Value Function Transfer for General Game Playing

Bikramjit Banerjee BANERJEE@CS.UTEXAS.EDU
Gregory Kuhlmann KUHLMANN@CS.UTEXAS.EDU
Peter Stone PSTONE@CS.UTEXAS.EDU
Department of Computer Sciences, The University of Texas at Austin

Abstract

We present value function transfer techniques for General Game Playing (GGP) by Reinforcement Learning. We focus on 2 player, alternate-move, complete information board games and use the GGP simulator and framework. Our approach is two-pronged: first we extract knowledge about crucial regions in the value-function space of any game in the genre. Then for each target game, we generate a smaller version of this game and extract symmetry information from the board setup. The combined knowledge of value function and symmetry allows us to achieve significant transfer via Reinforcement Learning, to larger board games using only a limited size of state-space by virtue of exploiting symmetry.

1. Introduction

We present two basic techniques for value function transfer in the General Game Playing (GGP) domain (Pell, 1993). This domain allows description of a wide range of games in a uniform language, called the Game Description Language (GDL). The challenge is to develop a player that can compete effectively in arbitrary games presented in the GDL format (Genesereth & Love, 2005). In this paper we focus on the problem of building a learning agent that can use knowledge gained from previous games to learn faster in new games in this framework.

We use afterstate Q-learning (Watkins & Dayan, 1992) as the basic learning mechanism. We use two approaches to knowledge transfer: first we learn the values of some hand generated structures (with intuitive meaning, see Figure 2) in the game-tree, that we call *features*. We use the Q-values of such structures learned in one game (tic-tac-toe) to initialize (some) Q-values in other games. This approach is not limited to board games, but only to 2-player, alternate move, complete information games. It belongs to the line of previous research using similar techniques in other domains (Liu & Stone, 2006; Taylor & Stone, 2005). The intention is to reuse portions of the value-function space that are independent of the game in our chosen domain. However, a common problem with value-function learning is that the size of the state space can be overwhelming. Our second approach is designed not only to address that, but also to enhance transfer. We automatically identify if a given target game is a board game, and if so identify what kind of symmetries are present in a *small version* of the target game. We then use this structural symmetry information to compact the value function space in the full-size target game by aliasing states that are identical by symmetry. Alternatively, we can look at this as exploiting symmetry to simultaneously update multiple states that are symmetric variations of each other, leading to faster learning.

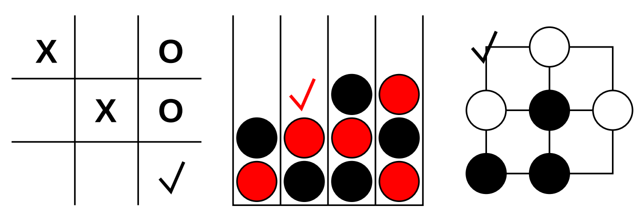

Figure 1. Illustration of feature F_1 in Tic-tac-toe, Connect-3 and CaptureGo. Note that in Tic-tac-toe, it also matches F_2

2. Feature extraction in value space

Our learning agent uses a private simulator based on the given game description, to lookahead a few levels of moves. This is not a strong assumption since novice human players routinely use this capability but they still need to learn from experience to become better players. We currently use four handcrafted *features* that serve to warn the player if a terminal (either favorable or not) state is in the vicinity of the current state in the game search tree. These four features are

F_1**:** Mark to win

F_2**:** Mark to block opponent from winning

F_3**:** Fail to block opponent from winning

Appearing in the *Proceedings of the ICML Workshop on Structural Knowledge Transfer for Machine Learning*, Pittsburgh, PA, 2006.

