State Space Complexity Management In Reinforcement Learning

A Thesis
in STS 401

Presented to
The Faculty of the
School of Engineering and Applied Sciences
University of Virginia

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Computer Science

Submitted by
Chris White

October, 2004

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in Science, Technology and Society Courses.

Signed: ____________________________

Approved __________________________ Date _____
Technical Advisor — David Brogan

Approved __________________________ Date _____
Science, Technology, and Society Advisor — Edmund Russell
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glossary</td>
<td></td>
<td>iii</td>
</tr>
<tr>
<td>I</td>
<td>Rationale and Objectives</td>
<td>1</td>
</tr>
<tr>
<td>II</td>
<td>Social and Ethical Contexts</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>A The Local Context</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>B The Wider Context</td>
<td>7</td>
</tr>
<tr>
<td>III</td>
<td>Literature Review</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>A Reinforcement Learning in Simulated Soccer</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>B State Space Reduction</td>
<td>10</td>
</tr>
<tr>
<td>IV</td>
<td>Statement of Project Activities</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>A Activities</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>B Schedule</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>C Personnel</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>D Resources</td>
<td>16</td>
</tr>
<tr>
<td>V</td>
<td>Expected Outcomes</td>
<td>17</td>
</tr>
</tbody>
</table>

A Equipment Checklist 20

B Biographical Sketch of Student 21
List of Figures

1  Simulation of a game of soccer ....................... 3
2  Project Schedule ........................................ 15
Glossary

A

**Action Space** A set of actions that an agent can perform. Actions in the action space may be high level behaviors or low level actuator outputs. The action space can be continuous or discrete, but this paper will only consider the discrete case.

**Actuator** An output device for an agent. Actuators effect the environment in which the agent acts. For example, a robot might have a gripping arm that can move blocks around. The arm would be an actuator.

**Agent** An autonomous entity that interacts with an environment. Agents have sensors to perceive the environment and actuators to modify the environment.

C

**Cardinality** The number of elements in a set.
I

Input Space  The set of sensor inputs that describe the current state of the environment in which the agent interacts. Note that there is a subtle difference between input space and state space. Input space refers to the raw sensor inputs that describe the environment but state space refers to any abstraction that describes the state. Many papers use the terms interchangeably, but it is not technically correct to do so. Input space is also sometimes referred to as observation space.

K

Kohonen Network  Also called a Kohonen self organizing map, a Kohonen network is an unsupervised pattern matching algorithm. A Kohonen network can organize large amounts of data into a relatively small number of patterns and learn to classify new examples into existing patterns.

L

Layered Learning  A learning paradigm in which low level actions are learned separately from high level behaviors. In the canonical sense, lower levels are learned first and higher levels learn using the previously trained lower levels. Any number of layers may be used.

S

Sensor  An input device for an agent. Sensors allow agents to perceive the environment. For example, a robot might have a camera that can take pictures of the world. The camera would be a sensor.
State Space
A set of states that describe abstractions of the environment in which the agent interacts.

Team Partitioned Opaque Transition Reinforcement Learning
A distributed reinforcement learning algorithm originally developed for simulated soccer. It is currently being used in network routing as well.

Tile Coding
Also called CMACs, tile coding is a state space reduction technique that works by overlaying multiple grids on the input space and using linear function approximation techniques to extract a state from a resulting feature vector. The full details of this algorithm exceed the scope of this paper, but an interested reader can find more information in the cited references.
Abstract
Reinforcement learning (RL) is an effective machine learning paradigm, but because the state space cardinality often grows exponentially as the dimensionality grows linearly, many domains are not well-suited for traditional RL techniques. Tile coding with linear function approximation has been widely used to circumvent the curse of dimensionality, but suffers from the drawback that trial and error plays a large role in deciding how best to create tilings over the state space. My motivation is to explore methods that can scale to large state spaces but do not require a large amount of domain knowledge and can form abstractions over the state space in a much more automated fashion. I have chosen Robocup simulated soccer as a domain because its 90-dimensional continuous-valued state space make it a formidable challenge for reinforcement learning algorithms. I plan to develop an algorithm that uses Kohonen Networks to manage the complexity of the state space in an automaticmanner. I expect results to show that our algorithm will outperform other reinforcement learning algorithms in the task of simulated soccer.
I Rationale and Objectives

Simulated soccer with the Robocup Soccer Server presents itself as a fairly substantial artificial intelligence challenge. Distributed decision making, noisy sensors and actuators, hidden state, and adversarial behavior all make simulated soccer a difficult domain. Many systems to date have relied heavily on the use of domain expertise to achieve a decent level of play.

