

# The Self Organization of Context for Learning in Multiagent Games

Chris White and David Brogan

University of Virginia Department of Computer Science

## Abstract

Reinforcement learning is an effective machine learning paradigm in domains represented by compact and discrete state-action spaces. In high-dimensional and continuous domains, tile coding with linear function approximation has been widely used to circumvent the curse of dimensionality, but it suffers from the drawback that human-guided identification of features is required to create effective tilings. The challenge is to find tilings that preserve the context necessary to evaluate the value of a state-action pair while limiting memory requirements. The technique presented in this paper addresses the difficulty of identifying context in high-dimensional domains. We have chosen RoboCup simulated soccer as a domain because its high-dimensional continuous state space makes it a formidable challenge for reinforcement learning algorithms. Using self-organizing maps and reinforcement learning in a two-pass process, our technique scales to large state spaces without requiring a large amount of domain knowledge to automatically form abstractions over the state space. Results show that our algorithm learns to play the game of soccer better than a contemporary hand-coded opponent.

## Introduction

Simulated soccer with the RoboCup Soccer Server (Kitano *et al.* 1997) presents a substantial artificial intelligence challenge. Distributed decision making, noisy sensors and actuators, hidden state, and adversarial behavior all make simulated soccer a complex game to master. Many systems to date have relied heavily on the use of domain expertise to achieve an adequate level of play.

While domain expertise has so far proven effective in the simulated soccer domain, such systems rely on a great deal of hand tuning and their underlying algorithms are not easily generalized. As the complexity of these multiagent games continues to increase, incorporating domain expertise into a system becomes more challenging and artful. In particular, program architects must know how to describe game states in sufficient detail that the action selection algorithms can situate the current game state in the proper context and extract the best action. We observe, for example, that selecting an action during a one-on-one confrontation between

a player and a defender depends primarily on the players' relative positions whereas selecting actions during a team-wide attack requires information including the position on the field, the list of players involved, and the identification of a prepared defense.

The results presented in this paper are motivated by the long-term goal of machine learning to create systems that can automatically learn to reproduce expert behavior without explicitly needing expert knowledge built into the system. We move in this direction by looking at the sub-problem of automating the creation of a state abstraction for a learning agent. Specifically, this research addresses the challenge of automatically eliciting intelligent behavior from the soccer player in possession of the ball. Examples of actions a player may undertake are to kick the ball to a teammate, to kick towards a region of the goal, or to change speed and direction. Due to the realism of the soccer simulation, noisy sensors and actuators cause imperfect perception of the environment and unexpected action outcomes. Nonetheless, simulated soccer teams seek to construct series of actions that lead to the ball being kicked into the goal. Reinforcement learning is a paradigm that naturally fits such a domain with imperfect models, sequential decisions, and delayed rewards.

We also recognize that robust, automated and scalable tools for performing machine learning may make machine learning a more viable option for commercial developers in the electronic entertainment industry. Learning systems are under-utilized in the gaming industry because of their reputation for being potentially unpredictable and requiring too much tweaking to achieve a useful result. We aim to show that it is possible to use reinforcement learning to achieve compelling results with a minimal amount of hand-holding.

There are multiple issues that complicate the use of reinforcement learning in simulated soccer. The 22 players and the ball are modeled with 90 continuous-valued variables that are updated 6,000 times per game. Each player confronts a limited view of the field due to a 90-degree view cone that restricts visibility to within the cone and degrades vision as a function distance<sup>1</sup>. Because the players are independent processes and communication is severely restricted,

<sup>1</sup>See the Soccer Manual at <http://sserver.sourceforge.net> for more information.

decision making is distributed and agents must learn independently.

