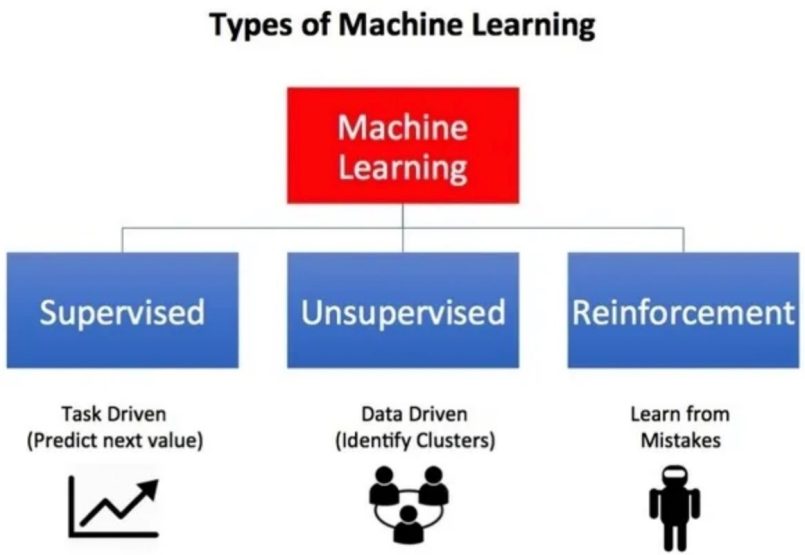
# CS 4100: Introduction to AI

Wayne Snyder
Northeastern University

---

Lecture 16: Reinforcement Learning; Genetic Algorithms



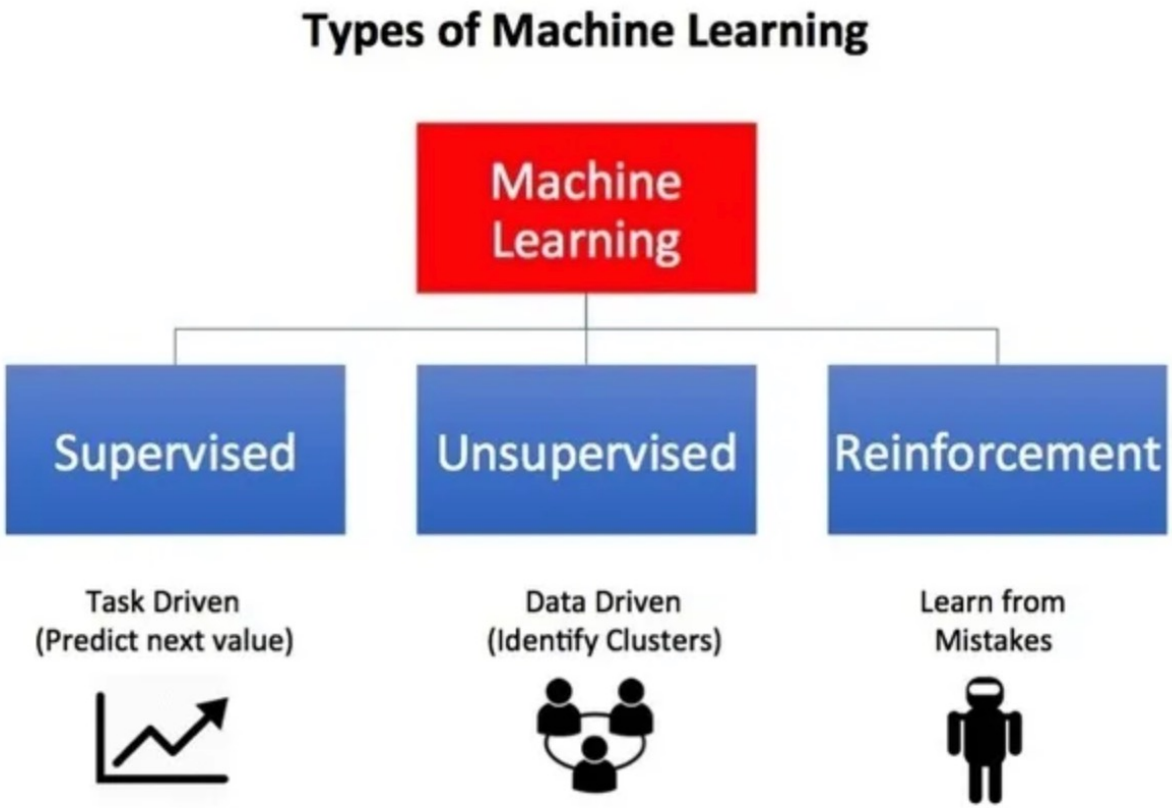
# Reinforcement Learning

Plan for today:

- Introduction to Reinforcement Learning
  - Relationship to Supervised and Unsupervised Learning
  - Machine Learning Framework
  - Basic Concepts and Terms
  - Applications

- Genetic Algorithms
  - Basic Concepts and Terms
  - Application: The Prisoner's Dilemma

