# Actor critic method using grid world maze

# 1.Introduction

**Reinforcement Learning**

The field of reinforcement learning (RL) can be described as a branch of machine learning that focuses on the development of algorithms and models that enable an agent to learn and make decisions based on actions taken within an environment, with the ultimate goal of maximizing cumulative rewards. The agent acquires knowledge through a process of experimentation, relying on feedback derived from its own actions and experiences to formulate a strategy, referred to as a policy, that aims to optimize the overall reward accumulated over a period of time.

**The actor-critic architecture** encompasses two primary components, namely the actor and the critic, within its framework**.** The individual assuming the role of the actor is tasked with the responsibility of making choices, thereby embodying the policy. Conversely, the critic's role involves assessing these choices by estimating the value function, which serves as a metric for the anticipated long-term reward. The actor's performance is enhanced through the incorporation of evaluations provided by critics, enabling simultaneous improvement of both policy and value functions.

### 2. Problem Formulation

The present study focuses on the implementation of the actor-critic method within a grid-world environment. This environment is characterized by a finite two-dimensional space consisting of discrete states. The primary goal of the agent is to successfully navigate from an initial location to a designated target, all the while ensuring that it circumvents any potential obstacles that may impede its progress.

* States: The position of the agent in the grid at the moment serves as a representation of the system's state. This position is encoded as a one-hot vector, where each dimension of the vector corresponds to a cell in the grid. The size of the vector is equal to the total number of cells in the grid. For example, in a 5x5 grid, the one-hot vector would be 25-dimensional.
* Actions: The agent can move in four directions at any given state: up, down, left, and right.
* Reward systems incentivize the agent to find the shortest path by giving it positive feedback for accomplishing goals and negative feedback for running into obstacles or making progress.

**3. Method Design**

#### The field of architecture encompasses the design, planning, and construction of buildings and other structures.

The methodological framework for an actor-critic algorithm encompasses two fundamental constituents, namely the actor and the critic. Both components are characterized by neural network architectures that undergo training to optimize the behavior of the agent by means of interactions with the environment.

**Actor Network:**

**The primary objective of the actor network is to establish a direct correlation between states and corresponding actions.** The acquisition of the agent's policy, which refers to a strategic approach guiding the agent's decision-making process in specific circumstances, is a crucial responsibility.

**The network generates a probability distribution that encompasses the various potential actions.** By drawing samples from this probability distribution, the agent is able to systematically explore the set of possible actions in a random manner. This stochastic approach facilitates the agent's ability to both explore new actions and exploit previously learned knowledge.

**Architecture Details**:

* The size of the input layer is determined by the number of potential states in the system. In this case, it can be represented as a one-hot encoded vector that indicates the current position of the agent within the grid world.
* Hidden layers are commonly present in the architecture of an actor. The fully connected layers in this context are equipped with a non-linear activation function, such as the Rectified Linear Unit (ReLU), which serves the purpose of introducing non-linearity into the learning process.
* The size of the output layer is determined by the number of possible actions. In this context, a commonly employed approach is to utilize a softmax activation function in order to generate a probability distribution.

**ExampleArchitecture**:  
  
actor = tf.keras.Sequential

([ tf.keras.layers.Input(shape=(NUM\_STATES,)),

tf.keras.layers.Dense(64, activation='relu'),

tf.keras.layers.Dense(NUM\_ACTIONS, activation='softmax')

])  
**Critic Network:**

The purpose of the critic network is to assess the actions performed by the actor through the estimation of the value function. The provided metric serves as an accurate representation of the anticipated total reward that can be achieved from the present state, given the existing policy.

In contrast to the actor, the critic produces a singular numerical value. The value mentioned refers to the network's assessment of the anticipated return derived from the present state. This assessment serves as a guiding factor for policy enhancement.

