Lucas Wilson

**Bibliography**

Sutton, Richard S., and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Chapter 6: Temporal-Difference Learning. Second edition, complete draft. 5 Nov 2017.

**Summary**

The 6th chapter of Sutton and Barto’s *Reinforcement Learning: An Introduction* contains relevant information about Temporal-Difference Learning (TD) in the context of reinforcement learning (RL), specifically in contrast to Monte Carlo and dynamic programming (DP) methods. The reinforcement problem is defined by some policy where moving between states by some action yields a value. The goal is to create a model to predict such values given state and available actions. Then, the action with optimal predicted reward can be chosen. TD, DP and Monte Carlo are methods which help the model converge to such a value function.

DP can calculate reward by adding the reward and expected value of the next state. It therefore doesn’t require reaching the terminal state to predict the value of the current state. However, it does require the ability to know the value of the next state pair, which the untrained model doesn’t have. Monte Carlo is capable of building such a model with approximations from sampling, but it requires reaching the terminal state to fully approximate the value. TD combines DP and Monte Carlo to solve the issues with each individually. Monte Carlo collects approximation values to be used by DP. The approximations become increasingly accurate with more sampling, and DP eliminates the requirement to reach terminal states.

They discuss several advantages of TD. TD doesn’t require a model of the environment (DP does) since it uses samples to create the distributions of the model. The model is incrementally built which allows for long episodes, which Monte Carlo would fail with. While both TD and Monte Carlos converge to the correct value function, TD seems to converge quicker for stochastic tasks. The reason for this is that TD better approximates the true certainty-equivalence. When batching is used, TD calculates it exactly. Batching collects increments and state-value pairs without updating the value function. Once the batch is collected, the value function is updated, and new increment values are calculated. The value function is updated with the new increments until the value function converges. They point out that non-batch methods don’t achieve certainty-equivalence or the minimum squared-error estimates, so batching performs better than non-batching, in some cases. For large state spaces, the certainty-equivalence solution is too complex to calculate, so the TD approximation becomes very useful in this case.

Finally, they discuss the SARSA, on-policy version of TD. On-policy means the current policy is applied to future values. This affects the value of states since it is assumed which action is taken. Instead of updating the current state, the current state-action pair is valued and updated with the next state-action pair’s value. SARSA comes from the state-action tuple of state, action, reward, next state, and next action. In the update function of SARSA, the value of the next state-action pair is used (the action chosen by policy); in the variations of SARSA, this is modified to be something else. Q-learning uses off-policy where it is replaced with the action which maximizes the expected value. Expected SARSA replaces it with the sum of the probabilities of an action at a state multiplied by its value, incorporating all values. The final is double learning which solves a model’s tendency towards positive values. Two Q-learning functions are used instead of just one. Which Q function is selected for action selection is chosen at random.

**Relation to Project**

This paper (including other chapters which I didn’t read or summarize) is extremely useful as a starting guide when it comes to reinforcement learning. It explains the different methods which can be used to update a utility function as well as their strengths and benefits. I can see how it could improve my project in terms of how I design my reinforcement agent. Although it doesn’t talk about how the Q-function being a neural network applies, it is still useful for understanding the algorithm. The algorithm used in my project was a very specific form of TD: learning rate and gamma were one. This paper implies that my RL agent could potentially perform better, if I used different learning rates and gamma values. Although, it still performs better than Monte Carlo. Given the nature of chaos, using different gamma values would probably be very useful since the current state has little effect on future states (assuming the agent interacts with the system in between, preserving the deterministic property of chaos).

Caballero, Lorena, Bob Hodge, and Sergio Hernandez. *Conway’s “Game of Life” and the Epigenetic Principle*.

**Summary**

Conway’s Game of Life (GoL) was created in 1970. It is played by creating some initial conditions and observing the results as the rules of survival, deaths and births are applied each generation. This complex system has emergent properties: “still life” is a stable, fixed structure; “oscillators” are cyclical structures; and “moves” are structures which move across the board.

Caballero, Hodge and Hernandez argue that his rules correspond to epigenetic rules. Von Neumann theorized that we can create universal constructors (a structure which can create more of itself). Watson and Crick’s theories on DNA replication turned out to be very similar to Neumann’s theory. There are structures in DNA which represent genes, and there are structures which regulate genes and duplicate DNA (which satisfies Neumann’s theory).Conway’s GoL inspired the theory of cellular automation, and Caballero, et al, argue that emergent behavior of GoL can also simulate the epigenetic properties like those found in DNA. Their argument is heavily based in Alan Turing’s work in biology as well as his work in computer science.

They point out four commonalities between GoL and Turing’s work. First, GoL produces “morphogens” which includes genes and other factors which affect the form of the overall system. Second, genes are cabalistic and remain unchanged after altering the system. Oscillators in GoL have this behavior. Third, Turning discusses the idea of creating or inhibiting morphogens, and the idea of creating and destroying cells is prominent in GoL. Fourth, there are smaller structures of these morphogens which Turning describes, and they correspond to similar structures in GoL. Turing has “stationary” and GoL has “still life.” Turing describes a periodic stationary as having longer wavelengths the longer the time period is. This corresponds to GoL’s “oscillators”. Oscillators with longer oscillation periods correspond with stationary morphogens with longer wave lengths.

To further these parallels, Edelman works in the field of topobiology, and he developed the field of epigenetic theory. Combining the work of Newton (classical physics) and Turing, he proposed 5 primary processes to describe the genetic and epigenetic process. Caballero, et al, describes these 5 processes in terms of GoL. First, cell division is simulated by the birth rule of GoL. Second, cell death is implemented likewise. Third, cell movement is possible, although rare. Fourth, cell adhesion is represented by the 2x2 still life square of GoL. Fith, differentiation and induction references the emergent systems of GoL interacting with each other, which they demonstrate as possible.

