

Agent-Based Modeling and Simulation


Introduction to Reinforcement Learning

Dr. Alejandro Guerra-Hernández

Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana
*Campus Sur, Calle Paseo Lote II, Sección Segunda No 112,
Nuevo Xalapa, Xalapa, Ver., México 91097*
<mailto:aguerra@uv.mx>
<https://www.uv.mx/personal/aguerra/abms>

Doctorado en Inteligencia Artificial 2024



- ▶ These slides are based on the book of Sutton and Barto [1], chapter 1. Any difference with this source is my responsibility.
- ▶ This work is licensed under **CC-BY-NC-SA 4.0** 
- ▶ To view a copy of this license, visit:

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Main Ideas

- ▶ The idea of **learning by interacting with our environment** is probably the first to occur to us about the **nature of learning**.
- ▶ This is based on **sensorimotor** connection to the environment that provides information about **cause and effect**, e.g., the consequences of actions and what to do in order to achieve goals.
- ▶ A major source of **knowledge** about our environment and ourselves.
- ▶ We will explore a **computational** approach to goal-directed learning from interaction.



Informal Definition

- ▶ Reinforcement learning (RL) is about **what to do**, *i.e.*, how to map situations to actions, to **maximize a numerical reward** signal.
- ▶ The learner is not told which actions to take, but instead must **discover** which actions yield the most reward by trying them.
- ▶ In the most interesting and challenging cases, **actions may affect** not only the immediate reward but also the next situation and, through that, **all subsequent rewards**.
- ▶ **Trial-and-error search** and **delayed reward** are the two most important distinguishing features of RL.



RL Problems, Methods, and Research Area

- ▶ RL is simultaneously a **problem**, a class of solution **methods** that work well on the problem, and the **field** that studies this problem and its solution methods.
- ▶ The **distinction** between problems and solution methods is very important in reinforcement learning; failing to make this distinction is the source of many confusions.

Formalization Choice

- ▶ Formalizing RL will use ideas from the optimal control from dynamic systems theory, e.g., incompletely-known **Markov Decision Processes (MDPs)**.
- ▶ MDPs are intended to include **three aspects of a learning agent** in their simplest possible forms, without trivializing any of them:
 - ▶ The capacity of **sensing** the states of the environment to some extent;
 - ▶ The capacity of **acting** in the environment, affecting their states; and
 - ▶ The capacity of achieving **goals** relating to the state of the environment.



Supervised Learning I

- ▶ Learning from a training set of **labeled examples** provided by a knowledgeable external supervisor.
- ▶ Each example is a description of a situation together with a specification, the label, of the correct action the system should take, which is often to identify a **category** to which the situation belongs, *i.e.*, a classification task.
- ▶ The objective of this kind of learning is to **generalize** its responses so that it acts correctly in situations not present in the training set.
- ▶ Although this is an important kind of learning, it is not adequate for learning from **interaction** by itself.



Supervised Learning II

- ▶ In interactive problems it is often **impractical to obtain examples** of the desired behavior that are both **correct** and **representative** of all the situations in which the agent has to act.
- ▶ In uncharted territory, where one would expect learning to be most beneficial, an agent must be able to learn from its **own experience**.



Unsupervised Learning

- ▶ This is about **finding patterns** hidden in collections of **unlabeled data**.
- ▶ Supervised and unsupervised learning would seem to **exhaustively classify machine learning** (ML) paradigms, but they do not.
- ▶ Although RL does not rely on examples of correct behavior, it is not a unsupervised ML method: It is trying to **maximize a reward** signal instead of trying to find hidden structure.
- ▶ Uncovering structure in an agent's experience can certainly be useful in RL, but by itself **does not address the RL problem** of maximizing a reward signal.
- ▶ Therefore RL is a **third ML paradigm**, alongside supervised and unsupervised learning, and perhaps other paradigms.



Exploration vs Exploitation

- ▶ A particular **dilemma** arises in RL, but not in other kinds of learning.
- ▶ To obtain a lot of reward, a RL agent must prefer actions tried in the past and found to be **effective**, but to discover such actions, it has to try **new** actions that it has not selected before.
- ▶ The agent has to **exploit** what it has already experienced in order to obtain reward, but it also has to **explore** in order to make better action selections in the future.
- ▶ Although the dilemma has been intensively studied, it remains **unsolved**.