While domain expertise has so far proven effective in the Robocup domain, such systems require a great deal of tweaking and hand-tuning. Incorporating domain expertise into a system is often much more of an art than a science. A long-term goal of machine learning is to create systems that can learn to reproduce expert behavior in an automatic fashion without explicitly needing expert knowledge built into the system.

I am interested in creating a learned strategy that minimizes the requirement of domain expertise. For my research, I will consider the sub-problem of deciding what an agent should do when he is in possession of the ball. The game of soccer consists of a series of such decisions made by the players on the field.

Reinforcement learning for control problems is a framework for learning sequential decision making to earn a reward. An episode in a reinforcement learning context consists of a set of actions taken sequentially by a learning agent at discrete time steps. 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.

Formally, reinforcement learning maps a set of states, $S$, and actions, $A$, to a reward in the set of real numbers. That is, performing action $a$ in state $s$ results in
reward $r$. This mapping, denoted $V^*(s,a)$, is approximated by experiencing rewards received from state-action pairs and averaging the rewards over time. The approximated function is denoted as $V^*$ as opposed to the actual value function $V^\pi$. In control problems, an action is selected for a given state by choosing the action $a$ from $A$ that maximizes $V^*$.

The chained sequence of actions that make up a game of soccer fit naturally into the reinforcement learning paradigm. Following the lead of other researchers in the field [1], [2], [3], I will take simulated soccer to be a semi-Markov decision process. 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 time step to occur when an agent from either team takes possession of the ball.

However, there are a number of issues that complicate the use of reinforcement learning in simulated soccer. (1) The input space is continuous and very high-dimensional\(^1\), (2) the agent has only a limited view of the field\(^2\), (3) decision making is distributed and agents must learn independently and (4) communication between agents is strictly limited by the soccer server.

The Robocup soccer simulator provides an environment that simulates the game of soccer. Twenty-two autonomous agents connect to the simulator to form two teams. The agents are all independent processes, and communication between the agents is moderated by the simulator. The simulator operates by sending out environmental information to each agent every time step, and accepting continuous valued commands from the agents. Examples of commands would be to turn a certain number of degrees, or to kick in a certain direction with a certain velocity. The simulator models noisy

---

\(^1\)There are approximately 90 continuous valued inputs that describe the current state of the game

\(^2\)The agent has a 90 degree view cone and the quality of the vision degrades with distance. See the Soccer Manual at http://sserver.sourceforge.net for more information.
sensors and actuators, meaning the agents do not perceive the environment perfectly nor do they have the ability to effect it exactly as they desire. The state of a game can be fully described with approximately 90 continuous variables, but only state variables in the agent’s visual range are provided to the agent.

Figure 1: Simulation of a game of soccer

I have chosen simulated soccer as a domain for our research because of its high level of complexity. Not much is known about how well reinforcement learning algorithms can perform in exceedingly complex tasks. Some work has been done elsewhere in simulated soccer by [1], [2], [3], [4], but still more experimental work is needed to answer important questions about the behavior of systems in complicated domains like Robocup.
The important research goals that I hope to achieve in this project are:

- Strive for an algorithm that is both domain independent and can outperform other similar algorithms in the simulated soccer domain.

- Answer questions about the scalability of reinforcement learning to complex domains with large state spaces, including questions about aliasing of $V^*$ as a result of undersampling. Can adaptive supersampling be of use? Can multi-resolution modeling improve results?

In order to meet these goals, I will have to review the background literature, plan the project, develop a research infrastructure, implement several algorithms, run tests on the algorithms, and write the final thesis report.

