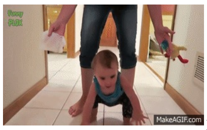
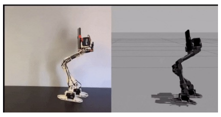
Artificial Intelligence

Module 7: Reinforcement Learning (PART-I)

Learning from Experience : Learning in human way





Dr. Chandra Prakash

Assistant Professor

Department of Computer Science and Engineering

(Slides adapted from StuartJ. Russell, B Ravindran, Mausam, Dan Klein and Pieter Abbeel, Partha P Chakrabarti, Saikishor Jangiti

Module 7: Reinforcement Learning

- PART 7.1 : Learning Agent
- PART 7.2 : Introduction to Machine Learning
- PART 7.3 : Types of Machine Learning
- PART 7.4 : Learning from experience :
  - Reinforcement Learning
  - Background
- PART 7.5 : Model based and Model free learning
- PART 7.6 : TD and Q Learning
- PART 7.7 : RL Applications

Agent Architecture Recap

- Simple Reflex Agent
- Model based Agent
- Goal based Agent
- Utility based Agent
- Learning based Agent

- An agent is **learning** if it improves its performance on future tasks after making observations about the world

- Learning happens through observations, i.e., Experience or data, etc.

What is Learning?

- Learning is an important area in AI, perhaps more so than planning.
  - Problems are hard -- harder than planning.
  - Recognised Solutions are not as common as planning.
  - A goal of AI is to enable computers that can be taught rather than programmed.
- Learning is an area of AI that focusses on processes of self-improvement.
- Information processes that improve their performance or enlarge their knowledge bases are said to learn.
- Why is it hard?
  - Intelligence implies that an organism / machine must be able to adapt to new situations.
  - It must be able to learn to do new things.
  - This requires knowledge acquisition, inference, updating/refinement of knowledge base, acquisition of heuristics, applying faster searches, etc.

How can we learn?

- Skill refinement
  - one can learn by practicing, e.g playing the piano.
- Knowledge acquisition
  - one can learn by experience and by storing the experience in a knowledge base.
  - One basic example of this type is rote learning (process of memorizing information based on repetition).
- Problem Solving
  - if we solve a problem one may learn from this experience. The next time we see a similar problem we can solve it more efficiently.
  - not usually involve gathering new knowledge but may involve reorganisation of data or remembering how to achieve to solution.
- Taking advice
  - Similar to rote learning although the knowledge that is input may need to be transformed (or operationalized) in order to be used effectively.
- Induction
  - One can learn from examples. Humans often classify things in the world without knowing explicit rules. Usually involves a teacher or trainer to aid the classification.
- Discovery
  - Here one learns knowledge without the aid of a teacher.
- Analogy
  - If a system can recognise similarities in information already stored then it may be able to transfer some knowledge to improve to solution of the task in hand.

Learning Agents

Why Learning ???

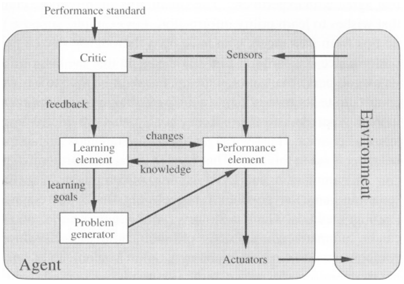
- The agent designer cannot anticipate all possible world states the agent need to handle and code them in the agent environment
  - E.g., A robot designed to navigate mazes need to learn the layout of new maze whenever encountered

- The agent designer cannot anticipate all changes over time
  - E.g., Stock market prediction

- Sometimes the agent designer have no idea how to program a solution
  - E.g., Stock market prediction

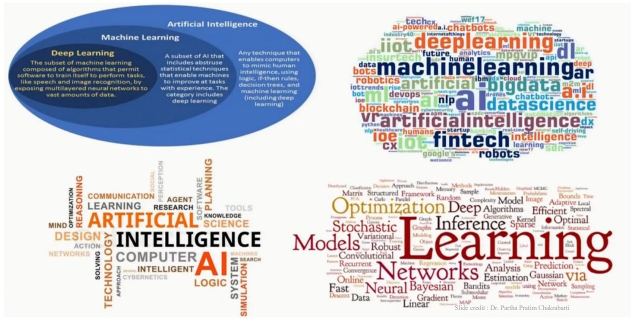
- Four conceptual components
  - Learning element
    - Making improvement
  - Performance element
    - Selecting external actions
  - Critic
    - Tells the Learning element how well the agent is doing with respect to fixed performance standard.
    - (Feedback from user or examples, good or not?)
  - Problem generator
    - Suggest actions that will lead to new and informative experiences.

Learning Agents



7

AI- ML Terminology



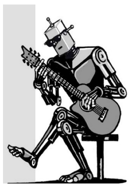
Introduction

Machine learning: Definition

- A scientific discipline that is concerned with the design and development of algorithms that allow computers to learn based on data, such as from sensor data or databases, etc.

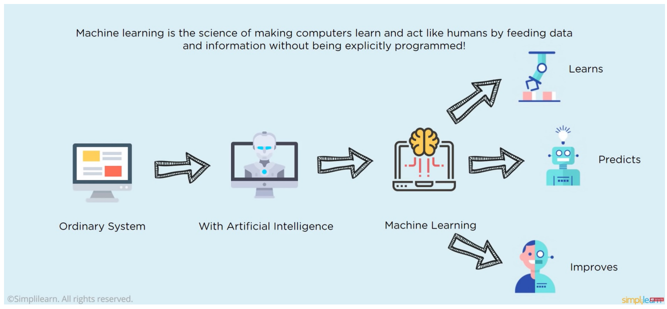
Major focus of machine learning research

- ✓ To automatically learn to recognize complex patterns and make intelligent decisions based on data .



9

What is Machine learning:

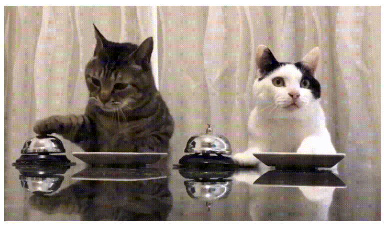


Can You Recognize these Pictures ?

Image or a sculpture ???

- If Yes, How do you Recognize it?

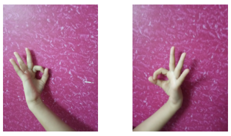
Learning



Learning is constructing or modifying representations of what is being experienced. -- McCarthy, 1968

Learning

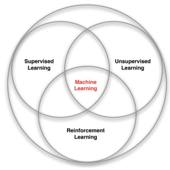
बायां हाथ   दायाँ हाथ   ????



Machine learning Type:

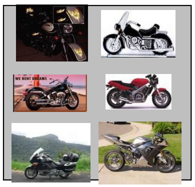
With respect to the feedback type to learner:

- **Supervised learning** :
  - Task Driven (Classification)
- **Unsupervised learning** :
  - Data Driven (Clustering)
- **Reinforcement learning**
  - Self learning (reward based)



Supervised learning

Cars



Motorcycles

Testing:
What is this?

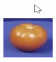
- Linear regression
- Logistic regression
- Perceptron
- Naive Bayes
- Neural Networks
- Decision trees; K-Nearest Neighbor
- Support Vector Machine (SVM)

Supervised learning

- Apply a prediction function to a feature representation of the image to get the desired output:

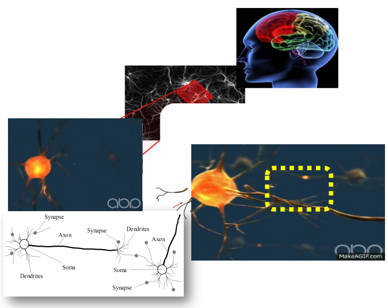
Training Images

$f($  $)$ = “apple”

$f($  $)$ = “tomato”

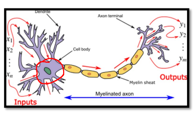
$f($  $)$ = “cow”

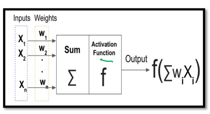
Slide credit: L. Lazebnik



Artificial Neural Network

- Consists of a number of very simple processors, also called **neurons**,
  - Analogous to the biological neurons in the brain.
- Neurons are connected by weighted links passing signals from one neuron to another.
- The output signal is transmitted through the neuron’s outgoing connection.
- The outgoing connection splits into a number of branches that transmit the same signal.
  - The outgoing branches terminate at the incoming connections of other neurons in the network.