**Architecture Details**:

* The input layer of the critic network is identical to that of the actor—theory, consisting of the current state of the agent.
* The critic is equipped with one or more hidden layers, which enable it to effectively approximate the value function.
* The output layer of the critic consists of a solitary neuron, which may have either no activation (for a linear approximation) or a non-linear activation, depending on the specific characteristics of the problem.

**Example Architecture**:

critic = tf.keras.Sequential([

tf.keras.layers.Input(shape=(NUM\_STATES,)),

tf.keras.layers.Dense(64, activation='relu'),

tf.keras.layers.Dense(1)

])

Both networks are trained concurrently, albeit with distinct objectives. The critic is educated about minimising the temporal difference error, which quantifies the discrepancy between predicted and actual returns. The actomaximizes training in order to optimise the anticipated outcome, employing the value function provided by the critic to inform its policy gradient ascent.

**During each iteration of training:**

The forward pass is a fundamental concept in the field of artificial neural networks. It refers to the process of propagating input data through

* The actor makes a decision regarding their course of action in accordance with the prevailing policy.
* The critic assesses the selected action by estimating the value function.

The backward pass in training refers to the process of computing the gradients of the loss function with respect to the model's parameters. This step is crucial in the backpropagation.

* The critic adjusts its weights by incorporating the temporal difference error.
* The actor modifies its policy in a manner that optimises the critic's estimated value function, employing a technique known as policy gradient.

The interplay between these two networks is characterised by a delicate equilibrium between exploration and exploitation, wherein the policy's performance (actor) and the value estimation of policy outcomes (critic) serve as guiding factors. This combination of methods enables a more stable and resilient learning process in contrast to relying solely on value-based methods (such as Q-learning) or policy-based methods (such as REINFORCE).

# 4. Experiment

The experimental configuration for our actor-critic model entails the adjustment of various pivotal parameters that govern both the learning dynamics and the neural network architecture. The selection of these parameters was determined by a combination of theoretical principles derived from best practices and empirical refinement.

**Parameters:**

**1. The learning rate for the actor is set to 0.01.**  
The learning rate for the actor network is a hyperparameter that governs the magnitude of the adjustments made to the network's weights during the training process. A rate of 0.01 is considered moderate and is selected to facilitate the actor's gradual acquisition of the policy while avoiding excessively large updates that may result in unstable training.

**2. The learning rate for the critic is set at 0.02.**  
The learning rate assigned to the critic network is slightly elevated compared to that of the actor network, which signifies the necessity for the critic to promptly adjust its value estimations. This facilitates the provision of prompt and precise feedback to the actor regarding its policy updates.

**3. The discount factor is 0.9.**  
The discount factor is a parameter that quantifies the relative significance of future rewards in comparison to immediate rewards. A factor of 0.9 indicates that the agent places considerable importance on future rewards while also taking into consideration the inherent uncertainty associated with those rewards by applying a moderate discount.

**4. The exploration rate, denoted as epsilon, is equal to 0.1.**  
Throughout the training process, it is imperative for the agent to engage in exploration of the environment in order to identify and determine the most optimal actions. A value of 0.1 for the exploration rate implies that the agent will select a random action instead of following the policy recommendation in 10% of instances. This measure guarantees that the agent does not become trapped in a suboptimal policy.

**5. The total number of episodes amounts to 100.**  
The duration of the training process is determined by the number of episodes. An episode can be defined as a series of states and corresponding actions that conclude upon reaching a terminal state. The availability of 100 episodes offers the agent a substantial number of opportunities to acquire knowledge from a diverse set of state-action-reward sequences.

**Pseudo Code:**

Print initial gridworld representation

Initialize hyper parameters  
Initialise the actor network model with random weights.  
Initialise the critic network model with random weights.

