DEEP REINFORCEMENT LEARNING

TABLE OF CONTENTS

Presentation 45-50 mins

- Brief History of Reinforcement Learning
- Differences between Reinforcement learning and Un/Supervised Learning
- What is Reinforcement Learning?
- Methodology, Notation and Parts of RL
- Deep Reinforcement Learning
- Future Research
- Additional Resources

Q&A 10-15; My Personal Projects

HISTORY OF REINFORCEMENT LEARNING

Founders

- Andrew Barto, Richard Sutton, Richard Bellman and Christopher Watkins

Foundational 3 corded braid of RL

- Animal Learning by trial and error
- Optimal Control/ Dynamic Programming
- Temporal Difference Learning

DIFFERENCES OF LEARNING METHODS

Supervised Learning

- Knowledge is classified (labeled) by an external Expert.
- Generalizes features from data to be able to classify outside of data set.

Unsupervised Learning

- Extracts hidden structures from unlabelled data.
- Does not have a goal or takes action.

Reinforcement Learning

- Uses Value Function and Reward Signals.
- It has a defined goal it is trying to accomplish.

DIFFERENCES OF LEARNING METHODS CONT.

Evolutionary Methods

- It is not goal based, only tries to increase fitness.
- Can’t see which actions lead to improvement and takes the whole model as the improvement.
- Has no long term value function, easily gets stuck on short term high rewards that have lower long term overall value.

Example:

Methods	Rewards at different time stamps					Total
RL	.1	.2	.6	.7	.8	2.4
Genetic	.9	.8	.01	.008	.00009	1.71809

WHAT IS REINFORCEMENT LEARNING?

Reinforcement Learning is an *Agent* with a *Goal* *Interacting* with an *Environment* and *Learning* from an *Experience.*

- An Agent is a network with two functions: a value-function and a policy function.
- A Goal is what you or the agent decides it wants to do. It is shaped by rewards.
- Interacting means the agent taking Actions in the environment.
- The Environment is the world in which the agent lives and is able to interact with.
- Learning is the agent evaluating the rewards it earned and optimizing its value and policy functions to better achieve its goal.
- Experience is everything the agent has done and seen. Its memory.

NOTATION

s, s_t, s' = States

a = An action

r = A reward

S = Set of all nonterminal states; S^+ is the same but includes the terminal states

R = Set of all possible rewards

A = Set of all available actions in state s

t = Discrete time step

π = Policy

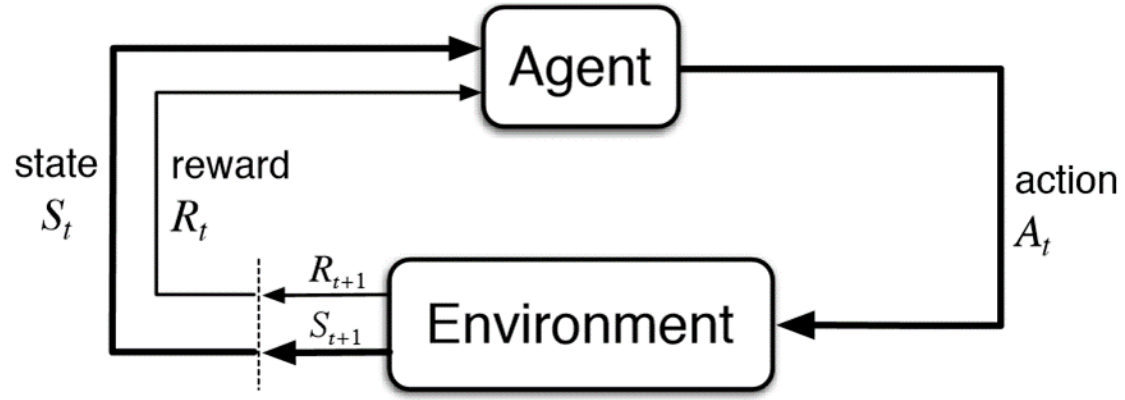
v_π = Value of state s under policy π

q_π = Value of taking action a at state s under policy π

λ = Discount Factor

α = Step-Size Parameter

MARKOV DECISION PROCESS



- The Markov Decision Process, MDP, is the core of RL.
- The Environment is set up and the agent looks at it in time steps as States. The agent then preforms an action in order to get closer to its goal. That action affects the Environment and changes it creating a new State, s_{t+1}.
- The reward signal is then updated and the agent learns from its behaviour. (Trail and Error)
- Our agent seeks to increase Value NOT Rewards.