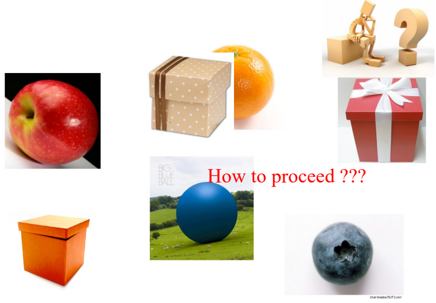




Source: Wikipedia

Real time problem –

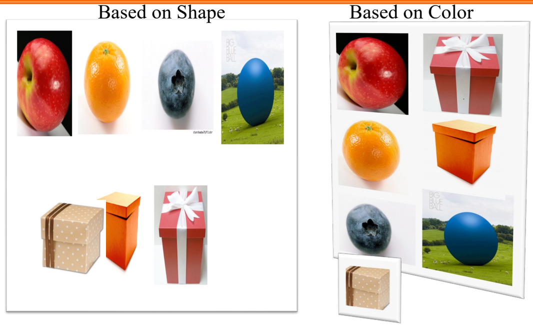
DATA available/unavailable – not what to do ??



How to proceed ???

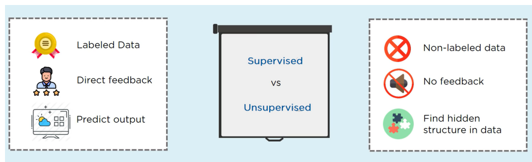
Based on Shape

Based on Color

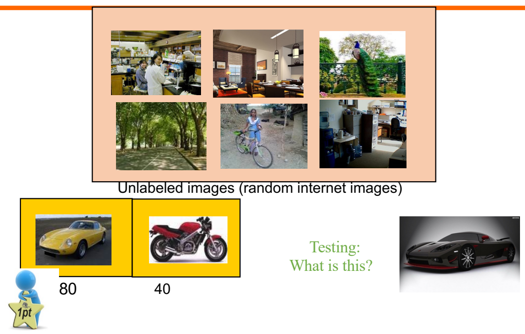


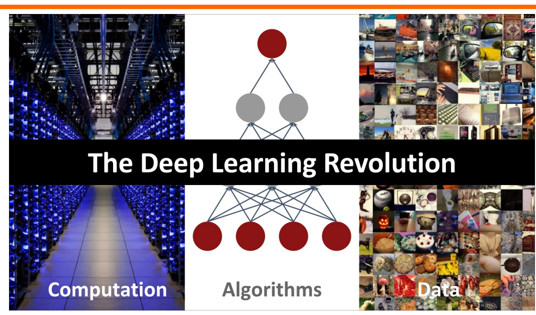
Un-Supervised learning

Supervised VS Unsupervised



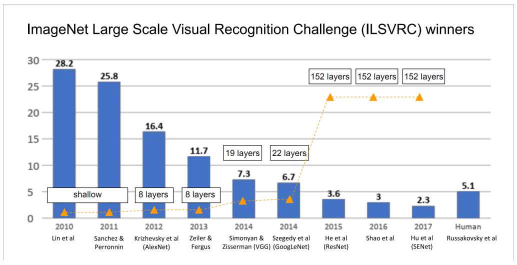
Reinforcement/Self Learning



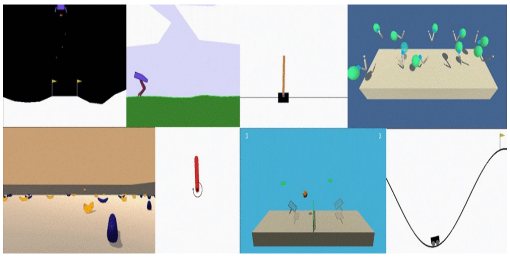


DATA

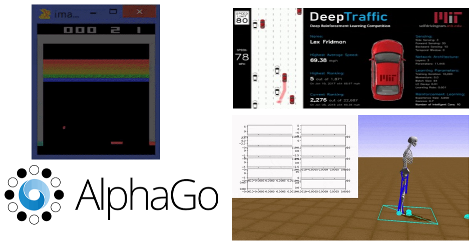
Vision and Deep Learning



Reinforcement Learning Examples



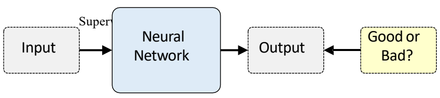
Deep Reinforcement Learning



- ATARI 2600
- Alpha Go
- Mnih, V. (2013). Playing atari with deep reinforcement learning Silver,
- D. (2016). Mastering the game of Go with deep neural networks and tree search
- Learning to Run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments

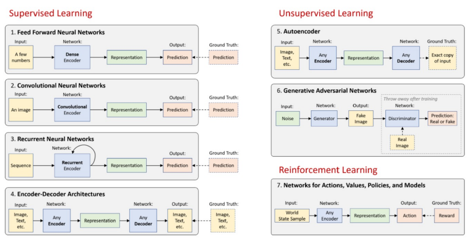
Supervised Learning

- It's all "supervised" by a loss function!

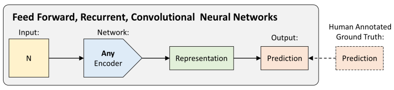


- **Someone has to say what's good and what's bad*

- At a high-level, neural networks are either encoders, decoders, or a combination of both:
  - **Encoders** find patterns in raw data to form compact, useful representations.
  - **Decoders** generate high-resolution data from those representations. The generated data is either new examples or descriptive knowledge.

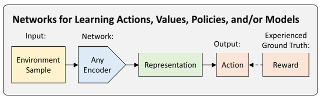


Deep learning - representation learning: the automated formation of useful representations from data.



- **Supervised learning** is "teach by example":
  Here's some examples, now learn patterns in these example.

- **Reinforcement learning** is "teach by experience":
  Here's a world, now learn patterns by exploring it.



Reinforcement

Dictionary meaning

Occurrence of an event, in the proper relation to a response, that tends to increase the probability that the response will occur again in the same situation.

Reinforcement Learning (RL)

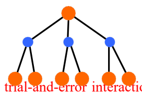
"a way of programming agents by reward and punishment without needing to specify how the task is to be achieved"

[Kaelbling, Littman,& Moore, 96]

- Imagine playing a new game whose rules we don't know; after a hundred or so moves, the referee tells that you lose.
- From the point of view as designers of AI systems
  - Providing a reward signal to the agent is usually much easier than providing labeled examples of how to behave.
  - Reward function is often very concise and easy to specify
  - We don't have to be experts, capable of supplying the correct action in any situation.

Reinforcement Learning (Cont..)

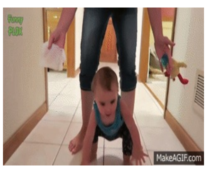
- Emphasizes learning **feedback** that evaluates the learner's performance **without providing standards of correctness** in the form of behavioral targets.
- Some researcher consider RL a form of unsupervised learning.
- An orthogonal approach for Learning Machine. :
- RL is training by
  - **rewards and punishments.**
  - **Good vs Bad**

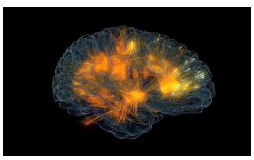


- RL agent learns by receiving a reward or reinforcement through trial-and-error interactions with a dynamic environment to achieve a goal, without any form of supervision other than its own decision-making policy.
- Reinforcement Learning is learning how to act in order to maximize a numerical reward.

Reinforcement Learning in Humans

- Human appear to learn to walk through "very few examples" of trial and error. **How** is an open question…
- Possible answers:
  - **Hardware:** 230 million years of bipedal movement data.
  - **Imitation Learning:** Observation of other humans walking.
  - **Algorithms:** Better than backpropagation and stochastic gradient descent

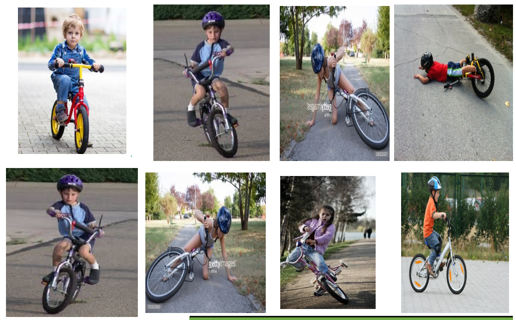




Reinforcement Learning (RL)

- Close to Human Learning.
- Agent learns a policy of how to act in a given environment.
- Every action has some impact in the environment, and the environment provides rewards that guides the learning algorithm

