Hierarchical Judgement Composition: Revisiting the Structural Credit Assignment Problem

Joshua Jones, Ashok Goel
College of Computing
Georgia Institute of Technology
Atlanta, USA 30332
{jkj, goel}@cc.gatech.edu

Abstract

Many agents need to learn to operate in dynamic environments characterized by occasional but significant changes. It is advantageous for such agents to have the capability to selectively retain appropriate knowledge while modifying obsolete knowledge after the environmental conditions change. Furthermore, it may be advantageous for agents to recognize revisitation of previously experienced environmental conditions, and revert to a knowledge state previously learned under those conditions. Many current function approximation techniques, while powerful in their generality, do not allow for such retention due to the fact that they do not explicitly relate domain knowledge with value estimation. We describe a technique, called hierarchical judgement composition, that does specify domain knowledge in the form of predictions about future events, and associates it with the intermediate representations used by the mechanism for generating state abstractions. Preliminary experimental results in the domain of turn-based strategy game playing show promise with respect to the desired characteristics.

Introduction

There are many situations in which agents need to make judgements about the quality of actions or world states. In simple environments, with relatively few available actions and states, it may be possible to search ahead through possible outcomes using a minimax strategy, or to gradually propagate quality values from reached states back to previous states through TD-learning methods (Sutton 1988). However, in highly complex environments with a great variety of actions available at various times with mutually interacting outcomes, straightforward application of these strategies becomes impractical. In these cases it may be more tractable to determine state value as a function of state features.

When there is a lack of *a priori* knowledge about the value function that should be used, it is advantageous for the agent to have the capacity to learn how these judgements should be made. Many statistical approaches to this problem treat the judgement generation mechanism as a "black box". Under these methods, the learning procedure does not consider internal mechanism structure in the translation from perceived features to judgements, but instead operates over completely general-purpose representational structures. This type of representation for acquired knowledge can be problematic when some features of the environment are subject to occasional significant change, as they offer no principled way to retain knowledge that remains accurate while selectively modifying obsolete knowledge. Game playing is one domain where such environments are common. As various opponents will adopt distinct strategies, an opponent's style is an environmental condition that may persist for a group of games, only to be supplanted by the differing style of an alternative opponent at other times, and perhaps reverted to once again at some later time. An agent that can quickly react to these types of situations will be much more successful than one that cannot. Another example that highlights the utility of responsiveness to dynamic environments is related to the design of large learning agents with modular structure, or agents that must work cooperatively with other learning agents. In such cases, cooperating agents or external modules within the large agent are treated as parts of the environment. As these entities learn and change, they constitute dynamic aspects of the agent's environment. These examples demonstrate the potentially widespread applicability of a method that reacts well in dynamic environments by selectively retaining knowledge.

Hierarchical Judgement Composition

We hypothesize that the attribution of some domain-specific semantics to intermediate representations used to derive judgments of quality will enable appropriate selective knowledge retention when applied in dynamic environments. Attaching domain-relevant meaning to parts of the estimate generation mechanism, in a form interpretable as a prediction, allows for knowledge-targeted structural credit assignment when learning in response to error. However, if a flat combination of underlying primitive factors is used to generate the top level assessment of state quality, each factor must be individually reexamined when modification is to occur. This process becomes prohibitively computationally costly when the judgment is complex, with a large number of underlying factors.

One response to this problem is to introduce a hierarchical representational structure for knowledge used by the judgement generation mechanism, while retaining the association

of domain knowledge with the internals of the representation. Rather than directly producing a complex estimate by combining a large set of primitive features, small groups of low level features are used to produce intermediate level judgements. Intermediate level judgements and primitive features are then combined into small groups to form successively higher level estimates. If each of the intermediate judgments can be interpreted as a prediction, and thus directly evaluated for correctness against observable environmental conditions at future times, more efficient knowledge update is possible when error is detected.

In this study, we evaluated these ideas for a fairly small subset of a larger domain to insure that the concepts are workable and allow for significant learning, and to determine what gains, if any, are realized in terms of appropriate knowledge retention in a changing environment. We specifically postulate that attaching domain knowledge that allows for *predictive* interpretation of intermediate judgements to a hierarchically composed state value computation mechanism will allow for scalable learning with good knowledge retention characteristics.