# Reinforcement Learning



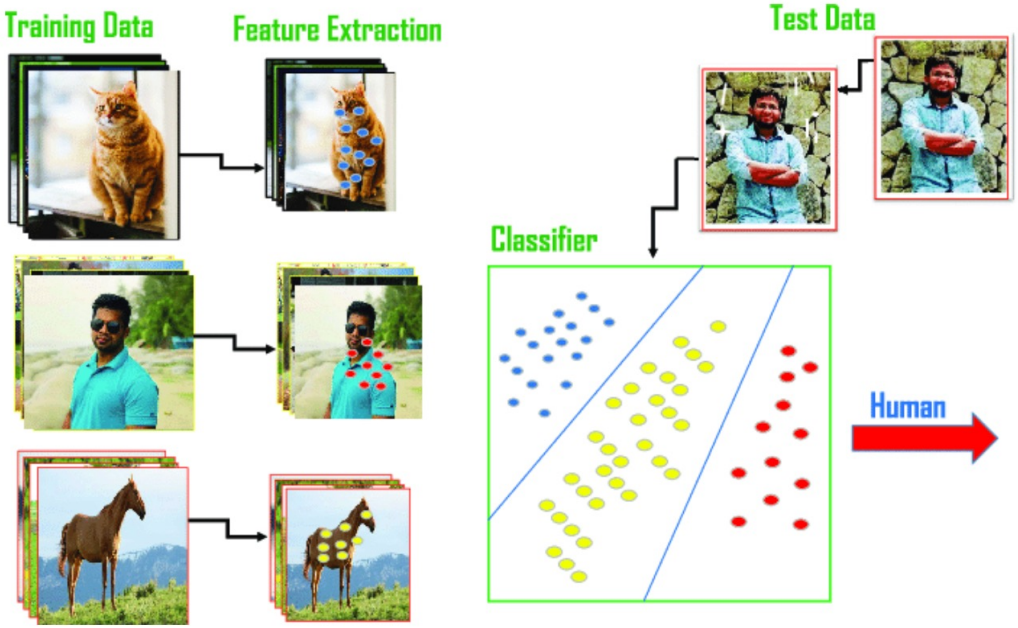
# Reinforcement Learning


In Supervised Learning we have

Data Source:  Labelled data (Lots of it!)

Framework: Train on labelled data, test on labelled data with labels withheld.

Problem to be Solved: given unlabelled data in future, label it!



# Reinforcement Learning

Unsupervised

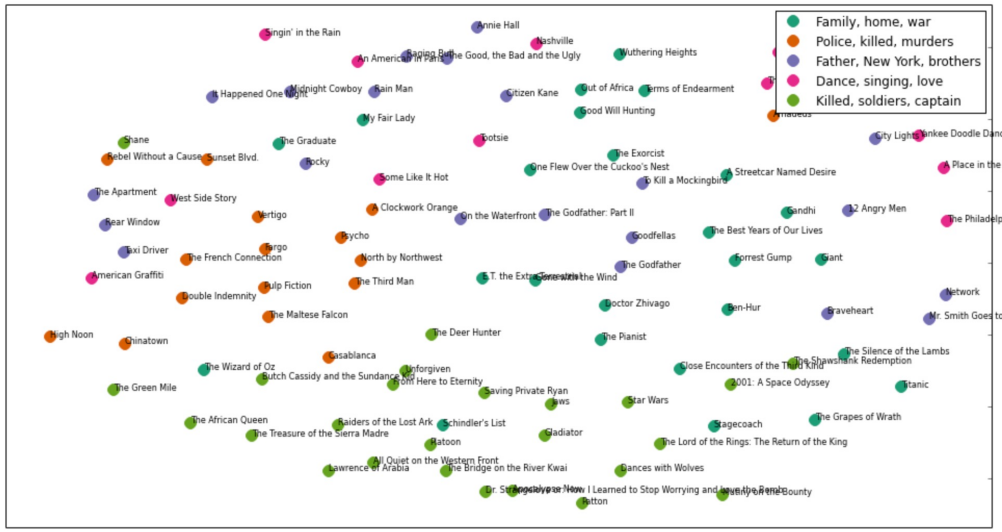
Data Driven
(Identify Clusters)

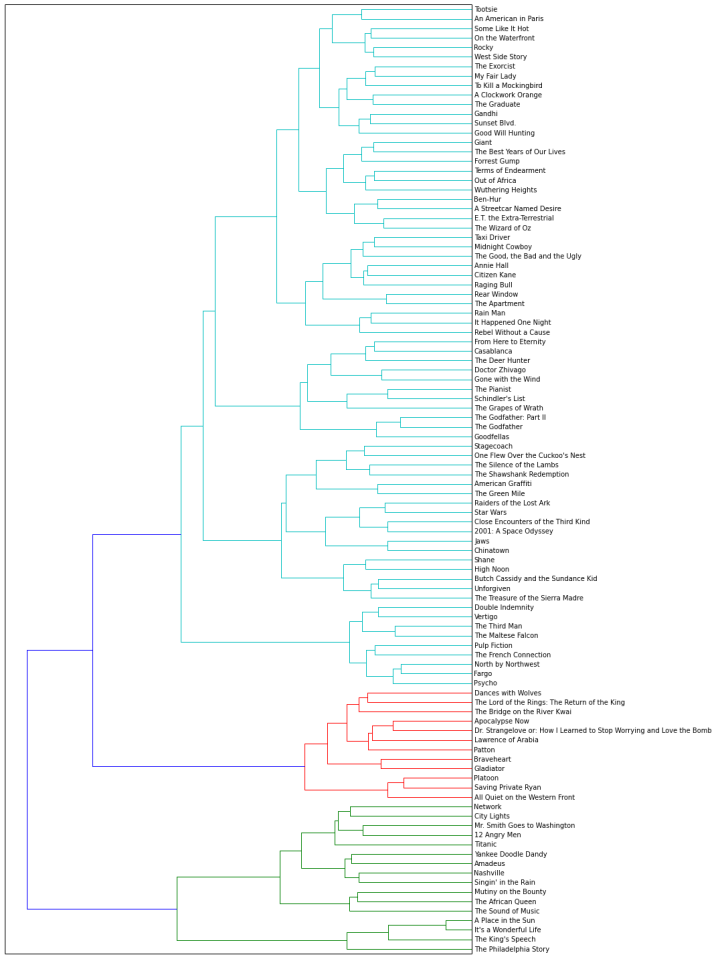
In Unsupervised Learning we have

Data source:  Unlabelled data (lots of it!)