Of these challenges, we primarily address the difficulty of learning in large, continuous state spaces. We manage the large input space of simulated soccer by training a self-organizing map, a Kohonen network, to recognize common patterns in the state of the soccer game. After this clustering pass, the state space of the soccer simulation is discrete and manageable for online reinforcement learning. Our two-pass learning algorithm is called Kohonen-RL. We demonstrate the success of Kohonen-RL by competing against teams that were constructed by hand and by learning algorithms. The ability to learn effectively using our algorithm speaks to interesting opportunities for clustering systems to capitalize on the sparse coverage of complex domains and for learning systems to generalize optimal behaviors across state abstractions.

### Related Work

The most relevant research has been conducted in the domain of 3v2 keepaway (Kuhlmann & Stone 2004; Stone & Sutton 2001; Stone, Sutton, & Singh 2001; Stone, Sutton, & Kuhlmann To Appear), a sub-domain of simulated soccer. In this domain, the researchers implement a team of three keepers and a team of two takers. The goal of the keepers is to keep the ball away from the takers for as long as possible without kicking the ball outside of a pre-defined region. To accomplish this, the system uses a one-dimensional tile coding scheme over thirteen state variables and a *Sarsa*( $\lambda$ ) RL algorithm to approximate the value function (Please see (Sutton & Barto 1998) for background on *Sarsa*( $\lambda$ ) and tile coding). Similar systems have been widely used in domains that have large, continuous-valued input spaces (Sutton & Barto 1998). Such systems are attractive because the size of the state-space depends only on the number of tiles, not the native state space of the application.

Despite the success of reinforcement learning of player behaviors in 3v2 keepaway, we see reason to continue looking for a method that generates a tractable state-space for the larger state space of simulated soccer. In order to apply the tile coding used in 3v2 keepaway, decisions about how to create the tilings over the soccer state space must be made by designers. While principled methods to lessen this burden on the designers do exist, it is still a long-term goal of machine learning to find ways of automatically generating compact abstractions of large, continuous state spaces.

Parti-game is a reinforcement learning algorithm that uses game theory and computational geometry techniques to create a state space with varying levels of resolution (Moore 1994). Parti-game only needs to allocate high resolution to areas of the state space where high resolution is needed, but it is limited to problems where the dynamics are deterministic.

State aggregation techniques show the most promise for reducing a problem’s state space in an automated manner. The underlying theory and convergence proofs for the use of state aggregation techniques in reinforcement learning are presented in (Singh, Jaakkola, & Jordan 1995), but the Kohonen networks utilized in this paper have not yet been

formally studied in the context of reinforcement learning. Furthermore, additional experiments are required to demonstrate the effectiveness of state aggregation in real-world applications and to qualify how well such methods scale.

### State-Space Reduction

Kohonen networks are widely used in unsupervised pattern recognition tasks (Pandya & Macy 1996). We look to Kohonen networks to simplify the large soccer state space because their pattern recognition abilities will capitalize on the game’s sparse sampling. Because the networks are self organizing, we achieve our goal of minimizing the need for domain expertise.

A Kohonen network consists of two fully connected layers of neurons; an input layer and an output layer, sometimes called the Kohonen layer. The size of the input layer is determined by the dimensionality of the state extracted from the problem domain. The size of the output layer is tunable, whereby a smaller layer promotes greater generalization. Each neuron in the output layer has a weight vector of dimensionality equal to the size of the input layer. For a given input, each neuron in the output layer generates an output according to a user-defined function that measures the distance between the network’s input and the output neuron’s weight vector. We use a modified L2 distance metric that normalizes the distance for each dimension by dividing by the standard deviation of that dimension before summing the distances of each dimension. Theoretically, the L1 distance metric creates more contrast between distant states in high dimensional spaces (Aggarwal, Hinneburg, & Klein 2001), but it is not clear whether or not this actually improves performance of the clustering algorithm. Choosing a distance metric for a clustering application is an important decision, but we do not go into the nuances of making such a decision in this paper.

