Reinforcement Learning:
A Primer, Multi-Task, Goal-Conditioned

CS 330
Logistics

Homework 2 due Wednesday.

Homework 3 out on Wednesday.

Project proposal due next Wednesday.
Why Reinforcement Learning?

When do you **not** need sequential decision making?

When your system is making a single isolated decision, e.g. classification, regression. When that decision does not affect future inputs or decisions.

Common applications

- robotics
- language & dialog
- autonomous driving
- business operations
- finance

(most deployed ML systems)

+ a key aspect of intelligence
The Plan

Multi-task reinforcement learning problem

Policy gradients & their multi-task/meta counterparts

Q-learning

Multi-task Q-learning

<— should be review
object classification

supervised learning

iid data

large labeled, curated dataset

well-defined notions of success

object manipulation

sequential decision making

action affects next state

how to collect data?
what are the labels?

what does success mean?
Terminology & notation

\[ o_t \quad \pi(\theta(a_t|o_t)) \quad a_t \]

- \( s_t \) – state
- \( o_t \) – observation
- \( a_t \) – action

\[ \pi(\theta(a_t|o_t)) \] – policy

\[ \pi(\theta(a_t|s_t)) \] – policy (fully observed)

\[ o_t \] – observation

\[ s_t \] – state

Slide adapted from Sergey Levine
Imitation Learning

\[ o_t \quad \pi_\theta(a_t|o_t) \quad a_t \]

Images: Bojarski et al. '16, NVIDIA

Slide adapted from Sergey Levine
Reward functions

which action is better or worse?

\( r(s, a) \): reward function
tells us which states and actions are better

\( s, a, r(s, a), \) and \( p(s'|s, a) \) define Markov decision process

high reward

low reward

Slide adapted from Sergey Levine
The goal of reinforcement learning

\[ \theta^* = \arg \max_\theta E_{(s,a) \sim p_\theta} [r(s, a)] \]

infinite horizon case

\[ \theta^* = \arg \max_\theta \sum_{t=1}^T E_{(s_t,a_t) \sim p_\theta} [r(s_t, a_t)] \]

finite horizon case

Markov property independent of \( s_{t-1} \)

Slide adapted from Sergey Levine
What is a reinforcement learning task?

**Recall:** supervised learning  
Data generating distributions, loss  
A task: \( \mathcal{T}_i \triangleq \{ p_i(x), p_i(y|x), \mathcal{L}_i \} \)

**Reinforcement learning**  
Action space \( \mathcal{S}_i \), \( \mathcal{A}_i \), state distribution \( p_i(s_1) \), initial state \( p_i(s'|s,a) \), reward \( r_i(s,a) \)}  
A task: \( \mathcal{T}_i \triangleq \{ \mathcal{S}_i, \mathcal{A}_i, p_i(s_1), p_i(s'|s,a), r_i(s,a) \} \)

A Markov decision process  
much more than the semantic meaning of task!
Examples Task Distributions

A task: \( \mathcal{T}_i \triangleq \{ \mathcal{S}_i, \mathcal{A}_i, p_i(s_1), p_i(s'|s, a), r_i(s, a) \} \)

Personalized recommendations: \( p_i(s'|s, a), r_i(s, a) \) vary across tasks

Character animation:
- across maneuvers
  \( r_i(s, a) \) vary
- across garments & initial states
  \( p_i(s_1), p_i(s'|s, a) \) vary

Multi-robot RL:
\( \mathcal{S}_i, \mathcal{A}_i, p_i(s_1), p_i(s'|s, a) \) vary
What is a reinforcement learning task?

Reinforcement learning

A task: \( \mathcal{T}_i \triangleq \{ \mathcal{S}_i, \mathcal{A}_i, p_i(s_1), p_i(s'|s, a), \mathcal{R}_i \} \)

An alternative view:

A task identifier is part of the state: \( s = (\bar{s}, z_i) \)

\[ \mathcal{T}_i \triangleq \{ \mathcal{S}_i, \mathcal{A}_i, p_i(s_1), p(s'|s, a), \mathcal{R}_i \} \quad \rightarrow \quad \{ \mathcal{T}_i \} = \left\{ \bigcup \mathcal{S}_i, \bigcup \mathcal{A}_i, \frac{1}{N} \sum_i p_i(s_1), p(s'|s, a), \mathcal{R}_i \right\} \]

It can be cast as a standard Markov decision process!
The goal of **multi-task** reinforcement learning

Multi-task RL
The same as before, except:

- A task identifier is part of the state: \( s = (\bar{s}, z_i) \)
  - E.g. one-hot task ID
  - Language description
  - Desired goal state, \( z_i = s_g \) ← "goal-conditioned RL"

If it's still a standard **Markov decision process**, then, why not apply standard RL algorithms? You can! You can often do better.