WHAT IS THE AGENT AND WHAT ISN'T?

An Agent-Environment boundary line.

In a robot or animal what is the agent and what is the environment?

The value function:

- Gives the model a sense of long term and short term goals and rewards.
- What the agents seeks to improve.

The policy function:

- Determines what actions the agent takes.
- What are NN seek to improve.

HOW CAN WE GIVE THE AGENT A GOAL?

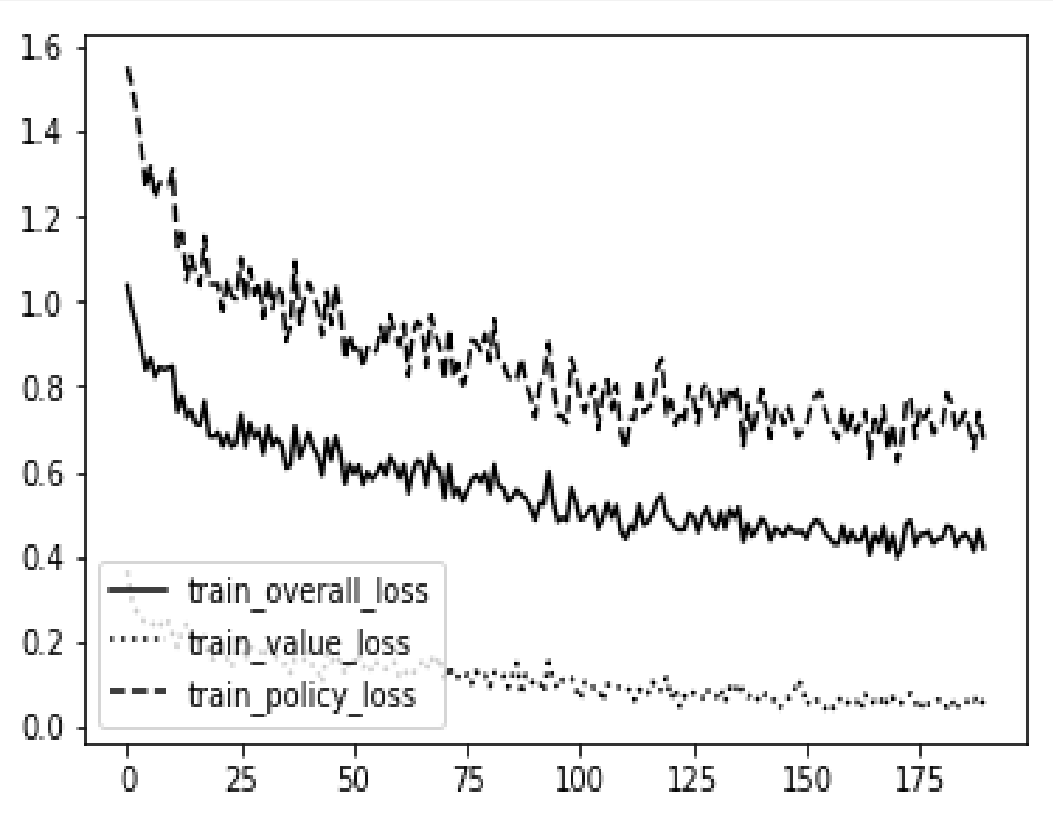
We can imply our desires to the agent through positive and negative rewards. This is the *Reinforce* part of RL.

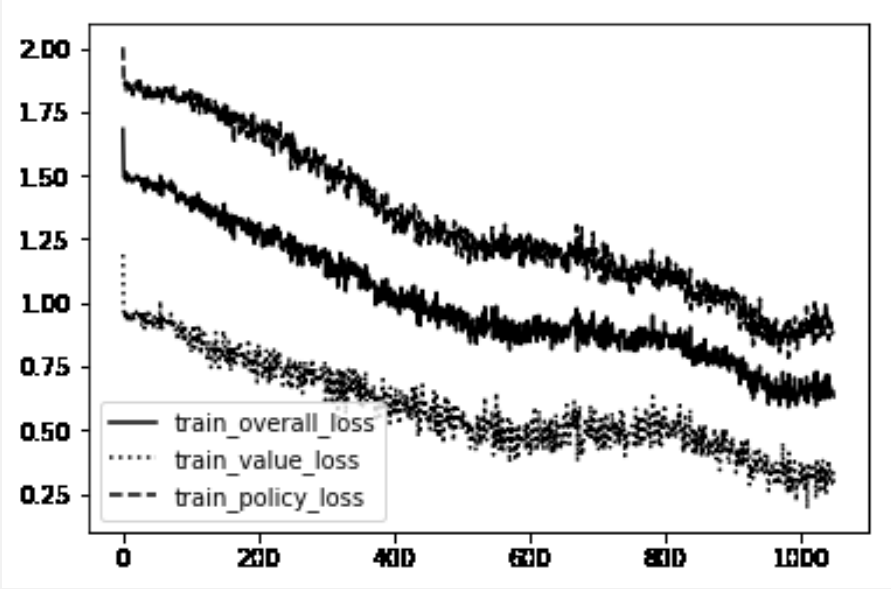
Ways to accomplish this:

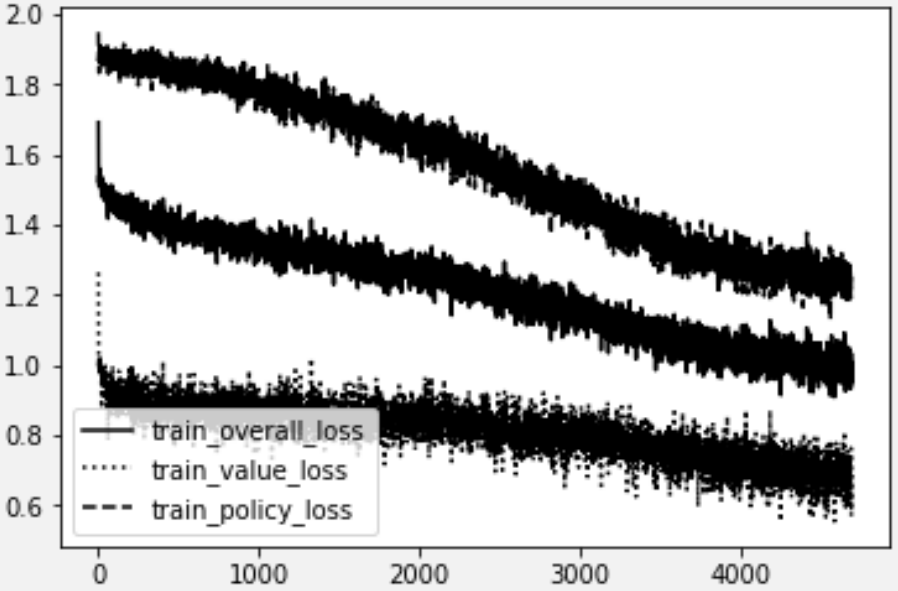
- Giving constant rewards per time step.
- Giving conditional rewards per action.
- Giving large rewards on completion of goal.

Be careful not to give a large reward to sub-goals or your agent might overfit to something stupid. Like this → → →

