*(*age*(X)) that increases from 0 for *age* < 50 to 1 for age > 80. The definition of such membership functions should match common usage, but is rather arbitrary.

The conclusion of a fuzzy rule may either be a typical fuzzy term (similar to the premise conditions), or some function of input variables (useful for approximation). The theory of fuzzy sets and fuzzy logic helps to draw conclusions from imprecise statements converting them into mathematical formulas. For example,

IF *old_age*(*age*(X)) AND *high_bp*(*blood_pressure*(X)) THEN *heart failure* of X is high

where X represents some description of a person. Conditions *old_age* and *high_bp* express membership degrees in [0,1] and have little to do with probabilities, while the conclusion is an implication of a discrete event and may be interpreted as high probability that this event occurs. In effect, imprecise natural language statements are converted into rules that use fuzzy concepts and soft implications.

Advantages: conclusions of F-rules change smoothly rather than being absent or present, and therefore approximate gradual changes, thereby avoiding pitfalls of crisp logic applied to real-valued measurements. The general form of F-rules captures imprecise knowledge contained in natural language statements and may help to add *a priori* knowledge defining initial structure of predictive systems that learn by parameter adaptation. Accurate statistical or other predictive systems may be more accurate in most cases, but F-rules may still be useful to provide linguistic comments summarizing and explaining the results.

Disadvantages: there is no unique way to define membership functions for linguistic variables (especially symbolic variables), or to define meaning of fuzzy operators, such as fuzzy implication. Real variables (such as *age* or *blood pressure*) are used by many classifiers directly providing accurate predictions without the need to create fuzzy variables. The complexity of the F-rule sets is frequently too high to understand what they really mean.

Threshold rules (T-rules)

C-rules or F-rules are not useful in situations when too many alternative conditions lead to the same conclusion. For example, if a sufficiently large number of dendritic inputs is activated, then the neuron will fire, which is a type of “majority votes for action” rule. Designating the number of total inputs by *N* and the active inputs by *M*, this rule may be expressed as:

IF M conditions out of N conditions are True THEN Conclusion is True

Such rules are also known as the **M of N rules**. The general form of T-rules assumes that each condition X_i is a real number in [0,1] associated with weight W_i measuring its importance, and conclusion follows if sufficient evidence is accumulated:

IF $\sum_{i=1}^{N} W_i X_i \geq \theta$ THEN Conclusion is True

For example, a majority vote may be expressed using binary weights and conditions with $\theta = 0.5$. An equivalent propositional logic or fuzzy rule would require $\binom{N}{M}$ terms. Decision

borders for such rules are hyperplanes in *N*-dimensional feature space. Sets of T-rules generate intersection of hyperplanes defining complex regions where conclusions are true. In this general form, T-rules include all C-rules (weights are 1 only for conditions used in the C-rule and zero otherwise, and the threshold is equal to the number of conditions).

Advantages: T-rules add a lot of flexibility to C-rules, represent all voting-like problems in natural way, and provide accurate classification whenever linear discrimination works well.

Disadvantages: linear mixing may not be the best way to aggregate evidence, and semantic interpretation of such rules may not be easy.

Prototype-based rules (P-rules)

Conclusions are frequently justified by recalling similar well-known cases, or prototypes. Brains estimate similarity in a way that may be hard to analyze using individual features, leading to "intuitive" decisions. Similarity and distance (or dissimilarity) usually can be used interchangeably. Given a case X and a prototype R a threshold P-rule has the form:

IF $D(X,R) \geq \theta$ THEN Conclusion is True

Similarity may also be used in a nearest neighbor rule. In the most general form, if X is similar to R, then conclusions should also be similar:

IFX ~ R THEN C(X) ~ C(R_k)

Frequently, this is simplified to: k = arg $\min_i$ D(X,R_i) therefore Conclude C_i.

P-rules have great expressive power, subsuming all other kinds of rules. C-rules may be obtained when distances are calculated using L_∞ norm (or Chebyshev norm) $D_\infty(\mathbf{X},\mathbf{R}) = \max_i |X_i - R_i|$, T-rules for voting are obtained using a zero reference vector with Manhattan distance, or in general case using cosine distance function $D_c(X,R) = \sum_{i=1}^{N} W_i X_i / |\mathbf{W}||\mathbf{X}|$ that gives hyperplane decision borders. F-rules are obtained from P-rules for additive distance functions with membership functions measuring similarity between feature values (Duch and Blachnik, 2004).

