Autonomous Learning in a Simulated Environment

CPR E 585 Final Report

Alexander Campbell
Chad Nelson
Daniel Stiner

Table of Contents

Abstract

This project uses a generic prediction learning model to develop a robot capable of playing the classic arcade game of Snake. Our project explores the possibilities of intrinsic motivation and reinforcement learning as applied to robots, and does so using Online Support Vector Machines. Online SVMs have the capability to learn behavior or technique, similar to other, more well-known AI algorithms, but with the additional capability of learning and forgetting as it is playing the game. In this research we compare and contrast different methods of learning and gameplay styles. In a broader context, we hope that our research will prove useful in the attempt to develop a more generalized artificial intelligence.

Introduction

Current machine learning research is dominated by a simple problem: the lack of generality. The body of research has been mostly focused on solving specific problems by extracting and using statistical properties in order to predict future events. Cornerstones to this process have traditionally included manual training and configuring of systems. This takes time and resources, and these systems do not provide a framework for future development.

Oudeyer, Kaplan and Hafner run into the problem of generality in their 2007 paper. They discuss a scenario where a computer is taught the grammar of a language by learning from a handmade sequence of increasingly difficult grammatical constructs. It develops a deeper and deeper understanding by having each successive step hand-crafted to take advantage of the prediction ability gained by the artificial intelligence system in the previous iterations. This concept, developmental learning through successively more complex steps, is a key part of creating an artificial intelligence with truly general properties.

Our project is intended to be a part of the solution to the problem of generality in developmental robotics in two ways. First, we demonstrably show that Support Vector Machines are an alternative to neural networks, with the possible benefit of scalability through parallel processing. Second, we give an example of how SVMs can be used as a generic predictor. The general predictor created in our project, along with existing research on internal motivation based on reinforcement learning, can be used in developing a framework for solving the generality problem in learning. In addition, the virtual aspect of our project is a model to follow in terms of rapid development.

In finer detail, we have implemented a robotic intelligence; one that is driven by intrinsic curiosity and is capable of both learning properties of its environment and adapting to a variable body. The relatively simple and well-known arcade game, "Snake," was the basis for testing the effectiveness of the project in achieving such intrinsic motivation for developmental learning. An algorithm called Intelligent Adaptive Curiosity was used to select behaviors which result in semi-predictable, yet novel events that are optimal for learning how to better predict future changes in the environment. Our hope was to see the continual emergence of new and more complex behavioral patterns as those typically seen in previous research[1] [2], despite the added complexity of a variable body schema. The preliminary work we completed has created the groundwork for a simple and extensible framework which will allow future research of similar motivation algorithms and other simulated environments.

With no handcrafted sequences of steps, our project could easily be expanded to applications where the rules of the system are relatively simple. An example of one area where our research could have been applied is Sutton's project of using learning algorithms to decrease the wait time for an elevator. In such a case, the system has simple set of rules and a highly predictable behaviour, just like our project of snake. Given an initial exploration period, the elevator system would learn to predict if stopping at a floor to pick-up waiting people would dramatically increase the average wait time for everyone. Then learning could be disabled or slowed and only optimal stops would be made. Our hope is that this methodology can be easily applied to such and similar problems with little to no modification of the learning algorithms.

Changes from the Proposal

- Instead of using standard Artificial Neural Networks, we switched to an SVM (Support Vector Machine) algorithm. This algorithm seems to work for our purposes and may serve as an interesting alternative to neural networks in many other similar cases
- The user interface has been redone slightly. Instead of always having a preferences window open in a corner of the screen, we have switched to using the Adime library (see http://admine.sf.net for details) for all settings dialogs. Adime works in Allegro, and looks much simpler and cleaner than implementing every bit of the dialog by hand. The change mostly means that we would have more time to concentrate on the AI of the robot. See the User Interface section for more information.
- While we were playing with the Allegro library in the early stages of the testing, we found that Allegro has native support for joysticks and gamepads. Therefore the robot could potentially control the game through a joystick.

[1] Marshall, Blank, and Meeden; An Emergent Framework for Self-Motivation in Developmental Robotics

[2] Oudeyer, P.-Y., F. Kaplan, and V. Hafner (2007). Intrinsic motivation systems for autonomous mental development. IEEE Transactions on Evolutionary Computation, 11(1):265–286.

After some preliminary experiments using SVMs, we were thrilled to see that the idea was going to work. Accuracy at predicting the game board soared as high as 95%. However, the process necessary to obtain these statistics took a considerable length of time and effort, was put together in multiple languages, and the snake could not begin prediction until the entire game had been played and the data was available. Thus we were faced with the problem of making the algorithm work *as* the snake was playing.

The answer to this problem came in slightly modified form of SVM, Online SVMs. Online SVMs are explained in more detail later in the paper, however the principle behind them is that you can modify what the robot has learned **during** a test. It can begin using new knowledge the instant the knowledge is obtained. With the Online SVM implementation, we were successfully able to put together a live game, which learned as the game was playing.

Prior Work

The concept of intrinsic motivation resulting from a desire to explore the world is not a recent thought. Robert White argued in essays as far back as 1959 that an internal drive exists beyond the traditional primary biological drives such as hunger and thirst. White called this drive "Competence Motivation," the idea that animals are driven by a constant internal need to interact more effectively with their environment[1]. Combining this need to continually learn how to more effectively manipulate the environment along with the previous definition by Hawkings[2], creates a theory that developing true intelligence is possible by a system that continually desires at least in part to explore the world and find situations from which it can learn to better predict the consequences of its actions. Such a robot fulfills the ideals of both White's and Hawking's theories.

Despite the relative simplicity of this concept, only in the last 20 years have scientists such as Oudeyer[3], Marshall[4], and Schidhuber[5] gone beyond thought experiments and started to develop descriptive algorithms, working simulations, and tangible robots. What follows is a listing of many different strategies taken by robotics scientists to this concept of curiosity-based motivation in creating intelligent robots. The goal of this project is to further current research by looking at variations of approaches that are thought to be feasible, yet do not have a large number of papers detailing them and the behaviors they can produce currently.

A paper on Intelligent Adaptive Curiosity defines IAC[6] as "a drive which pushes the robot towards situations in which it maximizes its learning process. It makes the robot focus on situations which are neither too predictable nor too unpredictable." This idea will undoubtedly play a crucial role in our project.

The following algorithms (explained by Oudeyer and Kaplan[7]) may be seen as different approaches to how the robot may focus on desirable situations and each will be covered in greater detail in the algorithms section. The authors also compared many of the currently existing methodologies for intrinsic motivation, many of which are driven by curiosity. These ideas support the view that using this kind of exploration in robotics is both promising enough to be worth investigating and has room for novel research. From Oudeyer and Kaplan's descriptions and other rankings we have drawn three of the overall best and least researched curiosity algorithms.

