Reinforcement Learning Cheat Sheet (add helpful images)

- Part 1: Watch the Alpha Go Movie. Deep Mind. **Reinforcement Learning** - learning to make a good sequence of decisions. Atari, Robotics, Education, Healthcare, NLP, Computer Vision all use RL. 4 components of RL: **optimization** - about yielding best outcomes, **delayed consequences** - planning, sacrificing reward now for reward later, **exploration** - you don't know the consequences of decisions not taken so you need to learn from experience, **generalization** - programming policies to act in certain ways given certain situations. **AI planning** - involves optimization, delayed consequences, generalization (you know the rules). **supervised and unsupervised learning** - only optimization and generalization, just that unsupervised learning doesn't have labels. **imitation learning** - learns from experience of others, involves optimization, delayed consequences, and generalization, assumes an input of good policies, reduces RL to supervised learning. **Issues with RL** - having the right rewards, robustness vs risk sensitivity (exploration vs reward trade-off), multi-agent RL. **sequential decision making** - cycle of: agent $\rightarrow$ (action a_t) $\rightarrow$ world $\rightarrow$ (observation o_t, reward r_t) $\rightarrow$ agent, goal is to select actions to maximize total future reward, may need to balance immediate and long term rewards, strategic behavior for high rewards, not the **discrete time step** is t. **history** - history of A, O, R that agent uses to make decisions. **world state** - full state of the world. **agent state** - state of world that agent needs to make decisions. **markov state** - state s_t is Markov iff $P(s_{t+1}|s_t, a_t) = P(s_{t+1}|h_t, a_t)$ i.e. that the future is independent of the past given the present. Setting $s_t = h_t$ allows any world to be markov. In practice, most recent observation is sufficient statistic of history i.e. setting $s_t = o_t$. **full observability / MDP markov decision process** - agent state same as world state **partial observability / POMDP partially observable MDP** - agent constructs its own state i.e. $s_t = h_t$, beliefs, RNN, etc (poker players only sees own cards, healthcare doesn't see all physiological processes). **bandits seq decision process** - actions have no influence on next observations, no delayed rewards. **deterministic** - given history and action, single observation and reward. **stochastic** - given history and action, many possible (probability distribution of) observations and reward. **model** - agent's understanding (model) of how the world changes in response the agent taking an action. **policy** - map from agent state to action to take. **value function** - future rewards of being in a state and/or action when following a particular policy. **transition / dynamics model** - predicts agent next state $P(s_{t+1} = s'|s_t = s, a_t = a)$. **reward model** - predicts immediate reward $R(s_t = s, a_t = a) = E[r_t|s_t = s, a_t = a]$ (reward will depend on which state one probabilistically ends up at, that's why it's an expectation?). **deterministic policy** - $\pi(s) = a$. **stochastic policy** - $\pi(a|s) = P(a_t = a|s_t = s)$. **value function** - $V^\pi(s_t = s) = E_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|s_t = s]$, **discount factor** γ weights immediate vs. future rewards, expectation is taken over all the different paths that can be taken from the state s by the policy π. Types of RL agents (what the agent/algo learns) - **value based**: explicitly learns value function, implicitly learns policy; **policy based**: explicitly learns policy, there is no value function; **actor critic**: explicitly learns policy, explicitly learns value function. Types of RL agents - **model based**: has an explicit model, may or may not have a policy or value function; **model free**: explicit value function and/or policy function, no model. **planning** - algo computes how to act given model of world. **RL** - agent doesn't know how world works, interacts with world to explicitly/implicitly learn how it works, improves its policy. **exploration vs exploitation** - as the trade-off sounds. **evaluation** - estimate/predict expected rewards from following a given policy. **control** - optimization to find the best policy.