What is the reward?
The same as before

Or, for goal-conditioned RL:

\[
r(s) = r(\bar{s}, s_g) = -d(\bar{s}, s_g)
\]

Distance function \( d \) examples:
- Euclidean \( \ell_2 \)
- Sparse 0/1
The Plan

Multi-task reinforcement learning problem

Policy gradients & their multi-task/meta counterparts

Q-learning

Multi-task Q-learning
The anatomy of a reinforcement learning algorithm

- **generate samples** (i.e. run the policy)
- **fit a model to estimate return**
  - fit $Q_{\phi}(s, a)$ (actor-critic, Q-learning)
  - estimate $p(s'|s, a)$ (model-based)
  - compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)
- **improve the policy**
  - $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$ (policy gradient)
  - $\pi(s) = \text{arg max} Q_{\phi}(s, a)$ (Q-learning)
  - optimize $\pi_{\theta}(a|s)$ (model-based)

**This lecture:** focus on model-free RL methods (policy gradient, Q-learning)

**11/6:** focus on model-based RL methods
Evaluating the objective

\[ \theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

\[ J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t}) \]

sum over samples from \( \pi_{\theta} \)
Direct policy differentiation

\[ \theta^* = \arg \max_{\theta} E_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

\[ J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)} [r(\tau)] = \sum_{t=1}^{T} r(s_t, a_t) \]

\[ \nabla_\theta J(\theta) = \int \nabla_\theta \pi_{\theta}(\tau) r(\tau) d\tau = \int \pi_{\theta}(\tau) \nabla_\theta \log \pi_{\theta}(\tau) r(\tau) d\tau = E_{\tau \sim \pi_{\theta}(\tau)} [\nabla_\theta \log \pi_{\theta}(\tau) r(\tau)] \]

A convenient identity:

\[ \pi_{\theta}(\tau) \nabla_\theta \log \pi_{\theta}(\tau) = \pi_{\theta}(\tau) \frac{\nabla_\theta \pi_{\theta}(\tau)}{\pi_{\theta}(\tau)} = \nabla_\theta \pi_{\theta}(\tau) \]
Direct policy differentiation

\[ \theta^* = \arg \max_\theta J(\theta) \]

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[r(\tau)] \]

\[ \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)] \]

\[ \nabla_\theta \left[ \log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t | s_t) + \log p(s_{t+1} | s_t, a_t) \right] \]

\[ \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t) \right) \left( \sum_{t=1}^{T} r(s_t, a_t) \right) \right] \]

\[ \pi_\theta(s_1, a_1, \ldots, s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t) \]

\[ \log \pi_\theta(\tau) = \log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t | s_t) + \log p(s_{t+1} | s_t, a_t) \]

Slide adapted from Sergey Levine
Evaluating the policy gradient

recall: $J(\theta) = E_{T \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t})$

$\nabla_\theta J(\theta) = E_{T \sim \pi_\theta(\tau)} \left[ \left( \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t | s_t) \right) \left( \sum_{t=1}^T r(s_t, a_t) \right) \right]$

$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left( \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \right) \left( \sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right)$

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(a_t | s_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left( \sum_t \nabla_\theta \log \pi_\theta(a^i_t | s^i_t) \right) \left( \sum_t r(s^i_t, a^i_t) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Slide adapted from Sergey Levine
Comparison to maximum likelihood

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

maximum likelihood:

\[ \nabla_{\theta} J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \right) \]

Multi-task learning algorithms can readily be applied!
What did we just do?

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i) r(\tau_i) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \]

Maximum likelihood:

\[ \nabla_\theta J_{ML}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i) \]

good stuff is made more likely

bad stuff is made less likely

simply formalizes the notion of “trial and error”!

REINFORCE algorithm:

1. sample \( \{\tau^i\} \) from \( \pi_\theta(a_t|s_t) \) (run it on the robot)
2. \( \nabla_\theta J(\theta) \approx \sum_i \left( \sum_t \nabla_\theta \log \pi_\theta(a^i_t|s^i_t) \right) \left( \sum_t r(s^i_t, a^i_t) \right) \)
3. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

Can we use policy gradients with meta-learning?
Example: MAML + policy gradient

two tasks: running backward, running forward

Example: MAML + policy gradient

There exists a representation under which RL is fast and efficient.