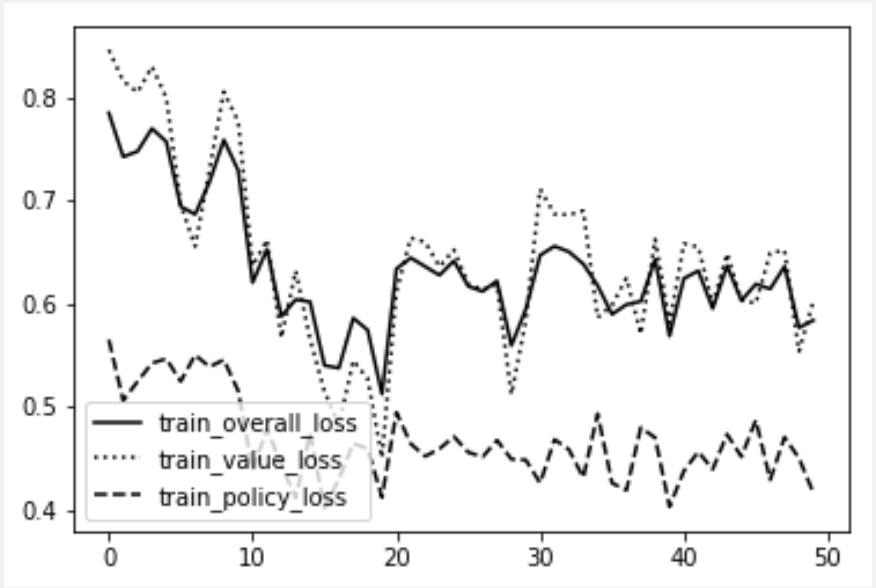
GOOD HYPERPARAMETERS

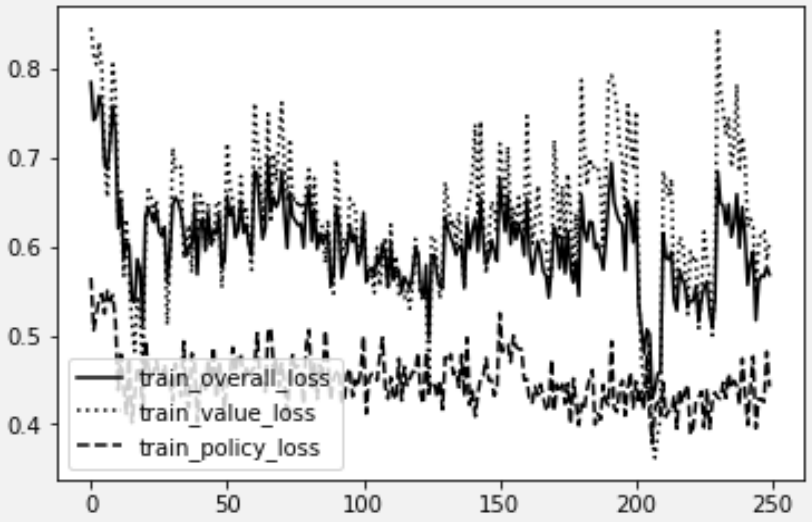


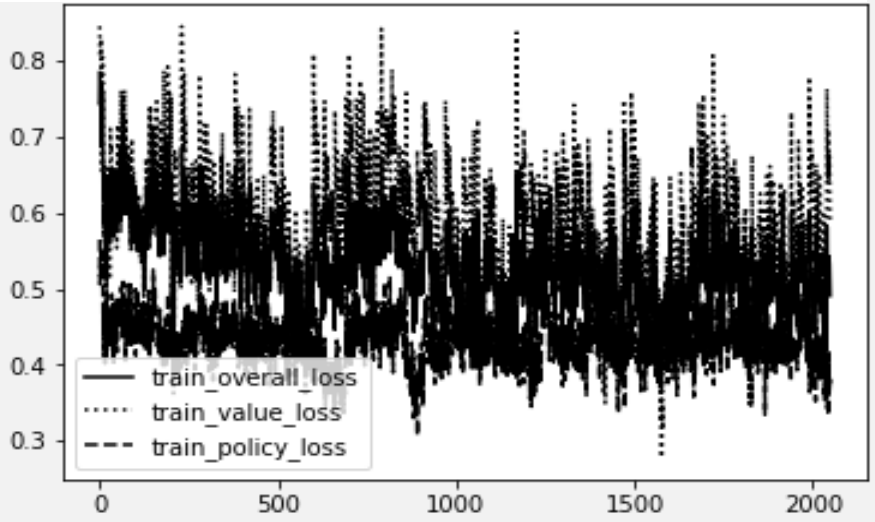


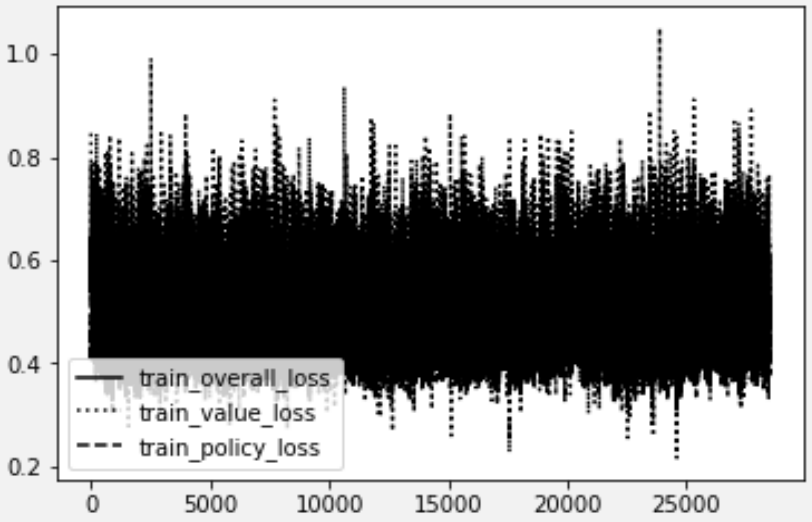


BAD HYPERPARAMETERS









HOW DOES IT LEARN?