Goal-oriented Behavior

- ▶ RL explicitly considers the **whole** problem of a goal-directed agent interacting with an uncertain environment.
- ▶ **Example:** Often ML is concerned with supervise learning without explicitly specifying how such an ability would finally be used for.
- ▶ **Example:** Theories of planning with general goals don't consider planning's role in real-time decision making; nor the question of where the predictive models necessary for planning would come from.
- ▶ These examples focus on **isolated sub-problems** of agency, being inherently limited.

Agency

- ▶ We start with a complete, interactive, goal-seeking **agent**.
- ▶ All RL agents have explicit goals, can sense aspects of their environment, and can choose actions to influence them.
- ▶ It is assumed that RL agents has to operate despite significant **uncertainty** about the environment.
- ▶ When **planning** is involved, the following questions must be addressed:
 - ▶ The interplay between planning and real-time action selection.
 - ▶ How are environment models acquired and improved?



RL agents as subsystems

- ▶ Completeness do not mean something like a complete organism or robot.
- ▶ Agents can also be **components** of a larger behaving system, interacting with the rest of the system and, indirectly, with the system's environment.
- ▶ **Example:** An agent that monitors the charge level of a robot's battery and sends commands to the robot's control architecture.

Background

- ▶ RL is part of a decades-long trend within AI and ML toward greater integration with **statistics**, **optimization** and other mathematical subjects.
- ▶ **Example:** The ability of some RL algorithms to learn with parameterized approximators addresses the classical “curse of dimensionality” in **operation research** and **control theory**.
- ▶ Interactions are stronger with **psychology** and **neuroscience**, with substantial benefits in both ways.
- ▶ RL research looks for general principles of **learning, search, and decision making**, simpler and fewer general principles of AI.



Chess player

- ▶ A master chess player **makes a move**.
- ▶ The choice is informed both by **planning**, anticipating possible replies and counterreplies, and by immediate, **intuitive judgments** of the desirability of particular positions and moves.

Adaptive Controller

- ▶ An adaptive controller **adjusts parameters** of a petroleum refinery's operation in real time.
- ▶ The controller **optimizes** the yield/cost/quality trade-off on the basis of specified marginal costs **without sticking strictly** to the set points originally suggested by engineers.

Animals

- ▶ A gazelle calf struggles to its feet minutes after being born.
- ▶ **Half an hour later** it is running at 20 miles per hour.

Robots

- ▶ A mobile robot **decides** whether it should enter a new room in search of more trash to collect or start trying to find its way back to its battery recharging station.
- ▶ It makes its decision based on the **current charge level** of its battery and **how quickly and easily** it has been able to find the recharger in the past.

We I

- ▶ Phil prepares his breakfast.
- ▶ This mundane activity reveals a complex web of **conditional behavior** and **interlocking goal-subgoal** relationships:
 - ▶ Walking to the cupboard.
 - ▶ Opening it.
 - ▶ Selecting a cereal box, then reaching for, grasping, and retrieving the box.
- ▶ The same for getting the bowl, spoon and milk carton.
- ▶ Each step involves a series of eye movements to **obtain information** and to guide reaching and locomotion.
- ▶ **Rapid judgments** are continually made about how to carry the objects or whether it is better to ferry some of them to the dinning table before obtaining others.



We II

- ▶ Each step is **guided by goals**, such as grasping a spoon to eat with once the cereal is prepared and ultimately obtaining nourishment.
- ▶ Whether he is aware of it or not, Phil is accessing **information about the state of his body** that determines his nutritional needs, level of hunger, and food preferences.



Interaction

- ▶ These examples involve **interaction** between an active decision-making agent and its environment, within which the agent seeks to achieve a **goal**, despite **uncertainty** about its environment.
- ▶ The agent's actions affect the **future state** of the environment, e.g., the next chess position; the reservoir level of the refinery; the robot's next location and the future charge level of its battery..
- ▶ Thereby affecting the actions and opportunities available to the agent at **later times**.
- ▶ Correct choice requires taking into account **indirect, delayed consequences** of actions, and thus may require foresight or planning.