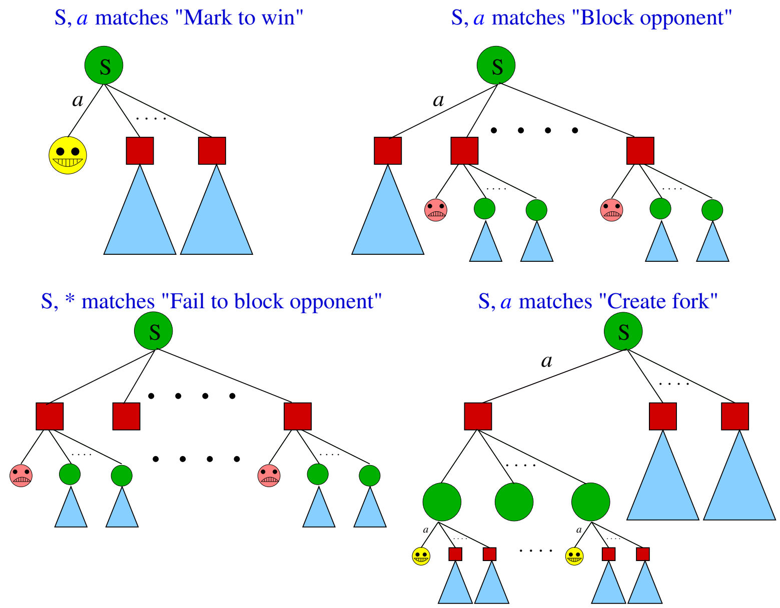

Figure 2. Illustration of features $F_1, \ldots, F_4$ in a game's search tree. Circular (green) nodes represent the learner's states, square (red) nodes are the opponent's states. Faces are terminal states and self-explanatory.

F_4**:** Create a fork

Feature F_1 is illustrated in Figure 1 in 3 different games. Figure 2 shows the common game tree structure of these features. None of the features require a deeper lookup than 2 moves by the learner.

Note that the features reveal a hierarchical nature, e.g., features F_2 and F_3 involve feature F_1 at the opponent level (in terms of the opponent's win), and similarly F_4 involves F_3. Once the training episodes are complete in the source game, we extract feature information from the acquired value-function space. This involves matching each state from this space against each of these features using the simulator for lookahead. If a state s matches a feature, we identify the candidate action a and note the value $Q(s, a)$ against that feature. For feature 3, all available actions of the learner are candidate actions, so we note their average Q-value. The value of a feature F_i is then calculated as

$$val(F_i) = \text{ avg } \{Q(s,a)|(s,a) \text{ matches } F_i\}$$

It is possible that an (s, a) matches multiple features (e.g., in Figure 1, left) and so each state can potentially contribute to multiple features. After the feature values have been computed, we use them to initialize $Q(s, a)$ in the target game for each (s, a) that matches F_i, $i = 1 \ldots 4$. Here again, an (s, a) may match multiple features, and we initialize as

$$Q_{init}(s,a) = \max_i \{val(F_i)|(s,a) \text{ matches } F_i\}$$

The states that remain uninitialized after this process are initialized to the default value. The idea behind this transfer mechanism is to save the cost of a few value-backup steps near terminal states (i.e., when the states gain predictive potential) and thus guide exploration to focus more in the regions where foresight is not usually available. In this way, our transfer learner behaves more like human learners.

Lookahead search has been shown to be an effective technique in conjunction with Reinforcement Learning (Tesauro, 1994). Although we depend on handcrafted features, this work should be looked upon as a proof of concept that feature transfer can be effective. Existing techniques for automated feature discovery in games (Fawcett, 1993) can be leveraged to strengthen feature transfer in future.

Characteristics of feature transfer

The features do not depend on the exact game, as long as it is within the genre of chosen games. Specifically, the size of the board, the number of available actions at each level, and the semantics of terminal states and win/loss criteria have been effectively abstracted away by exploiting the GDL. Consider the diverse natures of games in these aspects: in Tic-tac-toe the number of available moves steadily diminishes, in Connect-4 it diminishes at intervals, while in Othello it may actually increase. The winning criteria are widely varying in these games; they are similar in Tic-tac-toe and Connect-4 but completely different in Go or Othello. A key motivation behind this research is to develop simple techniques that can transfer knowledge effectively from one game to a markedly different game which is why we have focused on such a high level of abstraction.

Prior to developing the feature transfer concept, we have looked at the possibility of using a common feature space where we do both learning and transfer. In other words, we asked the question whether we can learn the quality values of features $Q(F_i)$ directly (so F_i forms the state space as well) instead of first learning $Q(s, a)$ and then extracting $val(F_i)$. We realized the futility of such an endeavor by posing a simple transfer problem that answered this question in the negative. Consider the problem of *role transfer*, i.e., learn to play a game as, say, the first mover and then use transferred knowledge to learn to play as the second mover. A feature space in which we learn must be discriminative enough so as not to alias semantically different states, but then if it is so discriminative then it is more than likely that we can identify the role of the player by just looking at the feature description. Now if the role is identifiable in a feature then the knowledge gained as the first mover would be useless when moving second. Consequently, we looked into different feature spaces for learning and transfer. Note that our features are also independent of the learner's role, but the state space used for learning is not.

