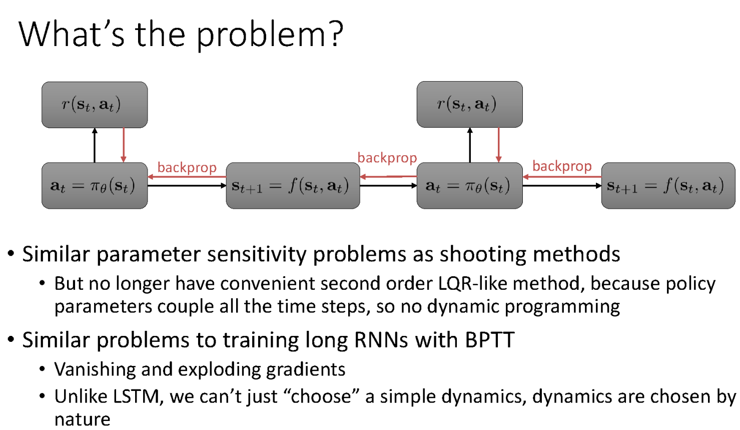
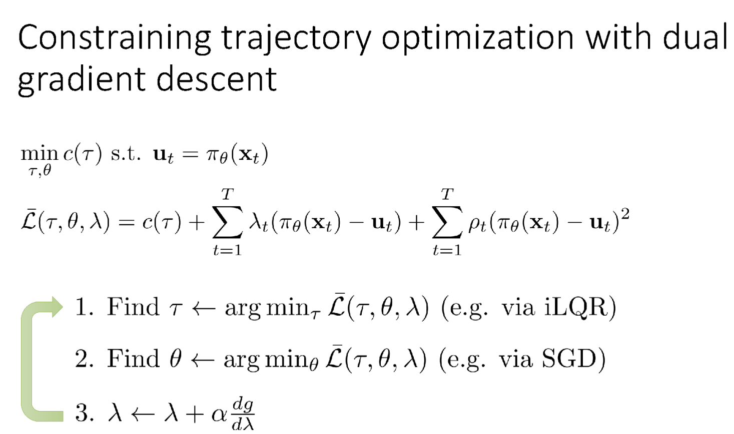
Lecture #8: Model-Based RL and Policy Learning

Nick, Nan and Halder

1 Introduction

Problems for backpropagating directly into policy





Deterministic case

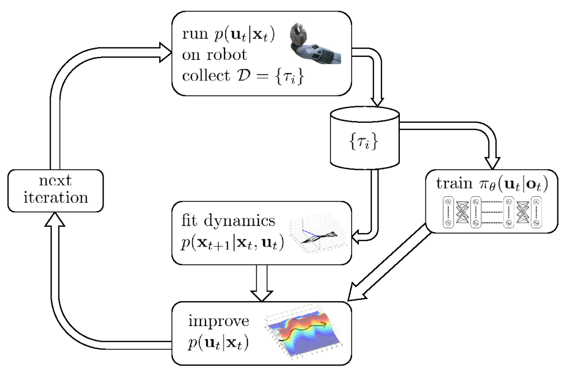
$$\min_{\tau,\theta} c(\tau) \text{ s.t. } \mathbf{u}_t = \pi_\theta(\mathbf{x}_t)$$

$$\bar{\mathcal{L}}(\tau,\theta,\lambda) = c(\tau) + \underbrace{\sum_{t=1}^{T} \lambda_t(\pi_\theta(\mathbf{x}_t) - \mathbf{u}_t) + \sum_{t=1}^{T} \rho_t(\pi_\theta(\mathbf{x}_t) - \mathbf{u}_t)^2}_{\tilde{c}(\tau)}$$

1. Optimize τ with respect to surrogate $\tilde{c}(\tau)$
2. Optimize θ with respect to supervised objective
3. Increment or modify dual variables λ

GPS

Stochastic (Gaussian) GPS with local models



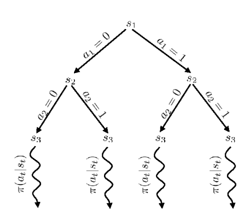
DAgger

Imitating optimal control with DAgger

Deep Learning for Real-Time Atari Game Play
Using Offline Monte-Carlo Tree Search Planning

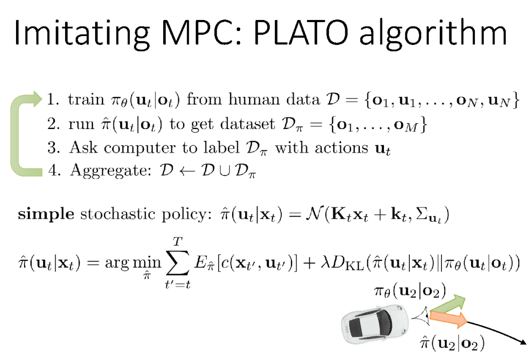
every N steps

1. from current state s_t, run MCTS to get $a_t, a_{t+1}, \ldots$
2. add (s_t, a_t) to dataset $\mathcal{D}$
3. execute action $a_t \sim \pi(a_t|s_t)$ (*not* MCTS action!)
4. update the policy by training on $\mathcal{D}$

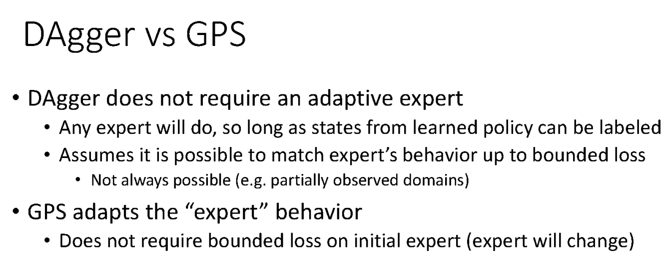


PLATO

Dagger does not care about how the actions are generated, it needs to make sure that actions are optimal with respect to the real reward function



DAgger vs GPS



Why imitate?

