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.

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.

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

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 Turning and emergent properties like those present in GoL, we can create these simulations.

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

Similarities, Differences, and how they relate to my project.