Machine Learning

Lesson 7

Reinforcement Learning

Lecturer: Dr. Msagha J Mbogholi, PhD

Flashback from Lesson 6

- All ensemble methods have in common is that they construct several base models from adapted versions of the training data, on top of which some technique is employed to combine the predictions or scores from the base models into a single prediction of the ensemble.
- The key idea in boosting is to train diverse models by increasing the weight of previously misclassified examples. This helps to reduce the bias of otherwise stable learners such as linear classifiers or decision stumps.
- Bagging trains diverse models from samples of the training data, These techniques are particularly useful to reduce the variance of low-bias models such as tree models.

Content

- Introduction
- Definitions and components
- Learning process
- Episodic vs continual tasks
- Action selection methods
- Learning methods
- RL techniques
- Application areas

Part 1

Introduction

7.1 Introduction

- Life is an endless journey of learning, no doubt you will agree.
- Your learning started from the minute you were born and will continue till you die (whether you like it or not!)
- To learn is defined as "gain or acquire knowledge of or skill in (something) by study, experience, or being taught." (google.com)
- To understand what reinforcement learning (RL) is let us first start by examining the human experience of learning.
- Let us do this by considering two examples.

7.1 Introduction (cont'd)

- Have you ever observed a child exploring the new world it finds itself in? Children will touch, taste, smell everything they come across.
- Consider the scenario of a child who observes a fire for the first time; chances are that they will try to touch the fire. What happens then?
- The child will experience pain and not go near the fire again; the same thing happens when it handles a sharp knife for the first time.
- Consider a farmer when s/he ventures into farming for the first time. They are drawn by the possible financial rewards they will get by investing in cash crops. However, it might not always go as planned; the rains might delay, the crop might be attacked by unexpected pests or diseases, and so on.
- In both scenarios above both child and farmer are presented with two possible outcomes; a reward (when the farmer cashes in on the crop) or a punishment/penalty (losing the crop or the child having a scar due to being burnt)
- These scenarios form the basis of what reinforcement learning is all about.

7.1 Introduction (cont'd)

- Reinforcement learning is a way by which a learner wishes to achieve a certain target (goal) and the steps that they take in order to achieve it.
- In the case of the farmer s/he wishes to have a healthy crop which they can sell at a profit in the marketplace (goal). In order to do so there are a series of steps that are followed: preparing the field, placing manure, sowing the seeds, using different chemicals, checking for diseases, and so on. These are the steps taken to achieve the goal.
- In the case of the child perhaps s/he approached the fire and felt warm; the goal was to enjoy the warmth but by going to near to the fire they got burnt.

7.1 Introduction (cont'd)

- Thus if the farmer loses the crop due to using wrong pesticide they learn that was a wrong step (miscalculation) on their part; s/he is not likely to make the same mistake with the next crop.
- Similarly the child also learns that fire is good if enjoyed from a distance; with time they learn the safest distance to keep (sooner rather than later).
- Reinforcement learning carries these same principles to teach systems (machines/programs) to achieve desired goals via a series of steps.