Similarity functions are related to dissimilarity, or distance functions, using differences between feature values. There is some freedom in choosing the precise functional relation;for example, it may be an exponential formula:

$$S(\mathbf{X},\mathbf{R}) = exp\left[-\sum_{i=1}^{N} W_i\, d(X_i - R_i)\right] = exp[-D(\mathbf{X},\mathbf{R})]$$

where function $d(x) = |x|$, or its normalized version, $d(x) = |x / |x_{max}-x_{min}|$ [[Please check: should "x" be in italics or not? Should "x" be a small letter or a capital letter? See the equation for S(X,R).]] is frequently used to measure distance (or dissimilarity) between feature values, but more sophisticated functions may be postulated. If D(X,R) is the square

of the Euclidean distance, then the similarity function S(X,R) becomes a product of Gaussian membership functions for each variable, frequently used to create fuzzy rules.

An example of P-rule is: if a new patient has symptoms sufficiently similar to a previously diagnosed patient, then apply the same diagnostic procedure. In the well-known benchmark data called Wisconsin breast cancer from the UCI repository, a single rule

IF $D(\mathbf{X}, R_{303}) < 62.7239$ THEN malignant ELSE benign

with Euclidean distance function gives 97.3% accuracy (sensitivity = 97.9% and specificity = 96.9%), where R_{303} denotes patient no. 303 with malignant cancer. The accuracy of this rule is not significantly worse (in a statistical sense) than that of any classifier on this data, and thus offers the simplest and most comprehensible description (Grąbczewski and Duch, 2002).

Advantages: P-rules can represent complex knowledge, are suitable for any type of features, even for complicated graph or hierarchical structures like networks without simple feature-based representation.

Disadvantages: distance or similarity functions may not have a natural interpretation, whereas propositional C-rules express classical logic assertions in a simpler way.

Finally, not all objects may be described by a simple attribute-value vector. Objects that have nested relational structures, such as chemical compounds, or sequential data in bioinformatics or natural language analysis require a more sophisticated approach. **First-order predicate calculus logic rules** (FOL rules) used in such cases are expressed as Prolog programs (Lavrac and Dzerosky, 1994).

Algorithms for extraction of logical rules from data

Many algorithms for extraction of various forms of rules from data have been developed (Duch et al. 2001, 2004). Although the philosophy behind all these approaches differs, their ultimate capability depends on the decision borders that they provide for classification. A natural category, such as a protein family, may have quite a complex shape in the feature space and thus may require several prototypes, each associated with different similarity functions, to describe it.

A good rule should cover many examples with high precision. There is a tradeoff between simplicity and accuracy of the set of rules. Simple but rough representations of data structures may be very useful, while optimal complexity of rules should satisfy the Bias-Variance Trade-Off. Confidence in rules may be regulated by another tradeoff, i.e., that between accuracy and the rejection rate. Leaving some parts of the data as unclassified allows for higher confidence in conclusions of rules that handle the remaining part.

Some of the methods that have been devised to generate sets of rules describing data structures are presented below.

Decision trees (Rokach and Maimon, 2009) represent rules in a hierarchical structure with each path/branch giving a single rule. The hierarchical structure of the path leads to many rules sharing the same initial conditions, which is often considered as a disadvantage of this type of rule extraction method (most of the rules share the same promises). Algorithms that simplify such rules by converting them into logical rules are known, for example, the C4 rules for the C4.5 decision trees (Quinlan 1993).

Machine learning (Mitchell, 1997) has focused on covering methods that try to create a hyperbox containing data from a single category. This includes a whole family of AQ covering algorithms that try to grow general rules starting from “seed examples” selected for each class. CN2 is a covering algorithm combining and extending the AQ algorithm using decision tree learning. RIPPER adds new features in a similar way as decision trees by creating conjunctive rule conditions, discarding examples already handled by existing rules. Version spaces (VS) work with symbolic inputs, formulating hypotheses about the data in the form of very general (single condition) and very specific (all features as conditions) conjunctive rules, and then specializing general hypotheses and generalizing the specific ones, until they match.