Because the main goals include striving for domain independence and testing the performance of reinforcement learning systems in complex domains, I will seek to avoid using simplifications to the game of soccer as a means to increase the tractability of simulated soccer for a reinforcement learning algorithm. Rather than try to scale soccer down to fit into reinforcement learning, I will aim to scale reinforcement learning up to simulated soccer.

II Social and Ethical Contexts

There are a wide variety of potential applications for research in robotics and machine learning. However, the benefits of this research can be classified into the local applications to the machine learning community and the wider applications to other fields of research and to society as a whole.
A The Local Context

The local context of my research includes the fields of robotics and machine learning. In this domain, the usefulness of my research contributions will be judged by their ability to further the current state of the art within these fields. This domain is the most important domain of interest because the only primary stakeholders in this project are other researchers. This project will not directly impact society because the final product will be a piece of knowledge that is only of interest to a small research community. The wider implications of this project are determined by how this small technical audience build upon and use this piece of knowledge.

Within the robotics community, there is a sentiment that as the problems roboticists would like to solve become more complicated, machine learning will become more and more important[4][1]. Many of the traditional AI techniques lack the ability to scale to the problems of tomorrow. Expert systems suffer from the fact that they can only be as powerful as the human expert who programs their knowledge bases. Some problems are simply too complicated for a human expert to program all his or her insight directly into the system. Graph search techniques suffer from exponential running time problems. It is becoming clear that the most effective way to solve the AI and robotics problems of tomorrow will be for machines to learn how to solve problems on their own.

The rapid and recent development of the developmental robotics community is evidence of the importance of machine learning on the future of robotics research. One of the reasons often cited for the rapid emergence of this field is the desire among researchers to create systems that are more autonomous and more adaptable than previous systems[5].

Within the machine learning community, there is a lot of concern over the ability
of reinforcement learning to scale to problems with large state spaces and/or complex value functions [6] [1] [2] [7]. The consensus in the community is that not much is known about the performance of reinforcement learning in such domains. While there have been a few landmark cases of reinforcement learning successfully solving large state space problems, such as Gerald Tesauro’s TD-Gammon backgammon algorithm[8], these algorithms lack generalizability and are therefore of limited use in answering the question of whether or not reinforcement learning can perform well in complex problem spaces.

Though TD-Gammon is a tour-de-force in showing the potential of the temporal difference reinforcement learning paradigm, the success of the algorithm stems from the properties of the game of backgammon. TD-Gammon is so powerful because the 198 features Tesauro uses to describe the state of a backgammon game are ideally suited for feed forward neural networks[7]. Tesauro acknowledges that in a game such as chess, where the value of a piece in a certain space is often contingent upon the location of other pieces on the board, the TD-Gammon algorithm would play significantly worse than a novice[8].

One goal of my research is to help answer some of the questions about the scalability of reinforcement learning to complicated domains. Robocup simulated soccer is an ideal domain for this research because the soccer server represents the current state of the game as a 90-dimensional continuous valued feature vector and because the complexity of the target value function to be learned by the agent is not known by the agent a priori. While these conditions are sufficient to guarantee that simulated soccer is complex enough to be a suitable domain to answer questions about the scalability of reinforcement learning, simulated soccer has a number of additional qualities that further increase its level of challenge. Agents have noisy sensors and actuators, only part of the world is visible at any given time, and the dynamics of
the system are non-deterministic and subject to noise.

Other authors have similarly chosen simulated soccer as a domain for machine learning and reinforcement learning research because of the complexity of the domain[4][1][2][3]. I intend to build upon their work.

B The Wider Context

One impact that research in machine learning has is in other fields of computer science. There are many fields of computer science that capitalize on machine learning research. [9] is an example of an algorithm developed for Robocup simulated soccer being used to improve the efficiency of packet routing in networks. [10] is an example of operating system scheduling being improved by a machine learning technique. [11] and [12] are examples of clustering being used to create realistic animations from large motion capture databases in real-time. Clearly, many fields stand to benefit from advancements in the field of machine learning.

