IMPLEMENTING CLASSIFIER SYSTEMS IN THE SAS® SYSTEM

Leonard S. Lutomski
Center for Applied Artificial Intelligence
The American Institutes for Research

Abstract

Classifier systems are parallel, rule-based machine learning systems. They are one member of a broad class of adaptive systems that includes neural networks, economic systems, and ecologies. Such systems involve a large number of diverse units that interact both competitively and cooperatively and that change over time so as to adapt to their environment. Classifier systems have been used, with varying success, to model the induction of expert knowledge required to regulate gas pipeline transmission, discover scheduling heuristics, and model idealized animal populations. This paper is an introductory discussion of classifier systems. It briefly discusses their structure, inherent problems, and some alternative approaches to genetic learning. The SAS code for a simple classifier system that learns to traverse a finite state world is presented in an appendix.

The Structure of Classifier Systems

A classifier system is an architecture for machine learning that consists of four components: (1) a rule and message system, (2) genetic heuristics for rule discovery, (3) a credit allocation algorithm, and (4) an environmental interface [Robeson87a,88b]. A classifier system can be conceived of as an abstract organism that modifies its rule set or 'learns' in response to the payoffs it receives from the actions it takes in its environment.

The environmental interface consists of both a perceptor and an effector system. Rules take perceptual messages as input to produce either (a) effector messages that alter some aspect of the environment or (b) internal messages that affect the future sequence of rule activation. Effector messages can result in a positive or negative environmental payoff. Each rule has an associated strength that is a measure of its utility to the classifier system in responding to the environment. On the basis of environmental payoffs, rule strengths are adjusted by the credit allocation algorithm. The algorithm used by classifier systems is the bucket brigade algorithm [Holland85] -- a simulated service economy. The automatic discovery of new rules is governed by genetic algorithms -- heuristics that simulate genetic recombination and natural selection.

The rule and message system

A rule and message system is the backbone of many artificial intelligence programs [Nilsson86]. Such systems cleanly separate three computational components: (1) a global database or message list that represents the system's knowledge about the problem at hand, (2) a rule list that represents the system's general knowledge, and (3) a control system or inference engine for regulating the application of rules to messages. Production systems are the most well-known type of rule and message system. Production-like systems were first formally described by Emil Post [Post361] and are commensurate with other models of computation [Sipser88a,85].

In production systems like YAPS [Allen82], the message language is symbolic and has a reasonably complex syntax. In classifier systems, messages have virtually no syntax, being composed of fixed-length words from the alphabet {0,1}.

Rules have two parts: (1) an antecedent that specifies the set of conditions that must be fulfilled for rule firing, and (2) a consequent that specifies the set of actions to execute when the rule fires. In ordinary production systems, the antecedents of rules are fulfilled when a consistent mapping is found of facts in the global data base to the expressions constituting the antecedent. The rules are of arbitrary length, can have considerable internal structure, and are capable of expressing negation, and complex or- and and-conditions. In a simple classifier system, on the other hand, a rule is a word two times the length of a message. The first half of the rule is the antecedent and the second half is the consequent. The rules are words from the alphabet {0,1}, where 0 is referred to as the wildcard symbol.

The inference engine or control system has three essential components, (a) a matcher, (b) a filter, and (c) a rule executor. In production systems, matching is regulated by either the RETE or the TREAT algorithm [Nayak88]. These algorithms construct and maintain a discrimination net that encodes both rule antecedents and their fulfillment by the current contents of the message list. On each cycle, the matcher forms the set of all those rules whose antecedents are satisfied. The resulting list of matches is passed to the rule filter that selects one rule for firing. The rule selected is then passed to the executor.

In classifier systems, the rule and message system operates within a loop that iterates over a number of trials in the environment. The matcher compares all messages to all rule antecedents. Sorting and/or indexing of rules and messages are sufficient to handle most efficiency problems. When the wildcard occurs in the rule antecedent it matches either 0 or 1. Any rules that match a message are considered for firing. Filtering is seldom more sophisticated than setting a minimum threshold for match 'strength' -- a quantity usually equivalent to an 'effective bid' that reflects the strength of the matching rule, the specificity of the rule, and a noise factor. Specificity -- typically a measure derived from the number of wildcard symbols in the rule antecedent -- is included to give specific rules an advantage over general rules. The noise factor -- that makes rule firing a stochastic process -- is included to allow the occasional firing of rules with low strength, giving them an opportunity to compete with those of higher strength.