Part 2

Definitions & Components

2.1 Definitions

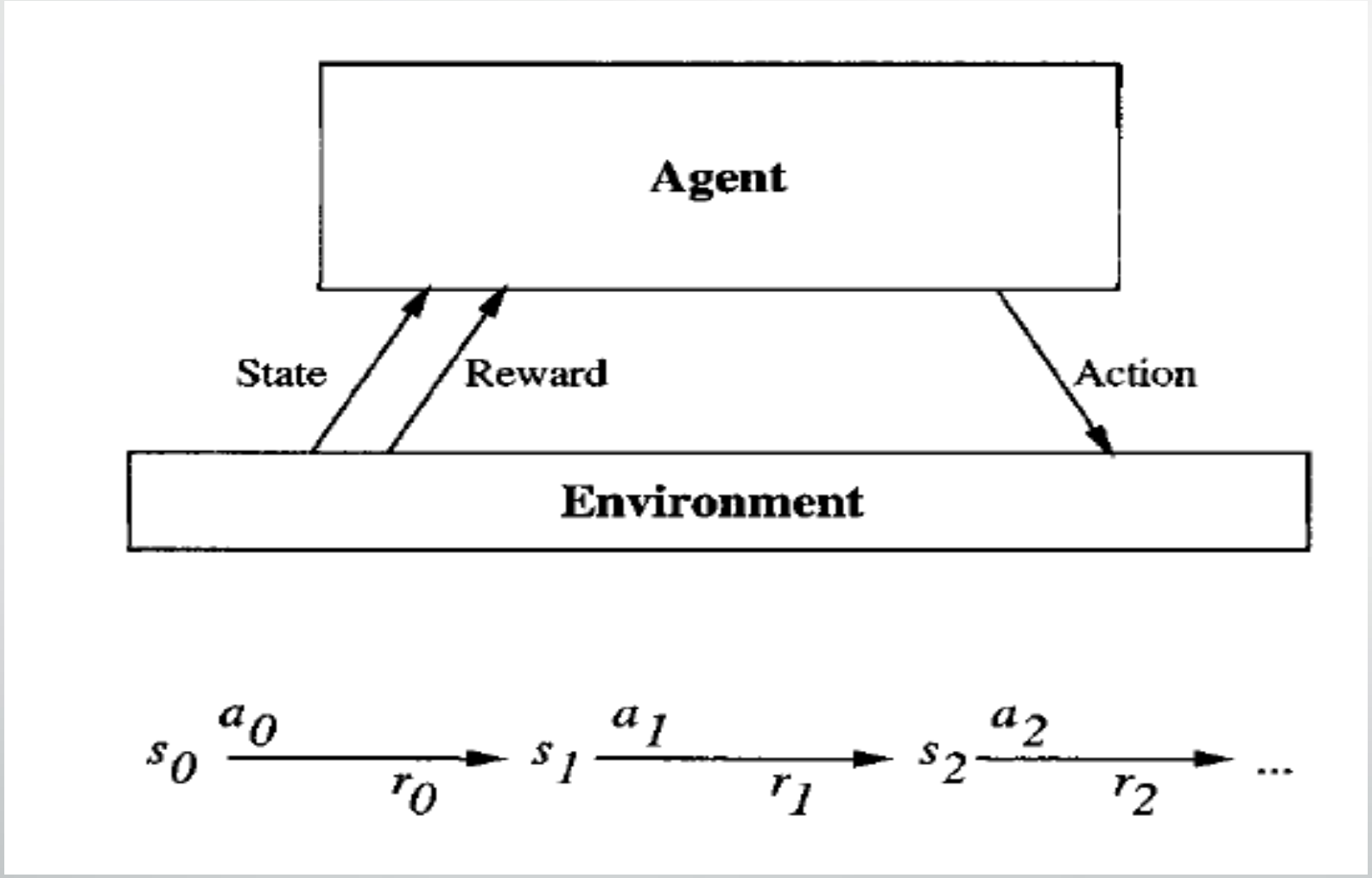

Fig 1. Agent interaction with environment (Mitchell, 1997)

2.1 Definitions (cont'd)

- Fig 1 captures all the main components of the reinforcement learning environment:
- Agent – this is the one who makes decisions based on the rewards and punishments; for example, the farmer or child in our introduction.
- Action – a step performed by the agent to change the current state of the environment; actions are discreet; thus a_0, a_1, and a_2 represent different actions performed by the agent
- State – the current condition of the environment; for example, the current state of the crop is damaged due to frost.
- Environment – everything outside the agent is referred to as the environment; the agent takes action and the environment will reward or punish the action; thus the agent takes actions in the environment.

2.1 Definitions (cont'd)

- t – refers to time measured discreetly; thus t = 0, 1, 2, 3…..which can be represented in a timeline as t_0, t_1, t_2, and so on.
- s – refers to the different states of the environment which the agent can change; thus s_0, s_1, and s_2 represent different progressive states of the environment.
- r – refers to the different rewards offered to an agent as it moves from one state to another; thus r_0, r_1, and r_2 are all rewards offered to the agent as it moves from one state to another. Reward are based on a score given by the reward function; the reward is a number.
- Policy – the function that is used by the agent to determine the next best action to take from the current state; the abbreviation used is π
- Fig 1 can now be described based on the definitions provided as follows:

2.2 Components

- The agent is in an environment constituting different states; let us call the set of all states S. The agent has a defined goal state but it does not know how to get there.
- This is what differentiates reinforcement learning from other forms of learning such as supervised learning. In reinforcement learning the agent must take a series of actions that will alter the state of the environment. In supervised learning the goal is known to the learner (we learnt this in lesson 1, remember?)
- The agent thus does not know the effect of its action(s) on the goal state; it must take the action and based on the reward offered it can tell whether that, or an alternative action will take it nearer to its goal state.
- In essence the reward function tells the agent how well it is doing in its pursuit of the goal state.
- In RL it is sometimes even a requirement that the reward is delayed until later; making it that more difficult for the learner.
- Whether the agent tracks its previous actions or not depends on the RL method used.
- These will be discussed later in the lesson.

2.3 Exploration & Exploitation

- Before examining the different methods and techniques used in RL it is important that we understand the difference between the two terms exploration and exploitation in this context.
- As described earlier the aim of the agent is to collect maximum reward in its pursuit of achieving the goal state.
- In so doing it can use either exploration or exploitation. What is the difference between the two? Let us demonstrate this using fig 2.