The neuron from the output layer that generates the lowest distance is defined to be the “winner” and the neuron’s identifier designates the transformed state of the input. The weight vector of the winning output neuron is adjusted to make that output neuron “closer” to the supplied input vector. The adjustment of the  $j^{th}$  weight of the winning output neuron is done according to the following equation:

$$W_j^{new} = W_j^{old} + \alpha(x_j - W_j^{old}) \quad (1)$$

Where  $\alpha$  is a step-size parameter. See (Pandya & Macy 1996) for more information about equation 1.

In our case, we are primarily concerned with which neuron wins because this serves to identify the transformed and simplified representation of the soccer game’s state. In more complicated cases, the network can be trained in a second pass to output a specific value and to group similar output neurons close together using localized lateral feedback. We omit a discussion of these features because although they contribute to the power of the Kohonen network, they do not affect the quality of the reinforcement learning.

A Kohonen Network is essentially a clustering tool. We chose to use Kohonen Networks because of ease of implementation and quick running time, but any number of clus-

tering algorithms would have worked as well. We refer the interested reader to the K-means, K-medians and K-centers algorithms. In general, increased running time efficiency sacrifices the quality of the clustering, but we omit a detailed discussion of algorithmic complexity of clustering as it exceeds the scope of this paper.

In our system, we specify 5,000 output neurons in the Kohonen network. Because the simulator models imperfect perception, it is not possible to use all the dimensions that describe a timestep of a game of soccer in the state space. Agents can only see part of the field and the accuracy of their perception degrades with distance, so we limit ourselves to the following 18 dimensional subset of the full dimensionality of the game of soccer as modeled by the simulator.

- The Cartesian coordinates of location of the ball on the field (2 dimensions)
- The polar coordinates of the closest four opponents to the ball (8 dimensions)
- The polar coordinates of the closest four teammates to the ball (8 dimensions)

Our choice of 5,000 neurons was arbitrary, but principled methods for choosing a good number of neurons exist. A developer could run the clustering algorithm many times in parallel and then plot the maximum cluster radius for each clustering run and use that data to infer what choice for the number of neurons gives the best balance between finding a tight clustering but keeping the number of clusters small.

To create our training set, we implemented an omniscient coach to watch 100 games between several teams from the 2002 Robocup World Championship tournament. The coach recorded approximately 70,000 input vectors corresponding to the spatial relationships described by the 18 dimensions above. We initialized the weights of our output neurons to the first 5,000 data points in our training set. We then performed 250 training epochs on our Kohonen network using all 70,000 samples, decrementing the step size parameter  $\alpha$  from equation 1 after each training epoch.

Kohonen networks can effectively compress a high-dimensional input space into a relatively compact state space for a reinforcement learning application because of their ability to capitalize on the sparseness of a system. The number of possible spatial configurations between players on the field grows exponentially with the dimensionality of the inputs that describe those spatial configurations (the curse of dimensionality). However, the spatial configurations of players on the field is not random. Player positioning is guided by the strategy of each team. Therefore, many spatial configurations have extremely low probabilities of occurring. The Kohonen network has the ability to locate configurations that have high probabilities of occurring and to generalize unseen configurations into configurations that it has seen before.

A reinforcement learning system based on a Kohonen network is generalizable to other sparse domains. The only decision a designer needs to make is how many output neurons the network should have. In principle, this decision can be replaced by specifying the maximum allowed distance from

any of the current output neurons before a new output neuron must be created for a new pattern. In this extension of the system, the resolution of the network is being specified a priori instead of the size of the network.

## Reinforcement Learning

Reinforcement learning for control problems is a framework for learning sequential decision making to maximize a reward. Each action taken is associated with the state in which that action was taken. After taking an action, the agent receives a reward which is then associated with the state-action pair that generated that reward. In some reinforcement learning algorithms, a state transition matrix is also learned.