Study Time as a Self Learning Model

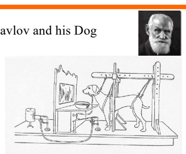


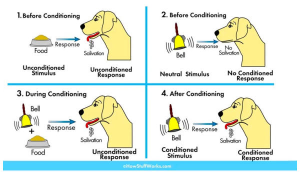
Bicycle Learning Experience Matrix

	Left	Right	Straight
Left	2	4	8
Right	3	1	7
Straight	6	11	50

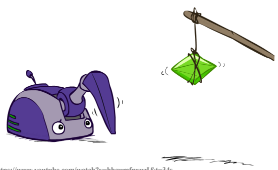
More Example (Experiment by Pavlov)

Pavlov and his Dog





- Early Results:
  - Classical (Pavlovian) conditioning experiments
  - Training: Bell→Food
  - After: Bell → Salivate
  - Conditioned stimulus (bell) predicts future reward (food)



Source : https://www.youtube.com/watch?v=hhqumfpxuzI&t=34s

RL and Animal Foraging

- RL studied experimentally for more than 80 years in psychology and brain science
  - Rewards: food, pain, hunger, drugs, etc.
  - Evidence for RL in the brain via a chemical called dopamine
- Example: foraging
  - Bees can learn near-optimal foraging policy in field of artificial flowers with controlled nectar supplies

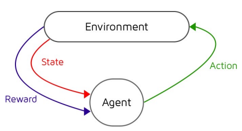
Reinforcement Learning Framework

At each step, the agent:

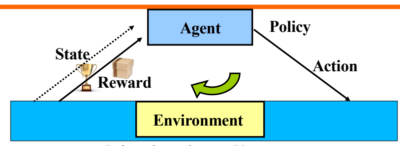
- Executes action
- Observe new state
- Receive reward
  - intrinsic rewards - food
  - Extrinsic rewards- money

Open Questions:

- What cannot be modeled in this way?
- What are the challenges of learning in this framework?

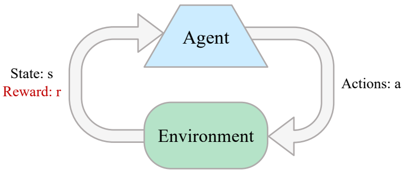


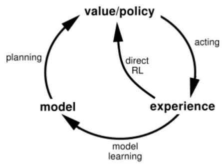
Element of Reinforcement Learning



- **Agent:** *Intelligent programs*
- **Environment:** *External condition*
- **Policy:**
  - *Agent's behavior at a given time*
  - *A mapping from states to actions*
    - *Lookup tables or simple function*
- **Reward function** :
  - *Defines the goal in an RL problem*
  - Policy is altered to achieve this goal
  - Feedback that measures the sucess or failure of the agent's action
- **Value function**:
  - *Specifies what is good in the long run while **Reward** function indicates what is good in an immediate sense.*
  - Value of a state - Total amount of reward an agent can expect to accumulate over the future, starting form that state.
- **Model of the environment** :
  - *Used for planning & if Know current state and action then predict the resultant next state and next reward.*

Reinforcement Learning





- Basic idea:
  - Receive feedback in the form of rewards
  - Agent's utility is defined by the reward function
  - Must (learn to) act so as to maximize expected rewards
  - All learning is based on observed samples of outcomes!

Example: Learning to Walk


Initial


A Learning Trial


After Learning [1K Trials]

[Kohl and Stone, ICRA 2004]

Environment and Actions

- **Fully Observable** (Chess) vs **Partially Observable** (Poker)
- **Single Agent** (Atari) vs **Multi Agent** (DeepTraffic)
- **Deterministic** (Cart Pole) vs **Stochastic** (DeepTraffic)
  - Deterministic system - no randomness is involved in the development of future states of the system.
  - Stochastic system - random probability distribution or pattern that may be analysed statistically but may not be predicted precisely.
- **Static** (Chess) vs **Dynamic** (DeepTraffic)
- **Discrete** (Chess) vs **Continuous** (Cart Pole)

Note: Real-world environment might not technically be stochastic or partially-observable but might as well be treated as such due to their complexity.

Learning vs Inference

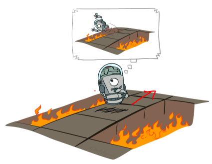
- Batch setting in Bayes Nets
  - Data → Model → Prediction
- Active setting in MDPs
  - Action → Data → (Model ?)
- Actions have two purposes
  - To maximize reward
  - To learn the model

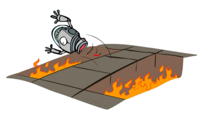
Reinforcement Learning Vs MDP

- Still assume a Markov decision process (MDP):
  - A set of states $s \in S$
  - A set of actions (per state) A
  - A model T(s,a,s')
  - A reward function R(s,a,s')
- Still looking for a policy $\pi(s)$

- New twist: don't know T or R
  - I.e. we don't know which states are good or what the actions do
  - Must actually try actions and states out to learn

Offline (MDPs) vs. Online (RL)


Offline Solution


Online Learning

Major Components of an RL Agent

An RL agent may be directly or indirectly trying to learn a:

- **Policy:** agent's behavior function
- **Value function:** how good is each state and/or action
- **Model:** agent's representation of the environment

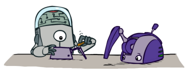
$$s_0, a_0, r_1, s_1, a_1, r_2, \ldots\ldots, s_{n-1}, a_{n-1}, r_n, s_n$$

state
action
reward
Terminal state

Main dimension in RL

Model-based vs. Model-free

- Model-based: learn the model (T, R)
- Model-free: directly learn what action to do when



Passive vs. Active

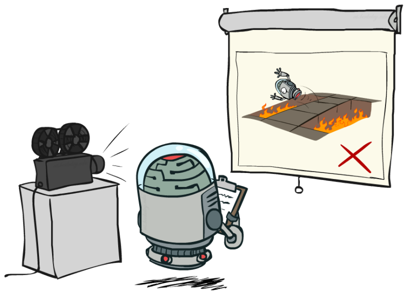
- Passive: learn state values evaluating a given policy
- Active: need to learn both optimal policy + state values

Dynamic programming
Model-based
Monte Carlo methods
No Model
Temporal-difference learning.

Strong vs Weak simulator

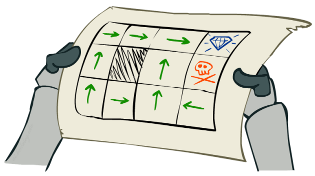
- Strong: can jump to any part of state space and simulate
- Weak: real world; can't teleport

Passive Reinforcement Learning



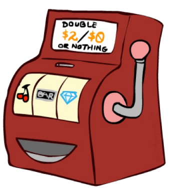
Passive Reinforcement Learning

- Simplified task: policy evaluation
  - Input: a fixed policy $\pi(s)$
  - You don't know the transitions T(s,a,s')
  - You don't know the rewards R(s,a,s')
  - Goal: learn the state values
- In this case:
  - Learner is "along for the ride"
  - No choice about what actions to take
  - Just execute the policy and learn from experience
  - This is NOT offline planning! You actually take actions in the world.

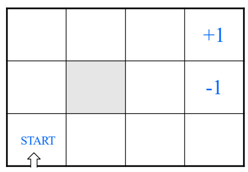


Passive Learning : Direct Evaluation

- Goal: Compute values for each state under π
- Idea: Average together observed sample values
  - Act according to π
  - Every time you visit a state, write down what the sum of discounted rewards turned out to be
  - Average those samples
- This is called direct evaluation



Polcity Selection for Robot in a Room

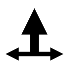


actions: UP, DOWN, LEFT, RIGHT

(Stochastic) model of the world:

Action: **UP**

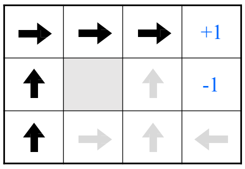
80% move UP
10% move LEFT
10% move RIGHT



- Reward +1 at [4,3], -1 at [4,2]
- Reward -0.04 for each step
- What's the strategy to achieve max reward?
  - We can learn the model and plan
  - We can learn the value of (action, state) pairs and act greed/non-greedy
  - We can learn the policy directly while sampling from it

Optimal Policy for a Deterministic World

Reward: **-0.04** for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are deterministic:

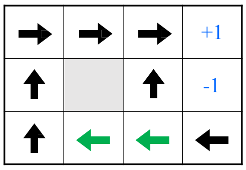
UP

100%	move UP
0%	move LEFT
0%	move RIGHT

Policy: Shortest path.

Optimal Policy for a Stochastic World

Reward: **-0.04** for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are stochastic:

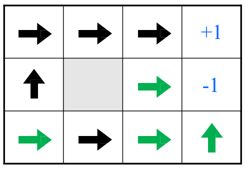
UP

80%	move UP
10%	move LEFT
10%	move RIGHT

Policy: Shortest path. Avoid -UP around -1 square.