---

[1] Robert W. White, Motivation reconsidered: The concept of competence, Psychological Review, 66:297–333, ISSN 0033-295X, DOI: 10.1037/h0040934.

[2] Hawkins, Jeff, and Sandra Blakeslee. *On Intelligence*. New York: Times, 2004. Print.

[3] Oudeyer, P.-Y., F. Kaplan, and V. Hafner (2007). Intrinsic motivation systems for autonomous mental development. IEEE Transactions on Evolutionary Computation, 11(1):265–286.

[4] Marshall, Blank, Kumar, and Meeden; Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture

[5] Schmidhuber, Jürgen. "Developmental Robotics, Optimal Artificial Curiosity, Creativity, Music, and the Fine Arts." Connection Science 18 (2006): 173-87.

[6] Oudeyer, P.-Y. and Kaplan, F. (2004). Intelligent adaptive curiosity: a source of self-development. In Proceedings of the 4th International Work-shop on Epigenetic Robotics, volume 117, pages 127–130.

[7] Oudeyer, P.-Y. and F. Kaplan. "What is intrinsic motivation? A typology of computational approaches", Frontiers Neurorobot., pp. 2007.

The simplest algorithm is Information Gain Motivation, which rewards a robot for learning. A typical implementation will check the accuracy of robot's predictions against the actual outcome of events, repeating behaviors which are more predictable than a specified lower bound but yet less predictable than a certain upper bound. The simplest way to do this is to take the ratio of successful predictions over total predictions to create a percentage that is easily bounded.

A slightly more complex algorithm called Learning Progress Motivation[1] favors learning to complete a task more efficiently by either solving a problem faster. One problem however, is the definition of efficiency which requires quantifying a measure of progress.

[1] Stout, A. and Barto, A., Competence Progress Intrinsic Motivation

The Game

We have defined a simple version of the game Snake to use as a standard test for our framework. It is very similar to the game traditionally played in arcades around the world and popularized by mobile phones, but with modifications to make algorithmic prediction of the game environment easier and accidental death by the snake more difficult. Traditionally, the game has the following properties:

Traditional game rules

- To start, the snake is a single block in the middle of the screen
- The snake can move in the four cardinal directions
- The game consists of a n * n square grid with each square being snake, apple, or blank
- There is always one apple on the grid, and only one apple
- “Eating” an apple increases the length of the snake by one
- Moving the head of the snake into any of the body segments results in death
- The edge of the board is surrounded by walls.
- If the snake moves into a wall, death occurs.
- Depending on the version of the game, if you lose a die the game is over. Sometimes there is an exception where the snake has three or five lives.
- Points are awarded for eating apples.

Modifications

The main modification we have made to make the game easier for a robot to explore is to remove the boundaries at the edges of the board, replacing them with “wraparound”. As the snake leaves one side of the playing field, it emerges from the opposing side heading in the same direction. This reduces the number of cases where death is possible, even to the point of death being impossible with a snake having less than five segments. This gives the robot a chance to explore the affordances of the game world without being reset continually by hitting an edge. However, as playing progresses to a snake of length five, the death affordance finally reappears with the ability to contort the snake in such a way as to hit its own tail and die. Thus this modification does not change the game in the long term, but is designed to give the robot time to learn in a safe environment and start to exhibit observable learning behavior.

Other modifications were made in attempts to simplify the game to better support learning and allow us to prove the generality of our approach to learning by testing the same algorithms against a variety of combinations. In all cases these modifications are interchangeable. The first modification is related to the movement behaviors of the snake. Instead of the traditional selection of a cardinal direction such as done with a joystick for movement, direction can be more simply controlled by selecting a turn left, turn right, or continue in the same direction behavior. The second modification is changing the view of the

game board. Instead of an absolute view looking over the top of the playing field, with the snake moving about inside, the view is instead centered over the snake's head, meaning that the center tile is always occupied by the snake's head.

All of these modifications were made to simplify the game during development and testing, and any or all could be disabled or configured to make the game as difficult to play as possible. The learning algorithms should be able to adapt and learn the unpredictability which results from hitting an apple tile. Even the affordance of death provided by turning off the "wrap-around" feature and re-enabled the traditional boundary walls would quickly be learned to be avoided, as running into them would produce a very predictable state, because in death the snake is always reset to the middle of the screen. Since very predictable events are defined as uninteresting, they are not repeated if another more interesting behavior is available and thus the snake would learn to not hit walls after some exploration.

Learning

The paper "*What is intrinsic motivation? A typology of computational approaches*"[1] provides an excellent overview of the various methodologies for implementing intrinsically motivated learning. The paper also supports our view that these kinds of exploration are both promising enough to be worth investigating and new enough to have room for novel research. Near the end of the paper, the authors rank each of the examined motivation algorithms according to various criteria. From this we have drawn the three overall best and least explored methodologies.

The first algorithm is called Information Gain Motivation. This algorithm attempts to simulate the basic pleasure of learning. Information Gain Motivation is useful in a few niche cases, but it is usually better if combined with another algorithm. One reason Information Gain Motivation can be useful is that it can be measured simply by checking the robot's predictions against real events. This will give solid evidence that the robot is learning. For more information on Information Gain Motivation, see Fedorov's 1972 paper on the subject[2] as well as the paper by Roy and McCallum[3] from 2001.

The second algorithm, Learning Progress Motivation, rewards the robot for doing a task more efficiently, either by solving a problem faster or learning to perform a task. The main road block in LPM is the difficulty in quantifying "progress". We can simplify this problem by only looking at a single context, vision. Two papers dealing with this algorithm are the paper by Oudeyer et al. (*Mean error for last tau contexts,* 2007) and the paper by Schmidhuber, (*Simply compare prediction of current state before and after a training*, 1991).

The third and final algorithm is Competence Progress Maximizing - also known as Flow Motivation. As an example, imagine you are working on a problem and are going along swiftly without any hitches; you are said to be "In the flow". When you are "In the flow" a single distraction can completely destroy your concentration and stop the tremendous burst of speed. Afterwards, it is difficult to re-achieve this state. As a possible extension to CPM we could simply encourage the robot by larger and larger amounts the longer it keeps making the right decisions. When it fails the streak is broken and it is no longer encouraged. Rewarding the snake in the way makes it favor sequences of optimal decisions.

The main focus of this project was and is Intrinsic Motivation, a combination of the algorithms above but motivated by an inside force. Intrinsic Motivation is when an agent does something *for the sake of doing it*, rather than because it is encouraged by another force[4]. If a human decides to learn Pig Latin for no reason at all, that is an instance of Intrinsic Motivation. No-one is forcing the person to spend time learning Pig Latin, or in a child's case, to play with objects and interact with the environment. Such a person or child is quite simply learning for the sake of enjoyment gained from the process of exploration and learning itself. They are learning

---

[1] Oudeyer, P.-Y. and Kaplan, F. (2004). Intelligent adaptive curiosity: a source of self-development. In Proceedings of the 4th International Work-shop on Epigenetic Robotics, volume 117, pages 127–130.