One concern when using complex feature spaces for transfer is that the time overhead for computing transfer knowledge should not overwhelm the learning time. By limiting the number of features and the depth of lookahead, we are ensuring a low computational complexity for transfer knowledge. The limited lookahead depth also serves to keep the features quite indicative of the outcome of the subsequent moves. Note however, this indication is not always unambiguous, e.g., while F_1 and F_3 are indicative of winning and losing the game respectively, F_2 provides no clearcut indication. Win, loss or draw are all possible outcomes in this case. This ambiguity justifies transfer learning; if merely looking ahead would give a concrete idea of the ultimate outcome of playing a in state s, then we could well have initialized the corresponding Q-value in the target game to the known value of that outcome.

3. Symmetry transfer

Many games that humans enjoy playing take place on a rectangular grid or board. These games often exhibit some kind of symmetry that makes the game simpler to reason about. One type is *reflectional* symmetry, which means that for every state, the board can be "flipped" without changing the value of that position for either player. Reflectional symmetry may exist across the board's vertical axis, horizontal axis, or both. An example of reflectional symmetry about the vertical axis is shown in Figure 3 for the Connect-3 game. Note that Connect-3 does not exhibit reflectional symmetry about the horizontal axis. In fact, for the state shown in the figure, the state that results from a flip about the horizontal axis is not even a valid state.

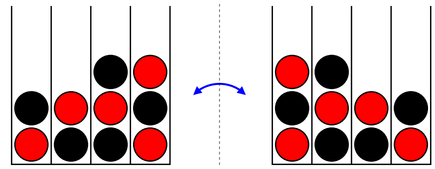

Figure 3. Reflectional symmetry in 4×4 Connect-3

In addition to reflectional symmetry, square boards may exhibit *rotational* symmetry. An example for the CaptureGo games is shown in Figure 4. The four states that result from rotating the board 90 degrees at a time are all strategically equivalent. We refer to the set of states that arise from a board's symmetric transformations to be the state's *symmetry set*.

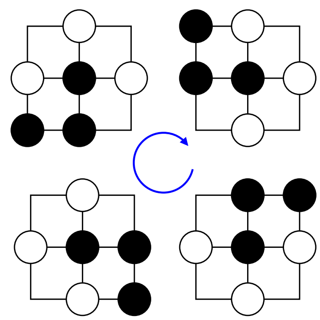

Figure 4. Rotational symmetry in 3×3 CaptureGo

A prerequisite for identifying board symmetries in a game is recognizing that the game contains a board. The agent identifies boards on its own by examining the formal game description. A board is essentially identified as a ternary predicate with two ordered inputs and one output. The full details can be found in (Kuhlmann et al., 2006).

Once the board has been identified, the agent tests it for each type of symmetry. For a board to exhibit a certain kind of symmetry, it must satisfy two conditions. First, for each terminal state, every state in its symmetry set must have the same outcome. Second, for every transition from non-terminal state s via action a to resulting state s', executing the symmetric transformation of a in the transformed state for s results in the transformation of s'.

Note that this symmetry identification method requires the enumeration of every valid state in the game. For large games, this would not be feasible. An important observation is that the symmetries exhibited by a smaller version of the same game are very likely to be the ones exhibited in the larger game. Therefore, the agent can use a smaller version of the target game it is going to learn to identify the game's symmetries and transfer that knowledge to the larger game.

The agent uses symmetry knowledge transferred from the smaller source task to speed up learning on the target task.

By treating symmetric states as identical, the agent avoids the need to learn about the same situation multiple times. By requiring the agent to learn fewer values in its value function, the hope is that it will be able to learn more efficiently.

