Deep Reinforcement Learning

2025 Second Semester, M.Tech (AIML)

Session #1:
Introduction to the Course

Instructors, Deep Reinforcement Learning Course

Some content for the slides may have been obtained from prescribed books and various other source on the Internet. The authors , hereby acknowledge all the contributors for their material and inputs and gratefully acknowledge those who made their course materials freely available online.

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Agenda

- Course Introduction

  Outline of Course, Evaluation & Operation

- Introducing Reinforcement Learning

What is Reinforcement Learning ?

- reward based learning / feedback based learning
- not a type of NN nor it is an alternative to NN. Rather it is an approach for learning
- Autonomous driving, gaming

Why Reinforcement Learning ?

- a goal-oriented learning based on interaction with environment

Course Objectives

Course Objectives:

1. Understand
   a. the conceptual, mathematical foundations of deep reinforcement learning
   b. various classic & state of the art Deep Reinforcement Learning algorithms
2. Implement and Evaluate the deep reinforcement learning solutions to various problems like planning, control and decision making in various domains
3. Provide conceptual, mathematical and practical exposure on DRL
   a. to understand the recent developments in deep reinforcement learning and
   b. to enable modelling new problems as DRL problems.

Learning Outcomes

1. Understand the fundamental concepts of reinforcement learning (RL), algorithms and apply them for solving problems including control, decision-making, and planning.
2. Implement DRL algorithms, handle challenges in training due to stability and convergence
3. Evaluate the performance of DRL algorithms, including metrics such as sample efficiency, robustness and generalization.
4. Understand the challenges and opportunities of applying DRL to real-world problems & model real life problems

Course Operation

- **Textbooks**
    1. Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto, Second Ed. , MIT Press

Course Operation

- **Evaluation**

  Two Quizzes for 5% each; **Best of two will be taken for 5%** (in final grading);

  Whatever be the points set for quizzes, the score will be scaled to 5%

  NO MAKEUP, for whatever be the reason. Ensure to attend at least one of the quizzes.

  Two Assignments - Tensorflow/ Pytorch / OpenAI Gym Toolkit → 25 %

  Assignment 1: Partially Numerical + Implementation of Classic Algorithms - 10%

  Assignment 2: Deep Learning based RL - 15%

  Mid-Term Exam - 30% [Only to be written in A4 pages, scanned and uploaded]

  Comprehensive Exam - 40% [Only to be written in A4 pages, scanned and uploaded]

- **Webinars/Tutorials**

  4 tutorials : 2 before mid-sem & 2 after mid-sem

Course Operation

- Schedule of Schedule of Quizzes

  See the announcements for details

- Schedule of Assignments

  See the announcements for details

- Schedule of Webinars

  See the announcements for details

Course Operation

- How to reach us ? (for any question on lab aspects, availability of slides on portal, quiz availability , assignment operations)

  See the announcements for details

- **Plagiarism [Important]**

  All submissions for graded components must be the result of your original effort. It is strictly prohibited to copy and paste verbatim from any sources, whether online or from your peers. The use of unauthorized sources or materials, as well as collusion or unauthorized collaboration to gain an unfair advantage, is also strictly prohibited. Please note that we will not distinguish between the person sharing their resources and the one receiving them for plagiarism, and the consequences will apply to both parties equally.

  In cases where suspicious circumstances arise, such as identical verbatim answers or a significant overlap of unreasonable similarities in a set of submissions, will be investigated, and severe punishments will be imposed on all those found guilty of plagiarism.

Reinforcement Learning

Reinforcement learning (RL) is based on rewarding desired behaviors or punishing undesired ones. Instead of one input producing one output, the algorithm produces a variety of outputs and is trained to select the right one based on certain variables – Gartner

When to use RL?

RL can be used in large environments in the following situations:

1. A model of the environment is known, but an analytic solution is not available;
2. Only a simulation model of the environment is given (the subject of simulation-based optimization)
3. The only way to collect information about the environment is to interact with it.

(Deep) Reinforcement Learning

Paradigm	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Objective	$p_\theta(y \mid x)$	$p_\theta(x)$	$\pi_\theta(a \mid s)$
Applications	→ Classification → Regression	→ Inference → Generation	→ Prediction → Control

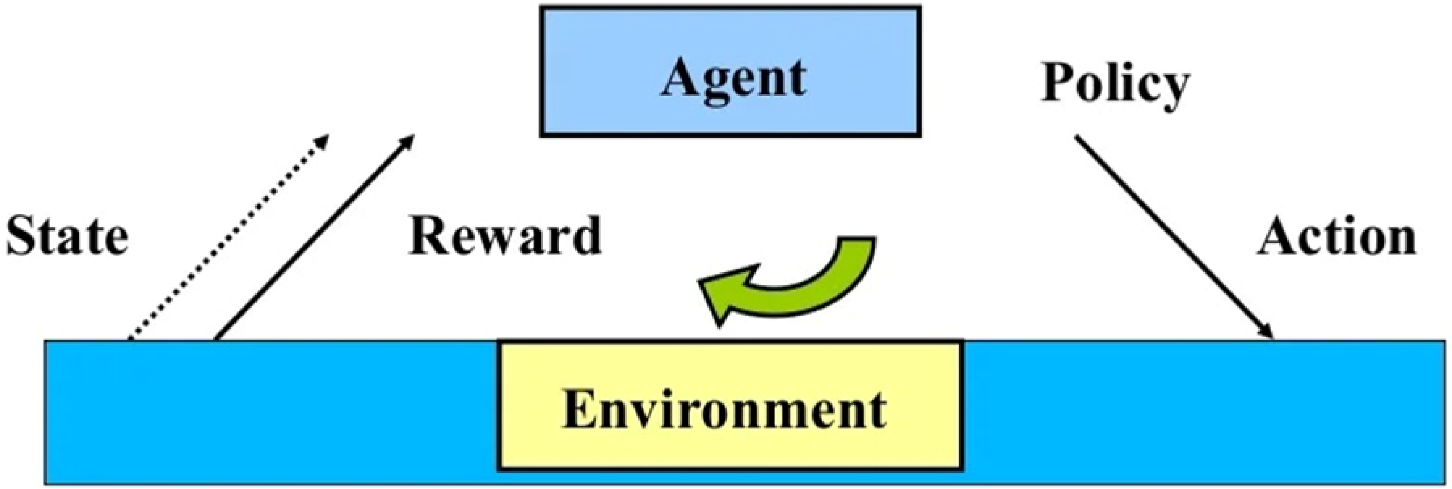
Types of Learning

Criteria	Supervised ML	Unsupervised ML	Reinforcement ML
Definition	Learns by using labelled data	Trained using unlabelled data without any guidance.	Works on interacting with the environment
Type of data	Labelled data	Unlabelled data	No – predefined data
Type of problems	Regression and classification	Association and Clustering	Exploitation or Exploration
Supervision	Extra supervision	No supervision	No supervision
Algorithms	Linear Regression, Logistic Regression, SVM, KNN etc.	K – Means, C – Means, Apriori	Q – Learning, SARSA
Aim	Calculate outcomes	Discover underlying patterns	Learn a series of action
Application	Risk Evaluation, Forecast Sales	Recommendation System, Anomaly Detection	Self Driving Cars, Gaming, Healthcare