Method: Clustering algorithms (many!)

Problem to be Solved : Find patterns in data, classification, exploratory data analysis, etc.



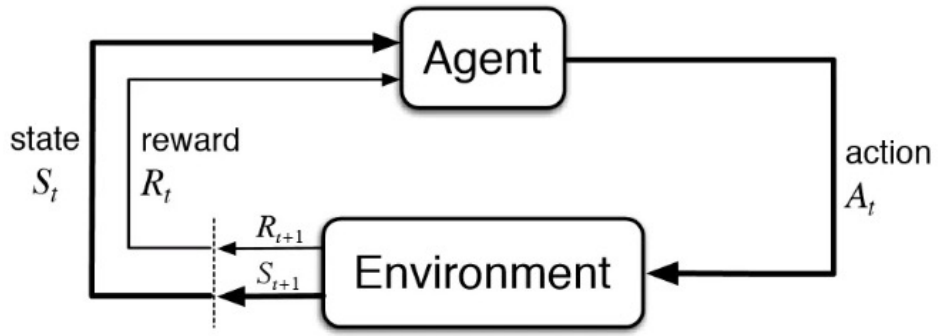
# Reinforcement Learning


How is Reinforcement Learning Different?

Data Source: An Agent interacting with an Environment.

Framework: The Agent performs Actions and changes its State over time following a Policy, and gets Rewards from the Environment. A heuristic function gives the projected Value of an action in a given state.

Problem to be Solved : Learn a policy which will maximize cumulative reward over time.



# Reinforcement Learning

Our main task today is to understand the following concepts:

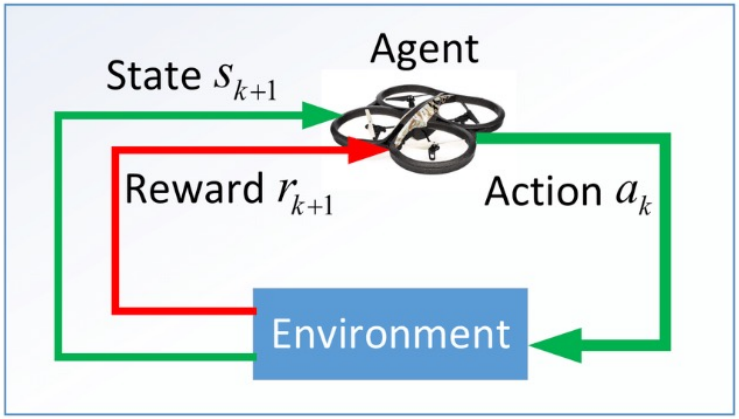
1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

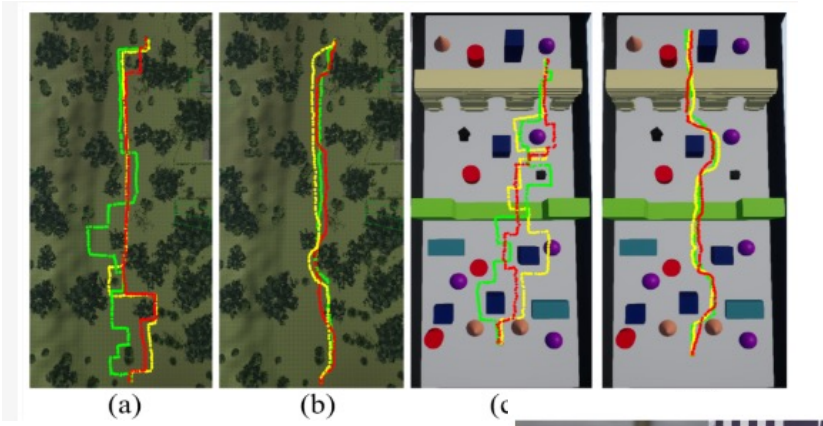
But before we get into the details, let's consider a number of situations where RL has been used.

# Reinforcement Learning: Examples

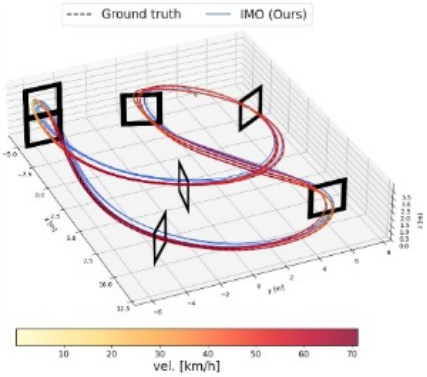
1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