Optimal Policy for a Stochastic World

Reward: **-2** for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are stochastic:

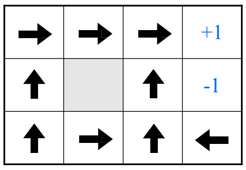
UP

80%	move UP
10%	move LEFT
10%	move RIGHT

Policy: Shortest path.

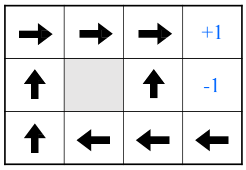
Optimal Policy for a Stochastic World

Reward: **-0.1** for each step



More urgent

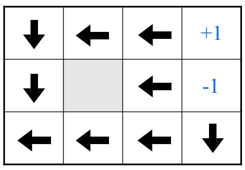
Reward: **-0.04** for each step



Less urgent

Optimal Policy for a Stochastic World

Reward: **+0.01** for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are stochastic:

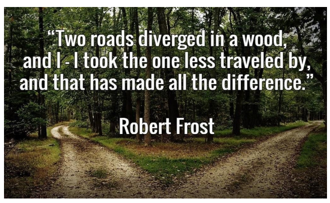
UP

80%	move UP
10%	move LEFT
10%	move RIGHT

Policy: Longest path.

Lessons from Robot in Room

- Environment model has big impact on optimal policy
- Reward structure has big impact on optimal policy



Reinforcement Learning (Cont..)

- Concept used in Reinforcement Learning
  - **Evaluative Vs. Instructive Feedback**
  - **Associative Vs. Non-Associative**
  - **Exploration and exploitation**

Action Selection Method

Exploration and exploitation

A. **Greedy action:** Action chosen with greatest estimated value.

Greedy action: a case of **Exploitation**.

- Ɛ -greedy
  - Most of the time the greediest action is chosen
  - Every once in a while, with a small probability **Ɛ**, an action is selected at random.

B. **Non-Greedy** action: a case of **Exploration**, as it enables us to improve estimate the non-greedy action's value.

- **ε -soft** - The best action is selected with probability (1 –ε) and the rest of the time a random action is chosen uniformly.

Ɛ-Greedy Action Selection Method :

Let the a* is the greedy action at time t and $Q_t(a)$ is the value of action a at time.

Greedy Action Selection:

$$a_t = a_t^* = \arg\max_a Q_t(a)$$

- Ɛ –greedy

$$a_t = \begin{cases} a_t & \text{with probability } 1-\epsilon \\ \text{random action with probability } \epsilon \end{cases}$$

Exploration vs Exploitation

- Deterministic/greedy policy won't explore all actions
  - Don't know anything about the environment at the beginning
  - Need to try all actions to find the optimal one
- ε-greedy policy
  - Every once in a while, with a small probability Ɛ, an action is selected at random.
  - **ε -soft** : With probability 1-ε perform the optimal/greedy action, otherwise random action
  - Slowly move it towards greedy policy: ε -> 0

Action Selection Policies (Cont…)

- **Softmax** –
  - Drawback of ε -greedy & ε -soft: Select **random actions uniformly**.
  - Softmax remedies this by:
    - Assigning a **weight with each actions**, according to their action-value estimate.
    - A random action is selected with regards to the weight associated with each action
    - The worst actions are unlikely to be chosen.
    - This is a good approach to take where the worst actions are very unfavorable.

Softmax Action Selection(Cont…)

- Problem with ε-greedy: Neglects action values
- Softmax idea: grade action probs. by estimated values.
- Gibbs, or Boltzmann action selection, or exponential weights:

$$\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}}$$

τ is the "computational temperature"

At τ → 0 the Softmax action selection method become the same as greedy action selection.

Some terms in Reinforcement Learning

- **The Agent Learns a Policy:**
  - Policy at step t, π_t : a mapping from states to action probabilities will be:

$$\pi_t(s, a) = \text{ probability that } a_t = a \text{ when } s_t = s$$

  - Agents changes their policy with Experience.
  - Objective: get as much reward as possible over a long run.
- **Goals and Rewards**
  - A goal should specify what we want to achieve, not how we want to achieve it.

Meaning of Life for RL Agent: Maximize Reward

- Future reward: $R_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_n$
- Discounted future reward:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-t} r_n$$

- A good strategy for an agent would be to always choose an action that maximizes the (discounted) future reward
- Why "discounted"?
  - Math trick to help analyze convergence
  - Uncertainty due to environment stochasticity, partial observability, or that life can end at any moment:

"If today were the last day of my life, would I want to do what I'm about to do today?" – Steve Jobs

Some terms in RL (Cont…)

- **Returns**
  - Rewards in long term
  - **Episodes**: Subsequence of interaction between agent-environment e.g., plays of a game, trips through a maze.
- **Discount return**
  - The geometrically discounted model of return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

where γ, $0 \le \gamma \le 1$, is the **discount rate**

  - Used to:
    - to make future rewards less important than immediate rewards
    - enforce short time learning in agent
    - To determine the present value of the future rewards
    - Give more weight to earlier rewards

UPDATE Rule

- **Common update rule form**:

  NewEstimate = OldEstimate + StepSize[Target –OldEstimate]

- The expression [Target - Old Estimate] is an error in the estimate.
- It is reduce by taking a step toward the target.
- In proceeding the (t+1)st reward for action a the step-size parameter will be 1\(t+1).

Value function

- States-action pairs function that estimate how good it is for the agent to be in a given state
- Type of value function
  - **State-Value function**
    - The **value of a state** is the expected return starting from that state; depends on the agent's policy:

State-value function for policy π :

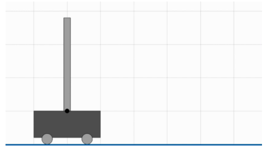
$$V^\pi(s) = E_\pi\{R_t \mid s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \;\middle|\; s_t = s\right\}$$

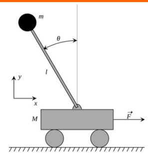
  - **Action-Value function**
    - The **value of taking an action in a state under policy π** is the expected return starting from that state, taking that action, and thereafter following π :

Action-value function for policy π :

$$Q^\pi(s,a) = E_\pi\{R_t \mid s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \;\middle|\; s_t = s, a_t = a\right\}$$

Examples of Reinforcement Learning



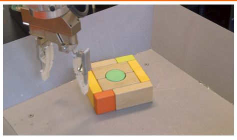
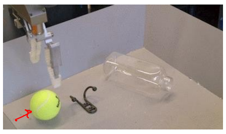


Identify :- G S A R ???

Cart-Pole Balancing

- **Goal** —Balance the pole on top of a moving cart
- **State** —Pole angle, angular speed. Cart position, horizontal velocity.
- **Actions** —horizontal force to the cart
- **Reward** —1 at each time step if the pole is upright

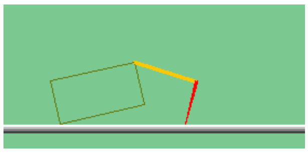
Examples of Reinforcement Learning

Grasping Objects with Robotic Arm

- **Goal** - Pick an object of different shapes
- **State** - Raw pixels from camera
- **Actions** – Move arm. Grasp.
- **Reward** - Positive when pickup is successful

The Crawler!

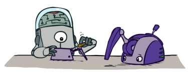


[Demo: Crawler Bot (L10D1)] [You, in Project 3]

Video of Demo Crawler Bot

Next Time

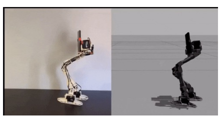
3 Types of Reinforcement Learning



Artificial Intelligence

Module 7: Reinforcement Learning (PART-II)

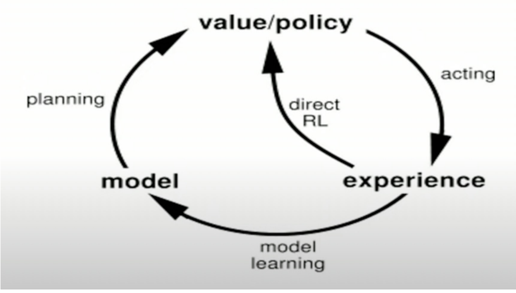
Learning from Experience : Learning in human way



Module 7: Reinforcement Learning

- PART 7.1 : Learning Agent
- PART 7.2 : Introduction to Machine Learning
- PART 7.3 : Types of Machine Learning
- PART 7.4 : Learning from experience :
  - Reinforcement Learning
  - Background
- PART 7.5 : Model based and Model free learning
- PART 7.6 : TD and Q Learning
- PART 7.7 : RL Applications

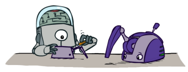
Learning/Planning / Acting



3 Types of Reinforcement Learning

Model-based

- Learn the model of the world, then plan using the model
- Update model often
- Re-plan often



Value-based

- Learn the state or state-action value
- Act by choosing best action in state
- Exploration is a necessary add-on

Policy-based

- Learn the stochastic policy function that maps state to action
- Act by sampling policy
- Exploration is baked in

Example: Expected Age

Goal: Compute expected age of CS 210 students

Known P(A)

$$E[A] = \sum_a P(a) \cdot a \quad = 0.35 \times 20 + \ldots$$

Without P(A), instead collect samples [a_1, a_2, … a_N]

Unknown P(A): "Model Based"

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Why does this work?