2.3 Exploration and Exploitation (cont'd)

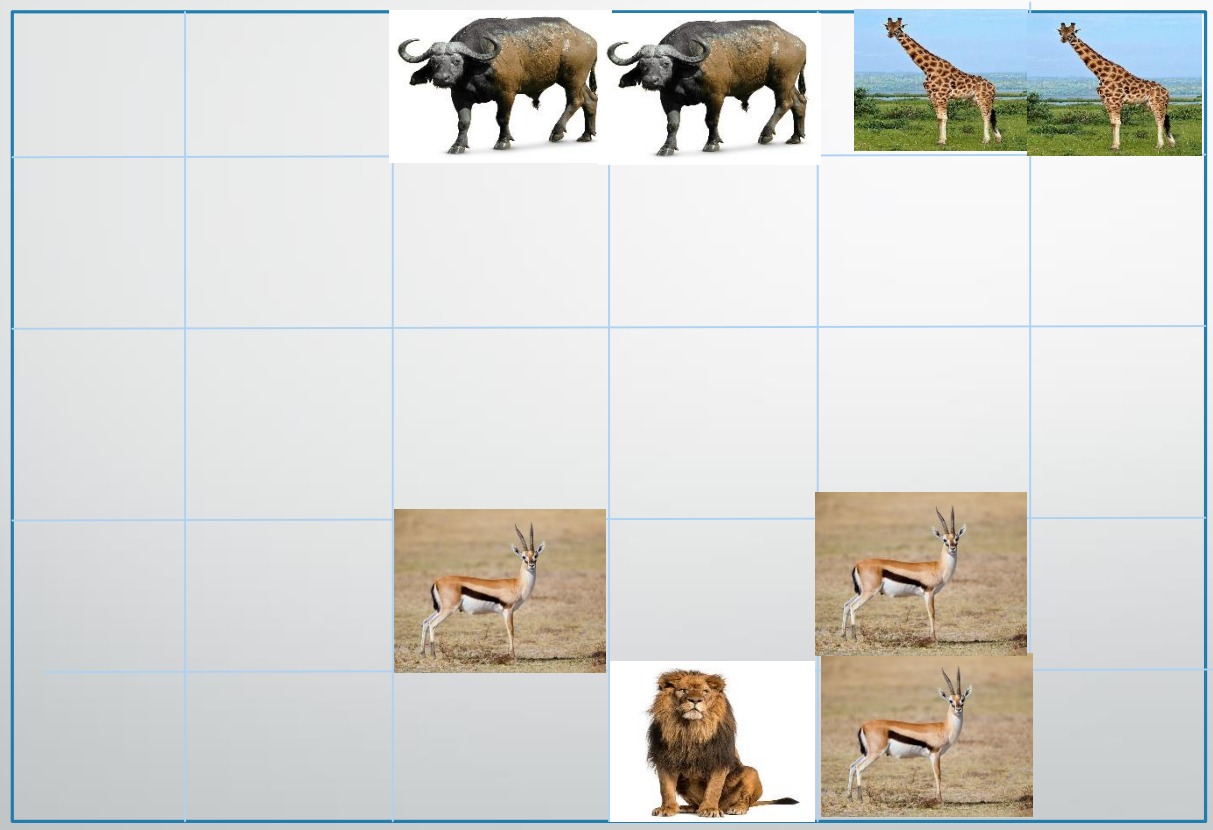

Fig 2. An exploration/exploitation scenario.

2.3 Exploitation and Exploration (cont'd)

- We all know the lion; it is known as the king of the jungle (jungle? Well not exactly, but story for another day).
- Lions normally hunt in a group known as a pride; it can consist of anything between 3 and 30 lions. However, as the males grow older they are chased away from the pride by younger males who take up the leadership mantle (and also get to have all the females to themselves!)
- So sometimes you will find 2 males roaming together so that they can hunt for food together (higher chances of success).
- In fig 2 we are presented with such a scenario. Our lions have found a cosy area where there are plenty of gazelles nearby that they can feast on. This is a situation of exploitation since they have known the gazelles are nearby and thus they exploit the opportunity. Exploitation is about getting reward from the known and reaping it repeatedly.

2.3 Exploitation and Exploration (cont'd)

- I am sure you can see the reward in this case is small; how many gazelles will they have to kill in a day in order to be fully satisfied? Consider that a gazelle weighs about 15kgs, these lions will need to kill at least 2 every single day. That's a lot of work isn't it?
- However, if the lions choose to explore they will find a greater reward in the form of giraffe and buffalo; now that is more satisfying for them. Consider that one buffalo can feed the lions for a whole week, and what about the giraffes? Did you know that giraffe is actually the lion's favorite meal?
- So this presents a big challenge in the trade off between exploration and exploitation, in terms of reward. Stay in one place and exploit rewards near to you, or explore and reap bigger rewards yonder.
- In RL there is the need to define functions (rules) that will help to exploit this trade-off between exploration and exploitation.

Part 3

Reinforcement Learning Process

3.1 The learning process

- By now we have gathered that the learner is a searching algorithm that looks for the goal state. We have also learnt that the learner purposes to achieve maximum reward in its search.
- But how does it go about searching (and gathering maximum reward) in its pursuit of goal state?
- Fig 1 helps us to understand better; at the beginning the agent receives a state s_0 from the environment; based on this the agent takes an action a_0. the environment then transitions to a new state s_1. The environment then gives a reward r_0 to the agent.
- The environment gives state s_1 to the agent; the agent will then take action a_1; the environment then transitions to a new state s_2. The environment then gives a reward r_1 to the agent.
- The agent purposes to achieve maximum reward and in it's search will avoid actions that will not do so. Thus as can be seen the loop is a series of state (s), action (a), and reward (r).