The agent learns in two ways:

- It can optimize the policy based off of the difference in expected reward and true reward.
- A NN can learn to map the Q-values and adjust weights along gradients and other methods to reduce error.

We use a discount parameter, to make the agent hedonistic.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

where γ is a parameter, $0 \leq \gamma \leq 1$, called the *discount rate*.

- This allows the agent to achieve more during life and not focus on death.
- A λ of 0 makes it only focused on the immediate reward.
- A λ of 1 makes it consider all rewards equally.

VARIETY OF RL MODELS

Types of States

- Discrete time steps, Continuous time and Infinite states.

Types of Environments

- Games against nature, Games against an opponent, Games against self, Games against multiple opponent agents and Games with teammates.

Types of Rewards

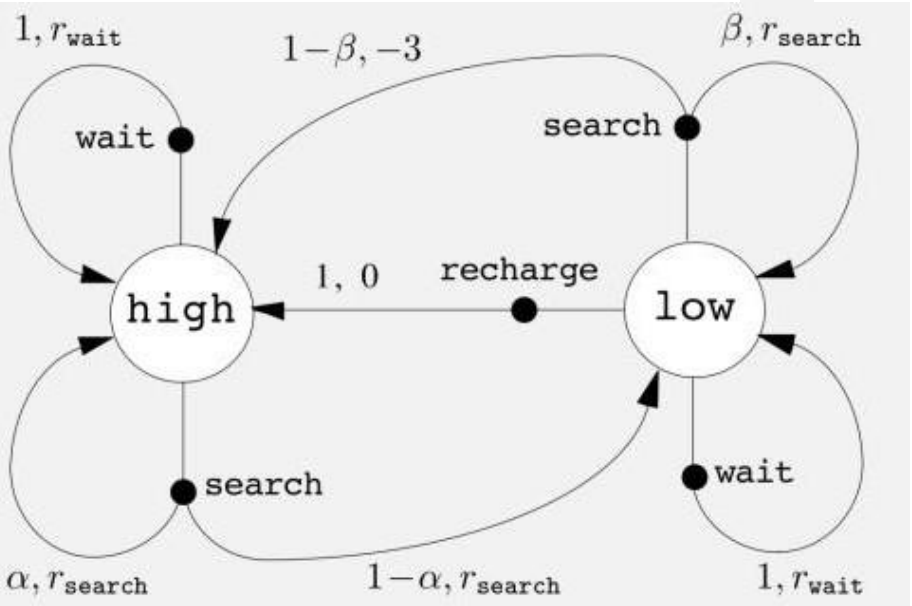
- Sparse Rewards, Instant Rewards, Random Rewards, Delayed Rewards and two-way rewards.

AN EXAMPLE; WALL-E

Imagine a robot whose job it is to pick up trash. It has a rechargeable battery with three states High, Low and Dead. It can do one of three things at a time. Search around for trash, Wait for a human to bring it trash or drop trash near it, or go home and recharge. Picking up trash gives it a +1 reward. Waiting a +0 reward. Recharging gives it a +0 reward as well. Running out of battery gives it a -3 reward. Every time the robot searches it uses some of its battery and has a probability, α - β, of switching from a higher state to a lower state.

What is the optimal policy and how would it learn?

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	r_{search}
high	search	low	$1-\alpha$	r_{search}
low	search	high	$1-\beta$	-3
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	-



THREE MAIN THEMES OF RL

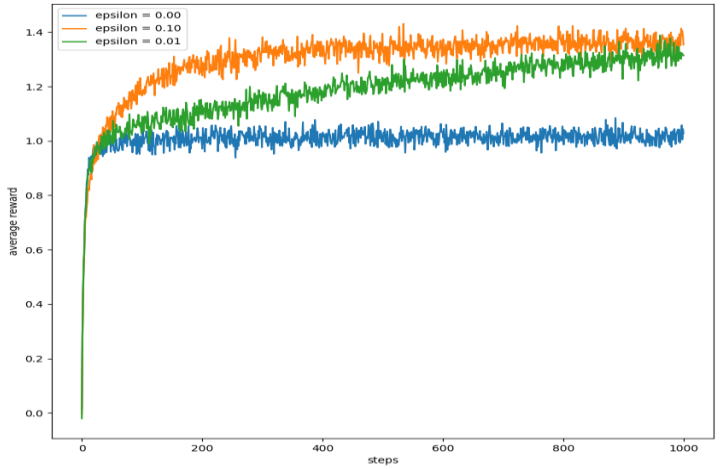
Every new model of RL seeks to improve three areas:

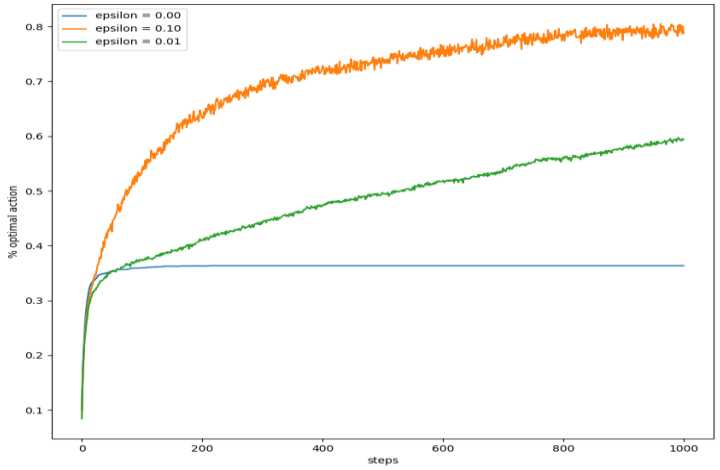
- Exploration/ Exploitation(Greedy).
- The way the agent creates goals and how it learns from it.
- The computation speed until convergence and the accuracy of results.

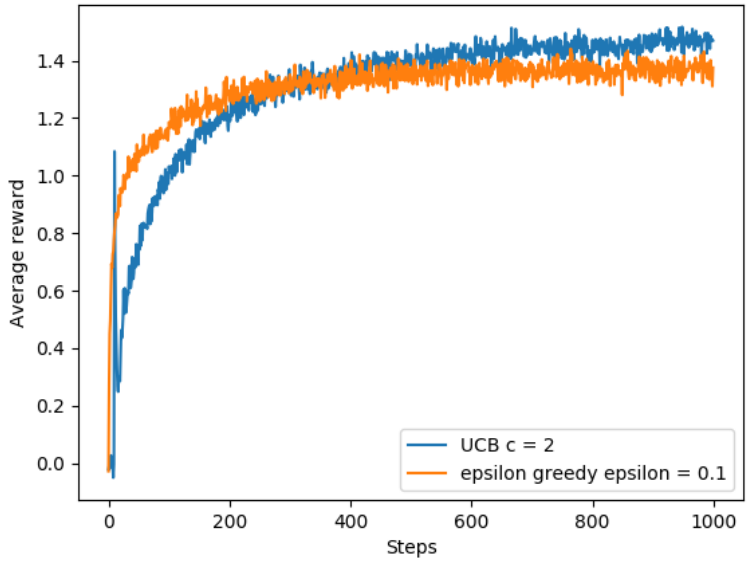
Exploration vs Greed

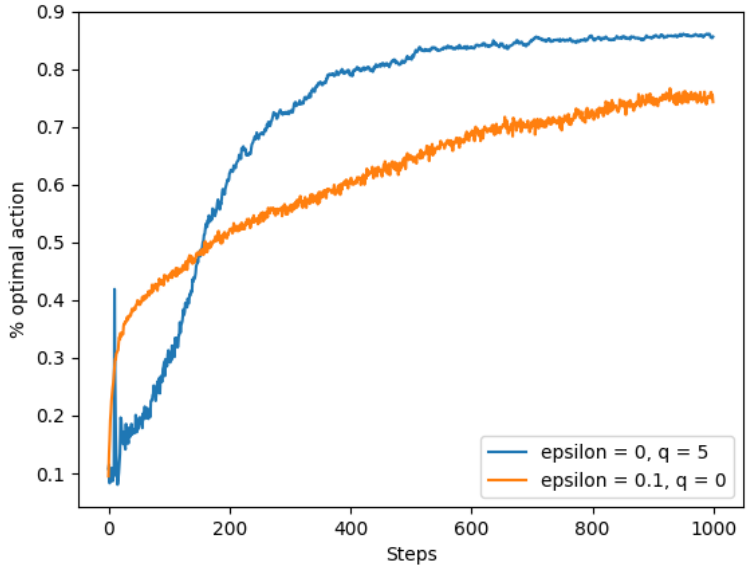
- Greedy: The agent only considers the immediate best reward
- Epsilon Greedy: The agent considers the best reward most of the time and a random reward ε times
  - Stationary epsilon: The value never changes. Usually .1 - .3 range with .2 being a goto.
  - Epsilon Decay: The value starts high then decays per time step to a min number. .99 to .1
- Optimistic: Set a very high starting value or *bias* which makes the agent only go after rewards higher then the starting bias.
- Upper Confidence Bounds: A policy selects actions repeatedly until it understands the potential and optimal paths better. Actions with lower value estimates or that have been selected often will be picked less frequently over time.