[2] Roy and A. McCallum, "Towards optimal active learning through sampling estimation of error reduction," in Proc. 18th Int. Conf. Mach. Learn., 2001, pp. 441–448.

[3] Fedorov, Theory of Optimal Experiment. New York, NY: Academic, 1972.

[4] Oudeyer, P.-Y. and F. Kaplan. "What is intrinsic motivation? A typology of computational approaches", Frontiers Neurorobot., pp. 2007.

for the very sake of learning, only unintentionally does learning such knowledge come in useful later in life.

We tried to recreate this in our program. The snake utilizes a “confusion horizon” or intermediate level of novelty, which simply means it concentrates on the point between what is too confusing and what is already well known. It is a very useful technique in robotics, as it keeps your robot on track and focused on what it needs to learn.

The snake originally starts by simply learning to predict the movements of its own body, which originally pose considerable difficulty for the snake. When this has been accomplished, it continues to learn and predict everything else about the game. It concentrates first on the most easily predictable part of the game, itself. Because it controls its own body, it can quite easily predict itself after a few trials.

While this works reasonably well, this would not be a complete paper without a section about the other theories involved in computational thought. These theories have been proven to work for simple tasks.

Reinforcement Learning

The first of these theories is Reinforcement Learning. There are many papers on reinforcement learning, and unfortunately we cannot cover all of them here.

The principle behind reinforcement learning is that the AI algorithm is” rewarded” when it chooses the correct answer, and “punished” when it is incorrect [citation needed]. Many algorithms have shown remarkable success using this method, comparable to traditional learning algorithms such as neural networks.

Design

During the creation of our program, we tried to structure the program cleanly and create a class based structure which can easily be extended to other simulated environments or even to accept input from sensors and control robots embodied in the physical world. The project was written in C++, using the Allegro game development library[1]. Subversion was used in order to facilitate collaboration, and makefiles were used to compilation. For a complete list of the development software please see the acknowledgements section in the conclusion at the end of this report.

Following is a description of the class structure in the project:

- Body class
    - Represents an embodied thing with sensors fed by its environment and behaviors which it can perform that affect its immediate environment in some way over time
- Sensor class
    - Abstraction for representing sensory inputs from a simulated or physical body
- Snake class
    - An implementation of Body which keeps track of the game board (its simulated environment) and has simple movement behaviors which can be carried out
    - Has sensors for all parts of game state, most important are the grid of game board tiles represented as float values and the status of each possible movement behavior as separate float values
- Predictor Class
    - Once a set of Sensors is attached, the predictor can be trained at each step of the simulation to better predict the current state based on previous states it has seen
    - Also at each step the current prediction of the next state can be querried
- OnlineSVRPredictor Class
    - Implementation of Predictor which trains a Support Vector Regression machine to predict each sensor value at each step of the simulation.
- snake-babbler
    - Selects a randomized behavior to perform at each step of the simulation
    - Outputs game board state each step for testing and training predictors offline
- snake-humanplayer

[1]The Allegro game development library is an open source set of tools which allow drawing on the screen and interpreting user input in a variety of forms. See alleg.sf.net

        - Normal keyboard or joystick based control
        - Eventually to be used to run a snake simulation while the robot controls it via its arm moving a joystick
- snake-predictor-svr
        - Uses the OnlineSVRPredictor to compute prediction error at each step of the simulation, and seeks to repeat behaviors which previously resulted in prediction errors above a certain threshold.

An important technical note to make is that the robot's learning algorithms are unaware of any meaning attached to the different sensor inputs or behaviors to perform, knowing only the raw data values over time. This supports our goal of a generalizable intelligence.

- The behaviors the Snake can perform
    - The snake has many behaviors, but there are two sets. The primary reason for having two sets is for comparison purposes, as comparison of various algorithms is an important part of our project.
    - Set 1:
        - The cardinal directions (north, west, south, and east).
    - Set 2:
        - Relative turning (straight ahead, left, and right).

We considered adding other, more complex behaviors such as those listed below:

- Moving to the west wall and then moving north.
- Moving south three squares and then moving east.
- Continue in the current direction until you wrap around.

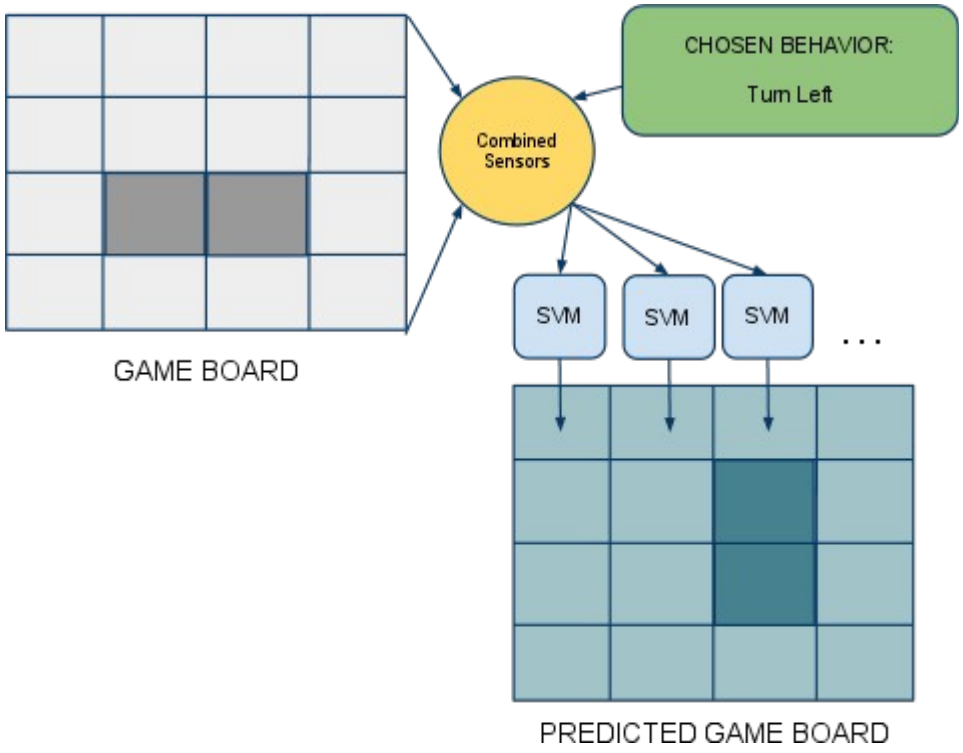
Due to time constraints we were unable to implement these and other similar complex behaviors in the project, however they would be relatively easy to implement in our framework and make for a interesting future experiments.