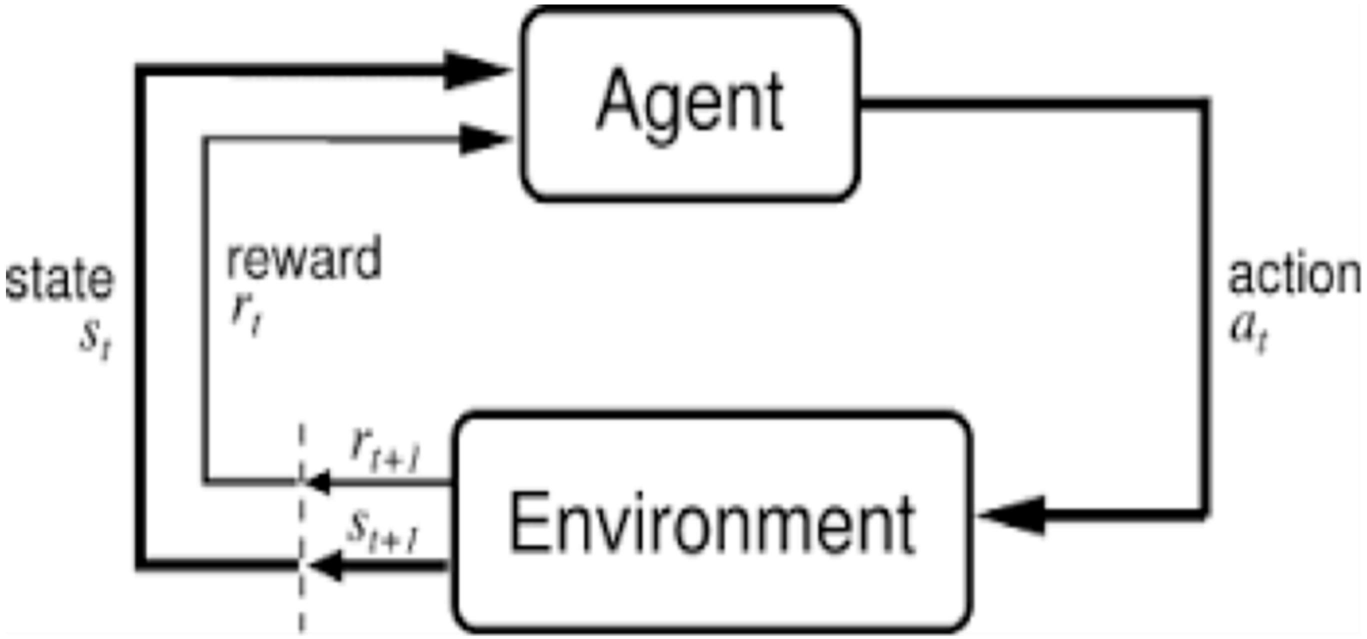
Characteristics of RL

- No supervision, only a real value or reward signal
- Decision making is sequential
- Time plays a major role in reinforcement problems
- Feedback isn’t prompt but delayed

Elements of Reinforcement Learning



Elements of Reinforcement Learning



Beyond the agent and the environment, one can identify four main sub-elements of a reinforcement learning system: *a policy*, a *reward* , a *value* function, and, optionally, a *model* of the environment.

Elements of Reinforcement Learning

- **Agent**
  - An **entity** that tries to learn the best way to perform a specific task.
  - In our example, the child is the agent who learns to ride a bicycle.

- **Action (A)** -
  - **What the agent does** at each time step.
  - In the example of a child learning to walk, the action would be “walking”.
  - A is the set of all possible moves.
  - In video games, the list might include running right or left, jumping high or low, crouching or standing still.

Elements of Reinforcement Learning

- **State (S)**
  - **Current situation** of the agent.
  - After doing performing an action, the agent can move to different states.
  - In the example of a child learning to walk, the child can take the action of taking a step and move to the next state (position).
- **Rewards (R)**
  - Feedback that is given to the agent based on the action of the agent.
  - If the action of the agent is good and can lead to winning or a positive side then a positive reward is given and vice versa.

Elements of Reinforcement Learning

- **Environment**
  - Outside world of an agent or physical world in which the agent operates.

Formal Definition - ***Reinforcement learning*** *(**RL**) is an area of machine learning concerned with how intelligent **agents** ought to take **actions** in an **environment** in order to maximize the notion of cumulative **reward**.*

Elements of Reinforcement Learning

- Policy
- Reward Signal
- Value Function
- Model (Optional)

Tic-Tac-Toc

X	O	O
O	X	X
		X

Tic-Tac-Toc

States	Initial Values
$\begin{bmatrix} X & & \\ & & \\ & & \end{bmatrix}$	**0.5**
$\begin{bmatrix} X & O & O \\ & X & \\ & & \end{bmatrix}$	**0.5**
$\begin{bmatrix} X & O & O \\ & X & \\ & & X \end{bmatrix}$	**1.0**
$\begin{bmatrix} X & & O \\ X & & O \\ & X & O \end{bmatrix}$	**0**
…	…

<u>Learning Task:</u> Play as many times against the opponent and learn the values