Incompleteness

- ▶ In all these examples, the effects of actions **cannot be fully predicted**; thus the agent must monitor its environment frequently and react appropriately.
- ▶ **Example:** Phil must watch the milk he pours into his cereal bowl to keep it from overflowing.

Goal orientedness

- ▶ All these examples involve goals that are **explicit** in the sense that the agent can **judge progress** toward its goal based on what it can sense directly.
- ▶ **Example:** The chess player knows whether or not he wins; the refinery controller knows how much petroleum is being produced; the gazelle calf knows when it falls; the mobile robot knows when its batteries run down; and Phil knows whether or not he is enjoying his breakfast.

Experience

- ▶ In all these examples the agent can use experience to **improve** its performance over time.
- ▶ **Example:** The chess player refines intuition he uses to evaluate positions, thereby improving his play; the gazelle calf improves the efficiency with which it can run; Phil learns to streamline making his breakfast.
- ▶ The **knowledge** the agent brings to the task at the start –either from previous experience with related tasks or built into it by design or evolution– influences what is useful or easy to learn, but interaction with the environment is essential for adjusting behavior to exploit specific features of the task.

Elements

- ▶ Beyond the agent and the environment, one can identify four main **sub-elements** of a RL system:
 - ▶ A policy.
 - ▶ A reward signal.
 - ▶ A value function.
 - ▶ Optionally, a model of the environment.

Policy

- ▶ Roughly speaking, a policy is a **mapping** from perceived states of the environment to actions to be taken when in those states, *i.e.*, a set of **stimulus–response rules** or **associations**.
- ▶ In some cases the policy may be a simple **function** or lookup **table**, whereas in others it may involve extensive computation such as a **search process**.
- ▶ The policy is the core of a RL agent in the sense that it alone is sufficient to **determine behavior**.
- ▶ In general, policies may be **stochastic**, specifying probabilities for each action.

Reward Signal

- ▶ A reward signal **defines the goal** of a RL problem.
- ▶ On each time step, the environment sends to the RL agent a **single number** called the reward.
- ▶ The agent's sole objective is to **maximize the total reward** it receives over the long run.
- ▶ It defines what are the good and bad events for the agent, analogous to **pleasure or pain**.
- ▶ The reward signal is the primary basis for **altering the policy**; if an action selected by the policy is followed by low reward, then the policy may be changed to select some other action in the future.
- ▶ In general, reward signals may be **stochastic functions** of the state of the environment and the actions taken.



Value Function

- ▶ A value function specifies what is **good in the long run**.
- ▶ Roughly speaking, the value of a state is the total amount of reward an agent can **expect to accumulate** over the future, starting there.
- ▶ Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the **long-term desirability** of states after taking into account the states that are likely to follow and the rewards available in those states.
- ▶ **Example:** Sacrificing the queen in a chess game has a very low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards, e.g., check mate.
- ▶ Values correspond to a more refined and **farsighted judgment**.



Rewards vs Value Functions I

- ▶ Rewards are in a sense primary, whereas values, as **predictions of rewards**, are secondary.
- ▶ Without rewards there could be no values, and the only **purpose** of estimating values is to achieve more reward.
- ▶ Nevertheless, it is values with which we are most concerned when **making and evaluating decisions**.
- ▶ Action choices are based on **value judgments** –We seek actions that bring about states of highest value, not highest reward, because these actions obtain the greatest amount of reward for us over the long run.
- ▶ Unfortunately, it is **much harder** to determine values than it is to determine rewards.



Rewards vs Value Functions II

- ▶ Rewards are basically given **directly** by the environment, but values must be **estimated** and re-estimated from the sequences of observations an agent makes over its entire lifetime.
- ▶ In fact, the most important component of almost all RL algorithms we consider is a method for **efficiently estimating values**.
- ▶ The central role of value estimation is arguably the most important thing that has been learned about RL over the last six decades.