Example: Black-box meta-learning + policy gradient

**Experiment:** Learning to visually navigate a maze
- train on 1000 small mazes
- test on held-out small mazes and large mazes
Policy Gradients

Policy gradient: \( \nabla_\theta J(\theta) = E_{T \sim \pi_\theta(\tau)} \left[ \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left( \sum_{t=1}^{T} r(s_t, a_t) \right) \right] \)

**Pros:**
+ Simple
+ Easy to combine with existing multi-task & meta-learning algorithms

**Cons:**
- Produces a high-variance gradient
  - Can be mitigated with baselines (used by all algorithms in practice), trust regions
- Requires on-policy data
  - Cannot reuse existing experience to estimate the gradient!
  - Importance weights can help, but also high variance
The Plan

Multi-task reinforcement learning problem

Policy gradients & their multi-task/meta counterparts

Q-learning

Multi-task Q-learning
Value-Based RL: Definitions

Value function: $V_\pi(s_t) = \sum_{t'=t}^{T} \mathbb{E}_\pi [r(s_{t'}, a_{t'}) \mid s_t]$  

"how good is a state"

Q function: $Q_\pi(s_t, a_t) = \sum_{t'=t}^{T} \mathbb{E}_\pi [r(s_{t'}, a_{t'}) \mid s_t, a_t]$  

"how good is a state-action pair"

They’re related:  
$$V_\pi(s_t) = \mathbb{E}_{a_t \sim \pi(\cdot \mid s_t)} [Q_\pi(s_t, a_t)]$$

If you know $Q_\pi$, you can use it to improve $\pi$.

Set $\pi(a \mid s) \leftarrow 1$ for $a = \arg \max_\bar{a} Q_\pi(s, \bar{a})$  
New policy is at least as good as old policy.
Value-Based RL: Definitions

Value function: \( V^\pi(s_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi} \left[ r(s_{t'}, a_{t'}) \mid s_t \right] \) total reward starting from \( s \) and following \( \pi \)

"how good is a state"

Q function: \( Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi} \left[ r(s_{t'}, a_{t'}) \mid s_t, a_t \right] \) total reward starting from \( s \), taking \( a \), and then following \( \pi \)

"how good is a state-action pair"

For the optimal policy \( \pi^* \):

\[
Q^*(s_t, a_t) = \mathbb{E}_{s' \sim p(\cdot|s, a)} \left[ r(s, a) + \gamma \max_{a'} Q^*(s', a') \right]
\]

Bellman equation
Fitted Q-iteration Algorithm

full fitted Q-iteration algorithm:

1. collect dataset \( \{(s_i, a_i, s'_i, r_i)\} \) using some policy
2. set \( y_i \leftarrow r(s_i, a_i) + \gamma \max_{a'_i} Q_\phi(s'_i, a'_i) \) for \( K \times \)
3. set \( \phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(s_i, a_i) - y_i\|^2 \)

Algorithm hyperparameters

- dataset size \( N \), collection policy
- iterations \( K \)
- gradient steps \( S \)

Result: get a policy \( \pi(a | s) \) from \( \arg \max_a Q_\phi(s, a) \)

Important notes:

- We can reuse data from previous policies!
- an off-policy algorithm using replay buffers
- This is not a gradient descent algorithm!

Can be readily extended to multi-task/goal-conditioned RL

Slide adapted from Sergey Levine
The Plan

Multi-task reinforcement learning problem

Policy gradients & their multi-task/meta counterparts

Q-learning

Multi-task Q-learning
Multi-Task RL Algorithms

Policy: \( \pi_\theta(a | \bar{s}) \rightarrow \pi_\theta(a | \bar{s}, z_i) \)

Q-function: \( Q_\phi(\bar{s}, a) \rightarrow Q_\phi(\bar{s}, a, z_i) \)

Analogous to multi-task supervised learning: stratified sampling, soft/hard weight sharing, etc.

What is different about reinforcement learning?

The data distribution is controlled by the agent!

You may know what aspect(s) of the MDP are changing across tasks.

Should we share data in addition to sharing weights?

Can we leverage this knowledge?
An example

Task 1: passing

Task 2: shooting goals

What if you accidentally perform a good pass when trying to shoot a goal?