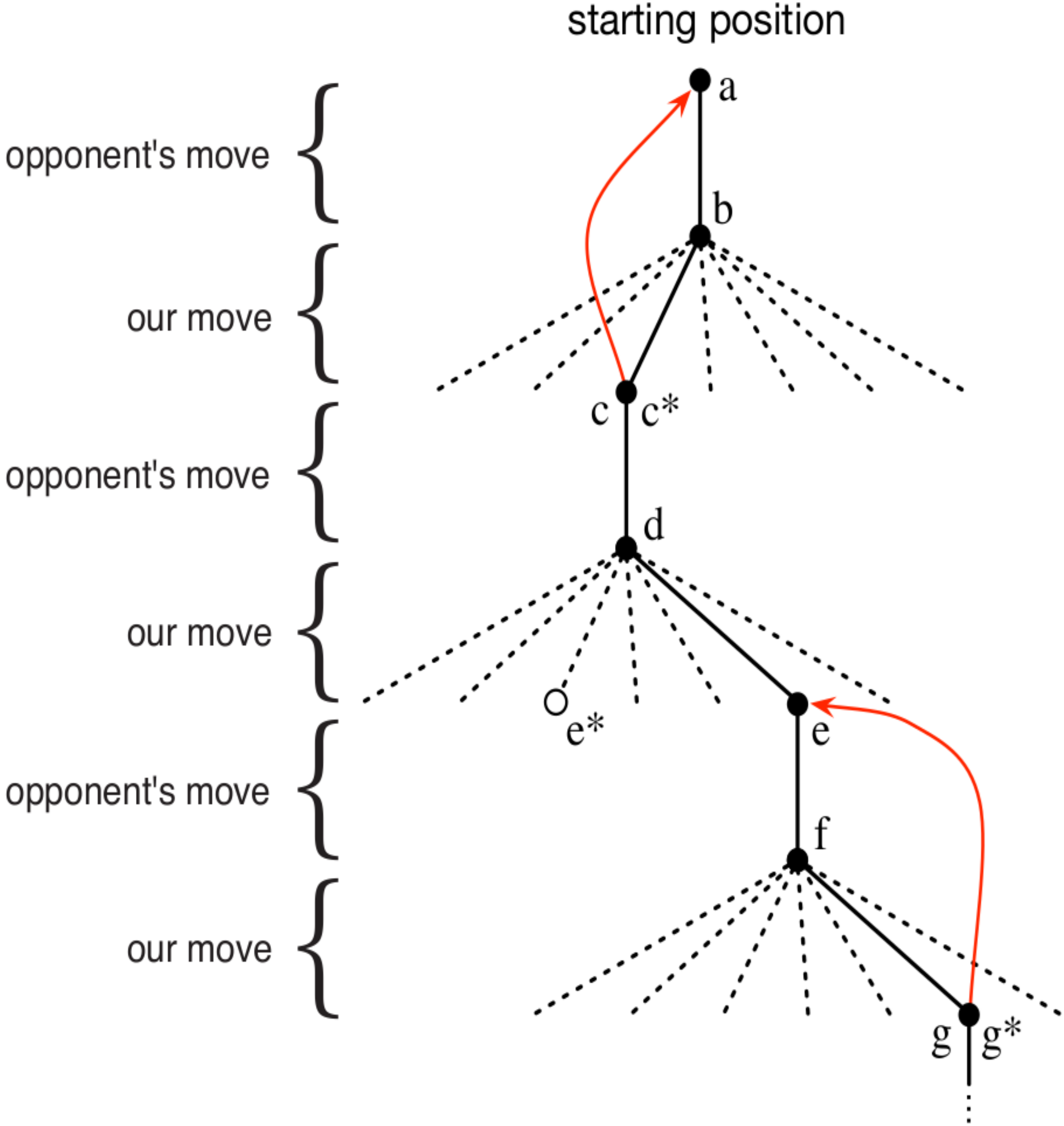
X	O	O
O	X	X
		X

Set up a table of states initial values

Tic-Tac-Toc

States	Initial Values
$\begin{bmatrix} X & & \\ & & \\ & & \end{bmatrix}$	**0.5**
$\begin{bmatrix} X & O & O \\ & X & \\ & & \end{bmatrix}$	**0.5**
$\begin{bmatrix} \boldsymbol{X} & O & O \\ & \boldsymbol{X} & \\ & & \boldsymbol{X} \end{bmatrix}$	**1.0**
$\begin{bmatrix} X & & O \\ \boldsymbol{X} & & O \\ & \boldsymbol{X} & O \end{bmatrix}$	**0**
…	…

S_t - state before greedy move
S_{t+1} - state after greedy move



$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

The Value Function

The Reinforcement Learning agent refines its value estimates using the rule:

$$V(S_t) \leftarrow V(S_t) + \alpha\big[V(S_{t+1}) - V(S_t)\big] \tag{1}$$

This equation is an instance of the general update formula used frequently throughout reinforcement learning:

$$\text{New Estimate} \leftarrow \text{Old Estimate} + \text{StepSize}\big[\text{Target} - \text{Old Estimate}\big] \tag{2}$$

Component Definitions

$V(S_t)$ **Old Estimate:** The current estimated value of the state S_t. This value is the estimated long-term desirability or expected future reward starting from that state.