Model of the Environment

- ▶ A model **mimics** the behavior of the environment, or more generally, allows **inferences** to be made about how the environment will behave.
- ▶ **Example:** Given a state and action, the model might predict the resultant **next state** and **next reward**.
- ▶ Models are used for planning, by which we mean any way of **deciding** on a course of action by considering possible **future situations before they actually happen**.
- ▶ Methods for solving RL problems that use models and planning are called **model-based methods**, as opposed to simpler **model-free methods** that are explicitly trial-and-error learners –viewed as almost the opposite of planning.

States I

- ▶ States are an **input** to the policy and value function, and both **input and output** of the model.
- ▶ Informally, a state as a **signal** conveying to the agent some **sense** of “how the environment is” at a particular time.
- ▶ The formal definition of state will be given by the framework of **MDPs**, however, follow the informal meaning and think of the state as whatever **information** is available to the agent about its environment.
- ▶ We assume that the state signal is produced by some **preprocessing system** that is nominally part of the agent’s **environment**.
- ▶ We **do not address** the issues of constructing, changing, or learning the state signal.



States II

- ▶ We take this approach not because we consider state representation to be unimportant, but in order to focus fully on the **decision-making issues**.
- ▶ In other words, our concern is not with designing the state signal, but with deciding what action to take as a function of **whatever** state signal is available.

Value Functions I

- ▶ Most of the RL methods we consider are structured around **estimating value functions**, but it is not strictly necessary to do this to solve RL problems.
- ▶ **Example:** Solution methods such as genetic algorithms, genetic programming, simulated annealing, and other **optimization methods** never estimate value functions.
- ▶ These methods apply multiple static policies each interacting over an extended period of time with a separate instance of the environment. The policies that obtain the most reward, and random variations of them, are carried over to the next generation of policies, and the process repeats.



Value Functions II

- ▶ We call these **evolutionary methods** because their operation is analogous to the way biological evolution produces organisms with skilled behavior even if they do not learn during their individual lifetimes.
- ▶ If the space of policies is sufficiently small, or can be structured so that good policies are common or easy to find—or if a lot of time is available for the search—then evolutionary methods can be **effective**.
- ▶ In addition, evolutionary methods have advantages on problems in which the learning agent cannot sense the **complete** state of its environment.



RL vs Evolutionary Methods I

- ▶ Our focus is on RL methods that learn while **interacting** with the environment, which evolutionary methods do not do.
- ▶ Methods able to take advantage of the details of individual behavioral interactions can be **much more efficient** than evolutionary methods in many cases.
- ▶ Evolutionary methods **ignore** much of the useful structure of the RL problem:
 - ▶ They do not use the fact that the policy they are searching for is a function from states to actions;
 - ▶ They do not notice which states an individual passes through during its lifetime, or which actions it selects.

RL vs Evolutionary Methods II

- ▶ In some cases this information can be **misleading**, e.g., when states are misperceived, but more often it should enable more efficient search.
- ▶ Although evolution and learning share many features and naturally work together, we **do not** consider evolutionary methods by themselves to be especially well suited to RL problems.

Assumptions

| | | |
|---|---|---|
| X | O | O |
| O | X | X |
| | | X |

- ▶ We are playing against an **imperfect** player, one whose play is sometimes incorrect and allows us to win.
- ▶ Draws and losses are equally **bad** for us.
- ▶ How might we construct a player that will find the imperfections in its opponent's play and **learn** to maximize its chances of winning?



Difficulties I

- ▶ Although this is a simple problem, it **cannot** readily be solved in a satisfactory way through classical techniques:
 - ▶ The classical **minimax** solution from game theory is not correct here because it assumes a particular way of playing by the opponent, e.g., a minimax player would never reach a game state from which it could lose, even if in fact it always won from that state because of incorrect play by the opponent.
 - ▶ Classical optimization methods for sequential decision problems, such as **dynamic programming**, can compute an optimal solution for any opponent, but require as input a complete specification of that opponent, including the probabilities with which the opponent makes each move in each board state.



Difficulties II

- ▶ Let us assume that this information is not available a priori for this problem, as it is not for the vast majority of problems of practical interest. On the other hand, such information can be estimated from **experience**, in this case by playing many games against the opponent.
- ▶ About the best one can do on this problem is first to learn a model of the opponent's behavior, up to some level of confidence, and then apply dynamic programming to compute an **optimal solution given the approximate opponent model**.