Drones



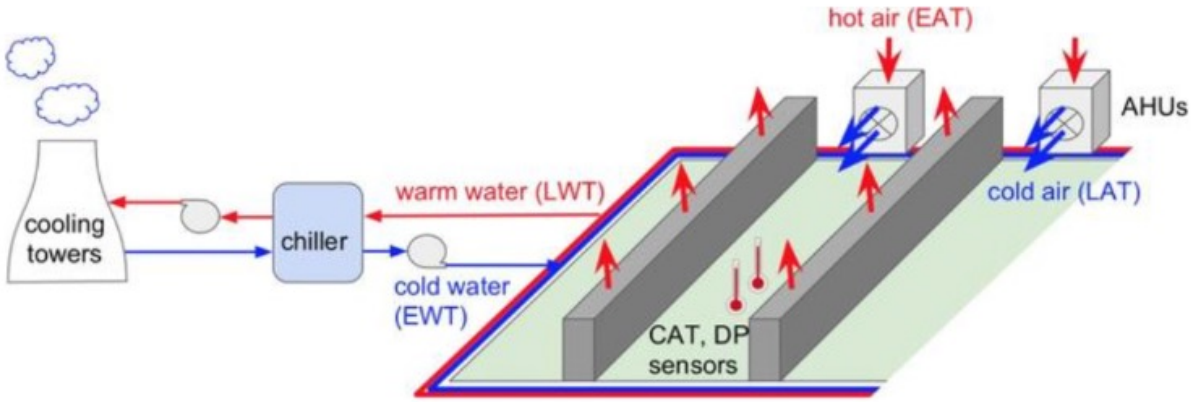




# Reinforcement Learning: Examples

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

## Cooling and Heating of Industrial Buildings

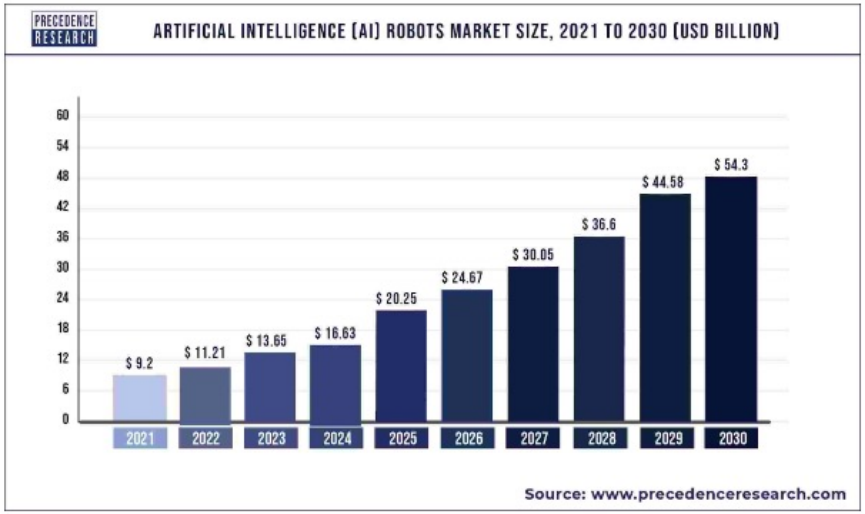



*Datacenters cooling*

# Reinforcement Learning: Examples

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

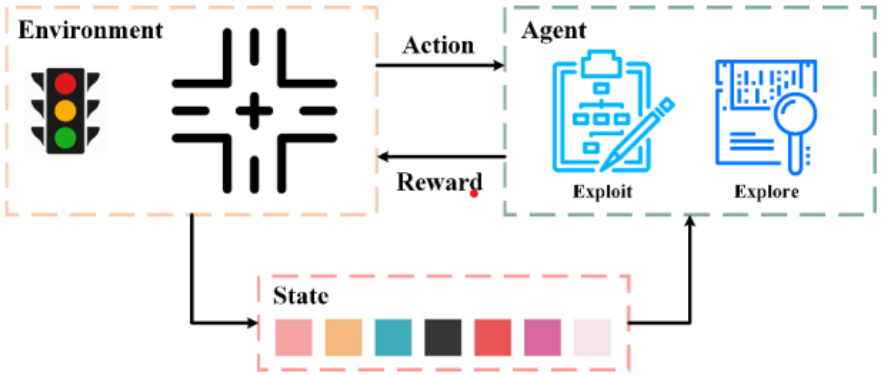
## Industrial Robots



# Reinforcement Learning: Examples

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

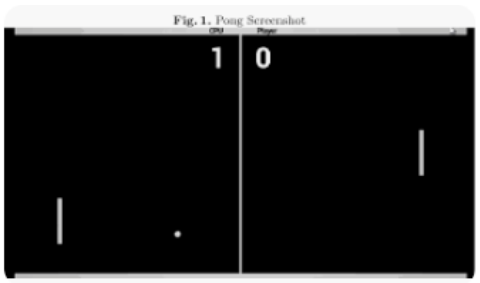
## Traffic Flow in Cities

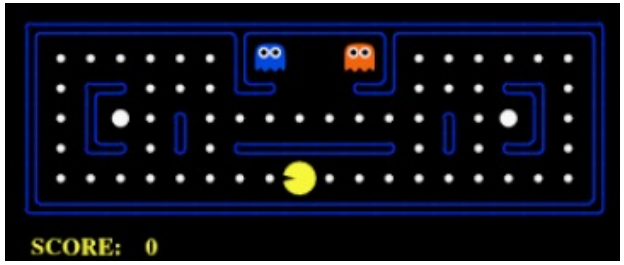



# Reinforcement Learning: Examples

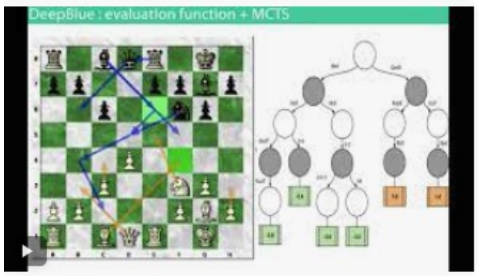
1. Environment
2. Agent
3. State
4. Action
5. Reward
6. Policy
7. Value

## Games





# Reinforcement Learning: Basic Concepts

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

Environment

The Environment is very broadly defined as the context in which the agent acts to achieve rewards. Generally it can be thought of as a physical space in which the agent moves, senses, acts, and receives rewards.