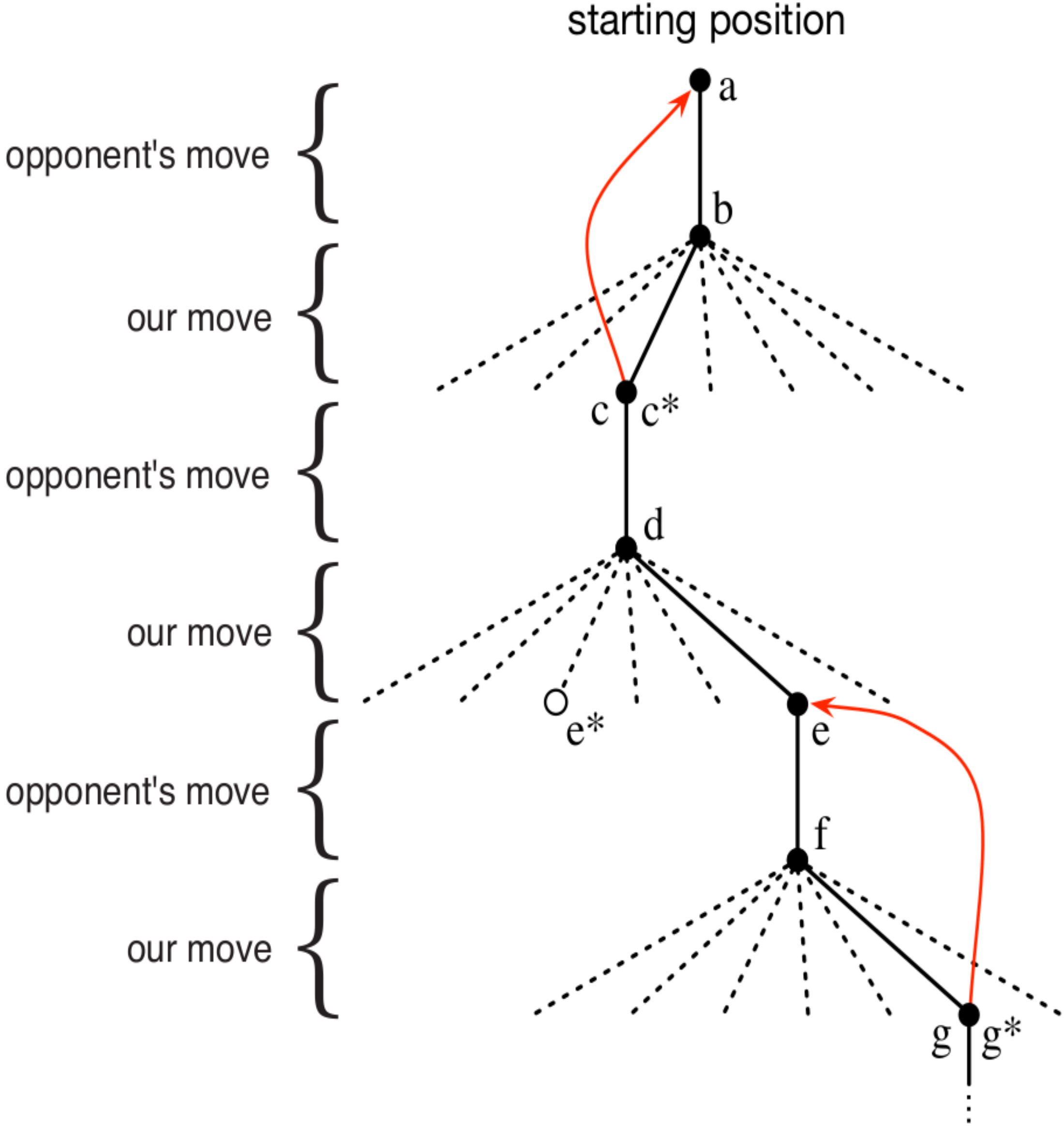
$S_{t+1})$ **Target Component:** The estimated value of the next state, S_{t+1}. In this form (Equation 1, commonly seen in the Tic-Tac-Toe example where intermediate rewards are zero), this value serves as the proxy for the future return.

Learning Parameters

α **Step-size Parameter:** A small positive fraction that controls the rate of learning. A smaller α leads to slower, more stable changes, while a larger α allows the estimate to adjust more rapidly to new information.

$\leftarrow$ **Assignment Operator:** Indicates that the value of $V(S_t)$ is being updated or assigned a new estimate.

Tic-Tac-Toc

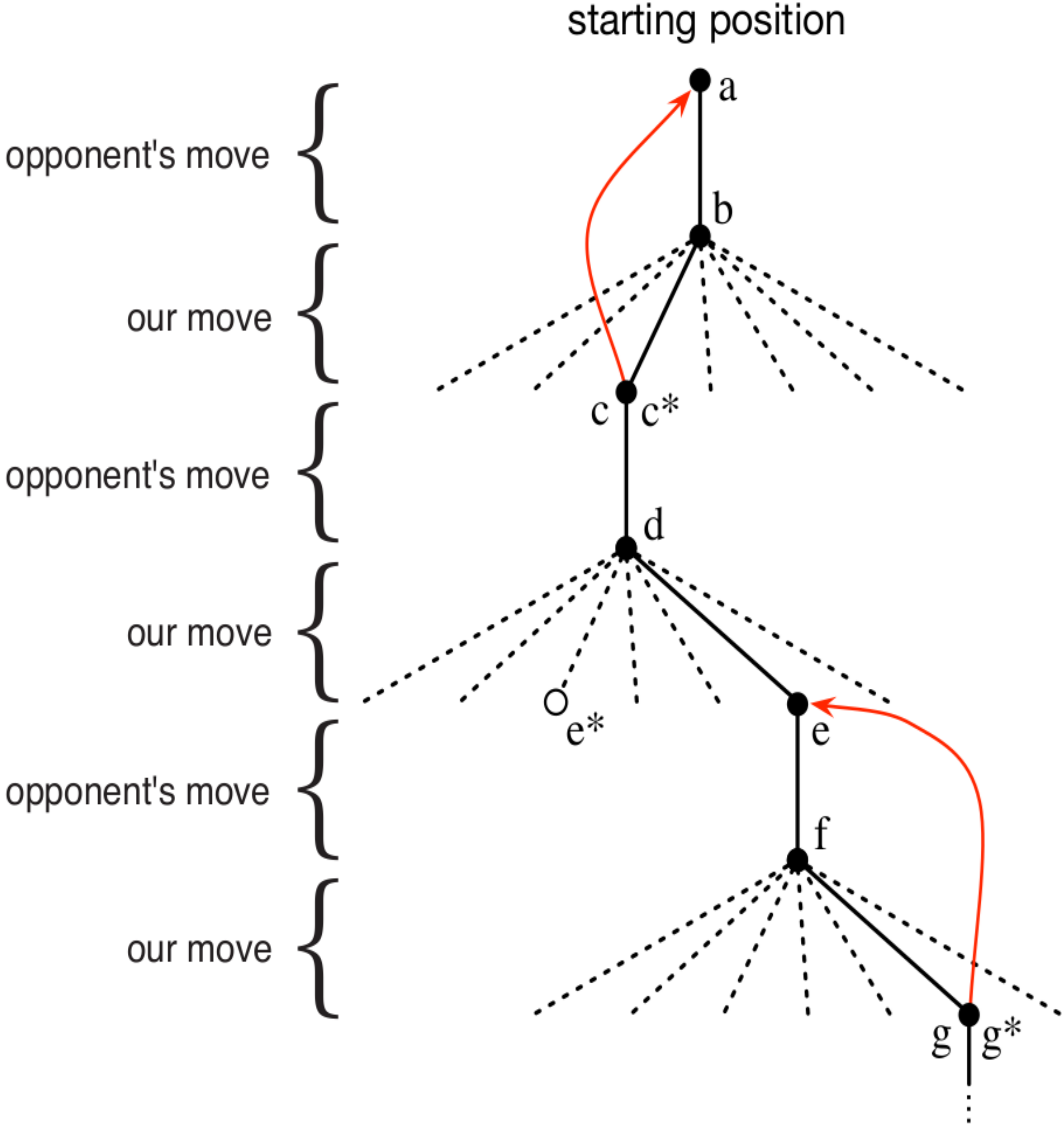


Temporal Difference Learning Rule

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

α- Step Size Parameter

Tic-Tac-Toc



Key Takeaways:

(1) Learning while interacting with the environment (opponent).
(2) We have a clear goal
(3) Our policy is to make moves that maximizes our chances of reaching goal
   - Use the values of states most of the time (exploration) and explore rest of the time.

Temporal Difference Learning Rule

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[V(S_{t+1}) - V(S_t) \Big]$$

α- Step Size Parameter

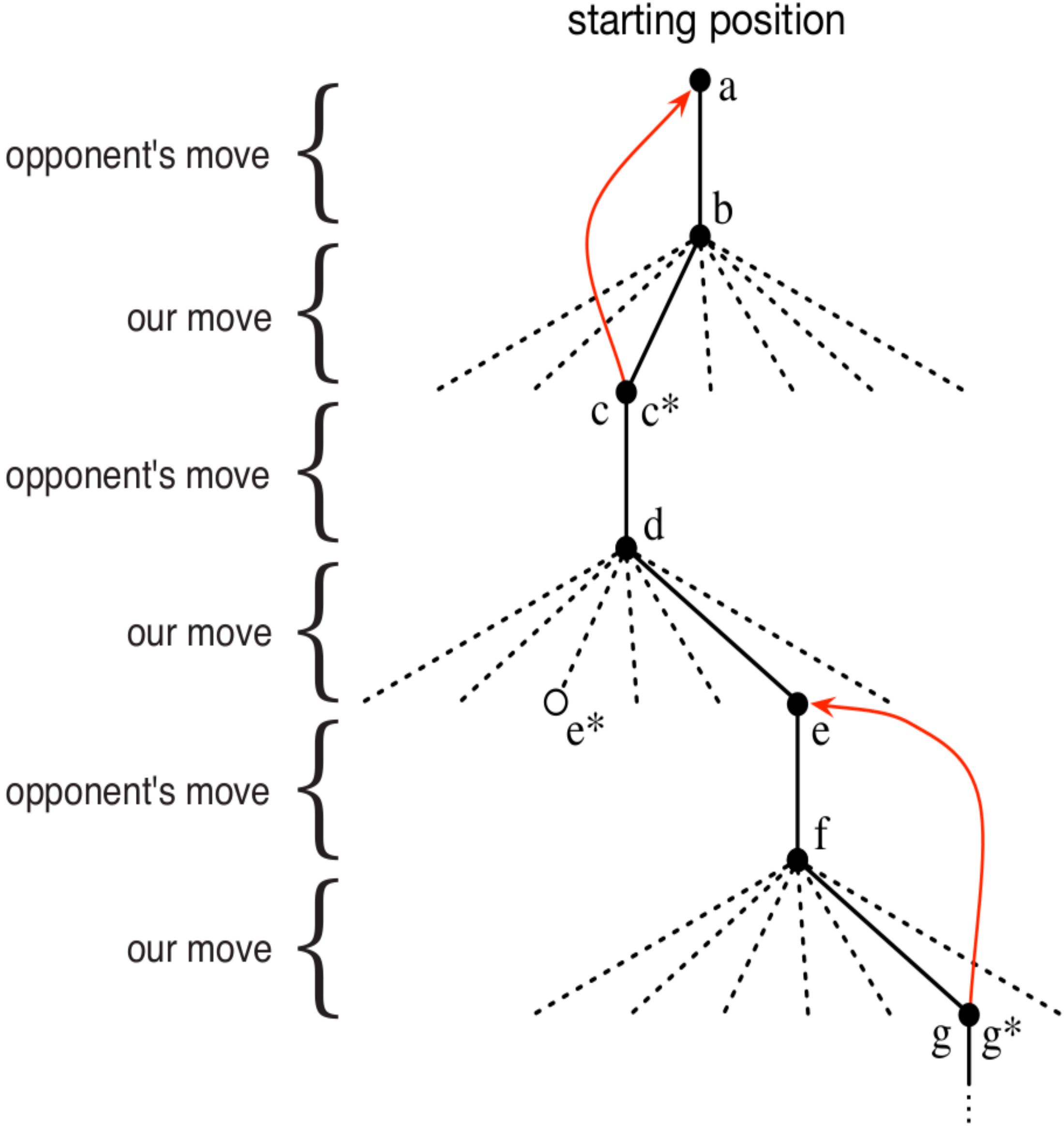
Tic-Tac-Toc

Questions:

(1) What happens if α is gradually made to 0 over many games with the opponent?

(2) What happens if α is gradually reduced over many games, but never made 0?

(3) What happens if α is kept constant throughout its life time?



Temporal Difference Learning Rule

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

***α*- Step Size Parameter**

Tic-Tac-Toc

Questions:

(1) What happens if α is gradually made to 0 over many games with the opponent?

(2) What happens if α is gradually reduced over many games, but never made 0?

(3) What happens if α is kept constant throughout its life time?