Algorithms

Early on we were encouraged to explore alternatives to exclusively relying on artificial neural networks for the project. In all previous work seen on the subject, training neural networks was the only method extensively used. So before beginning implementation of of prediction algorithms, we first examined other possible methods for predicting future states of the environment. An algorithm which came up many times in unrelated research was Support Vector Machines. They are similar to neural networks in that they prefer as inputs a vector of numbers scaled to be between zero and one which describe the major features of a problem and can be trained to give different outputs corresponding to different combinations of inputs. They differ in two major ways however. First, SVMs are simple in their operation, there are concrete mathematical functions called kernels that re-map the inputs into higher dimensional

space that makes it easy to draw a single hyperplane separating combinations of inputs into two disjoint regions. This is the second difference, a SVM has only a single binary output. A simple workaround is to create multiple SVMs, one for each possible kind of output. [1] This still does not allow for precise prediction of continuous data as might be seen in predictions of future sensorimotor flow from sensors embodied in the real world. However, by dividing such continuous values into a finite number of discrete regions it becomes possible to try SVMs in an application where neural networks may be used. Neural networks still work very differently, especially when the concept of hidden layers is introduced, but as seen in our preliminary results, in environments with relatively simple consequences for actions performed, SVMs may perform very well.

In our environment, each tile is represented by two binary features, one for the presence of an apple and one for the presence of the snake body. With a neural network this means a single network with double the number of tiles of outputs. With SVMs this means training a large number of separate vector machines, two for each tile. An example showing only inputs and predictions of the snake body is shown in the following diagram.



Before implementing an online learner which could learn to play snake in real time, we first used data from a random movement babbler to train neural networks and SVMs to test prediction accuracies. Both game good results, but we were more interested in the simplicity, speed and high accuracies that resulted from the SVM predictors for our simple game of snake. In the results section there are exact numbers for a five-by-five simulation. To be short, then results were excellent, with less than 5% nominal prediction error for each tile.

[1] "Multiclass SVMs." *The Stanford NLP (Natural Language Processing) Group*. Web. <http://nlp.stanford.edu/IR-book/html/htmledition/multiclass-svms-1.html>.

These promising results from offline training of SVMs encouraged further experimentation with training SVMs as the game was played in real-time. Sadly, there is currently a lack of such online SVM implementations. The Shogun toolkit and LaSVM were the only well developed projects we were able to find, and both seem far to complicated for a half-semester project. Instead a similar library, OnlineSVR was used to implement online learning. The main difference is that OnlineSVR uses incremental regression so as to reduce the time for adding each successive sample in a real-time training situation and outputs a continuous value instead of a binary classification. Otherwise using this library did not change the design of our project, and in theory either the Shogun toolkit or LaSVM could be used to develop alternate predictor which could be substituted for the OnlineSVR based one we have developed.

User Interface

The snake-predictor-svr module is the keystone of the project. When it starts, there is a snake of length one. The left side of the screen is the game board, the right side of the screen is the prediction.

The prediction is not a binary prediction due to the regression nature of OnlineSVR. Different shades of gray indicate how strongly it predicts the snake is in that location. If a square is completely white, the robot is predicting that the snake is there, with high confidence. If a square is light gray, then it means the robot is fair sure the snake is there. And if the square is black, it means the robot is very confident the snake is not there. The predictor is also able to predict the location of apples with a very high accuracy due to their lack of movement.

Hitting the <TAB> key will bring up a settings dialog which allows you to make changes to the game settings. Note that changing anything will cause a total reset of the game.

In the menu you can change:

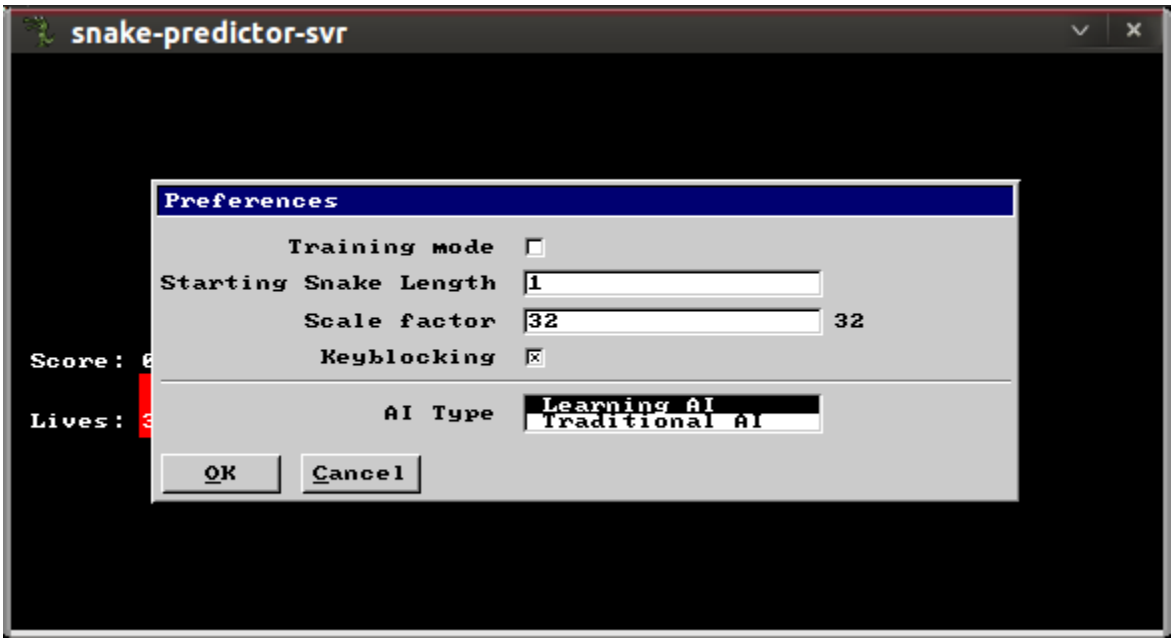
- Initial snake size (longer sizes will take longer to compute)
- Whether the learning algorithm is set to active
- What type of algorithm the snake is using
- The scale factor to draw the game board at. Currently the game board is 5 x 5 tiles
- Whether the program should require a keypress to continue each step of the simulation

The <T> key allows you to quickly toggle the learning algorithm on/off. Running the learning algorithm is very taxing on the system, and the longer the snake gets the more draining the program is. Using the T key allows you to have the snake learn for a while, and then run just using the knowledge it has learned. The snake will move much faster than with learning enabled.

The knowledge of the robot is saved in model files for the support vector machines and lists of past accuracies for certain states to be loaded the next time the program runs. By deleting the data files in the program's directory the snake will start from scratch and effectively forget all predictions and results learned from previous runs the next time the program runs.

Because we wished to give adequate time for the user to analyze each frame, the process is halted every frame. Pushing any key will continue. This option can be disabled in the

preferences dialog.



Graphical Settings Dialog, accessed via <TAB> key

Results

When we had finished debugging all of the compile time errors and activated the snake for the first time, it immediately began wandering around the game board. When it bumped into an apple, it immediately targeted that behavior of moving left as highly beneficial, and repeatedly ran across the screen leftwards, not recognizing the apple had reappeared somewhere else.