Outside of academic circles, there is also a broad range of possible applications for research in machine learning and robotics. One potential application of simulated soccer research is in the area of disaster relief and recovery. Because of the danger that can be associated with search and rescue missions in a disaster area, it would be nice to have autonomous robots that can perform search and rescue operations. Robots used in such a scenario would rarely be deployed in isolation, so they would need to communicate and coordinate with other robots on the same mission. These robots would also only have access to partial information of the environment in which they would act. There are enough similarities between simulated soccer and search and rescue that breakthroughs in one field can easily advance the other[13]. Because of these similarities, the Robocup Federation holds annual joint conferences on simulated
soccer and search and rescue.

Search and rescue is only one of many possible socially beneficial applications of research in machine learning and robotics. Other examples include space exploration, minefield clearing, and bomb defusal. In short, any application where the risk to human beings is great enough that one would prefer it be done by machine can benefit from machine learning and robotics research.

III Literature Review

A Reinforcement Learning in Simulated Soccer

The first attempt to use reinforcement learning in Robocup to choose an action when the agent has the ball was Team-Partitioned Opaque Transition Reinforcement Learning (TPOT-RL) [4]. The main contribution of team partitioned opaque transition reinforcement learning was the introduction of layered learning and action-dependent features. Action dependent features are used in this system to decouple the complexity of the state space from the dimensionality of the inputs. In this system, a decision tree is trained off-line to classify actions as likely successes or failures. The results of the classification on each action in the agent’s action-space are the features that determine the agent’s current state.

The benefit of this system is that for small action-spaces, the state-space is very compact. The state-space is also completely independent of the complexity of the input-space. TPOT-RL is able to achieve independence between the state-space size and the dimensionality of the input-space because the number of features that determine the state-space is tied to the action-space instead of the input-space. However, this method still has several important limitations. The first limitation is that the
state-space grows exponentially as the action-space increases linearly. This explosion occurs because when an action is added to the action-space, the number of possible combinations of classifications on the actions doubles. The original researchers included eight actions in the action space, all of which were passes. My motivation is to create an agent that can consider a larger, more diverse set of actions, but this limitation is a significant hindrance to the ability of the agent to maintain a balance between having a robust action-space and maintaining a tractable state-space.

The second limitation of team partitioned opaque transition reinforcement learning is that the system’s ability to identify the context of the game is limited to identifying which player on the team currently has the ball. This limitation is important because the optimal strategy for an agent often depends on information that is not relevant to the classification of an action as a likely success or likely failure. One such piece of information is the location of the ball on the field. For example, if the ball is in the corner of the opponent’s section of the field, the optimal choice of action is probably a centering pass. However, if the ball is in a different location on the field, the best choice for an agent may be completely different. Because each agent has a wide range of possible locations on the field, determining which agent on the field has the ball is insufficient for determining the current context of the game.

A more recent piece of related work has been done in the domain of 3v2 keep-away [2], [1], [3], a sub-domain of simulated soccer. In this domain, the researchers implemented a team of 3 keepers and a team of 2 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 create a tractable state space in the 3v2 keep-away problem, 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 [7] for background on
State Space Complexity Management In Reinforcement Learning

*sarsa*($\lambda$) and tile coding).

Similar systems have been widely used in domains that have large, continuous valued input spaces. Such systems are attractive because the size of the state-space depends only on the number of tiles. The system built for 3v2 keep-away is a significant advancement from TPOT-RL because the tile coding scheme can effectively generalize without losing track of the context of the game.

However, there is still reason to look elsewhere for a method to generate a tractable state-space for the simulated soccer reinforcement learning problem. Decisions about how to create the tilings over the input-space must be made by the 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 state representations[2]. With tile coding, this is extremely difficult if not impossible. I therefore look to another method that is more conducive to self-organization.

**B State Space Reduction**

Kohonen networks are widely used in unsupervised pattern recognition tasks. I hypothesize that Kohonen networks hold promise for solution to the large input-space problem because their pattern recognition abilities will be able to capitalize on the entropy of a system and because their self-organizing properties are conducive to achieving the goal of minimizing the requirement of domain expertise at design time.