Answers:

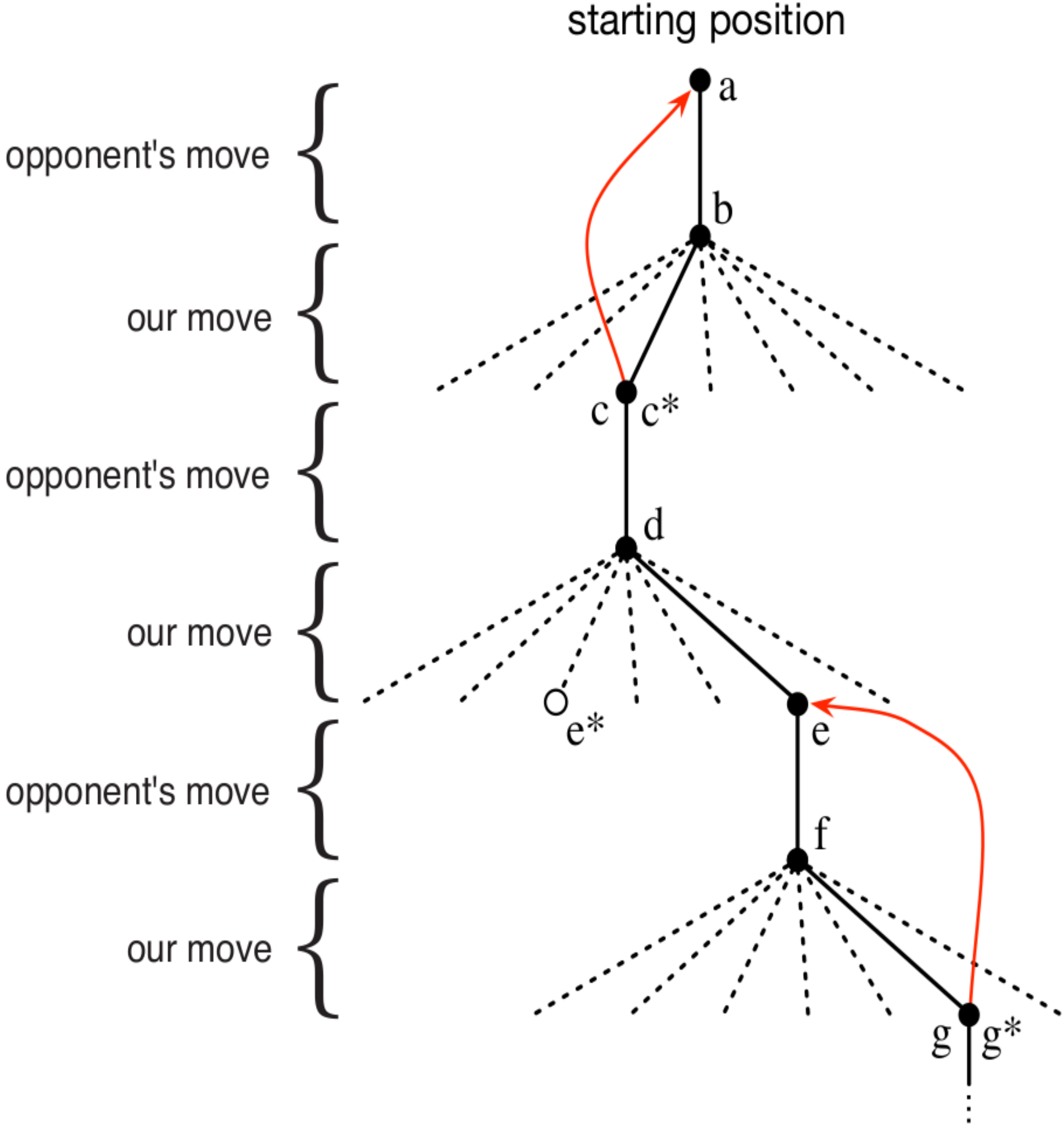
1) Learning **slows down and eventually stops**.

2)

- Early on: large updates → faster learning.
- Later: small updates → smoother convergence, still able to adapt slowly.

3)

- Agent **keeps learning at the same rate**.
- Never truly converges — can **oscillate** around optimal values.