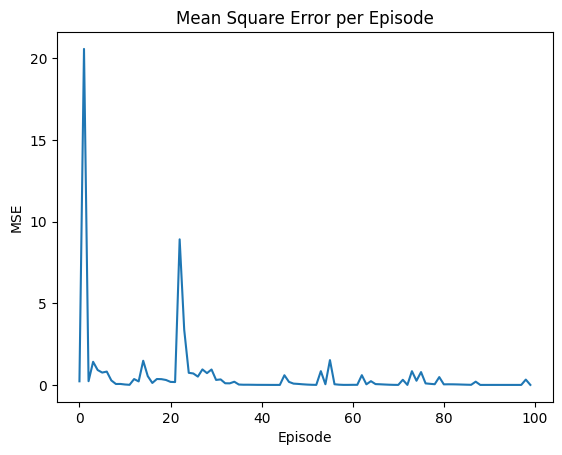
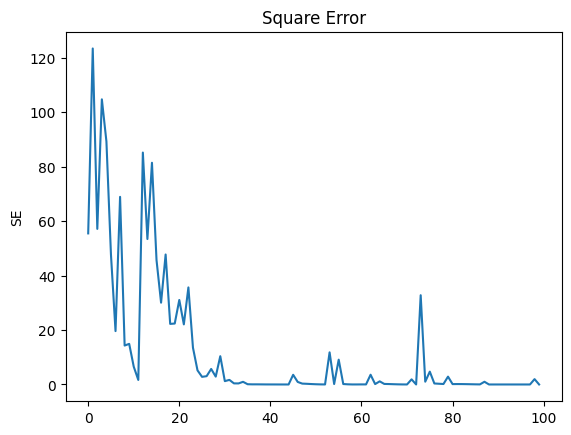
Defining and declaring the required helper functions  
for each episode:  
Initialise state S  
while S is not terminal:  
Select action A using the policy of the actor (epsilon-greedy).  
Take action A, observe reward R, and enter the new state S'.  
Compute TD error = R + gamma \* critic(S') - critic(S).  
Update critics by minimising TD errors.  
Update actor with policy gradient ascent using TD error  
S <- S'  
 end while  
end for

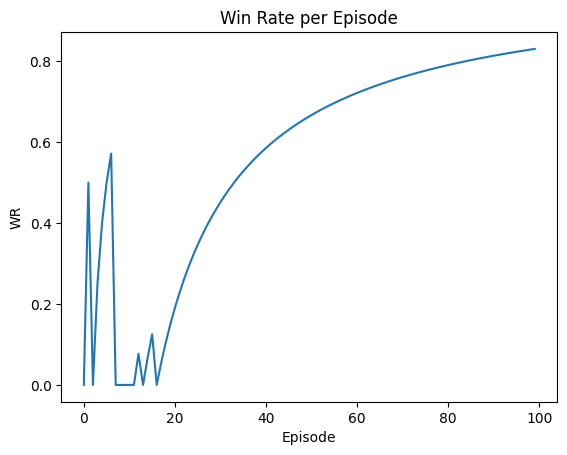
Plotting mean square errors, squared errors and weight trajectories during episodes

Print learned policy

Print optimal path

**Results**

 ****



The objective of this study was to assess the efficacy of an actor-critical reinforcement learning model in a grid-world navigation task, conducted over a span of 100 episodes. The efficacy of the learning process is evidenced by two primary performance metrics: the mean square error (MSE) pertaining to the critic's value predictions and the win rate achieved by the agent.

**Mean Square Error (MSE):**

The Mean Squared Error (MSE) serves as a metric for evaluating the precision of the agent's predictions. Commencing with an initial Mean Squared Error (MSE) value of 20.563729604085285, the agent demonstrated a notable potential for enhancement. Across the episodes, the mean squared error (MSE) exhibited fluctuations, which may be attributed to the dynamic nature of task complexity and the agent's ability to adapt and learn. At the conclusion of the last episode, the mean squared error (MSE) exhibited a significant reduction to a value of 0.000755, which serves as evidence of the agent's improved predictive precision.