3.2 The reward

- The goal of the agent is to maximize rewards. In so doing it will achieve the ultimate goal state. Let us call this goal state G which is achieved over time t. Thus the reward cumulatively at each time step t can be computed. We thus have
- $G_t = r_t + r_{t+1} + r_{t+2} + \ldots$, for example at t = 0, we get $G_0 = r_0$, corresponding to s_0, a_0, and r_0 in fig 1.
- This can be further summed up to:
- $G_t = \sum_{k=0}^{T} r_{t+k}$

3.3 Discounts

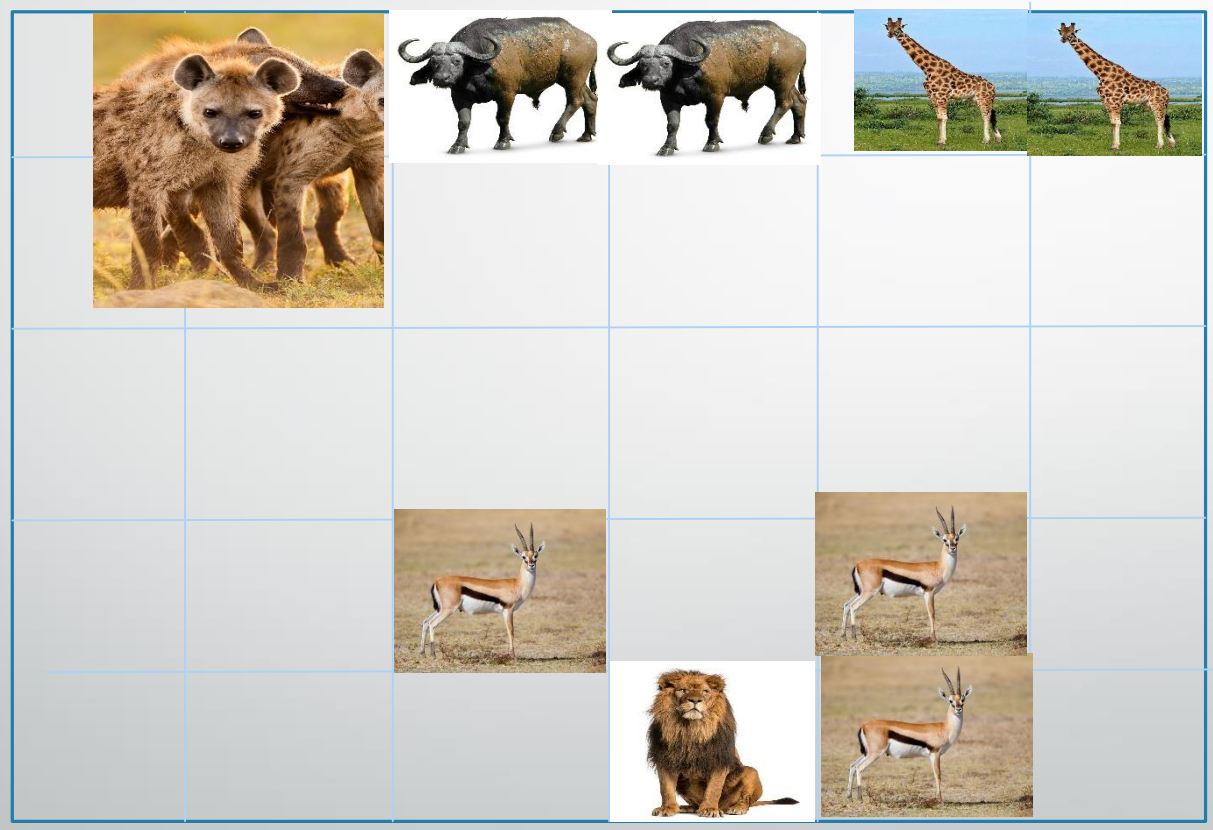

Fig 3 The concept of discounts

3.3 Discounts

- Fig 3 is the same as fig 2….well, almost.
- A pack of hyenas have made their way into the neighborhood of the giraffes and buffalo. This is very disturbing news for our two lions.
- Hyenas and lions are mortal enemies. It is good when the lions are many (safety in numbers) than when they are only 2. Should our two lions venture to where there favorite giraffe meat is they will most certainly be killed by the pack of hyenas (hyenas are second in size to lions among all the savanna predators)
- There is now a dilemma presented in this scenario. The lions have the option to eat the small rewards around them, or dare to go and hunt the buffalo and giraffes (and most likely get killed by the hyenas).
- This introduces the concept of a discount.

3.3 Discounts

- Essentially the lion can continue eating the small rewards around it; while venturing to eat the big game (buffalo and giraffe) can result in death. Thus the reward associated with big game has to be discounted (since there is no assurance they will survive should they even manage to kill the big game!)
- Let us call the discount rate gamma (γ) whose value will lie between 0 and 1;
- When the discount rate is high the discount will be small. This implies that the learner is focused on long term rewards. (in our example this would mean our lions will be focused on the big game which is a long term reward and of course riskier; therefore, not as accurate as munching on the gazelle).
- When the discount rate is low the discount is big. This will now imply that the learner is focused on short term rewards (what is known as sure bet in this part of the world). In our example this would mean that our lions will be focused on the gazelle and not on the big game as this is more assured and therefore more accurate (from an RL perspective).

