Constructive Induction of Features for Planning

Michael van Lent
Artificial Intelligence Laboratory
University of Michigan
1101 Beal Avenue
Ann Arbor, MI, 48109-2110
vanlent@umich.edu

Constructive induction techniques use constructors to combine existing features into new features. Usually the goal is to improve the accuracy and/or efficiency of classification. An alternate use of new features is to create representations which allow planning in more efficient state spaces. An inefficient state space may be too fine grained, requiring deep search for plans with many steps, may be too fragmented, requiring separate plans for similar cases, or may be unfocused, resulting in poorly directed search. Modifying the representation with constructive induction can improve the state space and overcome these inefficiencies. Additionally, since most learning systems depend on good domain features, constructive induction will compliment the action of other algorithms.

This abstract describes a system that uses constructive induction to generate new state features in the Tic-Tac-Toe (TTT) domain. The system generates features like win, block, and fork that are useful for planning in TTT. TTT has been chosen as an initial domain because it is simple, the features are well defined, and it is clear what domain knowledge has been added. Additionally, previous work on constructive induction in the TTT domain provides a starting point.

The CITRE system (Matheus & Rendell 1989) creates a decision tree with primitive TTT features (the contents of the board positions) and uses a binary *and* constructor to incrementally combine features selected to improve the decision tree. Domain knowledge filters out less promising features. CITRE minimally improves classification of board positions but cannot generate some types of planning features. Our system has fewer constraints and includes extensions that expand the space of constructible features. For example, n-ary conjunction (not binary) is used to combine existing features into more complex features and n-ary disjunction groups symmetrical versions of the same feature. Also, constructors are applied to all pairs of features, including a new "player to move" feature, without using domain knowledge as a filter.

The perfect "X win" feature is a disjunction of the 24 ways two X's in a row can appear in TTT. Each such row is represented by a conjunction of primitive features (e.g. *and* (pos11=X) (pos12=X) (to-move=X)). To construct new features, the system calculates the information gain of each existing feature. Due to the symmetry of TTT states, symmetrical features have equal information gains and the same parent primitive features and can be correctly grouped into disjunctions (e.g. *or* (pos11=X) (pos13=X) (pos31=X) (pos33=X)). Taking advantage of symmetrical features is one example of correcting a fragmented state space with constructive induction. The system then constructs more sets of features from the conjunction of elements of the disjunctive features (e.g. *and* (pos11=X) (pos12=X)). These sets of conjunctive features are again combined into symmetrical disjunctions by comparing information gain and parent features used. Conjunction and disjunction alternate, incrementally building complex features. This algorithm generates and selects features, such as win and block, which CITRE cannot generate. Win and block are each represented by 3 features covering the symmetries of the diagonal lines, the middle lines, and the outer lines. The fork and block-fork features are generated but criteria for selecting them need to be developed.

This constructive induction system creates useful features for planning in TTT. An immediate next step is to apply the system to more complex domains. Also, the present system inefficiently applies the conjunction constructor to all pairs of features. In the same way CITRE selects features to improve an existing decision tree, our system could use planning knowledge to direct its selection of features.

References

Matheus, C., and Rendell, L. 1989. Constructive induction on decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 645–650.