Greg Thomas

Lochana Palayangoda

DSCI/STAT 8950

05 March 2025

Reinforcement Learning and Q-Learning

Theory

Reinforcement learning (RL) stems from a strategy with parenting or having a pet. When potty training a puppy, you give them positive reinforcement with rewards such as treats and/or praise. Likewise with a child, if they do their chores some parents give them money or allow them to do something they want. This same concept works extremely well for machine learning, through making decisions. While a puppy might be near-sited and only considering how to get the treat now, RL's goal is to maximize the final reward.

Some main concepts for RL are environment, state, action, policy, and reward. Sahin Ahmed uses Pacman to describe how RL is used, I plan to use the game Snake to demonstrate it similarly to how he did in "A Beginner's Guide to Reinforcement Learning". With the case of Snake, it is a game where the player's goal is to get the longest snake without colliding into itself. The environment for this example would be the game itself. The initial state being observed is the current location of the snake and where the power-up to grow the snake's length is located. There are four actions that can be taken, moving up, down, left, and right. The policy tells the machine what to do, so if a power-up is near move towards it. With this knowledge the machine is aware to move towards the power-up, because then an immediate reward will be received, the snake length increasing to two. As the snake was moving towards the power-up the state changed since the player's location is changing. However, the policy did not change, until the reward was gained, and a new policy was added. The policy addition becomes, if the snake is near itself, move away from itself. This is also an addition to the state that the machine has to be aware of. In general, this process is going to repeat itself until there is no longer a space for a power-up to spawn and the machine wins (Ahmed). With the game being won, the machine needs to use exploration and exploitation to maximize the reward.

With the previous example, let's say the way to reach the maximum reward is to consider beating the game with the least number of moves. Given that our policy states to stay away from collisions, other actions will have to be considered to reach the maximum reward, this is known as exploration. Once the best-known action is understood, the machine can repeatedly take that action while experimenting with others to determine the maximized reward (Ahmed).

Rewards themself can be positive, neutral, or negative. Additionally, not all rewards are of equal size. This requires the model to consider whether it wants to consider an immediate reward or a larger reward within a set number of actions.

Parameter Meanings

To help the model be most prepared for its environment, there are different functions and agents to consider. The first is a discount factor, which can be used to guide the machine to chase a larger reward in future actions. This is a value between zero and one, where values closer to one tell the machine it is better to chase after the larger future rewards, rather than the smaller immediate ones (Ahmed). To help guide actions for the machine two functions exist that consider the discount factor, value function and reward function. The value function considers the discount factor and current state of the environment to predict future states and what actions to take to gain rewards. The reward function provides feedback based on the reward received for an action (GeeksForGeeks).

There are also different types of Agents to consider: Value-Based, Policy-Based, Actor-Critic, Model-Free, and Model-Based. "Value-based agents use an action value function, such as the Q-value function. Then the agent chooses the action with the highest Q-value at each state" (Ahmed). This is considered Q-learning, where agents can explore and update Q-values in a Q-table based on which value is higher. For Q-learning the Q-table is created upon the initial step, where all values in the table are zero. Exploration is done based on a policy, like the greedy policy, where the larger immediate action is chosen, the value in the table is updated. This iteratively happens to determine an optimal path (GeeksForGeeks). Policy-based agents use probability distribution instead of value functions, then the policy is updated based on gradient optimization. This works well for continuous actions like a self-driving car. Actor-critic agents use a policy network to take actions and use a value function to estimate how good the action the policy network decides to take is. The policy network is the actor, while the value function is the critic that is providing criticism to the actor (Ahmed). Model-Free agents learn through pure exploration, by seeing what works well. Q-Learning is an example of this, where no concept of what is in the Q-table exists to model off. While AlphaZero is a famous example of a model-based agent that figured how to play chess based on the model of chess and playing games against itself (Aditya).

Applications

With the Snake game and Pacman, RL has an application in video games. Especially as a great tool for learning about RL. Like video games examples have been done with board games as well, there are notable examples with Chess like AlphaZero.

Some applications that are being used to benefit society are self-driving cars, energy management, and personalized shopping recommendations. With self-driving cars route-optimization can be done on the fly. In a situation where a path is closed due to roadwork, the car can figure out the next best route. Agents need to consider pedestrians, other cars, stoplights and stop signs to account for safe arrival (Ahmed). For energy management, a smart home thermostat can figure out what a temperature should be for an air conditioner based on temperature outside, and the schedule of homeowners. A reward is provided in the form of less electricity used or running less, while providing a comfortable living experience. The last example would be

personalized shopping recommendations, which is a technology used on popular websites to provide targeted advertisements. Clicks on these advertisements let the machine know users are interested, while taking note of other criteria on the pages that are being followed.