All the rules that pass the minimum threshold are fired, permitting a degree of parallelism that in theory gives classifier systems some of the features of neural networks. The message list is completely replaced by the new messages formed from the consequent sections of the firing rules and the messages they matched. When the wildcard occurs in a consequent, it functions as a 'pass through' marker. It is replaced, in the new message, by the symbol at the same location in the matched message.

genetic algorithms

The constraints on the expressive power of rules and messages in classifier systems are accepted in order to support the use of genetic algorithms. Genetic algorithms are search and discovery algorithms based on the operations of genetic recombination and natural selection. Given a set of rules and their performance ratings, genetic algorithms create a new rule set that preserves the best features of the old while providing enough recombination and variation to support the automated discovery of novel rules.

The representation of rules as words in a limited alphabet permits the use of simple transformative operators on rules. The operators typically employed are variants of crossover, inversion, and mutation. The relative rates for these operations are usually set globally. The basic form of the three genetic operators is shown in Figure 1.
Crossover: 00000000 -> 0000 000 0 -> 0000 111 0 -> 00001110 11111111 -> 1111 111 1 -> 1111 000 0 -> 11110001
Inversion: 00110011 -> 00 1110 11 -> 00 0011 11 -> 00011111
Mutation: 00000000 -> 0000 000 0 -> 0000 # 000 -> 00000000

The operators are applied to the current rule list to create a new rule population after a specified number of iterations over the problem set that constitutes the system's environment. The creation of the new population is governed by a heuristic that insures that the rules with higher strengths are more likely to contribute to the new population. Additional heuristics may be employed to insure that the diversity of the new population is maintained, so that it does not prematurely converge on a solution that is only locally optimal. The classifier system typically iterates through a number of rule creation phases, up to a set maximum or until some performance criterion is achieved.

the credit allocation algorithm

The standard algorithm for handling credit allocation in classifier systems is the bucket brigade algorithm. In this algorithm, each rule, on firing, pays a percentage of its strength to the - bid - to the rule that issued the message fulfilling its antecedent. Existence and bidding rates may also be levied to hinder the survival of rules that never fire or create messages matched by only their antecedents. In this way, rules that get a positive or negative environmental payoff as the result of issuing an effector message, pass that payoff back to the rules that set the stage for them. The defect of this algorithm is that rule chains of length N must be executed N times for any payoff to reach the first rule in the sequence. This makes the bucket brigade algorithm slow and sensitive to any disturbance in a long chain of allocation. A 3-rule payoff sequence is illustrated in Figure 2.

the environmental interface

Classifier system explicitly possesses a perceptor and effector system that operate on a modeled environment. Each symbol in an antecedent condition of a rule is conceived of as representing the status of a particular perceptor. The execution of effector messages requires code that interprets the bit-pattern of the message to generate an action affecting the environment. The interaction of the environmental interface in creating a classifier system is a potent design strategy as it allows one to use the system in more than one domain with an alteration of only the interface code.

Classifier System Problems & An Alternative Approach

Classifier systems have two major problems: (1) premature convergence of the rule population on a local maximum, such that all members of the rule population are slight variants of a single individual, and (2) the inefficiency of the bucket brigade algorithm.

A number of corrections to the problem of premature convergence have been proposed and are of two types: (1) the ARGOT approach [Sheffer87] -- that keeps changing its representation of the environment so that the classifier system never converges, and (2) approaches that promote speciation among rules by making the fitness of an individual a function of its similarity to the rest of the population, or by explicitly preserving diversity when creating a new population, etc. Techniques to promote speciation are well surveyed in [Goldberg89].
If together these structures become large enough to exceed SAS's maximum record size, it becomes tempting to try to represent one of them as a temporary file that is rewritten a number of times during the course of the program. The problem with this tack is that the rewriting has to occur within a single DATA step because SAS cannot otherwise conveniently handle an unspecified number of iterations over rules, messages, and matches. The path taken in our simple system is to fix the size of the rule list and let both the message list and the match list grow up to a set maximum after which further matches or messages are refused. This permits the expression of the classifier system as single, though complex, data step. Macro variable space may be used as an additional storage buffer at some performance cost.

The basic constraint on the code presented is that it fit into the six pages allowed for submitted papers and still leave room for explanatory text. Many of the design options that can be implemented in a classifier system have been stripped out for this reason and efficiency has been sacrificed for clarity. The range of options available are well presented in [Goldberg89] and Holland's article on future directions in classifier systems [Holland88b]. Those options include features like (a) optional starts from pre-generated rule files, (b) multiple antecedents in rules, (c) Gray coding for numerical applications, and (d) rule replacement algorithms that explicitly measure and work to preserve the genetic diversity of the population. The code presented in the Appendix is further restricted to operate with environments represented as finite state worlds. It comes with a sample environment file and provides a working, modifiable, and simple classifier system for initial experimentation. With the data file and the global parameter settings shown, the system comes to a near-optimal rule set within 15 generations.