The chained sequence of actions that make up a game of soccer fit naturally into the reinforcement learning paradigm. Like other researchers in the field (Kuhlmann & Stone 2004; Stone & Sutton 2001; Stone, Sutton, & Singh 2001), we state simulated soccer is a semi-Markov decision process. By semi-Markov decision process, we mean that although the soccer simulator operates in discrete time steps of 100 milliseconds each, we do not consider each simulator time step to be a time step in the reinforcement learning sense. For the purpose of reinforcement learning, we consider a timestep to bound the time of when one agent from either team first takes possession of the ball until the time another agent takes possession.

As described in the previous section, the first pass of our action-selection algorithm uses data collected from 100 soccer games to create a Kohonen network that represents common patterns in the game of soccer. These patterns are then fixed and become the state space of the second pass of Kohonen-RL, the reinforcement learning stage. In this second pass, the agents play a series of 1,000 training games against an opponent in order to learn expected rewards for state-action pairs.

## The Action Space

In control problems, the learning system must choose an action from a set of actions called the action space. An agent's policy is the mapping of states in the state space to actions in the action space. We use a standard  $\epsilon - greedy$  policy to select an action, where the action  $a$  with the highest expected reward is selected with probability  $1 - \epsilon$  and a random action is selected with probability  $\epsilon$ . We use  $\epsilon = 0.125$  in our implementation.

Our action space consists of low-level skills that include dribbling, passing, and shooting. The following twelve actions were selected from those described in (de Boer & Kok 2002) because of their reliance on a rich state-space description:

- Take a shot on goal
- Pass to one of four closest teammates
- Make a through pass to the closest teammate to the offensive corner of the field
- Clear the ball in the widest angle between opponents
- Clear the ball down the sideline

- Dribble in the direction of the goal
- Dribble straight forward
- Dribble through the widest gap between opponents
- Outplay an opponent by kicking the ball behind it in a position where the agent will be more likely to reach the ball than the opponent

We select these actions in order to give the agent a diverse set of actions from which to choose in the learning process.

### Action Evaluation

It is a straightforward process to assign a reward to an action that is executed during a game. If the action results in a terminal state, an immediate reward can be given to the agent. For simulated soccer, we consider the following events to be terminal states:

- The ball goes out of bounds
- Play stops due to a penalty
- A goal is scored for either team
- The opposing team comes into possession of the ball

The action receives a reward of 100 if it scores a goal on the opposing team. For all other terminal states, the action receives a reward of 0. If the action does not result in a terminal state, the Sarsa algorithm described in the next section determines the reward.

### Updating the Value Function

Reinforcement learning algorithms are effective at learning in Markov Decision Problems because of their ability to take into account the expected reward of expected future states as well as the expected reward of the immediate state. In simulated soccer, the probability of transitioning from one state to any other state is unknown, so we must use a reinforcement learning algorithm that does not need a state transition model. Temporal Difference learning is one such class of algorithms. Sarsa is a specific Temporal Difference algorithm that bases its update on the 5-tuple  $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ , where  $s_t$  and  $a_t$  are the current state-action pair,  $s_{t+1}$  and  $a_{t+1}$  are the next state-action pair the agent encounters, and  $r_t$  is the reward for being in state  $s_t$ . The approximated value function  $V^*(s_t, a_t)$  is updated in the following manner:

$$V^*(s_t, a_t) = V^*(s_t, a_t) + \alpha[r_t + \gamma * V^*(s_{t+1}, a_{t+1}) - V^*(s_t, a_t)] \quad (2)$$

where  $\alpha$  is the learning rate and  $\gamma$  is the discount rate. Sarsa stores the value function in a data structure called a Q-table.

