**ReInforcement Learning workshop assets:**

**Introduce to RL**

**Supervised Learning**: learning from a training set of labeled examples provided by a knowledgeable external supervisor.

**Unsupervised Learning**: typically, about finding structure hidden in collections of unlabeled data.

**Reinforcement Learning:**

* Dynamic Systems theory: the optimal control of incompletely-known Markov decision processes
* Reward Hypothesis theory: all goals can be described by the maximation of expected cumulative reward.
* The Markov Decision Process (MDP) and Partially Observable MDP (POMDP)

[Dynamic Programming](file:///C:\Users\Chris\Documents\Amazon\RL\MarkovDecisionProcess\1960-howard-dynamicprogrammingmarkovprocesses.pdf) and [Markov Processes](file:///C:\Users\Chris\Documents\Amazon\RL\MarkovDecisionProcess\markov_process.pdf) – Ronald Howard

* Exploration and Exploitation (EvE). Exploring the environment in order to exploit a reward system. This is similar to the classic balance of Bias vs Variance in Supervised and Unsupervised learning. EvE must balance enough exploration for the agent to know the environment well enough to make a choice of optimal exploitation.

Markov Property = future from that state is conditionally independent of the past, given that we know the current state. i.e a ball flying through the air. If state = [position AND velocity] that can tell us where it has *been* AND where it will *go*. However, if we do not know velocity state is no longer Markov. The current state does NOT summarize all past states. All equations written in Sutton+Barto convention

P[ St+1 | St ] = P[ St+1 | S1,…, St]

Probability of the next state conditioned on the state you are in

P [ St+1 | St ] = P [ St+1 | S1,…, St ]

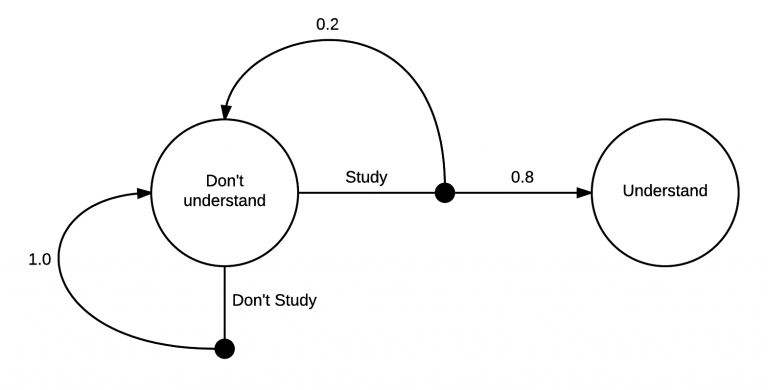
Is equal to the Probability of the next state if shown all previous states

“The future is independent of the past given the present” – David Silver

For a Markov state s and successor s’ (s-prime), the state transition probability is defined as:

**Pss’ =** P[St + 1 = s’ | St = s]

**MDP Transition** **Probability distribution**



A markov chain of two states (Understanding and Not understanding). At first you are in the state Don’t understand. From there, you have two possible actions, Study or Don’t Study. If you choose to not study, there is a 100% chance that you will end up back in the Don’t understand state. However, if you study, there is a 20% chance you’ll end up back where you started, but an 80% chance of ending up in the Understand state. The probability is distributed between 80% chance of going to the understand state, and 20% chance of the don’t understand state.

**Basic components of RL**

TODO: Insert Anatomy of an RL environment (diagram):

**Agent** – sometimes called an Actor – does the exploring and receives the reward. It may include one or more of:

**Environment** – The world, physical or virtual as it is perceived by an Agent. It could be completely or partially observable. Environments are either Stochastic (elements of the environment are constantly changing) or Deterministic (the environment has known “rules” or limits”)

* ***Policy*:** agent’s behavior function – how the agent picks its actions. It is the map from state to action.

Deterministic policy: a = π(*s*)

Stochastic policy: π(a|*s*) = P[A = a|S = s]

Probability of taking “some” action (A = a) conditioned on being in some state (S = s)

* ***Value Function***: how “good” (Quality) of each state and/or action. A prediction of future reward.

Vπ(s)

V = Value Function

π represents the idea that the function will return different results each time based on the agent’s quality/condition and the state **(S)** the agent is in. This is called indexing the function by π

i.e An agent that is “falling over” continuously will get a very different reward than an agent that is walking correctly)

Vπ(s) = E π[*R*t + γ *R*t+1 + γ 2*R*t+2 + … | St = s]

Value function for a policy = Expectation of the reward over the future as it is discounted more and more, being in some state (S = s)

γ = Gamma. Represents the discount in MDPs. Why discounts?

Mathematically convenient to discount rewards (making reward further into the future smaller)

Avoid finite returns in Cyclic MDPs

Future is uncertain

* ***Model*:** agent’s representation of the environment – may not be ACTUALLY how the environment works, but it is how the agent “THINKS” the environment works. A model predicts what the environment will do next.