EXPLORE / EXPLOIT









MODEL BASED VS MODEL FREE

Model Based

- Temporal Difference and Heuristic Search
- Models simulate reality and let the agent plan ahead of time and make predictions.
- The agent uses 2 functions, the transition function T and the reward function R.
- The agent learns by reducing the difference in the predicted value and the real value.

Model Free

- Monte Carlo Methods and Dynamic Programming
- Learns the policy directly using Q tables or policy gradients.
- Can only make predictions implicitly by learning from experience. Can not make a explicit call to a model to predict the future.

Both can be used together by having the Model create a simulated experience for the model free agent to learn by. This allows for greater ability of the RL model.

LIMITATIONS AND USE CASES

RL is at its best when it is being used with another NN. The strength of RL is that it can learn just about anything, and when paired with the right network it can out preform any standard NN.

That being said RL is extremely hard to implement in novel domains and the research is slow and painful.

Some common limitations are:

- Creating a Reward Function, Instructing the goal to the agent, Escaping Local Optima, Getting the right balance of exploration/exploitation, Overfitting to patterns in the environment, and Results are hard to reproduce and be unstable.

Common use cases for RL:

- Locomotion, Continuous Learning Problems, Operating Machinery, Genetics Research, Playing Games, Creating art (games, music, pictures), Video Prediction, Image Captioning, Language Translation, Robotic Grasping/Cooking, NLP, Speech Recognition/Synthesis.

DEEP REINFORCEMENT LEARNING

Every problem in machine learning can be solved by some combination of RL model and ANN architecture

Types of DRL:

- Policy Gradient:
  - Actor-Critic
  - TRPO
  - PPO
  - ACER
  - ACTKR
  - REINFORCE
- Q-Value:
  - Q-Learning / DQN
  - DDPG
  - SARSA

Tools for DRL:

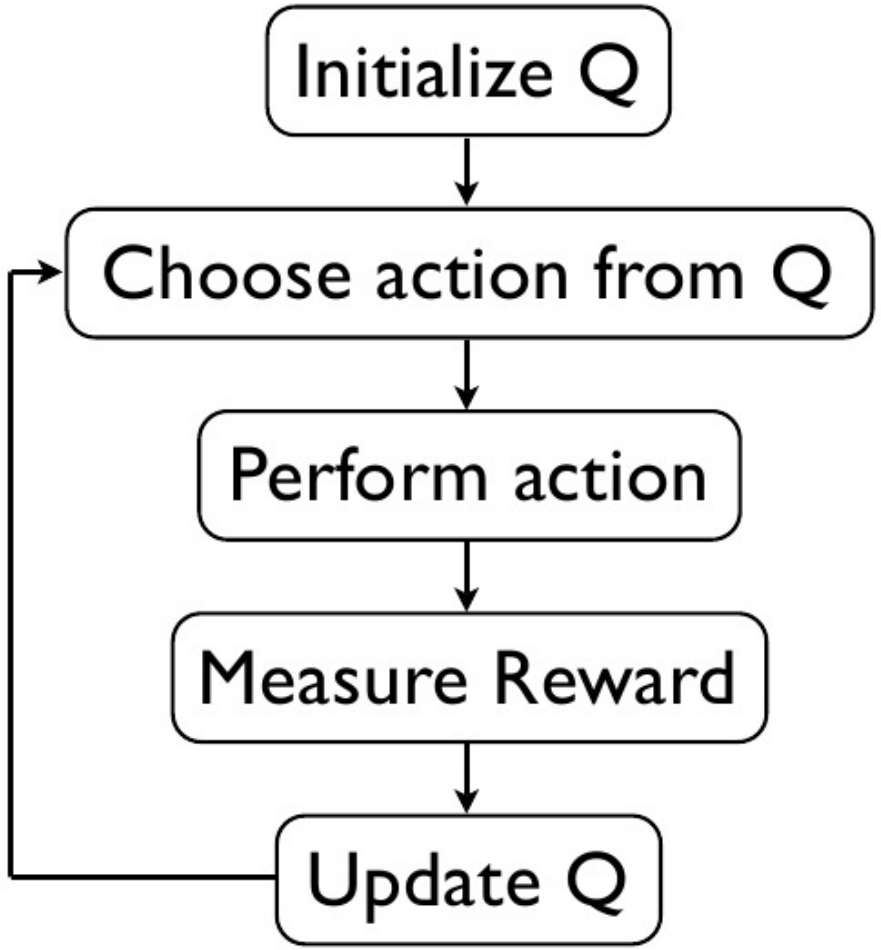
- Frameworks
  - Tensorflow: Tensorlayer and TRFL
  - Keras
  - Pytorch
- APIs
  - OpenAI Gym
  - OpenAI Roboschool
  - Deepmind Lab
  - Horizon