Kohonen-RL uses the Sarsa algorithm to approximate the optimal value function for each action. However, the distributed nature of simulated soccer requires a slight modification to the algorithm. Because the agent’s positioning on the field is dictated by its position in the team’s formation, each agent only traverses part of the field. This leads to a partitioning of the total 5,000-element state space among the teammates. In our experiments, each agent generally never visited more than one fifth of the 5,000 states. Therefore,

if  $player_i$  passes to  $player_k$ , it will often be the case that  $player_i$  has never visited the state  $player_k$  sees. When this occurs,  $player_i$  cannot accurately update its value function because it does not have an estimate for the value function of its teammate. We avoid this problem by giving each agent access to the Q-tables for each of its teammates. We keep memory requirements low by using hash tables to store the sparse Q-tables. To obtain an estimate for  $V^\pi(s_{t+1}, a_{t+1})$ , the agent need only look up the value of  $V^*(s_{t+1}, a_{t+1})$  in the Q-table corresponding to whichever teammate has the ball.

## Results

The algorithm described in the previous sections was trained through repeated playing against a simple opponent and tested against two contemporary soccer playing algorithms. We trained our algorithm by running it through 1,000 games against a team that uniformly samples the set of twelve available actions to select an action for the player with the ball. Kohonen-RL, by contrast, selects actions for the player with the ball according to the standard reinforcement learning  $\epsilon$ -greedy policy described previously. The two teams are otherwise identical with all non-ball-possessing players returning to a prespecified location on the field according to their team role (de Boer & Kok 2002).

During the 1,000 games, we tracked the cumulative goals scored to show the improvement in performance of Kohonen-RL’s learning (Figure 1). We also record the time that the ball is in the opponent’s half of the field. Although the number of goals scored is the only metric that really matters in the game of soccer, it is useful to have multiple metrics of success for the purpose of analyzing this algorithm. For example, if a team struggles to learn how to complete a play by scoring a goal but has no problem learning how to move the ball forward and keep it forward, that information would still be useful in analyzing the performance of the algorithm.

Figure 1 shows the cumulative scores for our learning team versus the uniformly random team. We terminate the graph at 200 games to highlight the bootstrapping period of the learning team’s performance. After 100 games, Kohonen-RL is winning most of the games and by the end of the 1,000-game training session, Kohonen-RL has outscored the random opponent by a 931 to 195 margin. Figure 2 shows the amount of time Kohonen-RL keeps the ball in its opponent’s half of the field for each of the games in the 1,000-game training session. Kohonen-RL typically keeps the ball in its opponent’s side of the field more than three-quarters of the 6,000-timestep game.

After completing this training period, we test the fully trained team against a hand-coded team we developed as a testbed. The hand-coded team uses a strategy that relies on the location of the ball on the field and implements a wing-attack offense that tries to push the ball into the corners and then center the ball for a shot on goal. This hand-coded team captures a robust strategy for winning simulated soccer games. Our learning algorithm is capable of playing as well as, if not better than, the hand-coded testbed. It averages 1.02 goals per game as opposed to 0.86 goals per game

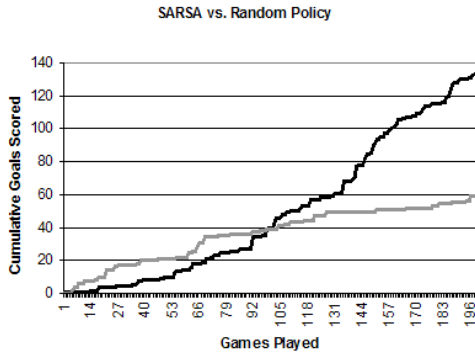


Figure 1: The cumulative goals during the first 200 games of a 1,000-game training period. The Kohonen-RL performance is in black and the random team’s performance is in grey.

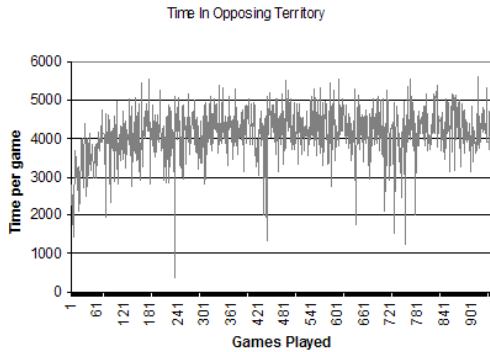


Figure 2: The time the learning team is able to keep the ball on the opposing team’s side of the field. Units of time are simulator timesteps. There are 6000 timesteps in a game of simulated soccer.

by the hand-coded team. Table 1 shows the results of 100 games against this hand-coded team.