3.3 Discounts

- Let us examine further the implications of γ on our discount rate. Since gamma lies between 0 and 1, rewards are multiplied by γ^t where t are the number of time steps in the future that the reward is from.
- Mathematically since γ is less than 1 (essentially a fraction) it means that γ^2 is smaller (think of say ½ squared gives ¼ for example) and therefore
- $\gamma^k \rightarrow 0$ as $k \rightarrow \infty$ which means that future predictions can be ignored (since their value will be very small).
- We can now show what the prediction of the total future reward will be:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^{k-1} r_k + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

Fig 4. Prediction of total future reward (Marsland, 2015)

- Thus in terms of rewards in our example it means that as time increases the chances of the hyenas catching the scent of the lions increases and the chances of reward for our lions (to eat) becomes less and less.

Part 4

Episodic vs Continual Tasks

4.1 Introduction

- An instance of an RL problem is called a task.
- As shall be seen at the end of this lesson there are so many application areas of RL such that it is impossible to list all of them; only the most interesting ones are discussed.
- However, RL tasks are classified as either being continual or episodic.

4.2 Episodic Tasks

- Episodic tasks have a starting and an ending (also known as a terminal state).
- An episode consists of a list of states (s), actions(a), and rewards (r), as depicted in fig 1. this is followed by a new state.
- If you watch TV series like Blacklist, Peaky Blinders, Lucifer, Squid games, and so on, you will have observed that they consist of a number of episodes.
- Every episode has a beginning and an ending (terminal state).
- In some of these series there is a grand finale episode which ends with the termination of the whole series (what did you think of squid games?)
- Another example could be when you play a game like Fortnite; an episode ends when you are either killed or when you finish a level. There are 1000 levels and only a handful of players have ever reached it in any chapter.

4.3 Continuous Tasks

- Continuous tasks have no terminal state.
- The agent continues by choosing the best actions (those with maximum rewards) and continuously interacts with the environment. Rewards for the agent are given periodically (with discounting) since there is no end.
- The agent will continue until an external force (person/program) stops it.
- An example of a continuous task is when you are driving your car; the onboard computer will keep monitoring everything until you stop; if you leave the car running for hours the learner will keep monitoring until you switch off the car.
- Another example is in software that monitors goings-on at the stock exchange, or foreign exchange; this will go on continuously till you stop it.

Part 5

Action Selection Methods

5. Action Selection Methods

- At this point it is already clear that RL is all about searching for the right actions that will give maximum reward.
- However, a key question arises: how does the learner choose which action to take?
- In each state the learner examines the value of each action it can undertake; that is the reward expected for carrying out the action.
- Chances are that the learner has been in this state before; it therefore makes sense that the simplest thing to do is to examine the average reward received for each action it took while in this state.
- The average reward can be computed and is denoted as $Q_{s,t}(a)$, where s is the state, a is the action, and t is the number of times (time steps) that the action has been taken in the given state before.

5. Action Selection Methods

- Using this technique eventually the true prediction for the reward will be found.
- Based on the average reward there are three methods that are available to the learner to choose the next action (a) to take :
- Greedy – the highest value of $Q_{s,t}(a)$ rules when this method is used. This essentially means using the most current knowledge (the most recent action taken before, that is, exploitation).
- ε greedy – similar to the greedy method; the only difference is incorporation of a small probability ε allowing the learner to pick some other action randomly. The idea is to explore the possibility of another action that might be better. Thus, the greedy method can be used more often, but occasionally this method is used to explore other possible actions.
- Soft-max – the ε greedy uses uniform probability to select alternatives. The soft-max method is a refinement of the ε greedy which uses a function to choose the next action to take.

Part 6

Learning Methods

6.1 Markov Decision Process (MDP)

- You are currently studying this course "machine learning". If you examine the course requirements in the syllabus you will find that there are some desirable prerequisites (not compulsory).
- However, there are other courses where there is a compulsory prerequisite in order for you to understand that particular course. This means that for you to do that course you must have come from another course.
- Does this tell you where to go next? Absolutely! Depending on what your future plans are, coming from another course to this one can help you determine what you want to do next (by the end of this course).
- Another example is a game called "Takeshi's castle" which I used to watch on TV (not sure if it's still going on). You have to pass a series of obstacles in order to get to Takeshi's castle and claim the prize.

6.1 Markov Decision Process (MDP)

- In one of the obstacles you had to find your way through several doors (like a maze) where if you opened the wrong door you would either fall into a pool of dirty water, or you would be found by the "demons" who would throw you out. The trick was to recognize the doors you had opened before and narrowly missed falling into (or being thrown into) the pool of dirty water.
- In both the above scenarios you needed to know where you're from in order to make a decision on where to go next.
- However, consider the game of chess and the state of the board at any given time. You do not need to know what the previous move was in order for you to make the next move. The next move is based on the current position of all the pieces on the board.
- See the difference between the two?
- This is what the Markov decision process is all about.