Domain Specification

The domain chosen to test this hypothesis is a computer-based strategy game known as FreeCiv (http://www.freeciv.org). The FreeCiv implementation is an open source variant of a class of Civilization games with similar properties, and there are many differently-themed games (e.g. outer space variations) that have roughly the same characteristics as well. The aim in these games is to build an empire in a competitive environment. The major tasks in this endeavor are exploration of the randomly initialized game environment, resource allocation and development, and warfare that may at times be either offensive or defensive in nature. Winning the game is achieved most directly by destroying the civilizations of all opponents, but can also be achieved through more peaceful means by building a civilization with superior non-military characteristics, such as scientific development.

FreeCiv makes an ideal domain for experimentation because it is a very complex game with a state space far too large to apply traditional statistical learning methods in a straightforward manner. Because an agent that is intended to play the game well must be quite complex, and must handle a variety of different types of tasks and judgments, a modular design that encapsulates various portions of the agent is needed. However, there is a high degree of interdependence between the various subgoals that comprise the top level goal of winning a game of FreeCiv. This means that the judgments required for rational action within a given module must have some knowledge about the competence and likely behavior of other modules. A design using hierarchical judgement composition allows a module to automatically incorporate information about the changes occurring in other modules as learning progresses, representing only the level of detail significant to the judgement in question.

It is not necessary to understand the game completely for the purpose of understanding this study, but some specifics are in order. The game is played on a virtual map that is divided into a grid. At the outset of a game, few squares on this grid are visible to the player. As the game progresses, the player can explore and reveal more information about the squares constituting the game map. Each square in this grid can be characterized by the type of terrain, presence of any special resources, and proximity to a source of water such as a river. In addition, each square in the grid may contain some improvement constructed by a player, including the computer agent designed as a part of this study. One of the fundamental actions taken while playing FreeCiv is the construction of cities on this game map, an action that requires resources. In return for this expenditure, each city produces resources on subsequent turns that can then be used by the player for other purposes, including but not limited to the production of more cities. The level of this production is based on several factors, including the terrain and special resources surrounding the city's location on the map, the construction of various improvements in the squares surrounding the city, and the skill with which the city's operations are managed. As city placement decisions are pivotal to success in the game, and the construction of a city is an action which itself requires resources, an intelligent player must make reasoned choices about where to construct cities. One characterization of the factors that inform this decision is as follows:

1. What is the expected quality, in terms of resource production, of each of the currently known grid locations, were a city to be built there?
2. What is the expected benefit, in terms of the discovery of higher quality potential city locations, of continued exploration of the map?
3. What is the expected cost of deferring the construction of a city to wait for further exploration?

In this study, we have designed an agent that plays FreeCiv, called CivDrone. Here, we are particularly focused on the module responsible for making decisions about city placement, and have specifically examined judgement (1) above, as it involves the estimation of a value that can be reevaluated in a fairly straightforward manner as each game instance progresses, and it relies upon both features drawn from the game state at estimation time and features related to knowledge of external modules. The intent is for the state abstraction produced by the module studied here to be combined with information from other such modules and used by a higher-level reasoner to select actions.

Agent Knowledge Structure

In order to implement the learning strategy examined in this study, the estimate of city location quality is decomposed into some underlying estimators and a tree of relationships that specifies the structure of the top level abstraction, the estimate of city location quality. This decomposition, and the relations among the various estimators, form the judgement model used by the agent to produce estimates and select modifications when error is detected. The decomposition tree of the top level estimate is depicted in Figure 1.

This decomposition is a simplification of actual game dynamics, but is sufficient for the purposes of this study. The