It may be:

- Discrete or Continuous (with corresponding notion of time)
- Deterministic or stochastic
- Fully observable or Partial observable
- Contain other agents or not
- Be static or dynamic

The environment is typically modeled as a state space.....

# Reinforcement Learning: Basic Concepts

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

## State

The state is "whatever information is available to the agent about its environment,", most typically, a representation of the position in the environment at a particular time step.

It may be produced by observation with sensors (where am I?) or maintained as an internal "memory."

Typically represented as a vector of floating-point numbers.

[Technical detail: States are assumed to have a Markov Property, meaning there is no "memory" of previous states. More on this next week, when we explore the notion of Markov Decision Processes, which is the basic mathematical framework for describing RL problems.]

## Action

An action is the mechanism by which the agent transitions to a new state in the environment.

# Reinforcement Learning: Basic Concepts

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

## Reward

Often called the Reward Signal, this is a floating-point number representing some unit of value.

The reward signal specifies what are "good" and "bad" outcomes for the agent through time.

Rewards may be received incrementally or all at once at the end.

## Value

The Value Function V(s) maps states to floating-point numbers indicating the expected total reward for an agent starting from state s.

Roughly, a value function is a heuristic function such as we have seen in best-first search and Min/Max, which estimates future cumulative rewards.

# Reinforcement Learning: Basic Concepts

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

Two caveats: First, carefully distinguish value and rewards; from the classic book on RL by Sutton and Barto:

> Whereas the reward signal indicates what is good in an immediate sense, 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 from that state. 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. For example, a state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards. Or the reverse could be true. To make a human analogy, rewards are somewhat like pleasure (if high) and pain (if low), whereas values correspond to a more refined and farsighted judgment of how pleased or displeased we are that our environment is in a particular state.

For now, just think of it like the h(x) function in A* or the eval(...) function in adversarial search.....

Second, there are significant classes of RL algorithms, such as Genetic Algorithms, which do not use a value function at all.

We will look at genetic algorithms later in the lecture....

# Reinforcement Learning: Basic Concepts

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

Also, there are significant classes of RL algorithm, such as Evolutionary Algorithms, which do not use a value function.

From the classic book on RL by Sutton and Barto:

> Whereas the reward signal indicates what is good in an immediate sense, 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 from that state. 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. For example, a state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards. Or the reverse could be true. To make a human analogy, rewards are somewhat like pleasure (if high) and pain (if low), whereas values correspond to a more refined and farsighted judgment of how pleased or displeased we are that our environment is in a particular state.

For now, just think of it like the h(x) function in A* or the eval(...) function in adversarial search.....

# Reinforcement Learning: Basic Concepts

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

## Policy

A Policy is an algorithm which takes a current state and outputs an action: given where I am, what to do next?

Learning an optimal policy is the entire point of an RL system; an optimal policy is one which maximizes the cumulative reward over time.

Policies can be

- Deterministic (mapping from states to actions); or
- Stochastic (given a state, return a probability distribution over possible actions).

## Agent

The Agent in all this is simply the algorithm that organizes all this activity: It makes state transitions (possibly based on observations), consults the Policy, and takes actions, moving to the next state. It updates the Policy based on rewards and values.

# Reinforcement Learning: Basic Concepts

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

One more term: Model

A concept that may or may not play a role in a RL program is a model of the environment, which gives information (e.g., probabilities) about what will happen (reward and next state).

The model may be:

- Fixed for all time;
- Something learned over time; or
- Unknown.

"Model Based" – could use planning and/or search using the model to predict next states and rewards.

"Model Free" or "Trial-and-Error Learning" – Can not search, can only update policy based on current state and reward.

Models are generally of two types:

- **Distribution Model**: Given a state s and possible action a, give the probability p of receiving reward r and transitioning to state s'.
- **Sampling Model**: Given a state s and possible action a, give the reward r and next state s', based on simulation, search, or a record of past experience.

# Reinforcement Learning: Basic Concepts

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

Wait, wait, one more: How is the training organized?

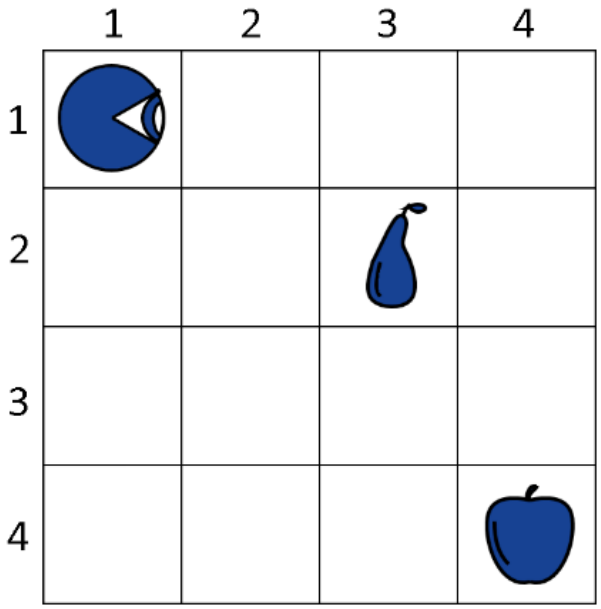
The possibilities are:

- Multiple trials with some resource limit (e.g., maximum number of time steps);
- No training: system self-organizes over time; or
- Some combination of the two.