Neural networks are good classifiers that can be used in various ways to generate logical rules. Some algorithms are aimed at logical approximation of functions that neural networks have learned; other algorithms try to enforce rule-like behavior of networks by changing their cost functions or defining special network structures and neural functions. Separable Basis Function (SBF) networks use the product of membership functions in their nodes, and thus are creating sets of F-rules in their nodes that are aggregated using linear combinations, which is adding a threshold rule to the conclusions of F-rules. Although many neural algorithms for C-rule and F-rule extraction from data have been described in the literature, they are not quite easy to use, and software is not readily available.

Nearest neighbor methods, or more similarity-based methods, may be presented in a framework (Duch, 2000) that includes many methods that are useful for the generation of P-rules: neural techniques based on the Learning Vector Quantization (LVQ) algorithms, or Radial Basis Functions (RBF) methods, or various methods to select prototypes for k-nearest neighbor.

Fuzzy systems are frequently tuned by hand, while neuro-fuzzy approaches (Nauck et al., 1997) use neural techniques for automatic optimization. Although these systems are capable of generating high-quality solutions, rarely the sets of rules they find are sufficiently simple to be used for understanding of data.

Support Vector Machines (SVM) may also be very useful to generate rules in all forms (Diederich, 2008). Linear SVMs implement T-rules, and non-linear SVMs may find good prototypes for P-rules.

Toolsfor rule extraction

Many data mining packages, such as Weka, RapidMiner, Knime, Orange, and others provide many algorithms that search for rules. A list of such packages can be found at the Open Directory project (http://www.dmoz.org/Computers/Software/Databases/Data_Mining/).

Cross-references

Artificial Intelligence
Bias-Variance Trade-Off
Classification, Machine Learning
Confidence, Machine Learning
Decision Rule, Machine Learning
Entropy, Information Theory
Generalization Ability, Machine Learning
Gini Index
Induction, Logics
Information Gain
Knowledge Discovery, Machine Learning
Kolmogorov-Smirnov Distance
Machine Learning
Overfitting, Machine Learning
R, Data Analysis Tool
Regression, Statistics
Stability, Machine Learning
Support, Machine Learning
Training Set, Machine Learning
Weka, Machine Learning Tool

References

Diederich J (ed.) (2008) Rule Extraction from Support Vector Machines. Springer Studies in Computational Intelligence, Vol. 80.

Duch W (2000) Similarity based methods: a general framework for classification, approximation and association. Control and Cybernetics 29(4): 937-968.

Duch W, Setiono R, Zurada JM (2004) Computational intelligence methods for understanding of data. Proc. of the IEEE 92(5): 771- 805.

Duch W, Blachnik M (2004) Fuzzy rule-based systems derived from similarity to prototypes. Lecture Notes in Computer Science, Vol. 3316: 912-917.

Duch W, Adamczak R, Grąbczewski K (2001) A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks 12: 277-306.

Grąbczewski K, Duch W (2002) Heterogeneous forests of decision trees. Lecture Notes in Computer Science Vol. 2415: 504-509.

Mitchell T (1997) Machine Learning. New York: McGraw Hill.

Nauck D, Klawonn F, Kruse R (1997) Foundations of Neuro-Fuzzy Systems. Chichester: Wiley.

Piatetsky-Shapiro G (1991) Discovery, analysis, and presentation of strong rules. In G. Piatetsky-Shapiro G., Frawley WJ (eds) Knowledge Discovery in Databases, AAAI/MIT Press, Cambridge, MA.

Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA, USA.

Rokach L, Maimon O (2009) Data Mining with Decision Trees: Theory and Applications, World Scientific Publishing Company.

Cite as:
W. Duch, *Rule discovery*, in: W. Dubitzky, O. Wolkenhauer, K. Cho & H. Yokota (eds.),Encyclopedia of Systems Biology, DOI 10.1007/978-1-4419-9863-7,#Springer Science+Business Media, LLC 2013