6.1 Markov Decision Process (MDP)

- In reinforcement learning a state that has the property such that it can compute reward without looking at previous states is known as a Markov state.
- Let us compare 2 equations, one showing the situation when previous states have to be relied on to compute reward, and one where the current state alone is enough (Markov state):

6.1 Markov Decision Process (MDP)

$$Pr(r_t = r', s_{t+1} = s' | s_t, a_t, r_{t-1}, s_{t-1}, a_{t-1}, \ldots r_1, s_1, a_1, r_0, s_0, a_0),$$

$$Pr(r_t = r', s_{t+1} = s' | s_t, a_t).$$

Fig 5 Probability calculation (Marsland, 2015)

- In the first equation, the probability in calculating the next state is determined by all the previous states up to the initial state so, ao. Clearly this is not ideal as there will also be need for storage of these previous states; it is not one that can be used for diverse, interesting problems.
- The second equation relies on the current state in determining the next action and by extension the next state.

6.1 Markov Decision Process (MDP)

- An RL problem that follows the second equation is known as a Markov Decision Process (MDP).
- The current state and action is enough to predict the reward, based on previous experience.

6.2 Monte Carlo

- This method is used in eposidic task scenarios.
- The rewards are received at the end of an episode.
- The learner gathers all the rewards and calculates the grand total of the rewards.
- The agent then begins a new episode.
- As is to be expected the learner will improve and make better decisions with each new episode.
- Let us examine how this method works in form of an equation.

6.2 Monte Carlo

- The following equation demonstrates how the method works (freecodecamp.org)

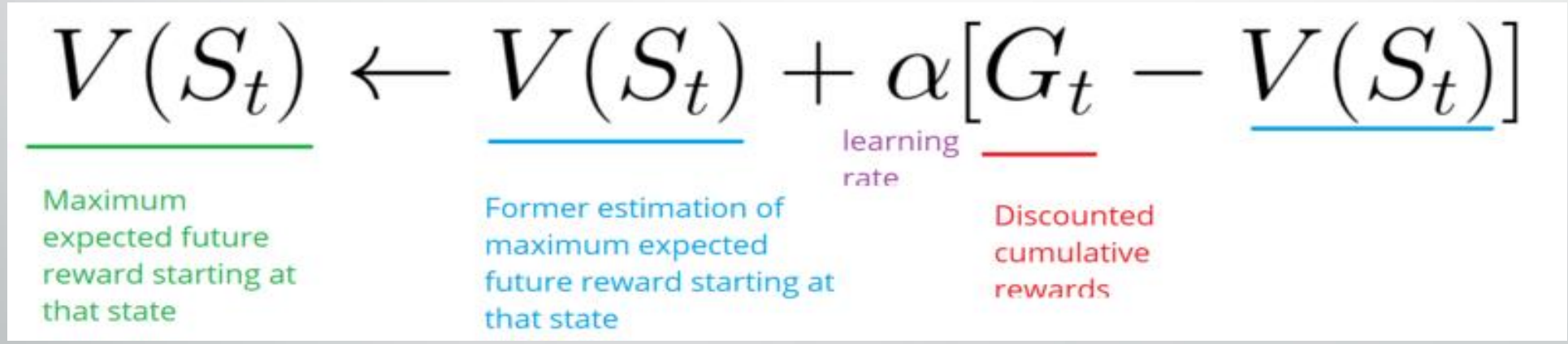

Fig 6 Monte Carlo method (Simonini, 2020)

6.2 Monte Carlo

- Let us consider two examples to further explain this method.
- Consider the game of chess. The game will normally end when the king is checkmated or when a draw is declared (neither player can win the game).
- Suppose you are playing against the learner; at the end of the game the learner will have collected a list of states (s), actions (a), rewards (r) and new states.
- The learner will now get the grand total sum of all the rewards gained throughout the game (G_t in the formula) to gauge its performance (that is how well, or badly it did). It will then update V (S_t) based on this.
- The learner will then start a new game. Consequently as more and more games are played the better the learner will become with each game.

6.2 Monte Carlo

- Revisit fig 3. Suppose this was also a game and not a real scenario.
- That would be a game whereby the lion might venture to eat the big game (giraffes and buffalo).
- To do so the lion would have to somehow snatch the big game without the hyenas killing it.
- So the game would end when the lion either successfully snatches a big game or if the hyenas kill it. (would be a very interesting game to play, yes?)
- So with each episode (game) the learner will get better at evading the hyenas and getting to the ultimate prize (the big game). Conversely if the learner is the hyenas then with each game they will get better at killing the lion.

6.3 Temporal Difference (TD)

- This method is also called the one step TD in that the future reward estimation is not done at the end of the episode (like in Monte Carlo).
- The value estimation (V) is updated for all non terminal states (S_t) as they occur, that is, at every step.
- This method is also called TD(0) or one step TD. A comparison of the two methods is presented (freecodecamp.org)

Monte Carlo $$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

TD Learning $$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

Previous estimate

Reward t+1

Discounted value on the next step

TD Target

Fig 7. Monte Carlo vs TD learning (Simonini, 2020)

6.3 Temporal Difference (TD)

- Thus at every time step the value estimate is updated by the method.
- From the formula it can be seen that for example at time step (t + 1) a TD target is formed using observed reward at that time step $R_{(t+1)}$ and the current estimate V ($S_{(t+1)}$)
- The TD (0) algorithm for Q values is known as the Q learning algorithm. Its steps are presented in fig 8. In this algorithm the value of V(s) is swappable with Q (s, a).