# Reinforcement Learning: Simple Examples

## Example One

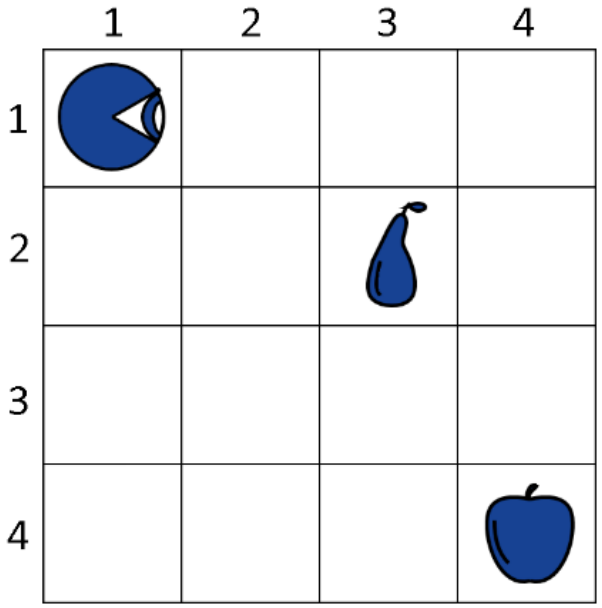
An agent is searching for food in a 4x4 grid laid out as follows:




- Environment: 4x4 grid as shown
- Agent: Hungry blob
- State: location of agent: (x,y)
- Action: move R,L,U,D
- Reward:
  - No fruit: -1 (goes hungry, uses energy)
  - Pear: 5
  - Apple: 10

# Reinforcement Learning: Simple Examples

At first, there is no policy other than making random moves.....

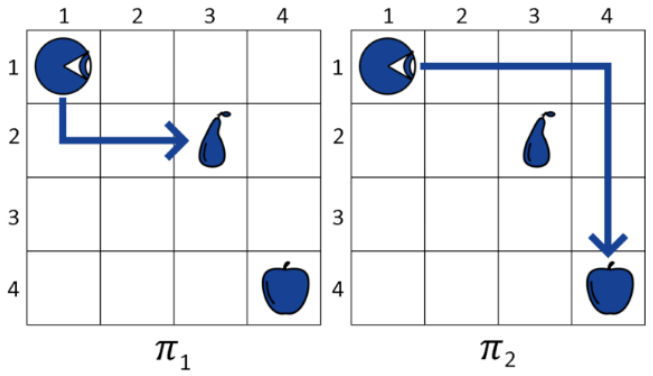



- Environment: 4x4 grid as shown
- Agent: Hungry blob
- State: location of agent: (x,y)
- Action: move R,L,U,D
- Reward:
  - No fruit: -1 (goes hungry, uses energy)
  - Pear: 5
  - Apple: 10

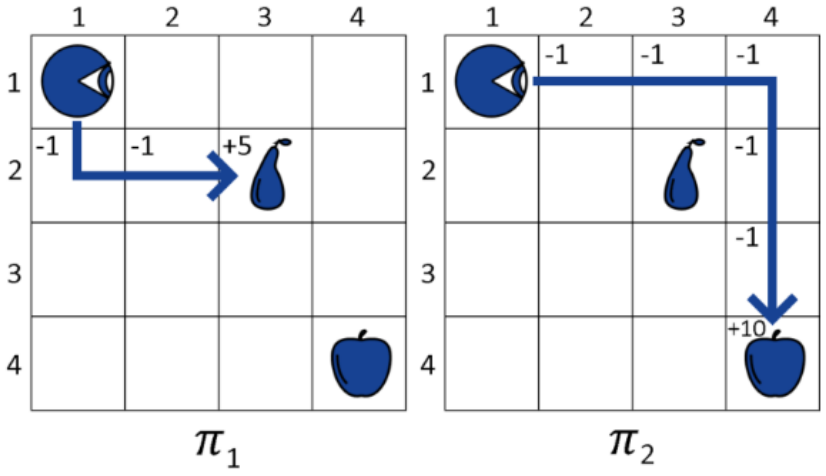
# Reinforcement Learning: Simple Examples

But after many trials, the agent compares policies, and selects the best one. Here are the two best policies:

1. $\pi_1$ = down, right, right → Pear
2. $\pi_2$ = right, right, right, down, down, down → Apple




- $U(\pi_1) = -1 - 1 + 5 = +3$
- $U(\pi_2) = -1 - 1 - 1 - 1 - 1 + 10 = +5$



# Reinforcement Learning: Simple Examples

We can add other components to speed up the learning....

1. A value function calculates the Euclidean distance $D_{pear}$ and $D_{apple}$ from the current state to each piece of fruit, and returns:

$$\frac{5 - D_{pear} + 10 - D_{apple}}{2}$$

2. The agent can make observations of a neighboring cell to determine whether it contains fruit at a cost of -0.25 per observation.

3. The agent learns a model of the environment by recording the locations of the pieces of fruit, and can use that information to plan a more direct route next time.