Q LEARNING

THE OG

- It is a model free, discrete action space and state space.
- For any finite MDP, Q-Learning can find an optimal policy by maximizing the expected value of the total reward over any and all successive steps. It can handle stochastic transitions and rewards.
- It creates a Q table, logging each Q value of all explored states and actions. It then pulls from this table to make actions from new states or old states.

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$



Q LEARNING CONT.

- The learning rate or step size determines to what extent newly acquired information overrides old information. Often a constant learning rate is used in practice.
- The discount factor determines the importance of future rewards.
- Initial conditions can be optimistic (high initial values), which encourages exploration.
- The Q function gets better at approximations over time by continuously updating the Q values in the table.

Can use two different methods to minimize the MSBE:

- Replay Buffers – is a set of previous experiences, it should be large enough to hold a wide range of experiences. If you keep only the most recent data you will overfit and if you keep too much experience the learning will be slow.

- Target Networks – uses a target as an example of what the MSBE loss should be. A second network called the target network is just a copy of the main network that is updated every n-steps.

PROXIMAL POLICY OPTIMIZATION

- Does not use Q-values, instead uses policy gradient methods. It also minimizes the cost function while ensuring the deviation from the previous policy is relatively small.

$$L^{CLIP}(\theta) = \hat{E}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right]$$

- θ is the policy parameter
- $\hat{E}_t$ denotes the empirical expectation over timesteps
- r_t is the ratio of the probability under the new and old policies, respectively
- $\hat{A}_t$ is the estimated advantage at time t
- ε is a hyperparameter, usually 0.1 or 0.2

- Has a continuous action space and state space, the operator is the Advantage function.

- Uses Trust Region optimization to update policies. If the probability ratio falls outside of the range of (1 - ε) and (1 + ε) then the advantage function will be clipped. This discourages large policy change if it is outside of our trust region.

- Can use Gradient Descent like Adam to optimize it, works well with many different NNs.

FUTURE RESEARCH

Problems still faced today

- Large action spaces reduces an agents effectiveness at specializing in a task.
- Adaptive methods don’t work well with fewer parameters under a large number of conditions.
- RL models suffer greatly from complexity and computation speeds.

Research directions

- Transfer Learning and having different agents be able to communication information.
- Learning and acting under partial information.
- Efficient sample-based planning (Monte Carlo Tree Search).
- Reducing dimensionality and improving computation speed.
- Improving learning speed of agent by decreasing time steps needed to learn.

THANK YOU

Chris Harvey ✉ *cjh@ku.edu*

ADDITIONAL LEARNING

- “Steps toward Artificial Intelligence” - Minsky
- “Reinforcement Learning: An Introduction 2nd Edition” - Sutton and Barto
- https://skymind.ai/wiki/deep-reinforcement-learning
- ‘Hindsight Experience Replay’ – OpenAI, Andrychowicz
- ‘Deep Reinforcement Learning: Pong from Pixels’ - Karpathy
- https://lilianweng.github.io/lil-log/
- http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
- https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html
- https://robotacademy.net.au
- https://towardsdatascience.com/proximal-policy-optimization-ppo-with-sonic-the-hedgehog-2-and-3-c9c21dbed5e
- https://towardsdatascience.com/curiosity-driven-learning-made-easy-part-i-d3e5a2263359