This was mostly due to a bug in prediction accuracies needing to be associated not only with the behavior which led to the large prediction errors, but also with the state which led to the the large prediction error. After adding this most interesting behavior started to emerge. In the basic state we were able to accomplish, the snake will look at each possible behavior at its current state, and will choose a previously tried action if it resulted in a not well predicted state. Otherwise it will choose an action at random. This simplistic behavior selection is enough to produce interesting patterns. For instance, the snake will rarely repeat actions in similar situations and will sometimes be temporarily stuck in cyclic loops as it attempts to learn the outcomes of a certain series of movements. This is most evident when learning is disabled but behaviors are still chosen based on results of past predictions. When the snake becomes stuck in such a cyclic loop, it is unable to learn and eventually extinguish the looping sequence and so is stuck in an infinite loop. This simple selection of behaviors is very similar to the intermediate and high novelty motivation algorithms proposed by Oudeyer[1], and would be easy to extend in the future to implement any of the other computational intrinsic motivations described in that 2007 paper[2].

Prediction Accuracy

The following table lists accuracies for predicting the absence/presence of the snake in each tile of a five by five game grid given the set of previous tile states and behavior performed. Prediction was done using libSVM to train support vector machines for each tile on a dataset of 10,000 simulation steps performed by a random movement babbler. Note that the worst score was 80.41% and the best score was 96.01%. Further examination of the raw data showed that the snake spent approximately one-quarter of the steps at each length one, two, three and four. Once reaching size five the snake is able to run into its own tail, and thus spends less than 2% of steps at length 5 or above before dying and being reset to length one. Slightly more than 2.5 tiles are occupied by the snake on average during a simulation, or about 10% of them on a five by five grid. The 4.21% median prediction error is well below this margin, and outside of the two 80% accuracies, all tiles were actually successfully predicted more than 95% of the time, meaning even in this rough test we were able to obtain accuracy errors half of that obtained by the best guessing strategy, that of assuming all tiles are blank all the time. This difference is significant enough to support the use of SVMs as accurate predictors, at least in our test case.

80.79%	95.78%	96.01%	95.77%	95.92%

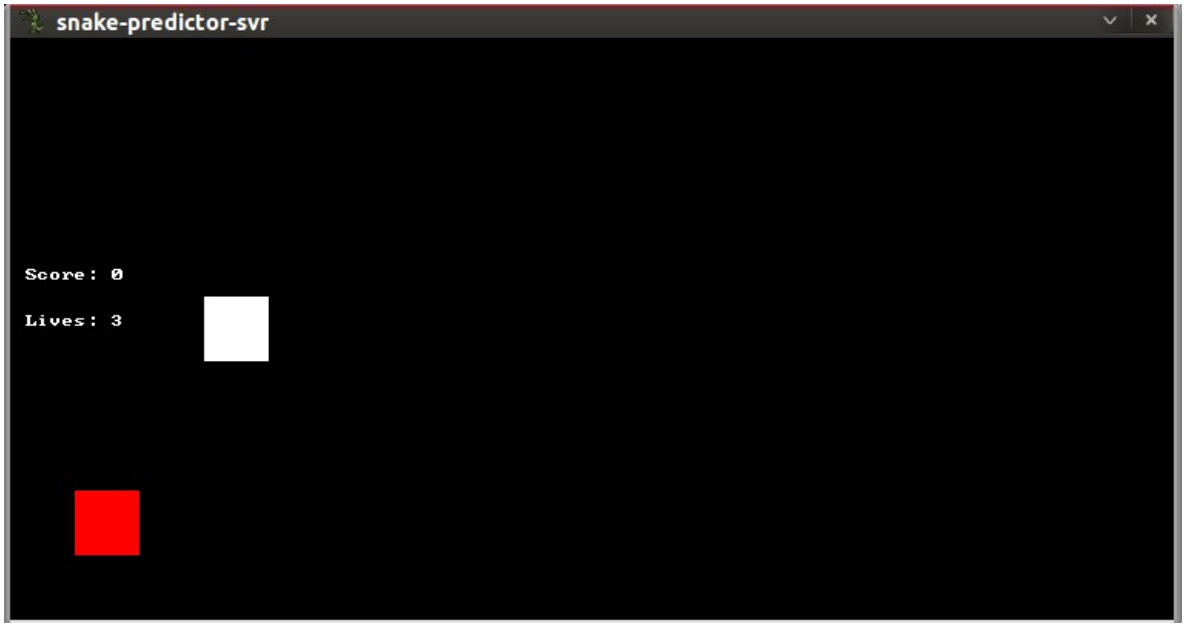
[1]Oudeyer, P.-Y. and F. Kaplan. "What is intrinsic motivation? A typology of computational approaches", Frontiers Neurorobot., pp. 2007.

[2]Oudeyer, P.-Y. and F. Kaplan. "What is intrinsic motivation? A typology of computational approaches", Frontiers Neurorobot., pp. 2007.

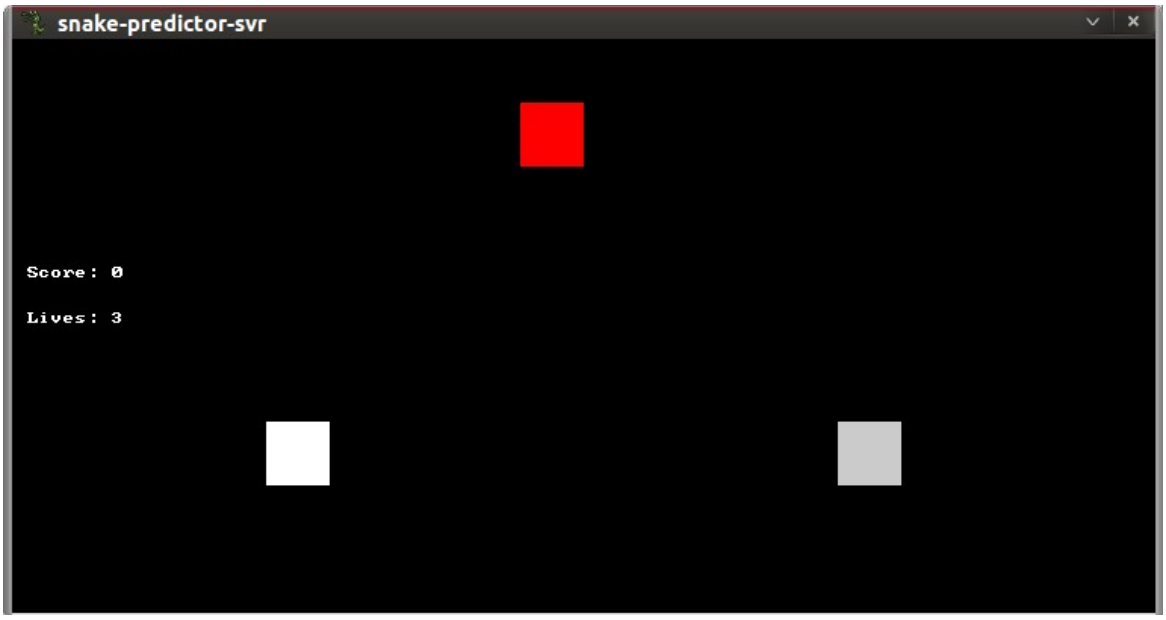
80.41%	95.95%	95.36%	95.91%	95.73%
95.77%	95.74%	95.80%	95.84%	96.00%
96.25%	95.72%	95.28%	95.79%	95.88%
95.96%	95.73%	95.77%	95.51%	95.83%