Transition(Model): **P** predicts the next state (Dynamics systems)

Rewards (Model): **R** predicts the next (immediate) reward

**Pass’ =** P[S’ = s’ | S = s, A= a]

**Ras =** E[Rt+1 | St = s, At = a]

**\***Some agents do not use models and they are optional

**Markov Reward Process**

**Return –** return Gt is the total discounted reward from time-step *t*.

Gt = Rt + 1 + γRt+2 + … = ∑∞ γkRt+k+1

**State Value function –** is the long term value (value from then on) of being in [concrete] state *s.* The State Value function v(s) of an MRP is the expected return starting from state *s*

v(s) = E[Gt | St = s]

**The Bellman Equation for MRPs**

(MRP Value Function can be decomposed into TWO parts)

* + Immediate reward Rt+1
  + Discounted value of successor state γv(St+1)

**Markov Decision Process**

An MDP is an MRP WITH DECISIONS. It is an environment in which all states are Markov.

Markov Decision Process is a tuple (S, A, P, R, γ)

* S is a finite set of states
* A is a finite set of actions
* P is a state transition probability matrix - **Pass’ =** P[S’ = s’ | S = s, A= a]
* R is a reward function - **Ras =** E[Rt+1 | St = s, At = a]
* γ is a discount factor – γ ϵ [0,1]

**Policies** – a policy π is a distribution over actions given states

π (a|s) = P[At = a | St = s]

in MDP polices depend on the current state (not the history)

**State Value Function** (how good is it to be in a particular state S) SVF of an MDP is the expected return starting from state s, and then following policy π

**Action Value function** (how good will it be to take this action)

V = telling us how good is it to be in a particular state

Q = telling us how good is it to take an action from a particular state

**Applied (hands on) RL**

**Early Concept: Q-Learning – Jupyter Notebook**

The Q-table

Q is for quality

Q-Value = the choice with the best reward given the available options in a current state

Q-Learning Algorithm (Action Value Function)

**First Evolution: Deep Q-Learning (DQN) – Jupyter Notebook**

Q Table values are determined by a Neural net (no memory, no framestacking) CartPole?

**Second Evolution: DQN with Memory – Jupyter Notebook**

Cartpole via Gym again – this time with deque memory

**Third Evolution: DQN with Memory and Frame Stacking – Jupyter Notebook**

**Current Evolution: Pixel based, A3C, PPO?**

Andrew Ng is wrong. (https://www.oreilly.com/ideas/practical-applications-of-reinforcement-learning-in-industry)

Assets:

Spreadsheet with landscape map of RL

Vocabulary

**Algo** – sometimes referred to as a “policy” the heuristics used to make a decision. i.e “Should I move forward or not? Is there a reward if I move forward?”

**State** – We could simply insert Markov Decision Processes here. State is meant to qualify the condition/location/status of the agent as a given point in time.

**Actions** – options available to the agent. I.e step forward, step sideways.

**Value/Reward** – The value of a state refers to how much reward the agent will receive for performing an action to achieve said state.

**ToolKit**

**Policy**

**On-Policy vs Off-Policy**

**Curriculum Learning**

Introduce a simplified version of a task to an agent, slowly increasing the complexity

**Domain Randomization**

Introducing variability into the simulation to assist models in generalizing to the real world.

**Temporal Limitation** – in certain environments, a snapshot in time (TEMPORAL) may not provide enough information to predict the best action. In these instances multiple snapshots are need to understand the direction/momentum/inertia of objects in motion to choose the best action.

**Entropy** – Entropy in RL is a measurement of the unpredictability of possible actions in a given policy. Higher Entropy = higher randomness in action choice

**Adaptive Task Selection**

**Behavioral Cloning**

**Single Agent vs multi Agent**

**Discrete vs continuous Actions**

**Dense vs Spare Rewards**

**Continuous Control**

**learning curricula**

**memory systems**

**Episode –** One complete agent training session on a distinct environment. Training ends when either the goal is achieved by the agent or a pre-determined number of actions have been taken and the agent training is terminated.

**Run(s)**

**Evaluative vs Instructive feeback**

Instructive feedback tells you HOW to achieve your goal

Evaluative feedback tells you HOW WELL you achieved your goal

**Episodic vs Continuous Task**

**Monte Carlo reward vs Temporal Difference Learning**

H1:t 🡪 St 🡪 Ht+1:∞

Full observability: Agent directly observes the environment state. (What You See Is All There Is)

Ot = Sat = Set

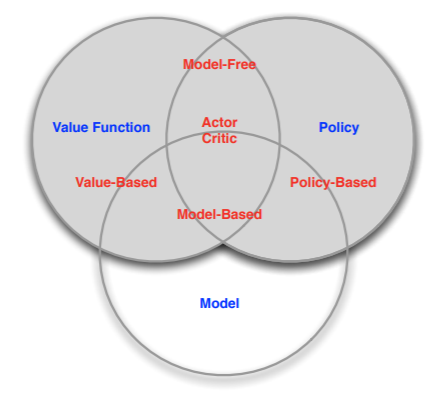
Agent state = environment state = information state

**RL Agent Taxonomy**

**Value-Based vs Policy-Based vs Actor Critic**

* Value Based = agent has a value function (Policy is implicit)
* Policy Based = agent explicitly uses policy without storing Value Function
* Actor Critic = agent that uses both Policy and Value Functions

**Model vs Model Free**

* Model Free = agent using Policy and/or Value Function
* Model = Model how the environment works then plan with Policy/Value functions****

<http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching_files/intro_RL.pdf> - David Silver