Kohonen networks can capitalize on the entropy of a system because input patterns with zero or near zero probability of occurring can be ignored and discarded by the network. Therefore, only patterns that the network will be likely to encounter are included in the network. In systems with high entropy, a large portion of all possible input combinations can be discarded.
A Kohonen network consists of two fully-connected layers of neurons; an input layer and an output layer, sometimes called the Kohonen layer. Each neuron in the output layer has a weight vector of dimensionality equal to the size of the input layer. The winning neuron in the output layer is determined by applying a distance function on the input layer and the weight vector of each neuron in the output layer to find the closest output neuron to the given input vector. In the simplest case, the distance function is just the euclidean distance function, but more complicated distance functions can be used if desired.

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:

$$w_{j}^{new} = w_{j}^{old} + \alpha(x_{j} - w_{j}^{old})$$  \hspace{1cm} (1)

Where $\alpha$ is a step-size parameter. See [14] for more information about 1.

In the simplest case, the winning neuron outputs a 1 while the rest of the neurons output a zero. However, in more complicated cases, the output neuron can be trained in a second pass to output a specific value. Kohonen networks can also be trained to group similar output neurons close together using localized lateral feedback. Lateral feedback can be useful because the agent will be able to perform frequency analysis on the learned value function easily if similar states are grouped together.

Frequency analysis of the learned value function is important because the agent has no a priori knowledge about the complexity of the value function. One of the elementary results of sampling and signal reconstruction theory is the sampling theorem, which states that a signal can be reconstructed from its samples only if the signal contains no frequencies greater than one half of the sampling frequency. The
Nyquist rate is the minimum rate of sampling at which a signal can be accurately reconstructed.

The output neurons of the Kohonen network represent samples of the state space. The size of the network represents the resolution of the network. One problem that could arise when using a Kohonen network in this context is the difficulty of finding the optimal size of the network for a given problem. Furthermore, the frequency of value function may not be uniform. For example, in the game of soccer, the value function for shooting the ball at the goal may be zero for any node in the network where the ball is not in the offensive third of the field but might oscillate frequently depending on the position of the goalie relative to the ball when the ball is in shooting range. In such a case, it would not make sense to uniformly sample the state space. All the nodes that were zero can be represented as a single node but there must be many nodes in the areas of the state space that oscillate frequently.

The lateral feedback function of a Kohonen network will provide the network with the unique ability to selectively and autonomously collapse or expand similar nodes based on frequency analysis of the learned value function. In addition to learning the value function for each node, an approximation of the standard deviation on the recorded returns can also be learned for each node. Collapsing nodes is defined as taking nodes that are next to each other and that have similar values and low standard deviations and consolidating them into a single node. Expanding nodes is defined as taking nodes that have a high standard deviation and re-clustering the training examples that fall into that node into multiple nodes. To my knowledge, no similar work has been done on sampling theory in the reinforcement learning context, but the theory and practice has been rigorously defined in computer graphics literature where adaptive supersampling is used as an anti-aliasing technique in ray tracers[15]. The network’s ability to perform this functionality gives it a significant advantage over
tile-coding based schemes that require sampling resolutions to be defined a priori.

IV Statement of Project Activities

A Activities

I expect that the work required to complete this project will fall into the following categories: review of background literature, planning the project, developing a research infrastructure, implementing the various algorithms to be tested, running simulations and benchmarking the algorithms, and creating the final report of the results.

Review of Background Literature

Much of this work has already been completed. I am well read in the field of machine learning and reinforcement learning and understand the impact that my work can have in these fields. I am also aware of the way in which other fields of research in computer science such as computer graphics, network routing, and operating systems use machine learning techniques to improve the state of the art.

Planning the Project

Planning the project involves coming up with research goals and then developing a plan of attack to accomplish those goals. At this stage in my project, both of these stages are complete.

Developing a Research Infrastructure

There are a number of issues that must be addressed in creating a research infrastructure. I obtained an open source base program that can interact with the Robocup
soccer server and implements the low-level actions of a soccer player such as kick in a certain direction or run to a certain point. This base code will allow me to concentrate solely on developing high-level behaviors without having to worry about low-level actions.

I have also set up a testing environment which includes an installation of the Robocup soccer server, several python scripts I wrote to automate and manage experiments, a program to collect and store data on the games, and several small tools I wrote to analyze the data.