Evolutionary Approach I

- ▶ An evolutionary method applied to this problem would directly search the **space of possible policies** for one with a high probability of winning against the opponent.
- ▶ Here, a policy is a **rule** that tells the player what move to make for every state of the game—every possible configuration of Xs and Os on the three-by-three board.
- ▶ For each policy considered, an estimate of its **winning probability** would be obtained by playing some number of games against the opponent.
- ▶ A typical evolutionary method would **hill-climb** in policy space, successively generating and evaluating policies in an attempt to obtain incremental improvements.



Evolutionary Approach II

- ▶ Or, perhaps, a genetic-style algorithm could be used that would maintain and evaluate a **population of policies**. Literally hundreds of different optimization methods could be applied.



Value Function Approach I

- ▶ First we would set up a **table** of numbers, one for each possible state of the game.
- ▶ Each number will be the latest estimate of the **probability of our winning** from that state.
- ▶ We treat this estimate as the **state's value**, and the whole table is the learned **value function**.
- ▶ State A has higher value than state B, or is considered **better** than state B, if the current estimate of the probability of our winning from A is higher than it is from B.

Value Function Approach II

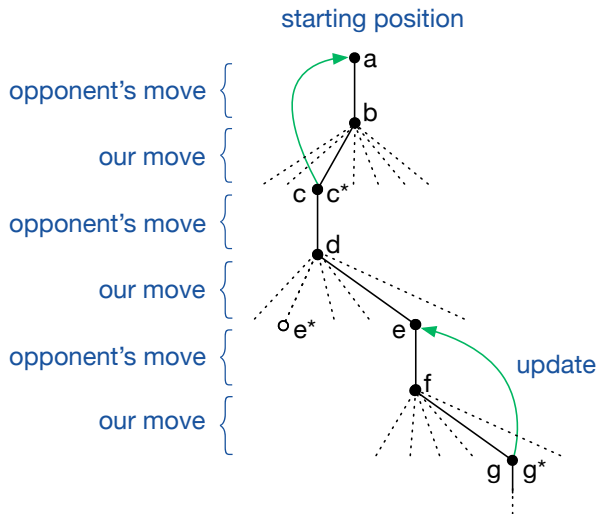
- ▶ Assuming we always play Xs, then for all states with three Xs in a row the **probability** of winning is 1, because we have already won. Similarly, for all states with three Os in a row, or that are filled up, the correct probability is 0, as we cannot win from them.
- ▶ We set the initial values of all the other states to 0.5, representing a **guess** that we have a 50% chance of winning.

Exploitation and Exploration

- ▶ We then **play many games** against the opponent.
- ▶ To select our moves we examine the states that would result from each of our possible moves (one for each blank space on the board) and look up their **current values** in the table.
- ▶ Most of the time we move **greedily**, selecting the move that leads to the state with greatest value, that is, with the highest estimated probability of winning.
- ▶ Occasionally, however, we select **randomly** from among the other moves instead. These are called exploratory moves because they cause us to experience states that we might otherwise never see.



Graphically



Updates I

- ▶ While we are playing, we **change** the values of the states in which we find ourselves during the game: We attempt to make them **more accurate** estimates of the probabilities of winning.
- ▶ To do this, we **back up** the value of the state after each greedy move to the state before the move (green arrows in the previous slide).
- ▶ More precisely, the current value of the earlier state is **updated** to be closer to the value of the later state, i.e., moving the earlier state's value a fraction of the way toward the value of the later state:

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

Updates II

- ▶ α is a small positive fraction called the **step-size** parameter, which influences the rate of learning.
- ▶ This update rule is an example of a **temporal-difference** learning method, so called because its changes are based on a difference $V(S_{t+1}) - V(S_t)$, between two estimates at two successive times.

Flexibility and Performance

- ▶ If the step-size parameter is reduced properly over time, then this method **converges**, for any fixed opponent, to the true probabilities of winning from each state given optimal play by our player.
- ▶ Furthermore, the moves then taken (except on exploratory moves) are in fact the **optimal moves** against this (imperfect) opponent.
- ▶ In other words, the method converges to an optimal policy for playing the game against **this opponent**.
- ▶ If the step-size parameter is not reduced all the way to zero over time, then this player also plays well against opponents that **slowly change** their way of playing.