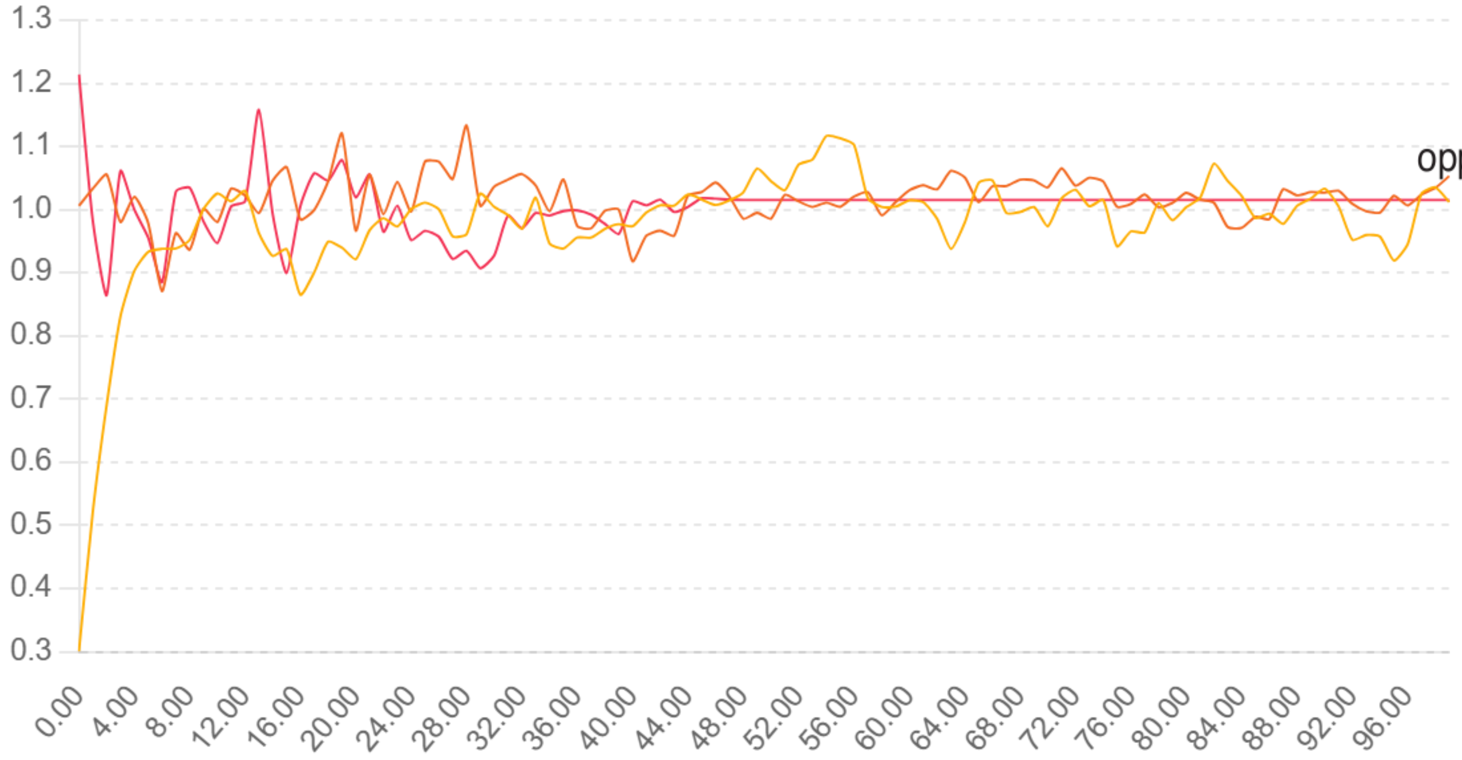
Tic-Tac-Toc

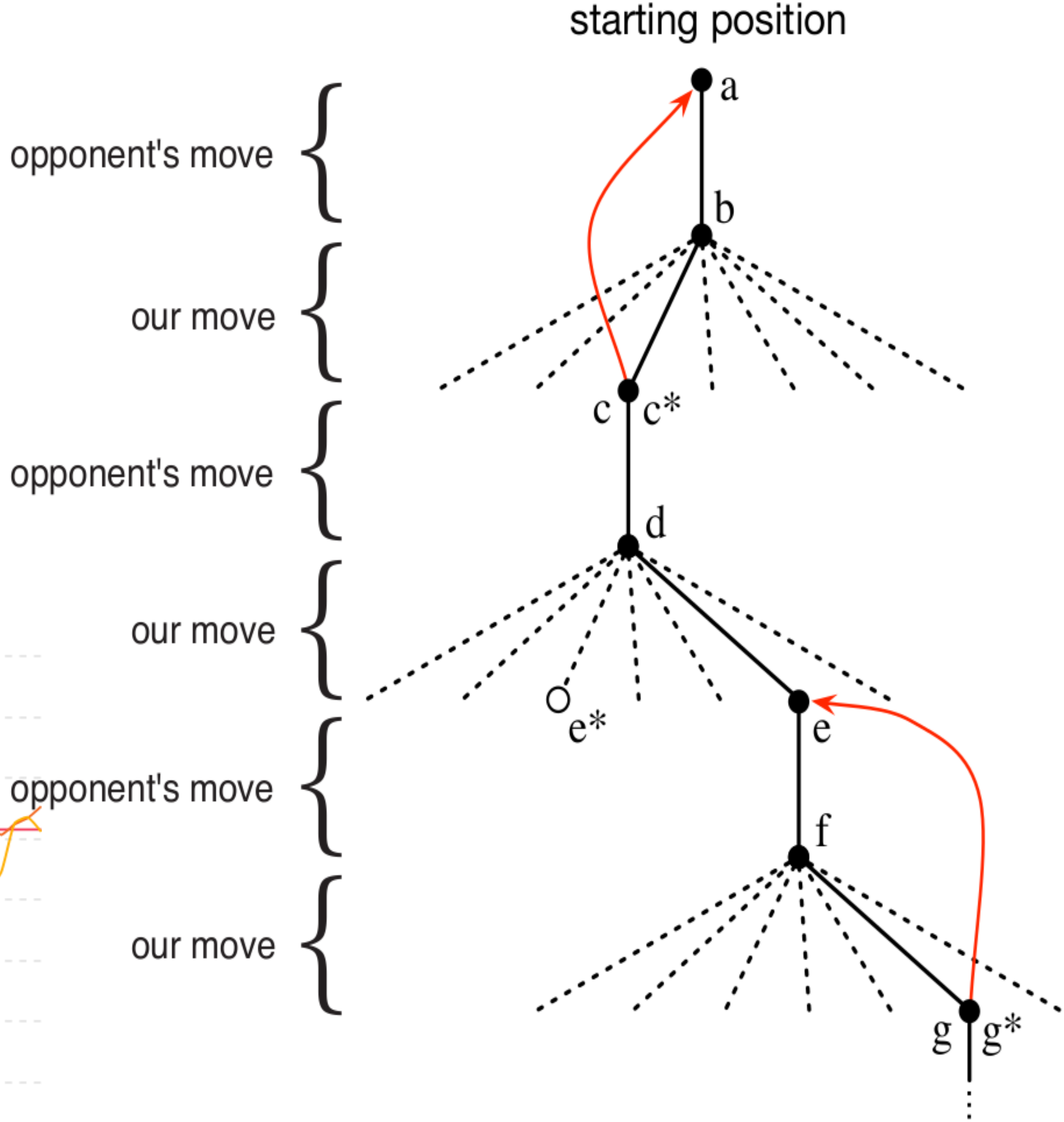
Questions:

(1) What happens if α is gradually made to 0 over many games with the opponent?

(2) What happens if α is gradually reduced over many games, but never made 0?

(3) What happens if α is kept constant throughout its life time?





Value Function Overview

Concept: A method for updating the estimated value of a state ($V(S_t)$) toward a target value based on the estimate of the next state ($V(S_{t+1})$).

The general update rule follows the pattern:

$$\text{New Estimate} \leftarrow \text{Old Estimate} + \text{Step Size} \times [\text{Target} - \text{Old Estimate}]$$

$$V_{\text{new}}(S_t) = V_{\text{old}}(S_t) + \alpha[V_{\text{target}} - V_{\text{old}}(S_t)]$$

In TD Learning:

$$V_{\text{target}} = V(S_{t+1}), \quad \alpha = \text{step-size parameter.}$$

Tic-Tac-Toe

Scenario: Agent ('X') is in current state S_t. Available moves are the empty squares (A1, A2, A3).

Current State (S_t)	Possible Action (A)	Resulting State (S_{t+1})	Estimated Value $V(S_{t+1})$
X O - - - - - O X	A_1: Top Right	X O **X** - - - - O X	0.65
S_t	A_2: Center	X O - - **X** - - O X	0.90
	A_3: Bottom Left	X O - - - - **X** O X	0.40

- **Decision:** The agent selects action A_2 because $V(S_{A2}) = 0.90$, which is the maximum value among all successor states.
- **RL Learning Mechanism (Update):** After taking action A_t and observing the next state S_{t+1}, the agent refines its estimate of the old state S_t using the Temporal-Difference method.
- The update follows the general learning rule:

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

- This process propagates value estimates backwards through the sequence of moves, linking the delayed reward (the win/loss at the end) to the state values in the middle of the game.