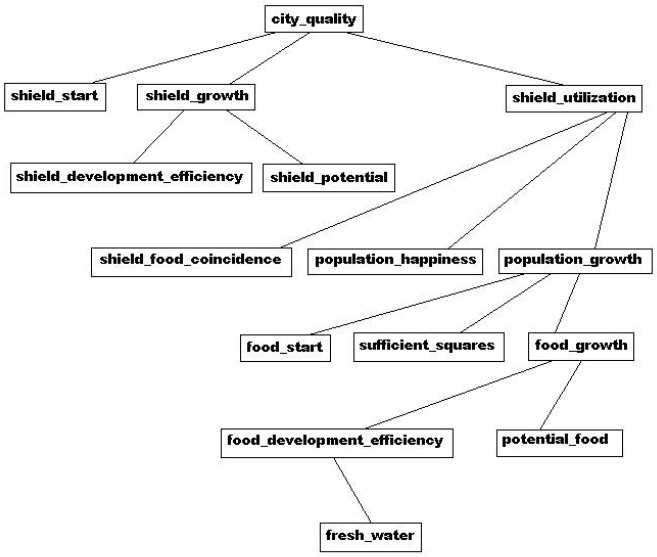

Figure 1: City Estimate Decomposition

top level estimate ranges over three possible values, called 'poor', 'sufficient' and 'good'. Each of the underlying estimators ranges over five possible values, adding 'very poor' and 'excellent' to the three values allowed for the high level estimate. The leaves in the decomposition tree represent either values that are drawn directly from game state or knowledge about portions of the agent considered external to the estimation subagent. The values drawn directly from game state are produced through a process of discretization that is applied to transform information from the game into one of the five allowed quality categories. This discretization process raises some issues of its own, but these problems are not addressed by this study. Here, hand tuning was used to determine the appropriate observation values to map to each quality category. Each higher level node in the decomposition tree represents a value that is derived by indexing a multidimensional table with the values of the node's children. One such table is stored for each non-leaf node, with a number of dimensions equal to the number of children. At the outset, each entry in these tables is initialized to 'satisfactory'. Through the process of repeated reevaluation and learning, these entries are adjusted to reflect observed environmental dynamics, increasing the accuracy of the estimates and thus minimizing the number of future reevaluations triggered.

Learning Technique

For this trial, the agent operates in an environment with no opponents, and always constructs exactly one city. On each turn of the game after city construction, the agent compares the previously computed prediction for resource production to the actual output of the city, taking into account the number of turns for which the city has been active. When the reevaluation process is triggered due to an accumulation of error between the anticipated evolving production value of a city and the actual production value of that city, the agent traverses the decomposition tree in an attempt to localize the origin of the error. In many cases, especially early in the training, there are multiple inaccuracies within the estimation tables. In the implementation used in this study, it is possible to modify multiple values in a single invocation of the reevaluation routine if multiple problems are found. After making modifications, the agent updates the estimate of the quality of the city's location based on the newly modified knowledge, to avoid falsely triggering the reevaluation process based on stale estimates.

```
procedure eval_node(n)

  problem = false

  for each child node c
    if prediction implied by judgement at c is inaccurate
      problem = true
      eval_node(c)
    end
  end

  if !problem
    adjust table value at n indexed by child judgement values
  end

end

eval_node(root)
```

Figure 2: Pseudo-code for judgement hierarchy adjustment.

The reevaluation process involves a traversal of a portion of the judgement decomposition tree. This process is illustrated in Figure 2.

First, each child of the top level node is examined. For each of these children, a relevant portion of the current game state is examined and compared to the prediction implied by the judgement that was made at the child node, again using a process of discretization like that discussed above. For this step, it is crucial that each node in the decomposition tree should represent a prediction about the evolution of some aspect of the game, excepting the leaf nodes that are drawn directly from game state at estimation time. These predictions are not verifiable at the time of estimation, but must be verifiable at future stages of the game. This characteristic allows each node in the judgment decomposition tree to be directly evaluated for correctness against the actual game state. Note that the percept-representing leaf nodes drawn from game state at estimation time are correct by definition, which is why they do not require this characteristic, and why eval_node is never called for such a leaf node. Leaf values representing knowledge about other parts of the agent *are* possibilities for modification. If the reevaluation process decides that one of these values is in error, it is modified directly, as there are no constituent values that could be responsible for the error.

If, for a given judgment in the tree undergoing reevaluation, no child node with an erroneous value can be located, the value stored in the table indexed by the values of the child nodes is adjusted, and the reevaluation process does

not visit those child nodes, reducing the number of node evaluations that must be performed. Otherwise, if one or more children with erroneous values have been located, each of these children is visited in turn, repeating the process described. This learning process does not guarantee convergence if the inputs at a given node are insufficient. This problem is the subject of ongoing research.

An example of the transformation of table values associated with the intermediate judgement nodes is shown in Figures 3 and 4.