We further validated our method by comparing it against two teams with identical  $\epsilon$ -greedy learning policies but with random state space abstractions. The first random state space abstraction is completely random; all parameters for all 5,000 cluster centers were generated with calls to `rand()` and normalized to fit into the appropriate range for each parameter. The second random state space abstraction is partially random; each of the 5,000 cluster centers were randomly selected from the set of training points used to train the Kohonen network. Figure 3 shows the learned state space abstraction used by Kohonen-RL performs sizably better than either of the random state spaces.

## Conclusions and Discussion

We have chosen simulated soccer as a domain for this research because of its high level of complexity. The contribution of this paper is a two-pass, data-driven technique

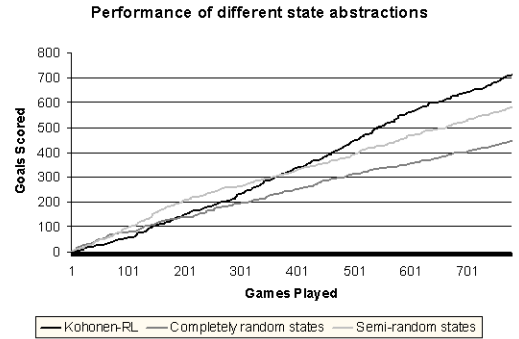


Figure 3: A comparison of Kohonen-RL versus identical reinforcement learning teams that use random state space abstractions instead of a trained Kohonen Network.

Kohonen-RL vs. Hand-coded Opponent					
Opponent	win	loss	tie	goals scored	against
HC	38	35	26	102	86

Table 1: Results of Kohonen-RL playing 100 games against a hand-coded testing team.

that learns an abstraction for the state space in its first pass and learns to associate rewards with state-action pairs during its second pass. Our Kohonen-RL system automatically identifies 5,000 representative states from a continuous, 18-dimensional space and uses those states to learn a value function for selecting effective actions in any state of the soccer simulation. The success of the Kohonen-RL team in the reported experiments demonstrates that reinforcement learning partnered with self-organizing maps can scale to complex domains such as simulated soccer with minimal injection of domain knowledge on the part of the designers. Despite this success, additional work in this field can tighten the relationship between state-space reduction and value function learning.

An unavoidable conflict in building such simplified simulations is the tradeoff between the ability of the system to abstract situations via the Kohonen network and the ability of the system to accurately approximate the value function  $V^\pi(s_t)$ . This is a well documented tradeoff (Sutton & Barto 1998; Santamaria, Sutton, & Ram 1996), but in many domains its effect is negligible on system performance. However, we found that in simulated soccer, particularly with certain actions, this tradeoff has a crippling effect on our system’s ability to to approximate the value function in an accurate way. We hypothesize that the value of performing some actions is intolerant of small changes in the state in which they are executed, thus conflicting with any benefits of state abstraction. For example, when shooting at the goal, a two-meter difference in the position of the goalie can make the difference between a very high  $V^\pi$  and a very low  $V^\pi$ . The Kohonen network’s generalization of the context of the game prevents reinforcement learning from discovering this very slight difference. The system can therefore never converge to the true value of  $V^\pi$ .

Sampling theory can provide a robust framework for minimizing the error introduced by generalization. A cluster point in the Kohonen network is essentially a sample point for the value function being approximated. Minimizing the error of the value function then becomes a question of deciding where best to put the cluster points. Other researchers (Santamaria, Sutton, & Ram 1996) who have done work on this problem have suggested that the state space can be transformed in order to provide high resolution only where it is needed. This research has primarily allocated higher resolution to frequently visited states, even though these are unrelated to the more important goal of allocating the high resolution to regions of the state space where the value function is the most complex. What is really needed is the ability to expand the resolution in regions of the state space where the value function experiences high-frequency oscillations. In future work, we will extend the Kohonen network's distance function to incorporate data about the difference between the approximated value function in a particular state and that state's neighbors in the network. The network will then be able to retrain itself to super sample the value function for a better approximation.