6.3 Temporal Difference

The Q-Learning Algorithm

- **Initialisation**
  - set $Q(s, a)$ to small random values for all s and a
- Repeat:
  - initialise s
  - repeat:
    - select action a using ϵ-greedy or another policy
    - take action a and receive reward r
    - sample new state s'
    - update $Q(s, a) \leftarrow Q(s, a) + \mu(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$
    - set $s \leftarrow s'$
  - For each step of the current episode
- Until there are no more episodes

Fig 8 Q learning algorithm (Marsland, 2015)

6.3 Temporal Difference

- Similar to Q learning algorithm is the Sarsa algorithm.
- With this algorithm in a current state s_t, and action a_t is taken, and the agent get the associated reward r_t; the agent will wind up in the next time step (state) $s_{(t+1)}$ and the action in that state will be $a_{(t+1)}$. Hence the abbreviation sarsa (that is, $s_t\, a_t\, r_t\, s_{(t+1)}\, a_{(t+1)}$).
- The algorithm for sarsa is presented in fig 9. The differences between the two algorithms is explained thereafter.

6.3 Temporal Difference

The Sarsa Algorithm

- **Initialisation**
  - set $Q(s, a)$ to small random values for all s and a
- Repeat:
  - initialise s
  - choose action a using the current policy
  - repeat:
    - take action a and receive reward r
    - sample new state s'
    - choose action a' using the current policy
    - update $Q(s, a) \leftarrow Q(s, a) + \mu(r + \gamma Q(s', a') - Q(s, a))$
    - $s \leftarrow s'$, $a \leftarrow a'$
  - for each step of the current episode
- Until there are no more episodes

Fig 9. Sarsa algorithm (Marsland, 2015)

6.3 Temporal Difference

- Q learning is called an off policy algorithm since when it is passing the reward for the next state ($s_{(t+1)}$, $a_{(t+1)}$) it picks the maximum reward for the next state $s_{(t+1)}$ and ignores whatever policy is being used.
- Sarsa on the other hand is a more conservative algorithm. It is an on-policy algorithm that will strictly follow what the policy says in the computation of the next state, that is, it will calculate the next action $a_{(t+1)}$ and pass the reward corresponding to that action.
- This means that Q learning is a greedy algorithm; the implication is that if sarsa uses the greedy algorithm as the policy then both sarsa and Q learning will be the same.

Part 7

Reinforcement Learning (RL) Techniques

Introduction

- There are 3 techniques (approaches) to reinforcement learning:
  - Value based
  - Policy based
  - Model based

7.1 Value based

- The theory behind this approach is to find the maximum value of the value function V(s).
- Recall that V(s) tells us the maximum expected future reward value that the learner will get at each state.
- To find the value at a given state it is the sum of the discounted rewards that the agent can expect to get (starting from the given state).
- Fig 10 presents the formula that is used:

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

Expected | Reward discounted | Given that state

Fig 10. Value function (Simonini, 2020)

7.1 Value based

- The learner uses the function to determine which step to take for the next state. It will choose the one with the highest value.
- In so doing it will achieve its goal. Fig 6 demonstrates how this will work. The path that the agent will choose is shown in red, as it picks the highest value at each step.