Figure 3 shows the table at the 'food_growth' node as initialized to contain values all equal to 'satisfactory', before any learning has occurred. This node represents a judgement about the amount of additional food resource that will become available to the city over time as improvements are constructed within the city's radius. The child judgements that inform this estimate are 'potential_food', which represents the raw potential for increase in food resource, and 'food_development_efficiency', which represents the success of an external module in realizing the potential for increase in the resource. Figure 4 illustrates the result of learning in varying environments, where city improvements are sometimes constructed and sometimes disabled. The results match expectations in a general way, though some of the specific entries initially appear strange. The value of 'food_development_efficiency' gradually changes from 'very_poor' to 'good' after the agent's city improvement building routine is enabled. This explains why some entries between these two values have been touched, but there are few changes. It also explains why the column representing a 'food_development_efficiency' value of 'excellent' has not been affected. One strange result is that the areas of the table representing desirable quantities of potential food and good food potential realization remain 'satisfactory'. This may be because few examples of such situations were experienced by the agent while city improvement construction was enabled.

food_development_efficiency

potential_food	VP	P	S	G	E
VP	Satisfactory	Satisfactory	Satisfactory	Satisfactory	Satisfactory
P	Satisfactory	Satisfactory	Satisfactory	Satisfactory	Satisfactory
S	Satisfactory	Satisfactory	Satisfactory	Satisfactory	Satisfactory
G	Satisfactory	Satisfactory	Satisfactory	Satisfactory	Satisfactory
E	Satisfactory	Satisfactory	Satisfactory	Satisfactory	Satisfactory

VP = Very Poor, P = Poor, S = Satisfactory, G = Good, E = Excellent

Figure 3: food_growth judgement node table before learning

Preliminary Results Summary

As stated above, the goals of the initial experiment were twofold; first, to determine the overall efficacy of the proposed technique, and second, to determine whether appropriate knowledge retention would speed relearning after a change in the environment. The environmental condition

food_development_efficiency

potential_food	VP	P	S	G	E
VP	Very Poor	Satisfactory	Poor	Satisfactory	Satisfactory
P	Very Poor	Poor	Satisfactory	Good	Satisfactory
S	Very Poor	Poor	Satisfactory	Excellent	Satisfactory
G	Very Poor	Satisfactory	Satisfactory	Satisfactory	Satisfactory
E	Very Poor	Satisfactory	Satisfactory	Satisfactory	Satisfactory

VP = Very Poor, P = Poor, S = Satisfactory, G = Good, E = Excellent

Figure 4: food_growth judgement node table after learning in multiple scenarios

manipulated involves the construction of improvements in the region surrounding the city on the game map. In some trials, the agent was configured so that no city improvements would be built, while in other trials the agent was configured to build such improvements according to a standard scheme. The city quality estimation subagent was not directly informed of this change, but was instead expected to learn of the modification by observing the progress of games. In this way, the portions of the agent responsible for improvement construction are treated as part of the environment.

Results show that the method does enable learning, as evidenced by a decrease in the number of reevaluations required (signalling a decrease in the error encountered) in all runs for the two tested configurations. The average decrease in error rate across all configurations was 25.6%. In trials where the agent starts with initial knowledge from different environmental conditions, error immediately decreases to a point significantly below that recorded when the agent starts from scratch, with an average initial benefit of 8%. This immediate decrease in error is evidence of appropriate knowledge retention from previous training in an alternate environment.

In other trials, the agent reenters a previously encountered environment that is not reflected in the current state of its judgement function due to an alternate intervening environment configuration. In these cases, the agent quickly identifies the specific factor that has changed, in this case the construction of city improvements, alters that piece of knowledge, and in so doing has reverted to the knowledge state previously learned for the appropriate environmental state, rapidly attaining the same lowered error rate achieved during the previous learning. The initial benefit realized when reentering a previously visited environment is a 22% decrease in starting error rate.

These preliminary results begin to support the hypothesis that this technique does speed learning when revisiting a set of environmental conditions that have been encountered previously. This characteristic may be useful for agents in dynamic environments that enter repeated configurations, such as game players that recognize opponent play styles. More complete results cannot be presented here due to space constraints, but may be released in a subsequent paper.

Related Work

The credit assignment problem has been characterized as a core problem in learning (Minsky 1961). Samuel (Samuel 1957) first identified the problem in his work on checkers playing programs. The major difference in our work is the explicit association of predictive domain knowledge with intermediate judgements such that underlying judgements can be directly evaluated for correctness at time points after estimation.