Because eventually you learn the right model.

Unknown P(A): "Model Free"

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Why does this work?

Because samples appear with the right frequencies.

Reinforcement Learning Methods

- **Model-based**
  - Learn an empirical model
  - Solve for V^π using policy evaluation
    - assuming that the learned model is correct
  - **Learning the model**
    - maintain estimates of T(s,a,s')
    - maintain estimates of R(s,a,s')
  - Dynamic programming
- **Model-Free**
  - Monte Carlo methods
- **Hybrid**
  - Temporal-difference learning.

1.Dynamic programming

- Classical solution method
- Require a <u>complete and accurate model of the environment.</u>
- Popular method for Dynamic programming
  - **Policy Evaluation** : Iterative computation of the value function for a given policy (prediction Problem)
  - **Policy Improvement**: Computation of improved policy for a given value function.

$$V(s_t) \leftarrow E_\pi\{r_{t+1} + \gamma V(s_t)\}$$

NewEstimate = OldEstimate + StepSize[Target – OldEstimate]

84

Model-Based Learning

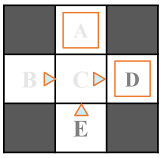
- Model-Based Idea:
  - Learn an approximate model based on experiences
  - Solve for values as if the learned model were correct

- Step 1: Learn empirical MDP model
  - Count outcomes s' for each s, a
  - Normalize to give an estimate of $\hat{T}(s,a,s')$
  - Discover each $\hat{R}(s,a,s')$ when we experience (s, a, s')

- Step 2: Solve the learned MDP
  - For example, use value iteration, as before

Example: Model-Based Learning

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1	Episode 2
B, east, C, -1 C, east, D, -1 D, exit, x,+10	B, east, C, -1 C, east, D, -1 D, exit, x,+10

Episode 3	Episode 4
E, north, C, -1 C, east, D, -1 D, exit, x, +10	E, north, C, -1 C, east, A, -1 A, exit, x, -10

Learned Model

$\widehat{T}(s,a,s')$

T(B, east, C) = 1.00
T(C, east, D) = 0.75
T(C, east, A) = 0.25
…

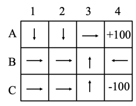
$\widehat{R}(s,a,s')$

R(B, east, C) = -1
R(C, east, D) = -1
R(D, exit, x) = +10
…

Exercise

Data on Executing π

(A1, D, -1)	(A1, D, -1)
(B1, R, -1)	(B1, R, -1)
(B2, R, -1)	(B2, R, -1)
(B3, U, -1)	(B3, U, -1)
(A3, R, -1)	(C3, U, -1)
(A2, D, -1)	(C4, -100)
(B2, R, -1)	
(B3, U, -1)	
(A3, R, -1)	
(A4, 100)	

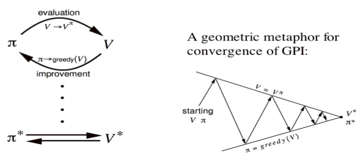


- T(A1, D, B1)
- T(B3, U, A3)

Generalized Policy Iteration (GPI)

Consist of two iteration process,

- **Policy Evaluation :** Making the value function consistent with the current policy
- **Policy Improvement:** Making the policy greedy with respect to the current value function



2. Monte Carlo Methods

- **Features of Monte Carlo Methods**
  - No need of Complete knowledge of environment
  - Based on averaging sample returns observed after visit to that state.
  - Experience is divided into **Episodes**
  - Only after completing an episode, value estimates and policies are changed.
  - Don't require a model
  - Not suited for step-by-step incremental computation

Model-Free Learning

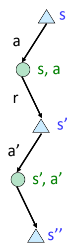
- Model-free (temporal difference) learning
  - Experience world through episodes

$$(s, a, r, s', a', r', s'', a'', r'', s''' \ldots)$$

  - Update estimates each transition

$$(s, a, r, s')$$

  - Over time, updates will mimic Bellman updates



MC and DP Methods

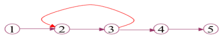
- Compute same value function
- Same step as in DP
  - Policy evaluation
  - Computation of state value (V_Π) and action value (Q_Π)for a fixed arbitrary policy (Π).
  - Policy Improvement
  - Generalized Policy Iteration

To find value of a State

- Estimate by experience, average the returns observed after visit to that state.
- More the return, more is the average converge to expected value

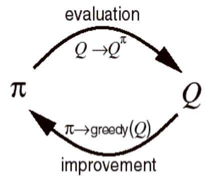
Monte Carlo Policy Evaluation

- *Goal:* learn $V^\pi(s)$
- *Given:* some number of episodes under π which contain s
- *Idea:* Average returns observed after visits to s



- *Every-Visit MC:* average returns for *every* time s is visited in an episode
- *First-visit MC:* average returns only for *first* time s is visited in an episode

Monte Carlo conrol

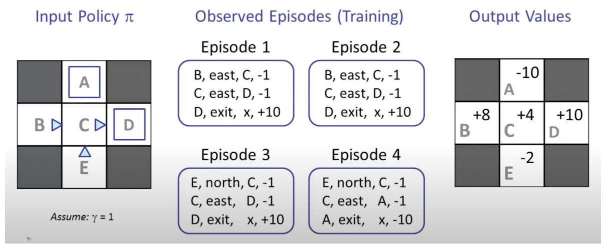


- MC policy iteration: Policy evaluation using MC methods followed by policy improvement
- Policy improvement step: greedify with respect to value (or action-value) function

Monte Carlo and Dynamic Programming

- MC has several advantage over DP:
  - Can learn from interaction with environment
  - No need of full models
  - No need to learn about ALL states
  - No bootstrapping
    - bootstrapping in RL means that you update a value based on some **estimates** and not on some *exact* values.

Direct Evaluation

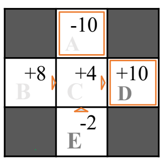


C= 9+9+9-11/4 = 4

Problems with Direct Evaluation

- What's good about direct evaluation?
  - It's easy to understand
  - It doesn't require any knowledge of T, R
  - It eventually computes the correct average values, using just sample transitions
- What bad about it?
  - It wastes information about state connections
  - Each state must be learned separately
  - So, it takes a long time to learn

Output Values



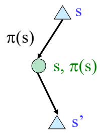
If B and E both go to C under this policy, how can their values be different?

3. Temporal Difference (TD) methods

- Learn from experience, like MC
  - Can learn directly from interaction with environment
  - No need for full models
- Estimate values based on estimated values of next states, like DP
- Bootstrapping (like DP)
- **Issue to watch for**:
  - maintaining sufficient exploration

Temporal Difference Learning

- Big idea: learn from every experience!
  - Update V(s) each time we experience a transition (s, a, s', r)
  - Likely outcomes s' will contribute updates more often
- Temporal difference learning of values
  - Policy still fixed, still doing evaluation!
  - Move values toward value of whatever successor occurs: running average



Sample of V(s): $sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

Temporal Difference (TD) Prediction

Policy Evaluation (the prediction problem):
for a given policy π, compute the state-value function V^{p}

Simple every - visit Monte Carlo method :

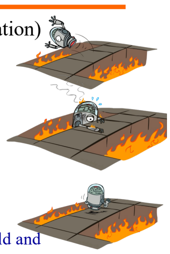
$$V(s_t) \leftarrow V(s_t) + \alpha\big[\underbrace{R_t}_{\text{target}} - V(s_t)\big]$$

The simplest TD method, TD(0) :

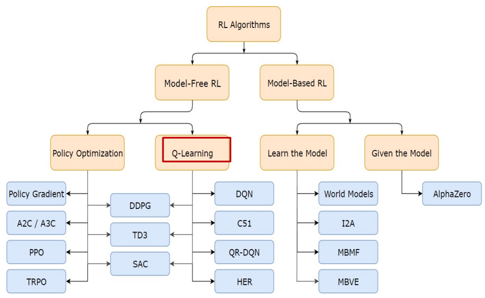
$$V(s_t) \leftarrow V(s_t) + \alpha\big[\underbrace{r_{t+1} + \gamma V(s_{t+1})}_{\text{target}} - V(s_t)\big]$$