- It combines supervised learning and control and planning, which are stable and reliable to use
- Input is o_t instead of x_t for handling real observation
- get rid of numerical instability

Why imitate?

- Relatively stable and easy to use
  - Supervised learning works very well
  - Control/planning (usually) works very well
  - The combination of the two (usually) works very well
- Input remapping trick: can exploit availability of additional information at training time to learn policy from raw observations
- Overcomes optimization challenges of backpropagating into policy directly
- Usually sample-efficient and viable for real physical systems

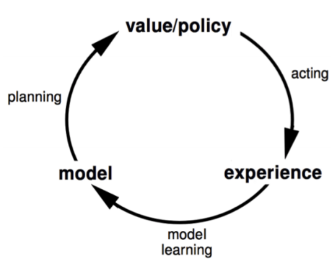
Dyna Algorithm

Dyna

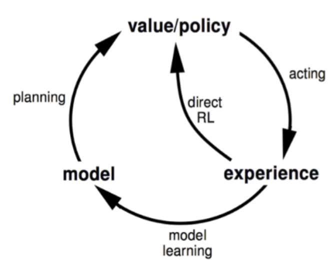
online Q-learning algorithm that performs model-free RL with a model

1. given state s, pick action a using exploration policy
2. observe s' and r, to get transition (s, a, s', r)
3. update model $\hat{p}(s'|s, a)$ and $\hat{r}(s, a)$ using (s, a, s')
4. Q-update: $Q(s, a) \leftarrow Q(s, a) + \alpha E_{s',r}[r + \max_{a'} Q(s', a') - Q(s, a)]$
5. repeat K times:
   6. sample $(s, a) \sim \mathcal{B}$ from buffer of past states and actions
   7. Q-update: $Q(s, a) \leftarrow Q(s, a) + \alpha E_{s',r}[r + \max_{a'} Q(s', a') - Q(s, a)]$

Comparison: Model-Based RL VS Integrated Architecture (Dyna)



Model-based RL

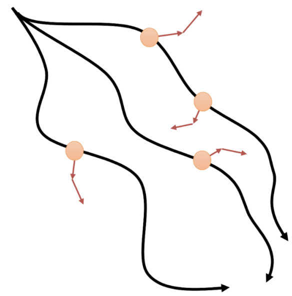


Dyna

Figures are taken from Richard Sutton's book: Reinforcement Learning: An Introduction

General "Dyna-style" model-based RL recipe

1. collect some data, consisting of transitions (s, a, s', r)
2. learn model $\hat{p}(s'|s, a)$ (and optionally, $\hat{r}(s, a)$)
3. repeat K times:
   4. sample $s \sim \mathcal{B}$ from buffer
   5. choose action a (from $\mathcal{B}$, from π, or random)
   6. simulate $s' \sim \hat{p}(s'|s, a)$ (and $r = \hat{r}(s, a)$)
   7. train on (s, a, s', r) with model-free RL
   8. (optional) take N more model-based steps



+ only requires short (as few as one step) rollouts from model
+ still sees diverse states

References

- https://dl.acm.org/citation.cfm?id=122377
- https://medium.com/@ranko.mosic/online-planning-agent-dyna-q-algorithm-and-dyna-maze-example-sutton-and-barto-2016-7ad84a6dc52b
- https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a-html/node29.html

2 Summary

Model-based RL algorithms summary

THIS WILL BE ON HW4!

- Learn model and plan (without policy)
  - Iteratively collect more data to overcome distribution mismatch
  - Replan every time step (MPC) to mitigate small model errors
- Learn policy
  - Backpropagate into policy (e.g., PILCO) – simple but potentially unstable
  - Imitate optimal control in a constrained optimization framework (e.g., GPS)
  - Imitate optimal control via DAgger-like process (e.g., PLATO)
  - Use model-free algorithm with a model (Dyna, etc.)

Limitations of model-based RL

- Need some kind of model
  - Not always available
  - Sometimes harder to learn than the policy
- Learning the model takes time & data
  - Sometimes expressive model classes (neural nets) are not fast
  - Sometimes fast model classes (linear models) are not expressive
- Some kind of additional assumptions
  - Linearizability/continuity
  - Ability to reset the system (for local linear models)
  - Smoothness (for GP-style global models)
  - Etc.

- Model-Free RL
  - No model
  - Learn value function (and/or policy) from real experience
- Model-Based RL (using Sample-Based Planning)
  - Learn a model from real experience
  - Plan value function (and/or policy) from simulated experience
- Dyna
  - Learn a model from real experience
  - Learn and plan value function (and/or policy) from real and simulated experience

3 Questions

1. Why quadratic loss in the second term

Deterministic case

$$\min_{\tau,\theta} c(\tau) \text{ s.t. } \mathbf{u}_t = \pi_\theta(\mathbf{x}_t)$$

$$\bar{\mathcal{L}}(\tau,\theta,\lambda) = \underbrace{c(\tau) + \sum_{t=1}^{T} \lambda_t(\pi_\theta(\mathbf{x}_t) - \mathbf{u}_t) + \sum_{t=1}^{T} \rho_t(\pi_\theta(\mathbf{x}_t) - \mathbf{u}_t)^2}_{\tilde{c}(\tau)}$$

2. Is iLQR a shooting method or a collocation method

https://people.eecs.berkeley.edu/ pabbeel/cs287-fa11/slides/NonlinearOptimizationForOptimalControl-part2.pdf