To economize the allocation of state space in the reinforcement learning process, our future work will develop action-dependent state spaces. We have presented results that use 18 dimensions to distinguish optimal circumstances for action selection. However, it may be the case that the value function for a particular action only depends on a subset of these 18 dimensions. For example, shooting at the goal may only depend on the position of the ball on the field and the position of the goalie. By automatically extracting the relevant state-action features, the dimensionality of the state space can be reduced and a finer mesh over the state-space can be used to allow the agent to perceive the variances in state that cause high-frequency changes in  $V^\pi(s_t, a)$ . There has been some recent work on learning to autonomously identify and ignore irrelevant state variables (Jong & Stone 2004) that could be applied to creating action dependent state-spaces, but the problem is still very much an open one.

### Acknowledgements

We gratefully acknowledge support from the National Science Foundation (ITR 0426971). We would like to thank David Luebke for allowing us to his his computing clusters for our simulations. We would like to thank the reviewers for their insightful and helpful comments.

### References

Aggarwal, C. C.; Hinneburg, A.; and Klein, D. 2001. On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the 8th International Conference on Database Theory*.

de Boer, R., and Kok, J. R. 2002. The incremental development of a synthetic multi-agent system: The uva trilearn 2001 robotic soccer simulation team. Master's thesis, University of Amsterdam, The Netherlands.

Jong, N. K., and Stone, P. 2004. Towards learning to ignore irrelevant state variables. In *The AAAI-2004 Workshop on Learning and Planning in Markov Processes – Advances and Challenges*.

Kitano, H.; Tambe, M.; Stone, P.; Veloso, M. M.; Coradeschi, S.; Osawa, E.; Matsubara, H.; Noda, I.; and Asada, M. 1997. The robocup synthetic agent challenge,97. In *International Joint Conference on Artificial Intelligence (IJCAI97)*, 24,30.

Kuhlmann, G., and Stone, P. 2004. Progress in learning 3 vs. 2 keepaway. In *RoboCup-2003: Robot Soccer World Cup VII*.

Moore, A. W. 1994. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In Cowan, J. D.; Tesauro, G.; and Alspector, J., eds., *Advances in Neural Information Processing Systems*, volume 6, 711–718. Morgan Kaufmann Publishers, Inc.

Pandya, A. S., and Macy, R. B. 1996. *Pattern Recognition with Neural Networks in C++*. IEEE Press.

Santamaria, J. C.; Sutton, R. S.; and Ram, A. 1996. Experiments with reinforcement learning in problems with continuous state and action spaces. Technical Report UM-CS-1996-088, University of Massachusetts.

Singh, S. P.; Jaakkola, T.; and Jordan, M. I. 1995. Reinforcement learning with soft state aggregation. In *Advances in Neural Information Processing Systems*, volume 7.

Stone, P., and Sutton, R. S. 2001. Scaling reinforcement learning toward robocup soccer. In *Proceedings of the 18th International Conference on Machine Learning*.

Stone, P., and Veloso, M. 1999. Team-partitioned, opaque-transition reinforcement learning. In *RoboCup-98: Robot Soccer World Cup II*.

Stone, P.; Sutton, R.; and Kuhlmann, G. To Appear. Scaling reinforcement learning toward robocup soccer. *Adaptive Behavior*.

Stone, P.; Sutton, R.; and Singh, S. 2001. Reinforcement learning for 3 vs. 2 keepaway. In *RoboCup-2000: Robot Soccer World Cup IV*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: MIT Press.