Active Reinforcement Learning

- Full reinforcement learning: optimal policies (like value iteration)
  - You don't know the transitions T(s,a,s')
  - You don't know the rewards R(s,a,s')
  - You choose the actions now
  - Goal: learn the optimal policy / values
- In this case:
  - Learner makes choices!
  - Fundamental tradeoff: exploration vs. exploitation
  - This is NOT offline planning! You actually take actions in the world and find out what happens…

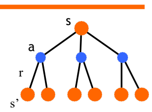


Taxonomy of RL Methods



Q-Learning

- State-action value function: $Q^{\pi}(s,a)$
  - Expected return when starting in *s*, performing *a*, and following π
- Q-Learning: Use **any policy** to estimate Q that maximizes future reward:
  - Q directly approximates Q* (Bellman optimality equation)
  - Independent of the policy being followed
  - Only requirement: keep updating each (s,a) pair



Learning Rate Discount Factor

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\left(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right)$$

New State Old State Reward

Q-Learning: Value Iteration

Learning Rate Discount Factor

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\left(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right)$$

New State Old State Reward

	A1	A2	A3	A4
S1	+1	+2	-1	0
S2	+2	0	+1	-2
S3	-1	+1	0	-2
S4	-2	0	+1	+1

```
initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
      select and carry out an action a
      observe reward r and new state s'
      Q[s,a] = Q[s,a] + α(r + γ max_a' Q[s',a'] - Q[s,a])
      s = s'
until terminated
```

Q-Learning: Off-Policy TD Control

One - step Q - learning :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

$$Q(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{old\ value} + \underbrace{\alpha_t(s_t,a_t)}_{learning\ rate} \times [\overbrace{\underbrace{r_t}_{reward} + \underbrace{\gamma}_{discount\ factor} \underbrace{\max_a Q(s_{t+1},a)}_{max\ future\ value}}^{expected\ discounted\ reward} - \overbrace{Q(s_t,a_t)}^{old\ value}]$$

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t)(1-\alpha_t(s_t,a_t)) + \alpha_t(s_t,a_t)[r_{t+1} + \gamma \max_a Q(s_{t+1},a)]$$

$$\boxed{Q(state, action) = R(state, action) + \gamma * Max[nextstate, allactions]}$$

> Initialize $Q(s,a)$ arbitrarily
> Repeat (for each episode):
>   Initialize s
>   Repeat (for each step of episode):
>   Choose a from s using policy derived from Q (e.g., ε-greedy)
>   Take action a, observe r, s'
> $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
> $s \leftarrow s'$;
>   until s is terminal

Sarsa: On-Policy TD Control

S A R S A: State Action Reward State Action

Turn this into a control method by always updating the policy to be greedy with respect to the current estimate:

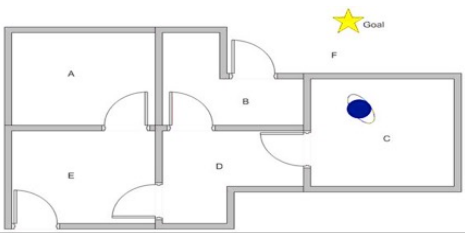
> Initialize $Q(s,a)$ arbitrarily
> Repeat (for each episode):
>   Initialize s
>   Choose a from s using policy derived from Q (e.g., ε-greedy)
>   Repeat (for each step of episode):
>   Take action a, observe r, s'
>   Choose a' from s' using policy derived from Q (e.g., ε-greedy)
> $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$
> $s \leftarrow s'$; $a \leftarrow a'$;
>   until s is terminal

Advantages of Temporal Difference (TD) Learning

- TD methods do not require a model of the environment, only experience
- TD, but not MC, methods can be fully incremental
  - You can learn before knowing the final outcome
    - Less memory
    - Less peak computation
  - You can learn without the final outcome
    - From incomplete sequences
- Both MC and TD converge

Example : PathFinder Bot using Reinforcement Learning

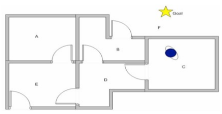
A Reinforcement Learning Example



Which path agent should choose???

A Reinforcement Learning Example

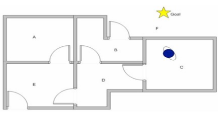
- Suppose we have 5 rooms A to E, in a building connected by certain doors :
- We can consider outside of the building as one big room say F to cover the building.
- There are two doors lead to the building from F, that is through room B and room E.
- Which path agent should choose???

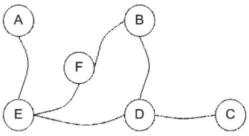


Solution using RL

Step 1: Modeling the environment-

- Represent the rooms by graph,
- Each room as a vertex (or node) and
- Each door as an edge (or link).
- Goal room is the node F

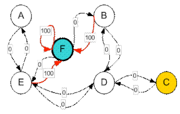




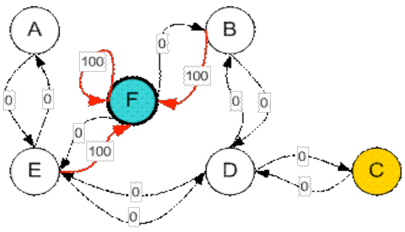
Step 1: Modelling the environment

- ✓ **Goal** –Outside the building – Node F
- ✓ Assign **Reward Value** to each room
- ✓ **State-** Each room (including outside building)
- ✓ **Action** – Agent's Movement from 1 room to next room
- ✓ **Initial state** – C (random)
- ✓ **Reward-** Goal Node - highest reward (100) rest – 0;

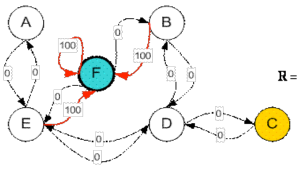
State Diagram



Reward table/ Matrix R



Reward table/ Matrix R



$$
\mathbf{R} = \begin{array}{c} state \setminus action \\ A \\ B \\ C \\ D \\ E \\ F \end{array}
\begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array}
$$

Q Matrix- Experience Table

- Q matrix – **Brain of agent** - represent the memory of what the agent have learned through experiences.
- In beginning, **agent know nothing, thus Q is zero matrix.**
- Let no of state is known (6).

$$
\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}
\begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}
$$

- <u>In more general case, start with zero matrix of single cell.</u>
- <u>It is a simple task to add more column and rows in **Q** matrix if a new state is found.</u>

Q Matrix- Experience Table

- To use the Q matrix, the agent traces the sequence of states, from the initial state until goal state. The algorithm is as simple as finding action that makes maximum Q for current state:

Algorithm to utilize the Q matrix
Input: Q matrix, initial state

1. **Set current state = initial state**
2. **From current state, find action that produce maximum Q value**
3. **Set current state = next state**
4. **Go to 2 until current state = goal state**

- The algorithm above will return sequence of current state from initial state until goal state.

Q learning

- **Given** : State diagram with a goal state (represented by matrix R)
- **Find :** Minimum path from any initial state to the goal state (represented by matrix Q)

Q Learning Algorithm goes as follow

1. **Set parameter , and environment reward matrix R**
2. **Initialize matrix Q as zero matrix**
3. **For each episode:**
    - **Select random initial state**
    - **Do while not reach goal state**
        - **Select one among all possible actions for the current state**
        - **Using this possible action, consider to go to the next state**
        - **Get maximum Q value of this next state based on all possible actions**
        - **Compute**

          $\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max[\mathbf{Q}(next\ state, all\ action$
        - **Set the next state as the current state**

End Do
End For

Step 2:

- Let us set the value of learning parameter=0.8 and initial state as room B.
- Set matrix Q as a zero matrix.
- Reward matrix R

$$\mathbf{Q} = \begin{array}{cc} & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad \mathbf{R} = \begin{array}{cc} state \setminus action & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} & \begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array}$$

Step 3: Update Q Matrix/Experience Table

- Randomly choose a state
- Let it select state B in matrix
- 2 possible action- D,F
- Consider now we are in state F.
- It has 3 possible actions to go to
    - State B, E or F.

$$\mathbf{R} = \begin{array}{cc} state \setminus action & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{array}{c} A \\ \boxed{B} \\ C \\ D \\ E \\ \boxed{F} \end{array} & \begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & \boxed{0} & - & \boxed{100} \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array}$$

Update Q Matrix

$$Q(state, action) = R(state, action) + \gamma * Max[nextstate, allactions]$$

$$Q(B,F) = R(B,F) + 0.8 * Max[Q(F,B), Q(F,E), Q(F,F)]$$
$$Q(B,F) = 100 + 0.8 * Max[0,0,0] = 100 + 0.8 * 0 = 100$$

$$\mathbf{Q} = \begin{array}{cc} & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

- F is final state – end of one episode.