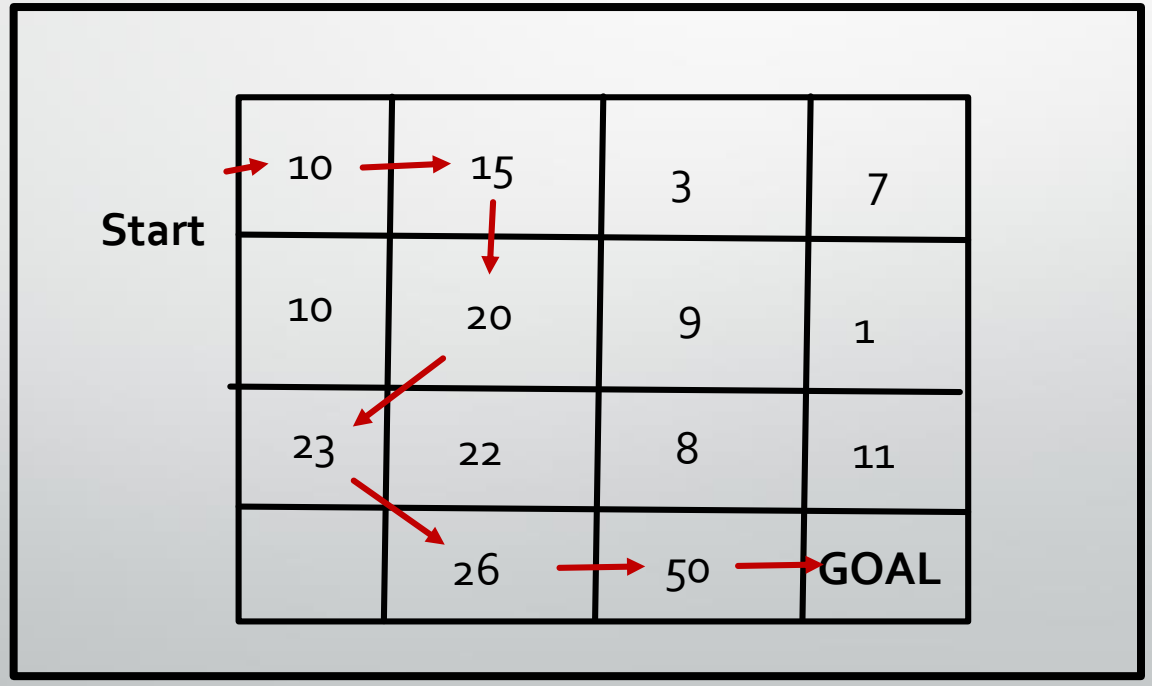

Fig 11.Value based path

7.2 Policy based

- A policy was defined earlier in the lesson as the function that is used by the agent to determine the next best action to take from the current state; the abbreviation used is π
- The aim when using a policy based approach is to optimize the policy function without using a value function V(s).
- For a given state (s) the policy alone should define which action (a) to take; thus
- $a = \pi(s)$; consequently, each policy should be used to determine best action for a given state.

7.2 Policy based

- There are 2 types of policy:
- Deterministic: as the name implies this a policy whereby a given state will always return the same action.
- Stochastic: this is where a probability distribution is used for output over actions. This can be presented as:
- $\Pi (a|s) = Pr [a_t = a \mid s_t = s]$
- Considering fig 11, a policy can be used instead of values that will direct the agent to follow steps in order for the goal to be reached.

7.3 Model based

- With this approach a model is developed capturing the behavior of the environment.
- This is particularly cumbersome and quite tedious. This is because a new model will have to be developed for every new environment.
- Thus it is not used much in RL.

Part 8

Application areas

Application areas

- RL is applied in many diverse areas. Some of them are:
- In development of video games
- Robotics
- Marketing – agents can make predictions on customer preferences, and so on
- Broadcast journalism – predicting readers behaviors
- Self driving cars
- Stock price predictions
- Natural Language Processing (NLP) – in text summarizations, question answering, machine translation, and so on.
- Healthcare systems – the agent can give treatment using policies it has learnt from previous experiences.

Summary

- Reinforcement learning is a way by which a learner wishes to achieve a certain target (goal) and the steps that they take in order to achieve it.
- In reinforcement learning the agent must take a series of actions that will alter the state of the environment. In supervised learning the goal is known to the learner; RL is essentially a search to achieve the goal.
- When the discount rate is high the discount will be small. Conversely, when the discount rate is low the discount is big.
- RL tasks are classified as either being continual or episodic.
- Action selection methods include greedy, ε greedy and soft-max.
- RL learning methods are markov decision process (MDP), monte carlo and temporal difference (TD).
- Approaches to RL are value based, policy based and model based.

References

- Alpaydin, E. (2010). *Introduction to machine learning*. MIT Press.
- Bajaj, P. (2022, May 18). *Reinforcement learning*. GeeksforGeeks. Retrieved May 25, 2022, from https://www.geeksforgeeks.org/what-is-reinforcement-learning/
- Duong, V. H. T. (2021, February 23). *Intro to Reinforcement Learning: Temporal Difference Learning, Sarsa vs. Q-learning*. Medium. Retrieved May 25, 2022, from https://towardsdatascience.com/intro-to-reinforcement-learning-temporal-difference-learning-sarsa-vs-q-learning-8b4184bb4978
- Gautam, A. (2019, March 15). *Introduction to reinforcement learning (coding sarsa) - part 4*. Medium. Retrieved May 25, 2022, from https://medium.com/swlh/introduction-to-reinforcement-learning-coding-sarsa-part-4-2d64d6e37617#:~:text=SARSA%20is%20an%20on%2Dpolicy,stands%20for%20the%20acronym%20SARSA.
- Gupta, A. (2021, November 29). *7 applications of reinforcement learning in Real World*. GeeksforGeeks. Retrieved May 25, 2022, from https://www.geeksforgeeks.org/7-applications-of-reinforcement-learning-in-real-world/

References

- Marsland, S. (2015). *Machine learning: An algorithmic perspective*. CRC Press.
- Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.
- Mwiti, D., & on, F. me. (2021, November 8). *10 real-life applications of reinforcement learning*. neptune.ai. Retrieved May 25, 2022, from https://neptune.ai/blog/reinforcement-learning-applications
- Simonini, T. (2020, April 7). *An introduction to reinforcement learning*. freeCodeCamp.org. Retrieved May 25, 2022, from https://www.freecodecamp.org/news/an-introduction-to-reinforcement-learning-4339519de419