RL vs Evolutionary Methods Revisited I

- ▶ To **evaluate** a policy an evolutionary method holds the policy fixed and plays many games against the opponent, or simulates many games using a model of the opponent.
- ▶ The **frequency** of wins gives an unbiased estimate of the probability of winning with that policy, and can be used to direct the next policy selection.
- ▶ But each policy change is made only **after** many games, and only the final outcome of each game is used: what happens during the games is ignored.

RL vs Evolutionary Methods Revisited II

- ▶ **Example:** If the player wins, then **all** of its behavior in the game is given credit, independently of how specific moves might have been critical to the win. Credit is even given to moves that **never occurred!**
- ▶ Value function methods, in contrast, allow **individual states** to be evaluated.
- ▶ In the end, evolutionary and value function methods both search the **space of policies**, but learning a value function takes advantage of information available **during** the course of play.



Apparent Limitations

- ▶ Although tic-tac-toe is a two-person game, RL also applies in the case in which there is no external adversary, that is, in the case of a **game against nature**.
- ▶ RL also is not restricted to problems in which behavior breaks down into separate **episodes**, like the separate games of tic-tac-toe, with reward only at the end of each episode. It is just as applicable when behavior continues indefinitely and when rewards of various magnitudes can be received at any time.
- ▶ RL is also applicable to problems that do not even break down into **discrete time steps** like the plays of tic-tac-toe. The general principles apply to continuous-time problems as well, although the theory gets more complicated.



Larger Problems

- ▶ Tic-tac-toe has a relatively small, **finite** state set, whereas RL can be used when the state set is very large, or even infinite.
- ▶ **Example:** Gerry Tesauro [2, 3] combined the algorithm described above with an artificial neural network to learn to play **backgammon**, which has approximately 10^{20} states. The neural network provides the ability to **generalize** from its experience, so that in new states it selects moves based on information saved from similar states faced in the past, as determined by its network.
- ▶ How well a RL can work in problems with such large state sets is intimately tied to how appropriately it can generalize from past experience.
- ▶ **Supervised learning** methods with RL. Neural networks and deep learning are not the only, or the best, way to do this.



Knowledge

- ▶ Learning started with no **prior knowledge** beyond the rules of the game, but reinforcement learning by no means entails a **tabula rasa view** of learning and intelligence.
- ▶ On the contrary, prior information can be **incorporated** into RL in a variety of ways that can be critical for efficient learning
- ▶ We also have access to the **true state** in the tic-tac-toe example, whereas RL can also be applied when part of the state is hidden, or when different states appear to the learner to be the same.



Model

- ▶ The tic-tac-toe player was able to **look ahead** and know the states that would result from each of its possible moves.
- ▶ To do this, it had to have a **model** of the game that allowed it to foresee how its environment would change in response to moves that it might never make.
- ▶ Many problems are like this, but in others even a short-term model of the effects of actions is **lacking**.
- ▶ RL can be applied in **either case**. A model is not required, but models can easily be used if they are available or can be learned.

Model-free Systems

- ▶ There are RL methods that **do not need** any kind of environment model at all.
- ▶ Model-free systems cannot even think about how their environments will change in response to a single action.
- ▶ The tic-tac-toe player is model-free in this sense with respect to its **opponent**: it has no model of its opponent of any kind.
- ▶ Because models have to be reasonably accurate to be useful, model-free methods can have advantages over more complex methods when the real bottleneck in solving a problem is the difficulty of constructing a sufficiently accurate environment model.
- ▶ Model-free methods are also important **building blocks** for model-based methods.



Referencias I

- [1] R Sutton and AG Barto. *Reinforcement Learning: An Introduction*. 2nd. Cambridge, MA, USA: The MIT Press, 2018.
- [2] G Tesauro. "Practical issues in temporal difference learning". In: *Advances in neural information processing systems*. 1992, pp. 259–266.
- [3] G Tesauro. "Temporal difference learning and TD-Gammon". In: *Communications of the ACM* 38.3 (1995), pp. 58–68.