Value Function in Tic-Tac-Toe - Numerical

Concept: The value function estimates the **long-term desirability** of a state by calculating the expected total future reward. In Tic-Tac-Toe (T-T-T), it represents the **estimated probability of winning** from that state.

Example: Value Function Table

State (Board Position)	Estimated Value (V)	Explanation
S_A (Agent Won)	**1.0**	Agent (X) has 3 in a row $\Rightarrow$ probability of winning $= 1$.
S_B (Opponent Won/Draw)	**0.0**	Opponent (O) wins or draw $\Rightarrow$ probability of winning $= 0$.
S_C (Unresolved Mid-Game)	**0.5**	Initialized to 0.5 (50% chance of winning).

Value Function in Tic-Tac-Toe - Numerical

When the agent plays, it updates state values using the **Temporal-Difference (TD) learning** rule:

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

Given:

- $V(S_t) = 0.6$
- $V(S_{t+1}) = 0.9$
- $\alpha = 0.1$

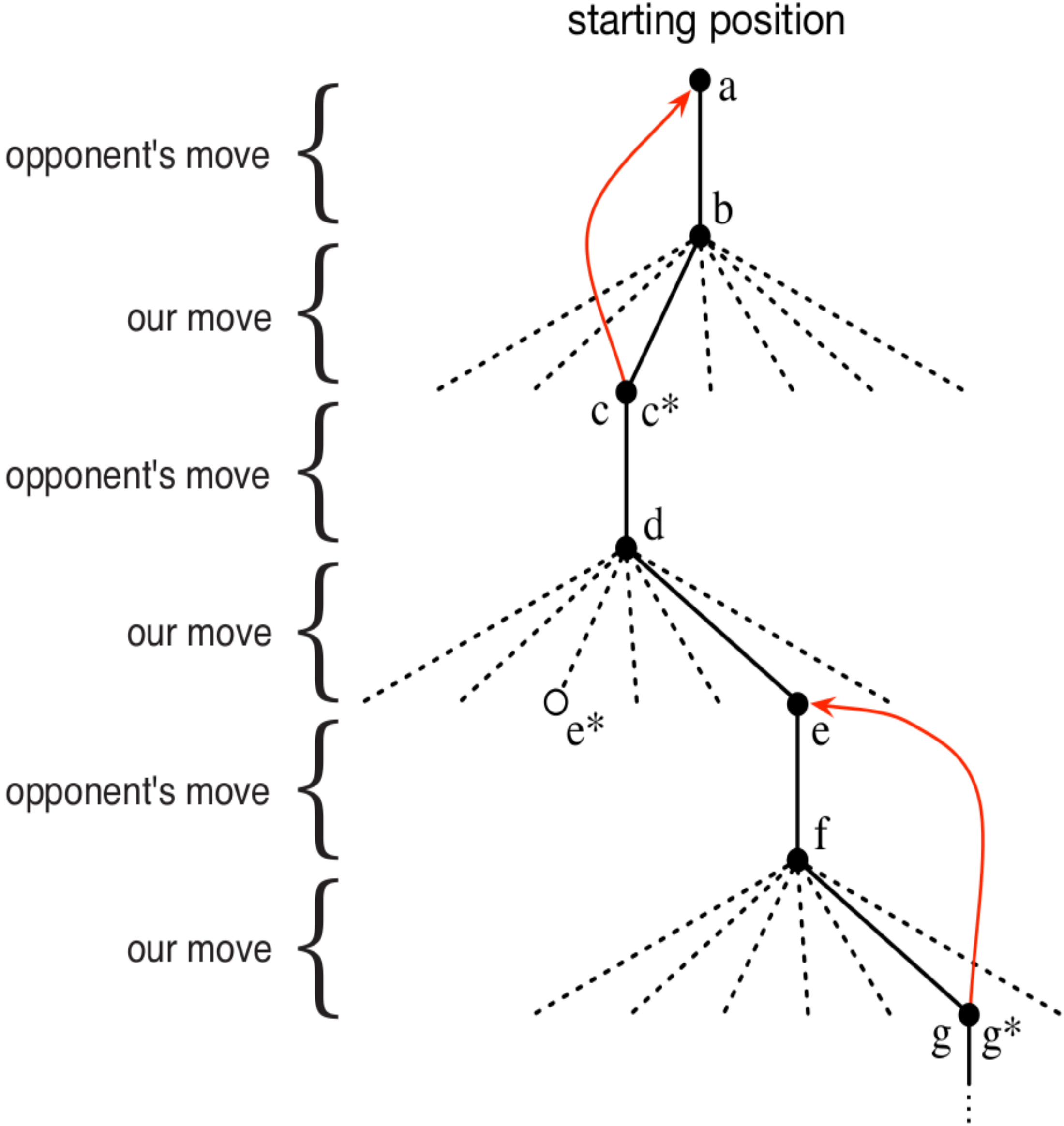
Steps:

1. TD Error: $V(S_{t+1}) - V(S_t) = 0.9 - 0.6 = 0.3$
2. Update Amount: $\alpha \times (\text{TD Error}) = 0.1 \times 0.3 = 0.03$
3. New Value: $V'(S_t) = 0.6 + 0.03 = 0.63$

Result: The value estimate for S_t improves from 0.6 to 0.63 by learning from the higher value of S_{t+1}.

Tic-Tac-Toc

Reading Assigned:
Identify how this reinforcement learning solution is different from solutions using minimax algorithm and genetic algorithms.
Post your answers in the discussion forum;



Temporal Difference Learning Rule

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

α- Step Size Parameter

Numerical Example #1

An agent in state S_t has an estimated value $V(S_t)$. After taking an action, it transitions to a successor state S_{t+1}, which has a higher estimated value. Compute $V_{\text{new}}(S_t)$

Given:

- $V(S_t) = 0.40$
- $V(S_{t+1}) = 0.80$
- $\alpha = 0.20$

Numerical Example # 2 - Homework

Now, the agent in state S_t encounters a successor state S_{t+1} with a **lower** estimated value. The update will decrease the current estimate. Compute $V_{new}(S_t)$

Given:

- $V(S_t) = 0.75$
- $V(S_{t+1}) = 0.50$
- $\alpha = 0.10$

References for today's session

1. Chapter 1 - Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto, Second Ed. , MIT Press

End of Session #1

Thank you