# Reinforcement Learning: Simple Examples

We can add other components to speed up the learning....

1. A value function calculates the Euclidean distance $D_{pear}$ and $D_{apple}$ from the current state to each piece of fruit, and returns:

$$\frac{5 - D_{pear} + 10 - D_{apple}}{2}$$

2. The agent can make observations of a neighboring cell to determine whether it contains fruit at a cost of -0.25 per observation. The agent is trading off search against following the existing policy.

3. The agent learns a model of the environment by recording the locations of the pieces of fruit, and can use that information to plan a more direct route next time.

# Reinforcement Learning: Simple Examples

Example Two:


The Multi-Armed Bandit is a generalization of the "one-armed bandits' that exist in all casinos.

The MAB problem is defined by a set of K random variables, each of which gives a reward based on some probability distribution:

$$MAB = ( R_1, R_2, ..., R_k)$$

And action is to choose an $i \in [1,2, \ldots, k]$ and "pull the level" on the random variable to receive an award. You typically have a budget of N pulls of a level. You may have no knowledge of the distributions, or partial knownledge (e.g., you may know that the RVs are distributed using a Bernoulli distribution with $1 result with probability p and $0 with probability 1-p).

This simple scenario can model many complex decision-making situations, such as resource allocation among research departments, clinical trials, or financial portfolio allocation.

# Reinforcement Learning: Simple Examples

This simple problem highlights an important distinction for RL algorithms:


How much should the agent exploit what knowledge it already has about the levers it has already pulled? (Sometimes called the greedy strategy.)

VS

How much should the agent explore new levers, with possible worst or better results?

Note that this is not a mutully-exclusive choice: your knowledge of levers already pulled depends on how many times: do you go with a well-known lever, or try one that you have not tried much, or a completely known lever?

Environment: ( $R_1$, $R_2$, ..., $R_k$)
State: How many times you pulled each lever and the results
Action: i
Reward: $R_i$
Value: ??
Model: Estimates of distributions $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_k$
Policy: Given the state, decide to what extend you pursue the greedy and exploratory actions to pull the next lever i.

# Reinforcement Learning: Simple Examples

Problem Three:

Board Games (e.g., TicTacToe or Connect4)

Note a significant problem with MinMax algorithms:

They assume that both players are following an identical strategy and have an identical level of expertise.

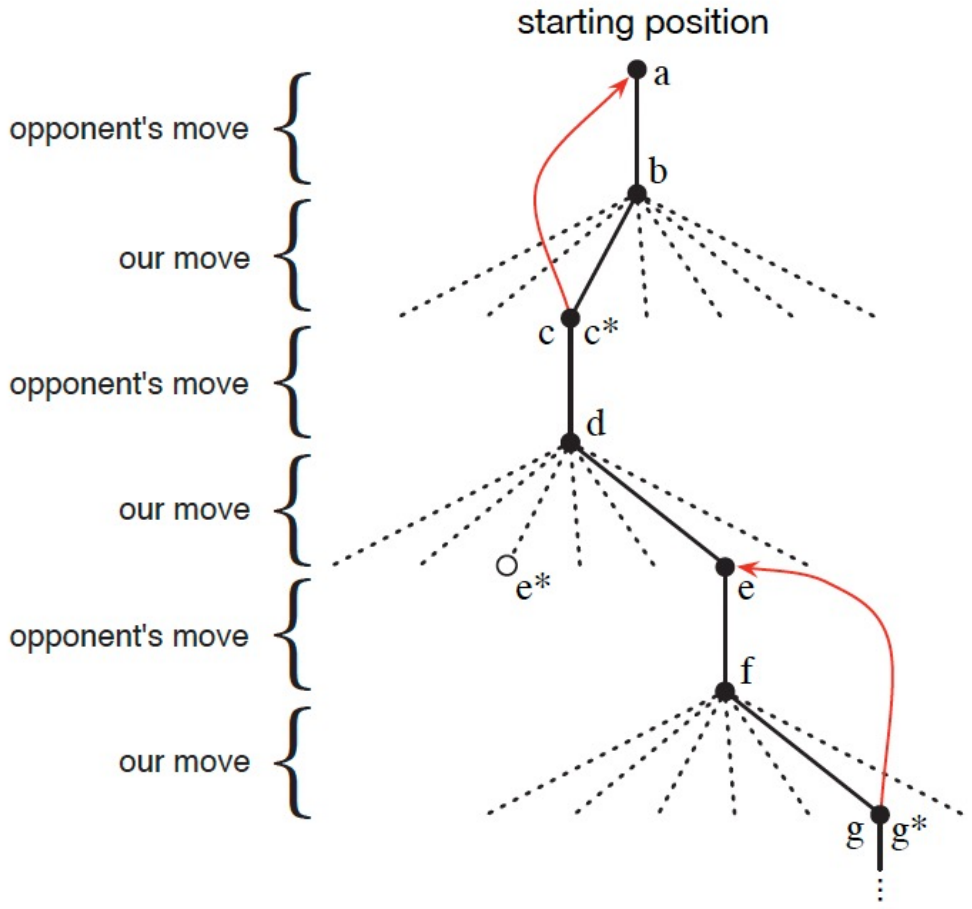
What if this is not the case? How can you exploit the weaknesses you notice in an opponent's strategy if you play them over and over?

Example: Humans do not see diagonal wins in Connect 4 as well as horizontal and vertical wins. Could you design a Connect 4 program to learn to exploit this weakness after playing many games against human opponents?