Screenshots

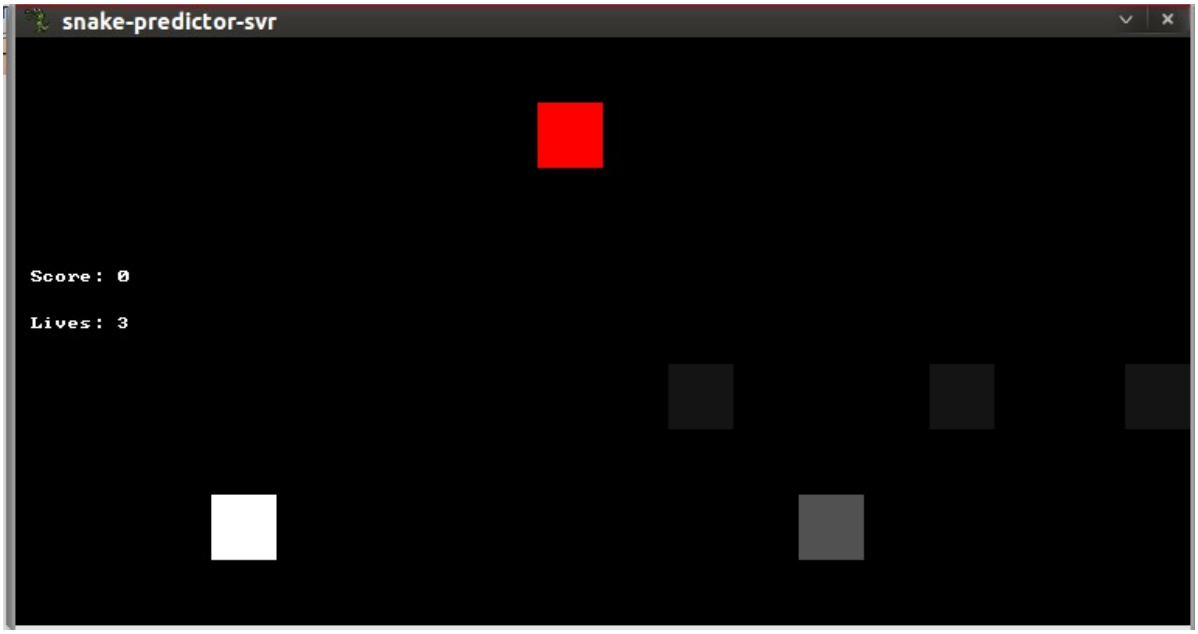
The following are a selection of sequential screenshots from the final program to highlight steps of the prediction and learning process:



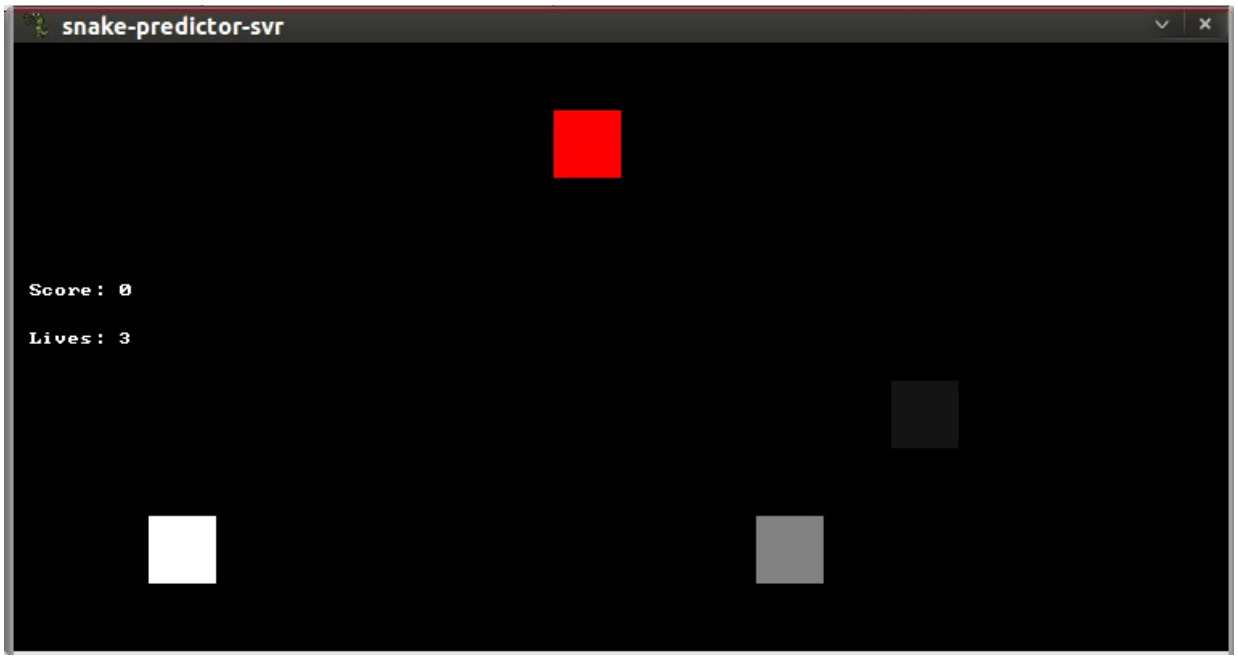
In this screenshot you can see the length one snake (the white square) the apple (the red square) and some text giving basic details of the game state. Prediction and learning is not enabled.