Kernel Density Estimation

Theory

Given a dataset that is independently distributed, one of the techniques I was taught for data analysis was to turn the dataset into a histogram. Likewise, Kernel Density Estimation(KDE) makes no assumptions about the data it is providing an estimate for. Infact the model can work with almost any set of univariate or bivariate data, where majority of histograms are univariate.

A kernel function is run against a singular point of data, to create a parabola for that point. This is repeated for every data point in the source data, and the summation of those results are known as the KDE for the dataset (Fessel). Another way to think about it would be to think about Legos. Every individual Lego piece is considered a data point, the instructions for the Lego set would be considered the kernel function, and the resulting structure would be the KDE (Drapala). KDE results in a smooth line that can be used to better understand data instead of using a histogram. When dealing with bivariate data, instead of a parabola rings end up getting created around a data point, which can be summed up to create a shape that can describe both variables (Waskom).

Parameter Meanings

The two main parameters that affect KDE are the chosen kernel function and the bandwidth. There are multiple different options that could be chosen for a kernel function, a few examples would be: Gaussian distribution, uniform distribution, triangular distribution, and Epanechnikov distribution. The two kernel functions that I noticed being used the most often were Gaussian(normal) distribution and Epanechnikov distribution. With larger datasets, the difference between kernel functions doesn't matter as much, since all of the kernels end up getting summed up (Drapala). It is important to mention that some kernel functions could produce fewer desirable results if the variable being measured is dependent on another variable. Bandwidth is the value the dictates how smooth the resulting KDE is. Smaller bandwidth values have more rigid KDEs that appear closer to histograms, while larger bandwidth values slowly remove the fluctuations in the dataset (Akinshin). It is important to avoid under smoothing or over smoothing. Different bandwidth algorithms exist such as: Silverman's rule of thumb, Biased cross-validation, and unbiased cross-validation. It is important to consider the distribution parameter that was chosen for the kernel function when picking a bandwidth algorithm. Gaussian distribution works better with Silverman's rule of thumb but might not work as well with biased cross-validation (Akinshin). There does not seem to be a general guideline for which bandwidth algorithm to pick, beyond trial and error. Another note for bandwidth, it is important to consider

if the dataset has outliers, because it can also affect the smoothness of the KDE, especially with higher bandwidth values.

Applications

The main application of KDE is for data analysis. KDE can allow you to understand a better distribution of your dataset than a histogram could provide. A couple of pros for KDE over histograms is their ability to handle missing data and having a more precise representation of the data since binning is avoided. However, histograms do handle outliers better.

Works Cited

Aditya. “Reinforcement Learning in Chess.” *Medium*, Medium, 23 Nov. 2023, medium.com/@samgill1256/reinforcement-learning-in-chess-73d97fad96b3.

Akinshin, Andrey. “The Importance of Kernel Density Estimation Bandwidth.” *The Importance of Kernel Density Estimation Bandwidth | Andrey Akinshin*, aakinshin.net/posts/kde-bw/. Accessed 4 Mar. 2025.

Comment, et al. “Reinforcement Learning.” *GeeksforGeeks*, 24 Feb. 2025, www.geeksforgeeks.org/what-is-reinforcement-learning/.

Drapala, Jaroslaw. “Kernel Density Estimation Explained Step by Step.” Towards Data Science, 28 Jan. 2025, towardsdatascience.com/kernel-density-estimation-explained-step-by-step-7cc5b5bc4517/.

Fessel, Kimberly. “What Is Kernel Density Estimation? And How to Build a KDE Plot in Python? | Seaborn KDEplot.” *YouTube*, YouTube, 29 June 2020, www.youtube.com/watch?v=DCgPRaIDYXA

“Q-Learning in Reinforcement Learning.” *GeeksforGeeks*, GeeksforGeeks, 25 Feb. 2025, www.geeksforgeeks.org/q-learning-in-python/.

Sahin Ahmed, Data Scientist. “A Beginner’s Guide to Reinforcement Learning.” *Medium*, Medium, 5 Feb. 2025, medium.com/@sahin.samia/a-beginners-guide-to-reinforcement-learning-d7cd477348fc.

“Seaborn.Kdeplot#.” *Seaborn.Kdeplot - Seaborn 0.13.2 Documentation*, seaborn.pydata.org/generated/seaborn.kdeplot.html. Accessed 4 Mar. 2025.