I have also created a CVS tree on my Computer Science department home directory to store and manage my documents and my source code. This tree is backed up by the department on a daily basis in case of a crash. This constitutes the entirety of the research infrastructure I need to complete this project in a timely manner.

**Implementing the Algorithms**

Most of the algorithms that I need to develop and test for this project I have already developed. Of the algorithms I wish to test, the only one that I have not yet implemented is the algorithm that uses adaptive super sampling and frequency analysis to autonomously collapse and expand nodes in the Kohonen network. Given the current state of my code base, I believe that this algorithm could be implemented in two to three weeks time during the semester or just one week during the semester break when other class work is not competing for my time.

**Running the Simulations**

For most of the algorithms that I plan to test, the simulations take approximately 10 days to run. However, these simulations can be run in parallel, so testing multiple algorithms does not extend this time period. The adaptive supersampling algorithm
will take much longer to simulate because it must make several training passes, re-
clustering the high frequency nodes between each pass. I estimate that a full sim-
ulation of this algorithm could take up to 30 days. Because of the time that this
algorithm could take to simulate, it is imperative that the algorithm be finished and
ready for simulation before the beginning of the spring semester. If this goal is not
met, then the adaptive supersampling algorithm will most likely have to be dropped
from the final report.

The Final Report

I expect to spend the spring semester writing and polishing the final report. Much
of the writing for this report can be done in parallel with other parts of the project,
and parts of the writing have already been completed.

B Schedule

![Project Schedule](image-url)

Figure 2: Project Schedule
C Personnel

In order to complete this project, I have set up weekly meetings with my technical adviser to keep the research on schedule and to address any problems that arise. I will also meet with him more frequently when simulations start to produce results that need to be analyzed. I expect my technical adviser to help me analyze background literature and to help me identify the research communities that will be most interested in my work. I also expect him to advise me on the publishability of results.

I meet with my Science, Technology and Society adviser twice a week during class. I will attend office hours if the need arises.

D Resources

The following resources are required for the successful completion of this project:

- A Linux development environment because the Robocup soccer simulator only runs on Linux,
- A Linux cluster to run long running simulations in parallel, and
- Office space to perform my work in close proximity to my technical adviser.

At this time, I have access to all of these resources. The U.Va Graphics group has allowed me to use one of their Linux machines in the Graphics Lab. Courtesy of Professor Dave Luebke, I have access to two of the Graphics group’s Linux clusters for the purpose of running simulations.
V Expected Outcomes

At the conclusion of this project, I intend to have several algorithms implemented and tested. I expect that the results of the tests will add to the body of knowledge about reinforcement learning. The results of my tests and analyses will include the rate of learning and graphs of the learning curve for the various systems I implement. The results will also include measures of total performance gain for each of the implemented systems. Finally, I will include analyses of conditions under which the algorithms have difficulty performing.

I will also re-implement several existing algorithms for the purpose of comparing my algorithms to the existing state of the art. One clear indicator of success will be whether or not my system can outperform current systems. Performance can be benchmarked in terms of goals scored or in terms of training time.

One measure of success in academic research is publication. In addition to having the final report published in Virgo, I would like to submit a condensed version of the thesis to one of the top three AI conferences. These conferences are the International Joint Conference on Artificial Intelligence (IJCAI), the International Conference on Machine Learning (ICML), and the Innovative Applications of Artificial Intelligence (IAAI) conference.

While my work on this project will likely end when my senior thesis is complete, it is important to note that research is never complete. As part of my final report, I plan to include a roadmap of possible future work that others can pursue to build upon my work.
Bibliography


Appendix A

Equipment Checklist

The following equipment has been secured for this project:

- Two 16 node Linux clusters.
- A Linux workstation.
- Office space in Olsson Hall.

No further equipment or monetary funding is expected to be necessary for the completion of this project.
Appendix B

Biographical Sketch of Student

As a candidate for a Bachelor of Science in Computer Science at the University of Virginia, I have acquired the technical abilities in software development, project management, and artificial intelligence required to complete this project. I have been doing work in the Robocup simulated soccer domain for two years.