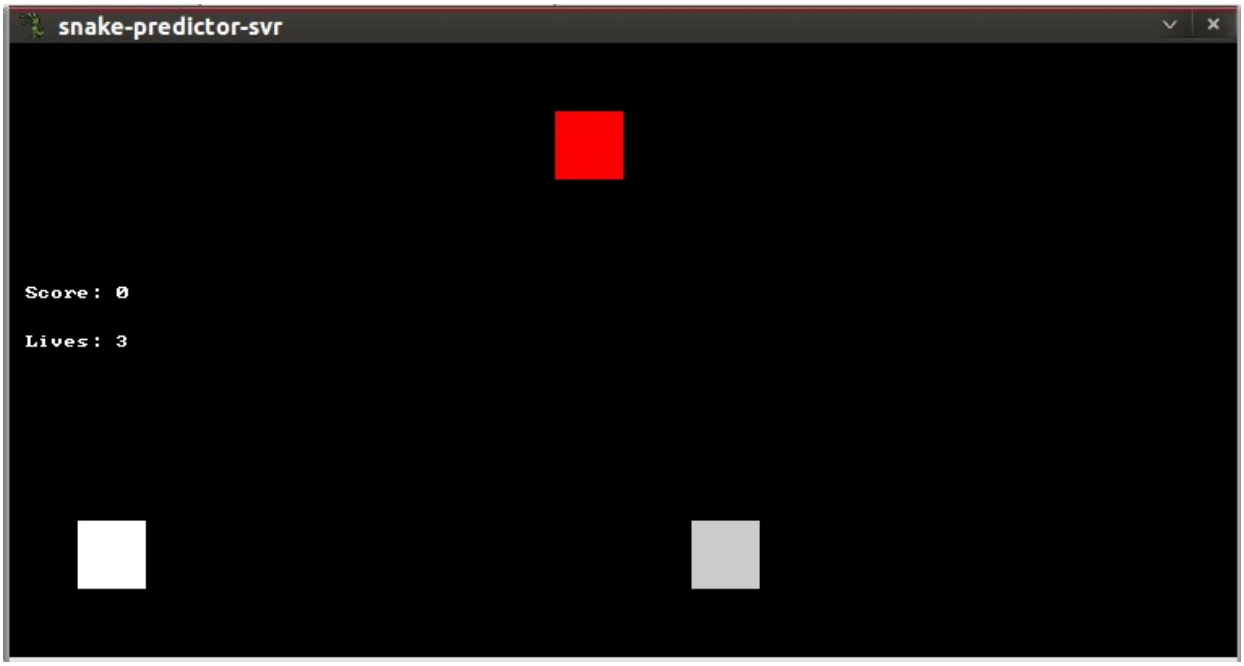
In this screenshot prediction and learning are enabled, the predicted game board appears to the right of the actual. The light gray color indicated a high confidence in the prediction in that tile.



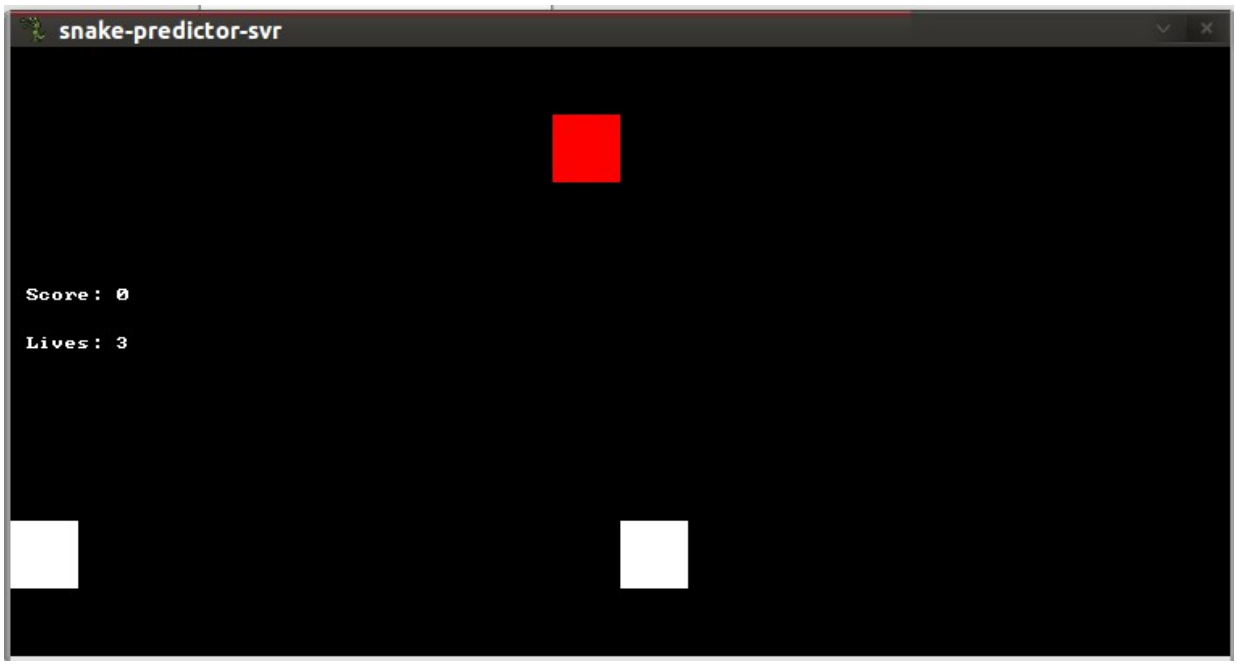
In this frame, multiple tiles have predicted the presence of snake body, but inspection shows the lightest tile is in fact correctly positioned. This is a situation our algorithm would be likely to re-visit, as it has a relatively high prediction error rate.



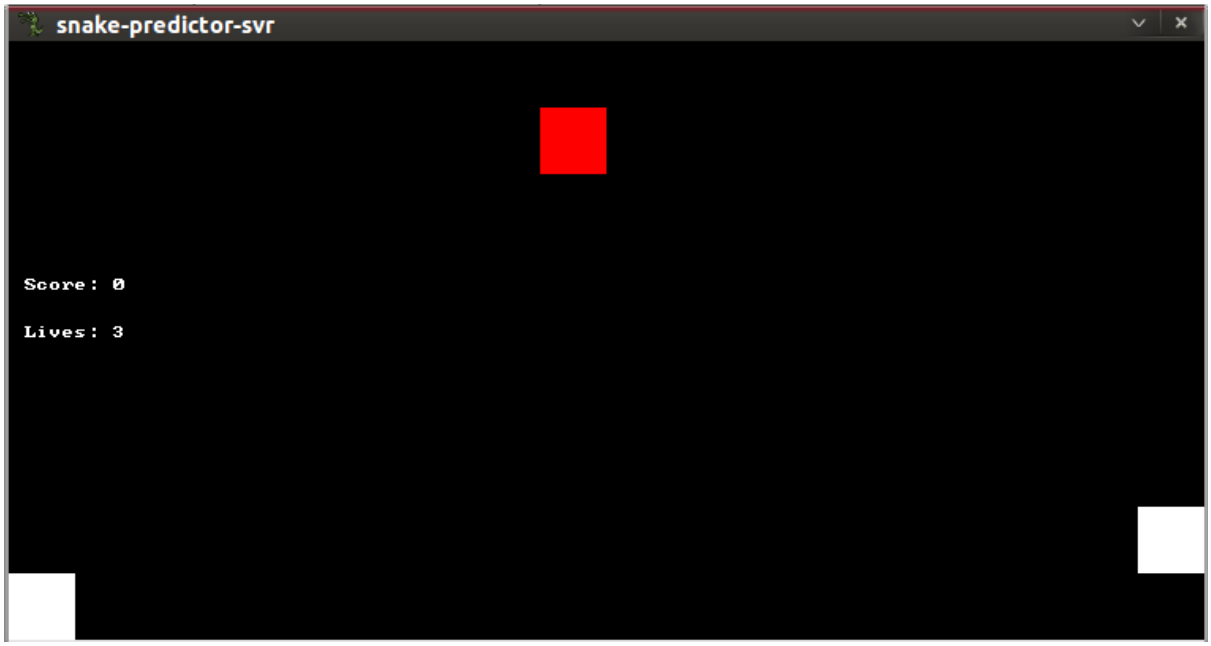
The snake has almost reached the edge and the predictor is still performing well.