- Part 2: MDPs can model a huge number of interesting problems and settings. **bandits** - use single state MDPs. **optimal control** - mostly about continuous state MDPs. **POMDP** - state is history. **markov process / markov chain** - memoryless random process, sequence of random states with markov property, S - finite set of states, P - transition model, no rewards, no actions. **markov reward process** - S (finite), P, R - reward function, γ - discount factor, no actions. **horizon** - number of time-steps in each episode in a process (can be a **finite** or infinite MRP). **return G_t of MRP** - discounted sum of rewards from time-step t to horizon $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...$ Note that $V(s) = E[G_t|s_t = s]$ (expectation over all paths that start at state s). **computing value of MRP empirically** - can simulate large number of episodes and average the returns. **bellman equation** - markov property of MRPs yields the following: $V(s) = R(s) + \gamma \sum_{s' \in S} P(s'|s)V(s')$, i.e. the value at a state is the sum of *the immediate reward* and *the discounted sum of future rewards*. **matrix form bellman equation for finite state MRP** - $V = R + \gamma PV$. **analytic solution for value of MRP** - $V = (I - \gamma P)^{-1}R$, can be computed in $O(n^3)$. **dynamic programming algo for computing value of an MRP** - (1) initialize $V_0(s) = 0$ for all s (2) for $k = 1$ until convergence, for all $s \in S$, $V_k(s) = R(s) + \gamma \sum_{s' \in S} P(s'|s)V_{k-1}(s')$, algo is $O(s^2)$ for each t. In finite horizon case, V_k^π is the exact value of k-horizon value of state s under policy π. In the infinite horizon case, $V_k^\pi(s)$ is an estimate of $E_\pi[r_t + \gamma V_{i-1}|s_t = s]$ **markov decision process MDP** - is a tuple: (S [finite], A - actions [finite], P, R, γ). **MDP + $\pi(a|s)$ is a MRP** - (S, R^π, P^π, γ) where $R^\pi(s) = \sum_{a \in A} \pi(a|s)R(s,a)$ and $P^\pi(s'|s) = \sum_{a \in A} \pi(a|s)P(s'|s,a)$. **modification of previous iterative algo for computing value of MDP + policy** - (1) initialize $V_0(s) = 0$ for all s (2) for $k = 1$ until convergence, for all $s \in S$, $V_k^\pi(s) = R^\pi(s) + \gamma \sum_{s' \in S} P^\pi(s'|s)V_{k-1}^\pi(s')$ – here a **bellman backup** for the policy π is applied. **MDP Control** - compute optimal policy: $\pi^*(s) = \underset{\pi}{\text{argmax}} V^\pi(s)$, the optimal policy for an MDP in an infinite horizon problem is deterministic, stationary (does not depend on time step), not necessarily unique. **policy search** - through enumeration searches $|A|^{|S|}$ deterministic policies. **state-action value of a policy** - take action a, then follow policy: $Q^\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} P^\pi(s'|s,a)V^\pi(s')$. **policy iteration PI** - (1) i=0; init $\pi_0(s)$ randomly for all states s (2) while i==0 or $|\pi_i - \pi_{i-1}| > 0$, do policy evaluation, policy improvement, i=i+1. **policy evaluation** - compute the value of π_i using iterative algo (equivalent to applying Bellman repeatedly till convergence). **policy improvement** - $\pi_{i+1}(s) = \underset{a}{\text{argmax}} Q^{\pi_i}(s,a) \forall s \in S$. We did this improvement step because we know $\max_a Q^{\pi_i}(s,a) >= V^{\pi_i}(s)$, but still that only suggests that following π_{i+1} for one action and then following π_i is better than just following π_i. However, it can be proved (by recursively plugging in the definition of Q) that π_{i+1} provides **monotonic**

improvement ($V^{\pi_{i+1}} >= V^{\pi_i}$ aka $V^{\pi_{i+1}}(s) >= V^{\pi_i}(s) \forall s \in S$) over π_i. PI can take at most $|A|^{|S|}$ iterations. **bellman backup** - applied to a value function, improves it if possible: $BV(s) = \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V(s')$ **value iteration VI** - considering longer and longer episodes to improve value function: (1) init $V_0(s) = 0 \forall s$ (2) set k = 1 (3) loop until finite horizon, convergence: (3a) for each state s, $V_{k+1}(s) = \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V_k(s')$ (essentially doing a bellman backup $V_{k+1} = BV_k$) (3b) $\pi_{k+1}(s) = \operatorname{argmax}_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V_k(s')$. **bellman for a particular policy** - $B^\pi V(s) = R^\pi(s) + \gamma \sum_{s' \in S} P^\pi(s'|s)V(s')$, for policy evaluation repeatedly apply B^π, i.e. $V^\pi = B^\pi B^\pi ... B^\pi V$. **contraction operator** - $|OV - OV'| <= |V - V'|$. Bellman backup is a contraction on V, $|V - V'|$ is the **infinity norm**, the max difference over states $max(s)|V(s) - V'(s)|$. In VI for finite horizon k, optimal policy in general is not stationary (depends on time step). **VI vs PI** - VI: compute optimal policy for horizon=k and increment k, PI: compute infinite horizon value of policy (policy eval), use it to select a better policy (policy improvement).