They conclude with the argument that it is possible to simulate complex systems of biology. The parallels enable the exploration of potential theories, but it can’t be used to justify them. Using the theories laid out by Turing and emergent properties like those present in GoL, we can create these simulations.

**Relation to Project**

This article to some degree demonstrates the utility of the research in my project. GoL is a good model of cellular automation and as they point out, it has several other applications. While GoL isn’t directly applicable, it is similar to others systems which are. Given that, my project serves as a speculation of the potential usefulness my RL agent has when it comes to predicting or controlling chaotic, emergent properties of system like GoL and those discussed in the paper. This means that my project doesn’t have any direct value, but if applied to more advanced and closer-to-nature simulations, it could be very useful.

Gadaleta, Sabino, and Gerhard Dangelmayr. *Reinforcement Learning Chaos Control Using Value Sensitive Vector-Quantization*.

**Summary**

Gadaleta (who happens to be an old grad student for CSU’s math department and did work with Dr. Gerhard and Dr. Kirby) wrote a paper specifically about applying RL algorithms to control chaotic systems. They use a process called vector-quantization which reformats data before feeding it to the model. To apply this method, a set of codebooks are chosen. Codebooks are vectors in the same dimensional space as the state vector. The preprocessing of an input point is choosing a codebook closest to the input point. These codebook values are the state values the RL model sees. The goal of their RL agent is to stabilize the unstable, chaotic point. The observer then is defined to reward subsequent states being nearest to the same codebook points, i.e., the state is unchanged, stabilized. They use SARSA learning to update the Q-function and define the action utility function. An addition to the RL algorithm, a major purpose for their research, is that the codebook is not static. The RL agent can add codebooks using “the growing neural-gas algorithm of Fritzke.” In this algorithm, codebooks accumulate local distortion errors (when a codebook is selected as being the nearest neighbor, the error is calculated and added to the codebook’s record of previous error). Codebooks are then added in areas where this local distortion error is high. This ensures that the space of codebooks is dense for values of the state space.

To evaluate the RL algorithm, they use a logistic map in one dimensional space. They show that the generation of codebooks is sensitive to the values learned by the RL agent. The agent has improved performance from their control (a previous algorithm) and can successfully stabilize the chaotic problem. Because of the common successful application of solutions to chaotic problems to more dynamical systems, they also argue that this algorithm will also work well.

This paper also includes several useful references to other sources on chaos control.

**Relation to Project**

This project is essentially what I am trying to do, but it is much better. The difference is that they define a different RL agent algorithm than I do. Additionally, they study the stabilization problem given unstable systems. Their RL agent tries to prevent the effects of instability while attempting to keep the system in a static state. My RL agent doesn’t stabilize but rather guide. My system can move very far away from the original state since it was unstable and my RL makes no effort to keep it there. It instead tries to guide the state as it’s moving away from the previous state (as GoL iterations occur). It’s a slightly different approach.

I don’t think my RL agent would benefit from their use of codebooks given that the problem space is discrete. If, however, my RL agent was applied to a cellular automation system like GoL but was continuous, it likely would be useful. In chaotic spaces, states close together are not necessarily near each other in later states. However, as the algorithm progresses, the density of codebooks would potentially mirror the sensitivity of a system. Codebooks could also be denser in areas where attractors live (the tendency of a system to draw a specific shape such as the Lorenz butterfly is an attractor). I wonder if generating codebooks is a means of reducing a problem space to a discrete space such as GoL. In that case, my RL’s effectiveness in GoL would relate to its effectiveness in a continuous space with codebook generation. I don’t, however, test this in my project.

**For reference:**

Sutton, et al: Introduction to RL

Caballero, et al: GoL’s relation to Epigenetic Principle

Gadaleta, et al: RL chaotic control problem

**Similarities**

Both Gadaleta and Sutton cover useful aspects in designing an RL agent. From Sutton’s work, we can see that Gadaleta’s choice of a TD Q-function was likely a good one. If chaos is a larger field, then it is possible that we could find a citation for Gadaleta in Sutton’s compilation.

Caballero’s work is not very similar to Sutton’s, but it is similar to Gadaleta’s. Caballero discusses the idea of cellular automation and GoL and its relation to very chaotic systems in biology (emergent properties). Gadaleta discusses chaotic systems as well (unstable points in chaotic systems).

**Differences**

They vary in technicality. Sutton’s article was a compilation of current works regarding Reinforcement Learning. As such, it was very general. It didn’t focus much on the relation to chaos, as in Gadaleta’s, nor the applications in biology, as in Caballero. Sutton’s was also much more of a guide from many different sources than it was a report on a specific finding from his team.

Caballero’s article doesn’t perform an experimentation or a compilation, but it instead argues a definition of GoL and equates it to definitions in relation to the epigenetic principle. They don’t interact with GoL systems but work with definitions of it. This is different from research by experimentation.

Gadaleta’s work was attempting to repeat similar experimentation done by others, in regard to codebooks, RL and chaos. This is different from Sutton’s report without explanation and Caballero’s argument of a definition.

Caballero’s work would justify the importance of the specific fields of cellular automation. If a model can control or understand a cellular automation system well, then it may also control actual cellular systems well based on Caballero’s equation. Gadaleta’s work is from the other side: they attempt to prove that such a model can control or understand chaotic systems, such as cellular automation, well.