Repeat again (Episode 2)

- Start with initial random state.
    - State D
    - 3 possible actions- B, C and E.
- By random selection, let
    - B is next state.
    - state B- 2 possible actions (D, F)
- Compute Q value

$$\mathbf{R} = \begin{array}{cc} state \setminus action & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{array}{c} A \\ \boxed{B} \\ C \\ \boxed{D} \\ E \\ F \end{array} & \begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array}$$

$$Q(state, action) = R(state, action) + \gamma * Max[nextstate, allactions]$$

$$\mathbf{Q}(D,B) = \mathbf{R}(D,B) + 0.8 \cdot Max\{\mathbf{Q}(B,D), \mathbf{Q}(B,F)\} = 0 + 0.8 \cdot Max\{0, 100\} = 80$$

$$\mathbf{Q} = \begin{array}{cc} & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Inner loop continue

Start Again with B state

$$\mathbf{R} = \begin{array}{cc} state \setminus action & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} & \begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array} \qquad \mathbf{Q} = \begin{array}{cc} & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max[\mathbf{Q}(next\ state, all\ actions)]$$
$$\mathbf{Q}(B,F) = \mathbf{R}(B,F) + 0.8 \cdot Max\{\mathbf{Q}(F,B), \mathbf{Q}(F,E), \mathbf{Q}(F,F)\}$$
$$= 100 + 0.8 \cdot Max\{0,0,0\} = 100$$

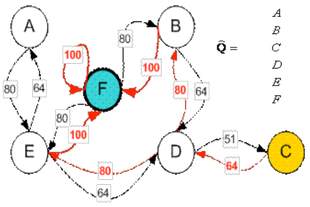
- No change in matrix Q – same value
- F goal state – Finish 2 episode

Continue for more episodes …..

- If agent learn more and more, experience through many episode,
- It reaches to convergence value of matrix Q

$$\mathbf{Q} = \begin{array}{cc} state \setminus action & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} & \begin{bmatrix} - & - & - & - & 400 & - \\ - & - & - & 320 & - & 500 \\ - & - & - & 320 & - & - \\ - & 400 & 256 & - & 400 & - \\ 320 & - & - & 320 & - & 500 \\ - & 400 & - & - & 400 & 500 \end{bmatrix} \end{array}$$

After Normalization



$$\hat{\mathbf{Q}} = \begin{array}{c|cccccc} state \setminus action & A & B & C & D & E & F \\ \hline A & - & - & - & - & 80 & - \\ B & - & - & - & 64 & - & 100 \\ C & - & - & - & 64 & - & - \\ D & - & 80 & 51 & - & 80 & - \\ E & 64 & - & - & 64 & - & 100 \\ F & - & 80 & - & - & 80 & 100 \end{array}$$

C -- D -- B -- F or C -- D -- E -- F

Q-Learning: Representation Matters

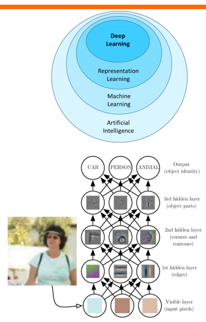
- In practice, Value Iteration is impractical
  - Very limited states/actions
  - Cannot generalize to unobserved states

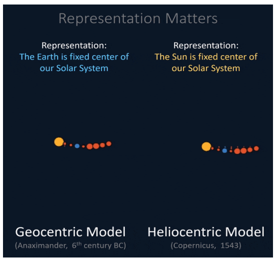
- Think about the **Breakout** game
  - State: screen pixels
    - Image size: **84 × 84** (resized)
    - Consecutive **4** images
    - Grayscale with **256** gray levels

$256^{84 \times 84 \times 4}$ rows in the Q-table!

$= 10^{69,970} >> 10^{82}$ atoms in the universe

Deep RL = RL + Neural Networks



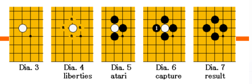


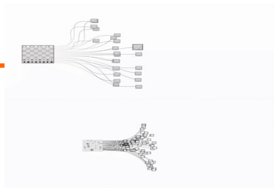
DeepMind Atari (©Two Minute Lectures)

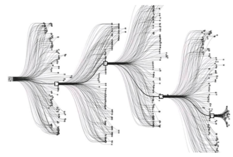
Alpha Go Story

Source: https://www.youtube.com/watch?time_continue=6&v=8tq1C8spV_g&feature=emb_title
https://www.youtube.com/watch?v=8dMFJpEGNLQ

Game of Go

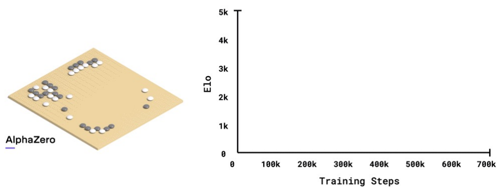






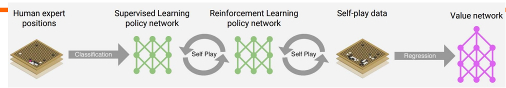
Game size	Board size N	3^N	Percent legal	legal game positions (A094777)[11]
1×1	1	3	33%	1
2×2	4	81	70%	57
3×3	9	19,683	64%	12,675
4×4	16	43,046,721	56%	24,318,165
5×5	25	8.47×10^{11}	49%	4.1×10^{11}
9×9	81	4.4×10^{38}	23.4%	1.039×10^{38}
13×13	169	4.3×10^{80}	8.66%	$3.72497923 \times 10^{79}$
19×19	361	1.74×10^{172}	1.196%	$2.08168199382 \times 10^{170}$

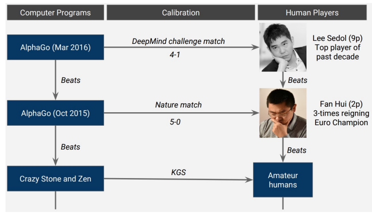
Deep Mind, acquired by Google in 2014, made headlines in 2016 after its AlphaGo program beat a human professional Go player Lee Sedol, the world champion, in a five-game match.



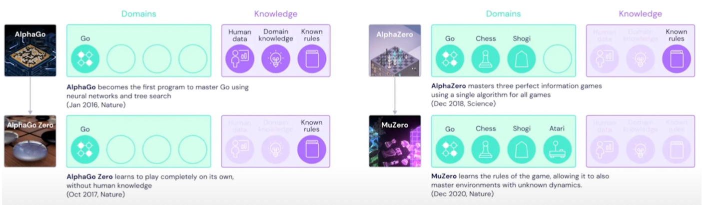
A more general program, AlphaZero, beat the most powerful programs playing go, chess and shogi (Japanese chess) after a few days of play against itself using reinforcement learning.

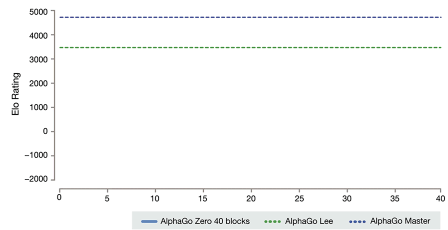
AlphaGo (2016) Beat Top Human at **Go**





MuZero: Learning dynamics for planning (2020)





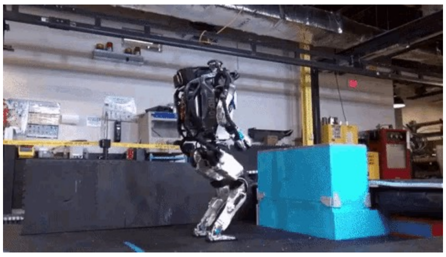
"In part because few real-world problems are as constrained as the games on which DeepMind has focused, DeepMind has yet to find any large-scale commercial application of deep reinforcement learning."