Further Information

Accessible overviews of classifier systems are presented in [Holland85] and [Goldberg89]. The proceedings of the international conferences on genetic algorithms [Grefenstette87, Grefenstette88a] and the special issue of Machine Intelligence devoted to genetic algorithms [cf. Robertson88a] are the best sources of current information on classifier systems. The C source code for CES-C [Riolo88] can be obtained from Rick Riolo. The GA-List that is moderated by John Grefenstette can be subscribed to by sending a BITNET message to ga-list-requests @aic.nrl.navy.mil.

References

[Allen83]

[Goldberg89]

[Grefenstette87]

[Grefenstette88a]

[Grefenstette88b]

[Holland85]

[Holland86]

[Holland87]

[Nayak88]

[Nilsson80]

[Riolo88]

[Robertson88a]

[Robertson88b]

[Sahomaa85]

[Shafer87]

SAS® is a registered trademark of SAS Institute Inc., Cary, NC, USA.

The author may be contacted c/o The American Institute for Research, P. O. Box #1113, 1791 Arasradero Rd., Palo Alto, CA, 94302, 415-493-3550.
do over all rules;
*----- mutate;
IF &Rnum(PR)={#match * &Eff;} THEN DO;
SUBSTR(HRUL(BUL),XR4NT{#match}) = SUBSTR("OFF",XR4NT{#match});
END;
*----- flip rule buffers, reinitialize strength, etc.;
CSTR{RUL} = BUL(RUL);
FTOT{RUL} = 0;
END;
*----- loop over all rules;
DO RUL = 1 TO &rnum;
VRO{RUL} = CRUL{RUL};
S7BSTR{RUL{RUL},#match} = "G{MSG} : ";
END;
00 RU1 .. 1 TO &rnl.lll;
do over all rules;
*-----
EHIl;
loop over
EtID;
"'--------- Initiate;
CR ....
EIIO;
flip rule buffers, reinitialize strength, etc.;
CSTR{RUL} = BUL(RUL);
FTOT{RUL} = 0;
END;
*-----
IF &AHU{#seed} THEN DO;
DO MS6 = 1 TO &num;
FTOT{RU1} = 0;
MHO = 1;
SET PTR{#seed};
initialize <KectJrnalists
TIMEHALT = 0;
DO UNTIL (TP={#seed} OR FLG{#seed} OR POS{#seed});
loop over intermediate states until a terminal state is reached or failure;
DO UNTIL TYP={#seed} OR TIMEHALT;
*--------- cover effector messages;
IF &lthr THEN PUT
RUl} = &sifli;
VRO + 1;
*---------
EN!);
*--------- cover perceptual messages;
IF TYP={#seed} OR &rnum THEN DO;
VRO = 1;
*--------- try to use unfired rule;
DO WHILE (RUL <= #arm & VRO = 0);
RUL = 1;
*---------
[...]
VRO = 1;
*--------- try to use unfired rule;
DO WHILE (RUL <= #arm & VRO = 0);
*---------
[...]
VRO = 1;
RUN;
VAR=FTOT(RUL);
END;
IF RUL<INIT THEN RUL = Xrandint(1000);
VAR = Square((VAR-Xrandint(1000)));
SUBSTR(CUR(RUL),1,BLEN) = 
SUBSTR(CUR(RUL),2,BLEN);
SUBSTR(CUR(RUL),2,100000); 
FTOT(RUL)=1;
SUBSTR(CUR(RUL),1,BLEN)=Square(BLEN),1); 
SUBSTR(CUR(RUL),1,BLEN+BLEN)="I" ;
END;
END;
ELSE IF EFX = 0 THEN DO;
**---- convert message to output value;
IF SUBSTR(MS1,1) = SUBSTR(RUL,1,1) THEN OUT = '0';/* OUT: effector */
ELSE OUT = '1';
**-------- interpret & change environment accordingly;
TRM = RANINT(EFX); /* TRM: transtate determinant */
IF OUT = '0' THEN DO;
IF TRM = 0.50 THEN ID='L'; ELSE ID='R';
IF TRM < TRM THEN PTR = BSL; ADDR='L'; END;
ELSE DO;
PTR = BSR; ADDR='R'; END;
END;
ELSE DO;
IF TRM >= 0.50 THEN ID='L'; ELSE ID='R';
IF TRM > TRM THEN PTR = BSL; ADDR='L'; END;
ELSE DO;
PTR = BSR; ADDR='R'; END;
END; /*--------- log state transition data;
IF TRM == THEN PUT "off";
END OUT TO TPO TP1 TPR FOR PTR;
**-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;
SET WORLD POINT=POS;
CPT = Maximize(PAC,Square(OS));
NMD = 1; /*-------- allocate credit for state entry;
TRM, TRM =
IF RUL THEN DO MSG = 1 TO BLEN;
MSG(OS) = " "
END;