Store experience as normal.  *and*  **Relabel** experience with task 2 ID & reward and store.  
“hindsight relabeling”  "hindsight experience replay” (HER)
Goal-conditioned RL with hindsight relabeling

1. Collect data $\mathcal{D}_k = \{(s_{1:T}, a_{1:T}, s_g, r_{1:T})\}$ using some policy
2. Store data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$
3. Perform hindsight relabeling:
   a. Relabel experience in $\mathcal{D}_k$ using last state as goal:
      $\mathcal{D}'_k = \{(s_{1:T}, a_{1:T}, s_T, r'_{1:T})\}$ where $r'_t = -d(s_t, s_T)$
   b. Store relabeled data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'_k$
4. Update policy using replay buffer $\mathcal{D}$

Result: exploration challenges alleviated

$K$aelbling. Learning to Achieve Goals. IJCAI ’93
Multi-task RL with relabeling

1. Collect data $\mathcal{D}_k = \{(s_{1:T}, a_{1:T}, z_i, r_{1:T})\}$ using some policy
2. Store data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$
3. Perform hindsight relabeling:
   a. Relabel experience in $\mathcal{D}_k$ for task $\mathcal{T}_j$:
      $$\mathcal{D}_k' = \{(s_{1:T}, a_{1:T}, z_j, r'_{1:T})\} \text{ where } r'_t = r_j(s_t)$$
   b. Store relabeled data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k'$
4. Update policy using replay buffer $\mathcal{D}$

When can we apply relabeling?
- reward function form is known, evaluable
- dynamics consistent across goals/tasks
- using an off-policy algorithm*

---

Kaelbling. Learning to Achieve Goals. IJCAI ’93
Hindsight relabeling for goal-conditioned RL

Example: goal-conditioned RL, simulated robot manipulation

Figure 2: Different tasks: pushing (top row), sliding (middle row) and pick-and-place (bottom row). The red ball denotes the goal position.
Time Permitting: What about image observations?

Recall: need a distance function between current and goal state!

\[ r'_t = - d(s_t, s_T) \]

Use binary 0/1 reward? Sparse, but accurate.

Random, unlabeled interaction is optimal under the 0/1 reward of reaching the last state.
Can we use this insight for **better learning**?

If the data is **optimal**, can we use supervised imitation learning?

1. Collect data $\mathcal{D}_k = \{(s_{1:T}, a_{1:T})\}$ using some policy

2. Perform **hindsight relabeling**:
   a. Relabel experience in $\mathcal{D}_k$ using last state as goal:
   
   $$\mathcal{D}_k' = \{(s_{1:T}, a_{1:T}, s_T, r'_{1:T})\} \text{ where } r'_t = -d(s_t, s_T)$$
   b. Store relabeled data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k'$

3. Update policy using **supervised imitation** on replay buffer $\mathcal{D}$
Collect data from "human play", perform goal-conditioned imitation.

Lynch, Khansari, Xiao, Kumar, Tompson, Levine, Sermanet. Learning Latent Plans from Play. '19
Can we use this insight to learn a better goal representation?

Which representation, when used as a reward function, will cause a planner to choose the observed actions?

1. Collect random, unlabeled interaction data: \( \{(s_1, a_1, \ldots, a_{t-1}, s_t)\} \)
2. Train a latent state representation \( s \mapsto x \) & latent state model \( f(x' | x, a) \) s.t. if we plan a sequence of actions w.r.t. goal state \( s_t \), we recover the observed action sequence.
3. Throw away latent space model, return goal representation \( x \).

“distributional planning networks”

Srinivas, Jabri, Abbeel, Levine Finn. Universal Planning Networks. ICML ’18
Yu, Shevchuk, Sadigh, Finn. Unsupervised Visuomotor Control through Distributional Planning Networks. RSS ’19
Evaluate metrics on achieving variety of goal images

reaching, rope manipulation, pushing

Compare:
- metric from DPN (ours)
- pixel distance
- distance in VAE latent space
- distance in inverse model latent space

Yu, Shevchuk, Sadigh, Finn. Unsupervised Visuomotor Control through Distributional Planning Networks. RSS ‘19
Evaluate metrics on achieving variety of goal images
The Plan

Multi-task reinforcement learning problem

Policy gradients & their multi-task/meta counterparts

Q-learning

Multi-task Q-learning

How data can be shared across tasks.
Many Remaining Questions: The Next Two Weeks

Can we use **auxiliary tasks** to accelerate learning?

What about **hierarchies** of tasks?

Can we learn **exploration strategies** across tasks?

What **do** meta-RL algorithms learn?

**Wednesday paper presentations**
Auxiliary tasks & state representation learning

**Monday paper presentations**
Hierarchical reinforcement learning

**Next Wednesday:**
Meta-Reinforcement Learning
*(Kate Rakelly guest lecture)*

**Monday 11/4:**
Emergent Phenomenon
Additional RL Resources

**Stanford CS234:** Reinforcement Learning
**UCL Course from David Silver:** Reinforcement Learning
**Berkeley CS285:** Deep Reinforcement Learning

Reminders

Homework 2 due **Wednesday**.

Homework 3 out on **Wednesday**.

Project proposal due **next Wednesday**.