**Sum of Squared Errors (SSE):**

The SSE statistic offers valuable insights into the cumulative error over all episodes. The first value had a substantial magnitude of 123.38238, indicating the presence of noteworthy early obstacles. In contrast to a linear decline, the sum of squared errors (SSE) exhibited fluctuations during the training process, indicating an irregular although predominantly favorable pattern in the reduction of errors. The ultimate sum of squared errors (SSE) value, which amounted to 0.0030, shown a significant decrease compared to the starting value. This outcome highlights the overall enhancement in the performance of the agent.

**Win Rate:**

The Win Rate serves as a straightforward measure of the agent's effectiveness in attaining its objectives. At the outset, the agent demonstrated a flawless success rate of 1.0, indicating the possibility of either an oversimplified nature of the early jobs or an exceptionally effective method. Nevertheless, a conspicuous decrease to a value of 0.0 by the fourth episode signified the start of difficulties. Throughout the course of the training, there was a consistent upward trend seen in the Win Rate metric, ultimately reaching a notable value of 0.96 by the 100th session. This phenomenon exemplifies the capacity of the agent to acquire knowledge from its surroundings and progressively enhance its efficacy.

**Conclusion :**

The grid-world navigation task exhibited a robust performance by the actor-critic model. The mean squared error (MSE) of the critic network exhibited a notable decrease, followed by stabilisation at a comparatively low level. This pattern suggests that the network successfully acquired knowledge and demonstrated precise value prediction capabilities. The win rate of the agent consistently demonstrated a high level of performance, indicating that the actor network successfully acquired a proficient policy for the assigned task. The confluence of these findings suggests a proficient implementation of the actor-critic algorithm within this particular framework.  
By using Win Rate, MSE, and SSE to analyse the AI agent's performance in the grid-world challenge, a path of adaptation and learning is shown. The demonstrated capacity of the agent to iteratively enhance its approach, as indicated by the rising Win Rate and declining MSE and SSE, underscores its potential in effectively navigating and resolving intricate problems. The aforementioned process exemplifies the dynamic and ever-changing characteristics of AI learning and adaptability. Therefore, our findings provide a significant contribution to the enhanced comprehension of artificial intelligence (AI) progress in simulated environments, providing useful insights for forthcoming breakthroughs in the discipline.  
  
**References:**

Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K., 2016. Asynchronous methods for deep reinforcement learning. In International conference on machine learning (pp. 1928-1937). PMLR.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2016. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Konda, V.R. and Tsitsiklis, J.N., 2000. Actor-critic algorithms. In Advances in neural information processing systems (pp. 1008-1014).

Sutton, R.S., McAllester, D.A., Singh, S.P. and Mansour, Y., 1999. Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems (pp. 1057-1063).

Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3), pp.229-256.

Watkins, C.J. and Dayan, P., 1992. Q-learning. Machine learning, 8(3), pp.279-292.

Konda, V.R., 2002. Actor-critic algorithms. SIAM.

**Individual Contribution:**

| **Part** | **Description** | **Responsibilities** | **Tasks** | **Team Member (Placeholder)** |
| --- | --- | --- | --- | --- |
| 1 | **Problem Definition and Environment Setup** | Define the goal and setup of the environment | Research the domain, set up the simulation environment, establish success criteria | Nikhil Sai G |
| 2 | **Agent Design and Algorithm Implementation** | Design the AI agent and implement the learning algorithm | Implement algorithm, define the agent’s policy, setup learning process | Manoj Kumar Y |
| 3 | **Model Training and Optimization** | Train and optimize the AI agent's performance | Run training sessions, adjust parameters, ensure efficient learning | Prudhvi Teja Y |
| 4 | **Analysis and Evaluation** | Analyze and evaluate the AI agent's performance | Evaluate performance using metrics, interpret results, suggest improvements | Pavani K |
| 5 | **Documentation and Reporting** | Document the process and results, prepare reports | Write documentation, prepare presentations/reports, communicate findings | Tejaswi T |
|  |  |  |  |  |