- Part 3: G_t, $V^\pi(s)$, and $Q^\pi(s,a)$ can be defined with respect to taking a particular policy π. **bootstrapping** - in the dynamic programming algo for policy eval, when you use a value estimate (in cache) for the future value V_{i-1}. **Monte Carlo policy evaluation** - generate a number of trajectories (state action paths till episode terminates) following policy π, average their returns to create a value estimate for the state – doesn't need MDP dynamics/rewards?, no bootstrapping, state doesn't have to be Markov, only for episodic MDPs. **first visit MC on policy eval** - (1) after each episode i (1.1) define $G_{i,t}$ as the return from timestep t onwards in the i^{th} episode (1.2) for each state s visited in episode i, for the first time t that state s is visited in episode i, increment counter of total first visits $N(s) = N(s) + 1$, increment total return $S(s) = S(s) + G_{i,t}$, update estimate $V^\pi(s) = S(s)/N(s)$ (2) By law of large numbers, as $N(s) \Rightarrow \inf, V^\pi(s) \Rightarrow E_\pi[G_t|s_t = s]$. **every visit MC on policy eval** - replace each instance of "first" in "first visit MC" description with "every". **incremental MC on policy eval** - same as "every visit MC" except update estimate should be done in the following way: $V^\pi(s) = V^\pi(s) * \frac{N(s)-1}{N(s)} + \frac{G_{i,t}}{N(s)} = V^\pi(s) + \frac{1}{N(s)} * (G_{i,t} - V^\pi(s))$. **incremental MC on policy eval running mean** - same as "incremental MC on policy eval" except update is: $V^\pi(s) = V^\pi(s) + \alpha * (G_{i,t} - V^\pi(s))$ where alpha can be manipulated – if $\alpha > \frac{1}{N(s)}$, then you are forgetting older data. **MC off policy eval** - even though a behavior (old) policy will have different distribution of rewards across episodes, we use the behavior policy to estimate the value of the new policy – useful when we don't have history for new policy like in medical field (not like *we have an entirely new treatment for cancer* type of situation, but a situation like *how would the outcome change if we had*). **bias** - $Bias_\theta(\hat{\theta}) = E_{x|\theta}[\hat{\theta}] - \theta$ **variance** - $Var(\hat{\theta}) = E_{x|\theta}[(\hat{\theta} - E[\hat{\theta}])^2]$ **MSE** - $MSE(\hat{\theta}) = Var(\hat{\theta}) + Bias_\theta(\hat{\theta})^2$ **Importance Sampling** - you have data $x_1 ... x_n$ sampled from distribution $p(x)$ and you have $E_{x \sim p}[f(x)]$ but you want $E_{x \sim q}[f(x)]$, we can show $E_{x \sim q}[f(x)] = \int_x q(x)f(x)dx$ which turns out to be $\approx \frac{1}{N}\sum_{i=1}^N \frac{q(x_i)}{p(x_i)} f(x_i)$ Note we could always evaluate q and p at particular data points, but we originally could not find the expectation of f with respect to the q distribution which is what we wanted. **Importance Sampling for Policy Evaluation (also off policy)** - $V^{\pi_1}(s) \approx \frac{1}{N}\sum_{j=1}^N \prod_t \frac{\pi_1(a_t|s_t)}{\pi_2(a_t|s_t)} G(h_j)$ so in MC policy eval we can use the empirical average or we can use a reweighted empirical average (importance sampling) as necessary. **temporal difference learning for estimating V** - $V^\pi(s_t) = V^\pi(s_t) + \alpha * ([r_t + \gamma * V^\pi(s_{t+1})] - V^\pi(s_t))$ (note *td error* is the expression in parens that α scales). **usability of methods** - no model: MC, TD; non episodic: DP, TD; non markov: MC, converges (when tabular, alpha ¡ 1): DP, MC, TD; unbiased: MC; **tradeoffs** - TD lower variance because only 1 random decision, MC high variance no bias, TD only converges with tabular representation but MC converges even with functional representation. **MC/TD convergence** - MC converges to minimize MSE whereas TD converges to DP policy with maximum likelihood estimates for P(s'—s,a) and r(s,a). **certainty equivalence MLE MDP model estimates** - after each (s,a,r,s') tuple, recompute MLE MDP model for s,a; compute V^π using MLE MDP model; data efficient but computationally expensive to update model.
- Part 4: **model free policy iteration** - we already know how to do policy eval model free; need to modify policy eval for deterministic policies because you can't compute $Q(s,a)$ when $\pi(s) \neq a \Rightarrow$ so we need to try all (s,a) pairs; have to interleave policy eval and policy improvement $\Rightarrow$ so we need to ensure that Q estimate is good enough so policy improvement is a monotonic operator. ϵ **greedy policy** -