The algorithm has almost perfect confidence and 100% accuracy. It has nearly reached the edge.



Prediction is nearly perfect in this frame.



In this case the snake previously was moved left and "wrapped" around to the opposing side, as can be seen from the very confident prediction of such a result. However, the learning algorithm remembered this previous accurate prediction given such movement and chose a different behavior to try, in this case moving south. When encountering this situation again, the learning algorithm would likely choose to repeat this south-ward behavior until the proper outcome is learned.

Code Versioning Statistics

Alexander committed many smaller bug patches and design improvements, contributing to his relatively large number of commits. Both Daniel and Chad contributed relatively large feature commits with Chad implementing much of the snake simulation code and Daniel implementing the majority of the prediction and learning algorithms.

Name	**Number of Commits**
Chad Nelson	13
Daniel Stiner	35
Alexander Campbell	64

Conclusion

While current research in machine learning lacks generality, we have found that this is likely because creating methodologies capable of generalized learning in any situation is no easy task. The focus on solving specific problems by extracting and using statistical properties in order produce classifications or other outputs has yielded useful and interesting results. However this process includes large amounts of manual training and tuning of systems to produce accurate and useful results. This takes valuable time and resources, and does not provide a development framework for future projects.

We have researched and implemented a basic version of an alternative methodology. One that allows a learning machine to choose behaviors and goals to maximize its ability to learn about the environment and the ways it can be manipulated. The biggest advantage of such internally motivated learning is that it requires no supervision to train. We were not successful in proving the effectiveness of such learning motivations but were able to provide some novel approaches to the problem in our use of support vector machines to predict future states. Also, we implemented our project in such a way as to maximize flexibility to allow future variations or applications to similar problems.

Future work

In the project proposal we discussed what might be fun to add to the project in the future. During the creation of the project, we found a great many opportunities for experimentation and it was unfortunate we only had a semester to explore the possibilities inherent in learning even as simple a game as Snake.

What follows is a brief table of a few of the improvements and offshoots we would make given the time:

- Add more behaviors to the Snake, as mentioned in the Structure/Design section.
- Apply these intrinsically motivated learning strategies to interacting with objects to allow the robot to decide when it had successfully categorized an object based on its properties and is ready to play with a novel one
- Make a physical robot play Snake from a computer screen using the OpenCV library.
- Make a physical robot play the game from a physical screen, but this time with a joystick. Previous experimentation by Pasha Kazatsker in the Developmental Robotics Lab has actually accomplished this for controlling the pong game.
- Make two physical robots or two arm of the same robot play a 2-person game like pong against each other with joysticks.
- Make the robot play a similar, but slightly different game. Our project would generalize nicely to any grid-based game such as tetris, pong, or pac-man.
- Make the robot play a completely different game running at a low resolution, and treat the pixels of the screen as the game board. An example of such a game would be Wolfenstein 3D.

Some things we did do to make our program exceed expectations:

- We made the user interface clean, simple, and easy to understand for even novice computer users.
- The program can be customized in many ways through the use of the preferences dialog box.
- We wrote it entirely using free, open source software.

There are many different ways to approach and further expand upon this project, but we feel we've taken advantage of the short time we had to put together a solid project. Thanks for taking the time to read and review our final project report!

Acknowledgements

We would like to thank the following parties and projects for their contributions and work:

- Dr. Alexander Stoytchev, for assistance in planning the project and for sharing knowledge from the field of developmental robotics.
- The anonymous reviewers of our original proposal who suggested improvements
- The LibSVM and OnlineSVR (http://onlinesvr.altervista.org/) packages. These were the distinguishing features of our project.
- The GNU C++ Compiler.
- The Allegro game development library.
- The Adime library for Allegro GUIs.
- The Make project.
- The Subversion project

Bibliography

[1]Marshall, Blank, and Meeden; An Emergent Framework for Self-Motivation in Developmental Robotics

[2]Oudeyer, P.-Y., F. Kaplan, and V. Hafner (2007). Intrinsic motivation systems for autonomous mental development. IEEE Transactions on Evolutionary Computation, 11(1):265–286.

[3] Robert W. White, Motivation reconsidered: The concept of competence, Psychological Review, 66:297–333, ISSN 0033-295X, DOI: 10.1037/h0040934.

[4] Hawkins, Jeff, and Sandra Blakeslee. On Intelligence. New York: Times, 2004. Print.

[5] Oudeyer, P.-Y., F. Kaplan, and V. Hafner (2007). Intrinsic motivation systems for autonomous mental development. IEEE Transactions on Evolutionary Computation, 11(1):265–286.

[6] Marshall, Blank, Kumar, and Meeden; Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture

[7] Schmidhuber, Jürgen. "Developmental Robotics, Optimal Artificial Curiosity, Creativity, Music, and the Fine Arts." Connection Science 18 (2006): 173-87.

[8] Oudeyer, P.-Y. and Kaplan, F. (2004). Intelligent adaptive curiosity: a source of self-development. In Proceedings of the 4th International Work-shop on Epigenetic Robotics, volume 117, pages 127–130.

[9] Oudeyer, P.-Y. and F. Kaplan. "What is intrinsic motivation? A typology of computational approaches", Frontiers Neurorobot., pp. 2007.

[10]Stout, A. and Barto, A., Competence Progress Intrinsic Motivation

[11]Oudeyer, P.-Y. and Kaplan, F. (2004). Intelligent adaptive curiosity: a source of self-development. In Proceedings of the 4th International Work-shop on Epigenetic Robotics, volume 117, pages 127–130.

[12]Roy and A. McCallum, “Towards optimal active learning through sampling estimation of error reduction,” in Proc. 18th Int. Conf. Mach. Learn., 2001, pp. 441–448.

[13]Fedorov, Theory of Optimal Experiment. New York, NY: Academic, 1972.

[14]Oudeyer, P.-Y. and F. Kaplan. "What is intrinsic motivation? A typology of computational approaches", Frontiers Neurorobot., pp. 2007.

[15]The Allegro game development library is an open source set of tools which allow drawing on the screen and interpreting user input in a variety of forms. See alleg.sf.net

[16] "Multiclass SVMs." The Stanford NLP (Natural Language Processing) Group