Davis (Davis 1979) developed a scheme for interactive acquisition of expertise in the form of rules by credit assignment. In his scheme, the system provided a trace of its processing to a human expert, who assigned credit and identified the incorrect rules. The system also provided the syntax of rules (in the form of meta-rules) which were used by the expert to enter correct rules. In contrast, our work focuses on autonomous credit assignment, which requires a different approach.

Mitchell (Mitchell 1982) described a method for learning control heuristics for solving problems based in part on credit assignment. Our work differs in that the problems in Mitchell's work were generated by a human trainer who had access to the current state of the problem-solver's knowledge while in our work the agent is completely unsupervised. Further, the number of features that go into making a decision in our work is much larger than in Mitchell's.

Techniques for hierarchical reinforcement learning proposed by Dietterich (Dietterich 2000) also involve hierarchical task decomposition and state abstraction. In hierarchical RL, rewards are presumed to be returned from the environment upon direct execution of a primitive action. Reward values for delegated actions are equal to the accumulated reward received from primitive and further delegated subtasks during the execution of the subtask. In contrast to this bottom-up propagation of reward, JCT learning employs a selective top-down reward (punishment) propagation. This is because the choice output actions of subtrees are significant only insofar as they contribute to the perceived state, and thus output action, of the parent node. These actions are best seen as *internal* actions, that affect knowledge state, rather than affecting external world state. For this reason, the success or failure of an action chosen by a subtree can only be evaluated in light of the success or failure of the parent. This is a key difference in the type of tasks addressed by hierarchical RL vs. JCTs. State abstraction is also more aggressively attacked by JCTs; while Dietterich discusses directly discarding (ignoring) subsets of features at points in the hierarchy where it is clear that *no* information from the feature is ever required for the subtask, JCTs can summarize collections of features based on task-relevance.

McCallum's work on producing "utile state distinctions" (McCallum 1995) also deals with state abstraction and uses hierarchical representations. However, the specifics of the representation and the associated learning mechanisms are quite different from those described here. In particular, intermediate equivalence classes with predictive semantics are central to the JCT learning mechanism. The relative advantages and applicability of the methods is not yet known. More broadly, McCallum's focus is on learning useful equivalence classes with respect to a task, whereas the learning mechanisms described here are related to learning the contents of equivalence classes in a way that provides for retention and transfer in a dynamic environment. In the spirit of McCallum's work, current research seeks to extend the flexibility of the JCT representation by beginning to allow equivalence classes themselves to be learned as well.

Work on Predictive State Representations (PSRs) (Littman, Sutton, & Singh 2001) outlines rationale for using state representations that directly encode predictions about future events. A similar predictive notion of state appears in this work. However, the specifics of the PSR representation differ significantly from JCTs, as does the learning process. The predictive interpretation of state is used for structural credit assignment in the JCT learning process, while structural credit assignment is not a focus of work on PSR learning.

Conclusions

We have described a method for assigning credit based on the association of explicit, predictively interpretable domain knowledge with both intermediate and top level judgements in a hierarchically structured composition of the function computing state quality from state features. Preliminary results support the feasibility of the method, as well as the hypothesis that the use of hierarchical judgement decomposition with predictive domain semantics allows for appropriate knowledge retention and transfer across environments. In future work we plan to directly compare these results to the performance of alternative machine learning strategies applied to the same set of training tasks. We also intend to apply the technique on a larger scale and verify its generality and scalability.

References

Davis, R. 1979. Interactive transfer of expertise: Acquisition of new inference rules. *Artificial Intelligence* 12(2):121–157.

Dietterich, T. G. 2000. An overview of MAXQ hierarchical reinforcement learning. *Lecture Notes in Computer Science* 1864.

Littman, M.; Sutton, R.; and Singh, S. 2001. Predictive representations of state.

McCallum, A. 1995. *Reinforcement Learning with Selective Perception and Hidden State*. Ph.D. Dissertation, University of Rochester, Rochester, New York.

Minsky, M. 1961. Steps toward artificial intelligence. *Proceedings Institute of Radio Engineers* 49:8–30.

Mitchell, T. 1982. Generalization as search. *Artificial Intelligence* 18:203–226.

Samuel, A. 1957. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development* 3:210–229.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine Learning* 3:9–44.