4. Experimental Results

We conducted a set of experiments to measure the impact of each type of transfer on learning speed. We tested the transfer methods on two different target games: Connect-3 and CaptureGo. For both games, we compared the learning speeds of a *baseline* learner to learners that use transferred knowledge from other tasks. The baseline learner uses afterstate Q-learning with a value function initialized uniformly to the default value. The *symmetry* learner is identical to the baseline learner except that value function updates are performed for all afterstates that are in the symmetry set for the current afterstate. Finally, the *feature* learner uses the same backups as baseline, but initializes its value function to the values transferred from the source game. We do not report results from the combined (feature + symmetry) learner since they were not found to be significantly improved over the feature learner, given the shallow goals in these games.

Both of the target tasks take place on a 4×4 board. For symmetry transfer, the 3×3 versions of the games were chosen as the source games. The values for feature transfer were learned from standard tic-tac-toe as the source task. For both domains, the learner competes against an opponent that takes winning moves and avoids losing moves by looking ahead one full turn, but otherwise plays randomly.

Learning curves for each of the three learners in the Connect-3 game are shown in Figure 5. Both symmetry and feature transfer help the agent reach optimal play in fewer games than the baseline learner. The speed increase from symmetry transfer is significant, but not nearly as dramatic as feature transfer. The main reason for the limited impact of symmetry transfer is that only a single type of symmetry is present in Connect-3. Also, feature transfer reaches such a high level of performance so early, that there is little room for improvement. Games, such as Othello, where goals are usually deep in the game-tree could be more demonstrative of the superiority of the unified approach. We intend to run more experiments in the future.

The results of the experiments in the CaptureGo game are shown in Figure 6. Again, feature transfer achieves a high-level of performance from the start. This result is likely due to the fact that the calculated features examine the game tree at a greater depth than is explored by the one-move-lookahead opponent. Because CaptureGo exhibits both rotational and reflectional symmetry, the benefits of symmetry transfer in this game are more pronounced than in Connect-3. Also the advantage of symmetry transfer in compacting the state-space is compelling.

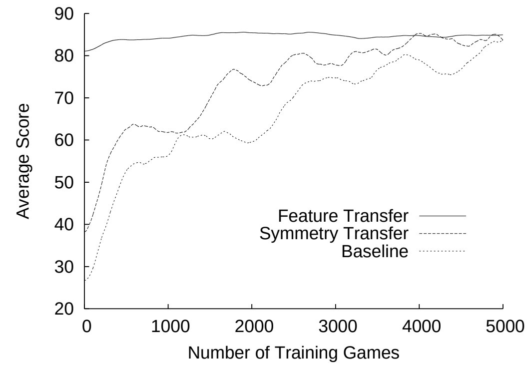

Figure 5. Results for 4×4 Connect-3

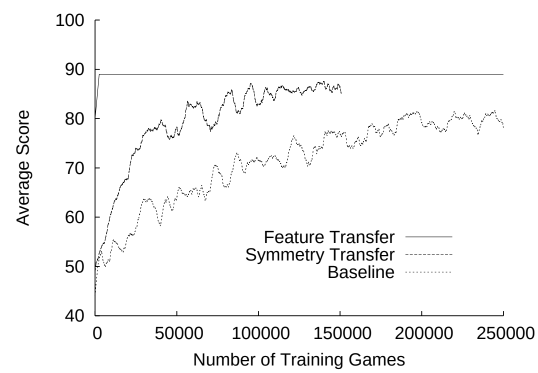

Figure 6. Results for 4×4 CaptureGo

References

Fawcett, T. E. (1993). Feature discovery for problem solving systems, PhD thesis, University of Massachusetts, Amherst.

Genesereth, M., & Love, N. (2005). General game playing: Overview of the AAAI competition. *AI Magazine*, *26*.

Kuhlmann, G., Dresner, K., & Stone, P. (2006). Automatic heuristic contruction in a complete general game player. *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. To appear.

Liu, Y., & Stone, P. (2006). Value-function-based transfer for reinforcement learning using structure mapping. *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. To appear.

Pell, B. (1993). Strategy generation and evaluation for meta-game playing. PhD thesis, University of Cambridge.

Taylor, M. E., & Stone, P. (2005). Behavior transfer for value-function-based reinforcement learning. *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems* (pp. 53–59). New York, NY: ACM Press.

Tesauro, G. (1994). Td-gammon, a self-teaching backgammon program, achieves masterlevel play. *Neural Computation*, *6*, 215–219.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.