[AlphaGo's designers used RL extensively to train the system, and an essential part of GPT-3 and GPT-4 (just released!) training used human interactions.]

# Reinforcement Learning: Simple Examples

RL for Board Games

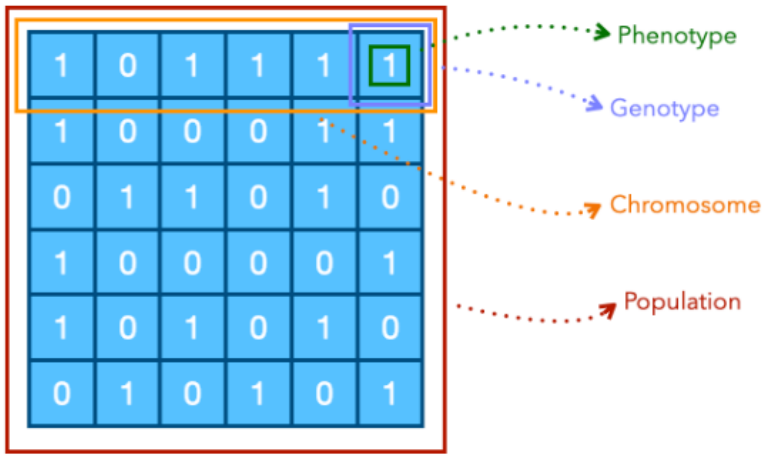


# Genetic Algorithms as RL

1. Environment
2. State
3. Action
4. Reward
5. Value
6. Policy
7. Agent
8. Model

Genetic Algorithms are a special subclass of RL algorithms with the following characteristics:

No model
No Value function
Multiple agents
Environment = a collection of agents
Policy = array of numbers



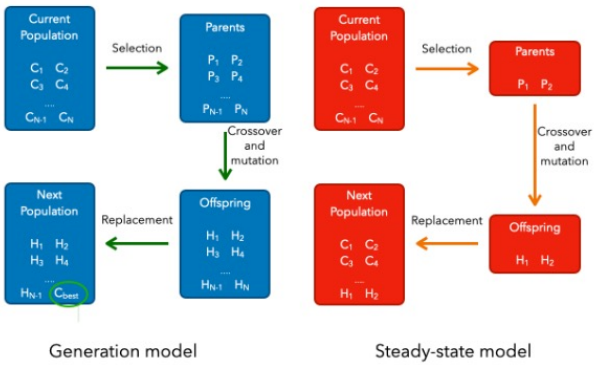
# Genetic Algorithms as RL

Framework for learning:

Multiple agents (arrays of numbers) compete with each other, and receive immediate rewards.

On the basis of the rewards, the agents can

- Die
- Produce offspring through:
  - Mutation: Make small (usually random) changes in the numbers
  - Crossover: Breed with other successful agents



# Prisoner's Dilemma

To explore genetic algorithms, we'll use a simple but powerful game called the Prisoner's Dilemma....

## Rules of the Game:

- There are two players, A and B;
- A move is C (cooperate/help) or D (defect/trick);
- Each player chooses a move without knowing the other player's move in advance;
- There are four possible outcomes (C & C, C & D, D & C, D & D);
- Rewards and punishments are determined as follows….

# Prisoner's Dilemma

| Listed are MY rewards or fines; you have the same! | | What You do |
|---|---|---|
| | | Cooperate (Help) | Defect (Trick) |
| What I do | Cooperate | Pretty Good! Reward for mutual cooperation: **$300** | Horrible! Fine for being a sucker: **-$100** |
| | Defect | Excellent! Reward for fooling the sucker: **$500** | Pretty bad! Punishment for mutual defection: **-$10** |

# Prisoner's Dilemma

Why this has been studied for many decades!

A RATIONAL player trying to maximize his own profit never cooperates!

Why?

- If your partner Cooperates, your best payoff is with Defect;
- If your partner Defects, your least fine is with Defect;

Ergo…. No matter what your partner does, you should defect. Defect is the only rational strategy!

| | | What I do: Cooperate | What I do: Defect |
|---|---|---|---|
| What you do | Cooperate | Fairly Good : Reward for mutual cooperation: $300 | Very bad! Fine for being a sucker: -$100 |
| | Defect | Very good! Reward for fooling the sucker: $500 | Fairly bad! Punishment for mutual defection: -$10 |

So don't blame the hand raisers, they're just being rational……

# Prisoner's Dilemma

Why this has been studied for many decades!

The Prisoner's Dilemma is a familiar paradigm in modern life:

- Should nations build up their store of weapons or agree to disarm?
- Should nations curb their emission of greenhouse gasses?
- Should athletes take steroids or performance-enhancing drugs?
- Should you cheat on your schoolwork?
- Should organisms share food and mates?

# Prisoner's Dilemma

## Iterated Prisoner's Dilemma

But altruism exists widely at all levels (from genes to societies)! How can this be? Maybe our model is not complex enough!

- We (and our genes) live in communities, and such games are rarely played just once and for all, but are repeated over and over;
- If you repeat the game, you can CHANGE your choices based on what the other players did in the past; you can have a more complex strategy!

Iterated Prisoner's Dilemma = Play some number of rounds and total your winnings/losses at the end.

# Prisoner's Dilemma

## Strategies for the IPD

A STRATEGY is now

- What to do the first time;
- What to do given the history of past rounds in this one game.

# Prisoner's Dilemma