Aug 14, 2019 Wired : https://www.wired.com/story/deepminds-losses-future-artificial-intelligence/

Source : Simulation and Automated Deep Learning

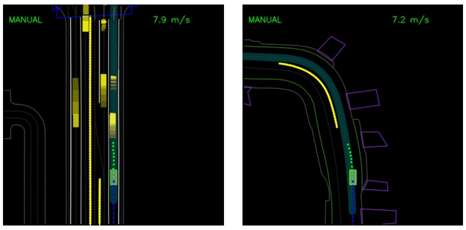
To date, for most successful robots operating in the real world: Deep RL is not involved

To date, for most successful robots operating in the real world: Deep RL is not involved

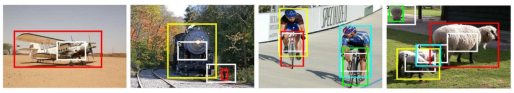


But… that's slowly changing:
Learning Control Dynamics

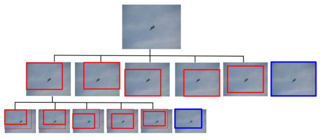
But… that's slowly changing:
Learning to Drive: Beyond Pure Imitation (Waymo)



But… that's slowly changing:
Object detection using DRL

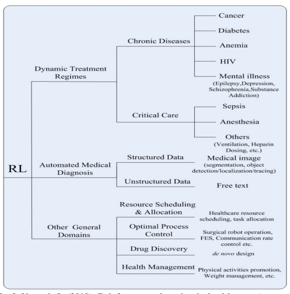


Deep Reinforcement Learning of Region Proposal Networks for Object Detection, 2018



- Hierarchical Object Detection wi

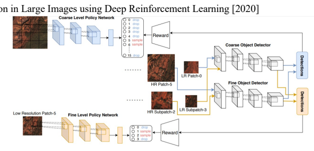
The outline of application domains of RL in healthcare



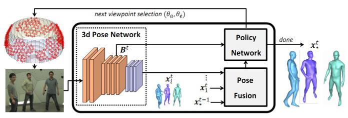
Source : Yu, C., Liu, J., & Nemati, S. (2019). Reinforcement learning in healthcare: A survey. arXiv preprint arXiv:1908.08796.

Deep Reinforcement Learning

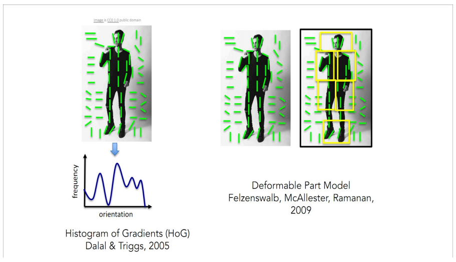
Efficient Object Detection in Large Images using Deep Reinforcement Learning [2020]



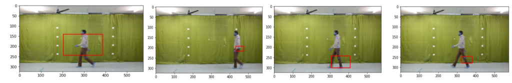
- Deep Reinforcement Learning for Active Human Pose Estimation [2020]



Pilot study for walking person detection using Reinforcement Learning



Test results of Walking person dataset:



- The bounding box shape is not correct but it is observable that the model has got the idea of how to detect a person in an image.
- Sometimes it zooms in too much on the person.

Test results of Walking person dataset:

Precision-Recall curves

Average precision-Recall score: **0.60**

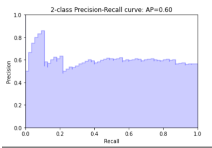
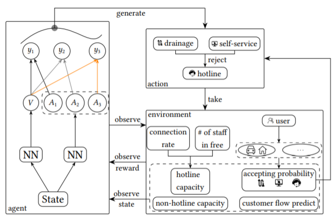


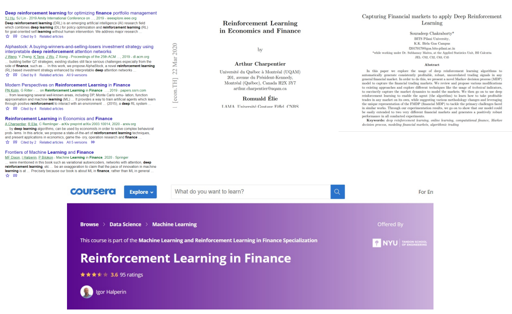
Fig: Precision vs Recall graph

Deep-RL in Call Centre

CRSRL: Customer Routing System Using deep Reinforcement Learning [2019]



Deep-RL in Financial markets



Deep Reinforcement learning in Electrical Engineering

Deep reinforcement learning for strategic bidding in **electricity** markets
Y Ye, D Qiu, M Sun ... - ... on Smart Grid, 2019 - ieeexplore.ieee.org
Bi-level optimization and **reinforcement learning** (RL) constitute the state-of-the-art frameworks for modeling strategic bidding decisions in deregulated **electricity** markets. However, the former neglects the market participants' physical non-convex operating ...
Cited by 7 Related articles

Increasing performance of **electric** vehicles in ride-hailing services using **deep reinforcement learning**
JF Pettit, R Glatt, JR Donadee, BK Petersen - arXiv preprint arXiv ..., 2019 - arxiv.org
New forms of on-demand transportation such as ride-hailing and connected autonomous vehicles are proliferating, yet are a challenging use case for **electric** vehicles (EV). This paper explores the feasibility of using **deep reinforcement learning** (DRL) to optimize a ...
Cited by 3 Related articles All 4 versions

[HTML] **Deep reinforcement learning** of energy management with continuous control strategy and traffic information for a series-parallel plug-in hybrid **electric** bus
Y Wu, H Tan, J Peng, H Zhang, H He - Applied Energy, 2019 - Elsevier
Hybrid **electric** vehicles offer an immediate solution for emissions reduction and fuel displacement under the current technique level. Energy management strategies are critical for improving fuel economy of hybrid **electric** vehicles. In this paper we propose a energy ...
Cited by 32 Related articles All 8 versions

Effective Charging Planning Based on **Deep Reinforcement Learning** for **Electric** Vehicles
C Zhang, Y Liu, F Wu, B Tang ... - IEEE Transactions on ..., 2020 - ieeexplore.ieee.org
Electric vehicles (EVs) are viewed as an attractive option to reduce carbon emission and fuel consumption, but the popularization of EVs has been hindered by the cruising range limitation and the inconvenient charging process. In public charging stations, EVs usually ...

Challenge: RL & Real-World Applications

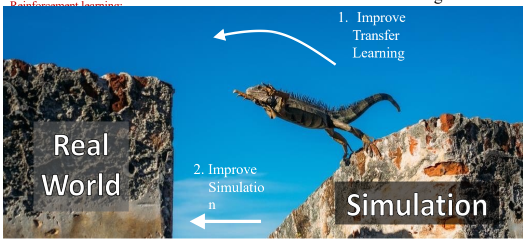
Reminder:

Supervised learning: teach by example

Reinforcement learning:

Open **Challenges**. Two Options:

1. Real world observation + one-shot trial & error
2. Realistic simulation + transfer learning

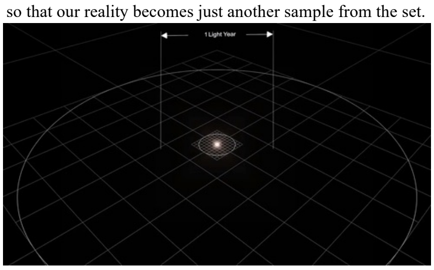


Conclusion

- Reinforcement learning addresses a very broad and relevant question: How can we learn to survive in our environment?
- We have looked at Q-learning, which simply learns from experience.
  - No model of the world is needed.
- We made simplifying assumptions: e.g. state of the world only depends on last state and action. This is the Markov assumption. The model is called a Markov Decision Process (MDP).
- There are many extensions to speed up learning.
- There have been many successful real world applications.

Thinking Outside the Box:
Multiverse Theory and the Simulation Hypothesis

- Create an (infinite) set of simulation environments to learn in so that our reality becomes just another sample from the set.



Key Takeaways for Real-World Impact

- Deep Learning:
  - **Fun part:** Good algorithms that learn from data.
  - **Hard part:** Good questions, huge amounts of representative data.
- Deep Reinforcement Learning:
  - **Fun part:** Good algorithms that learn from data.
  - **Hard part:** Defining a useful state space, action space, and reward.
  - **Hardest part:** Getting meaningful data for the above formalization.

References

- MIT Deep Learning Basics: Introduction and Overview with TensorFlow
- Univ. of Alberta
  - http://www.cs.ualberta.ca/~sutton/book/ebook/node1.html
  - www.cs.ualberta.ca/~sutton/book/the-book.html
  - Sutton and barto,"Reinforcement Learning an introduction."
- Univ. of South Wales
  - http://www.cse.unsw.edu.au/~cs9417ml/RL1/tdlearning.html
- https://people.revoledu.com/kardi/
- http://mnemstudio.org/path-finding-q-learning-tutorial.htm
- MIT Deep Learning and Artificial Intelligence Lectures
- https://www.analyticsvidhya.com/blog/2017/01/introduction-to-reinforcement-learning-implementation/
- https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/
- https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/