Control-Oriented Model-Based Reinforcement Learning with Implicit Differentiation

Evgenii Nikishin

with Romina Abachi, Rishabh Agarwal, Pierre-Luc Bacon



July 16, 2021

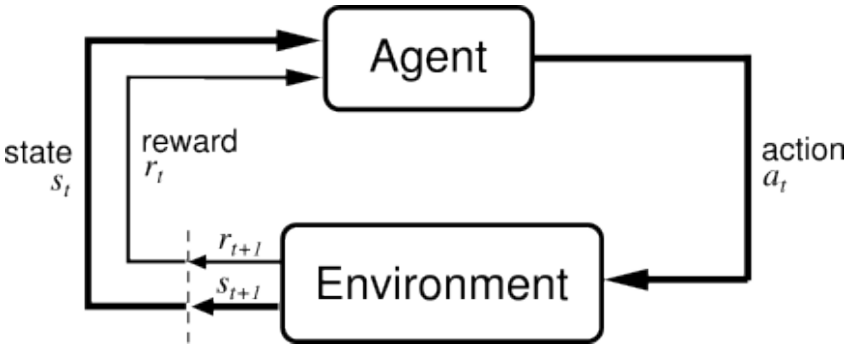
Main messages

- Maximum likelihood models in MBRL are often suboptimal
- Bi-level optimization problems could be solved using implicit differentiation

1. Intro to MBRL
2. Learning models that optimize returns
3. Background on IFT
4. Analyses and experiments
5. Discussion

RL problem statement

Markov Decision Process (MDP):



- Environment states $s_t \in \mathcal{S}$
- Agent actions $a_t \in \mathcal{A}$
- Rewards $r(s_t, a_t) \in \mathbb{R}$
- Initial state $s_0 \sim \rho_0$
- Agent policy $a_t \sim \pi(a_t|s_t)$
- State transitions $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
- Discount factor $\gamma \in [0, 1)$

Objective agent seeks to maximize:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$$

Q-learning: model-free RL

Action-value function:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t} r(s_t,a_t)|s_0 = s, a_0 = a\right]$$

$$Q^{*}(s,a) = \max_{\pi} Q_{\pi}(s,a) = Q_{\pi_*}(s,a)$$

Bellman optimality equation:

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

Simplest RL method: apply fixed point iteration to get Q^{*}.

Dyna: model-based RL

Q-learning does not assume the knowledge of $p(s'|s,a)$ and $r(s,a)$.

What if we estimate them from data?

[1][Rajeswaran et al., 2020]

Dyna: model-based RL

Q-learning does not assume the knowledge of $p(s'|s,a)$ and $r(s,a)$.

What if we estimate them from data?

The optimization problem becomes[1]:

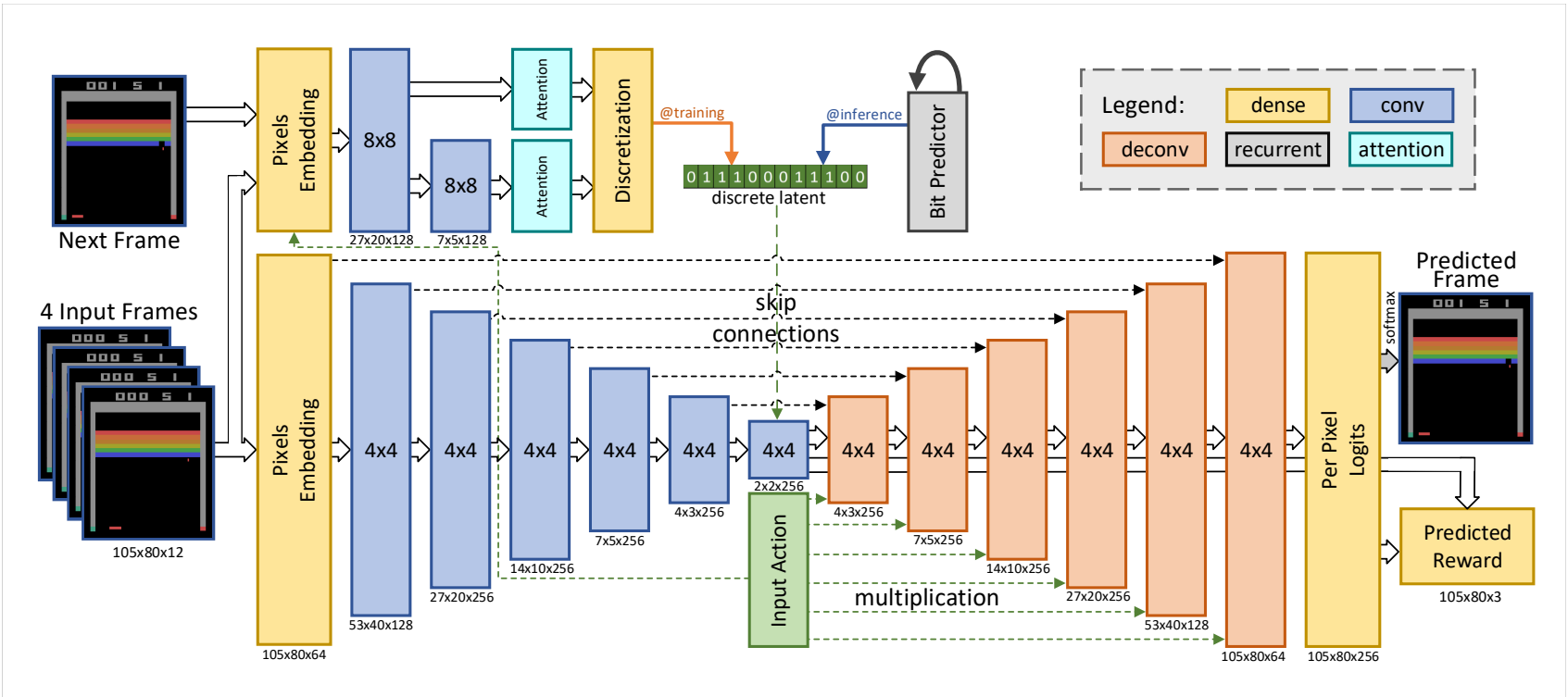
$$\max_{\pi} J(\pi, \theta), \quad \min_{\theta} \ell(\pi, \theta),$$

where ℓ is typically the negative log-likelihood.

[1] [Rajeswaran et al., 2020]

Discussion

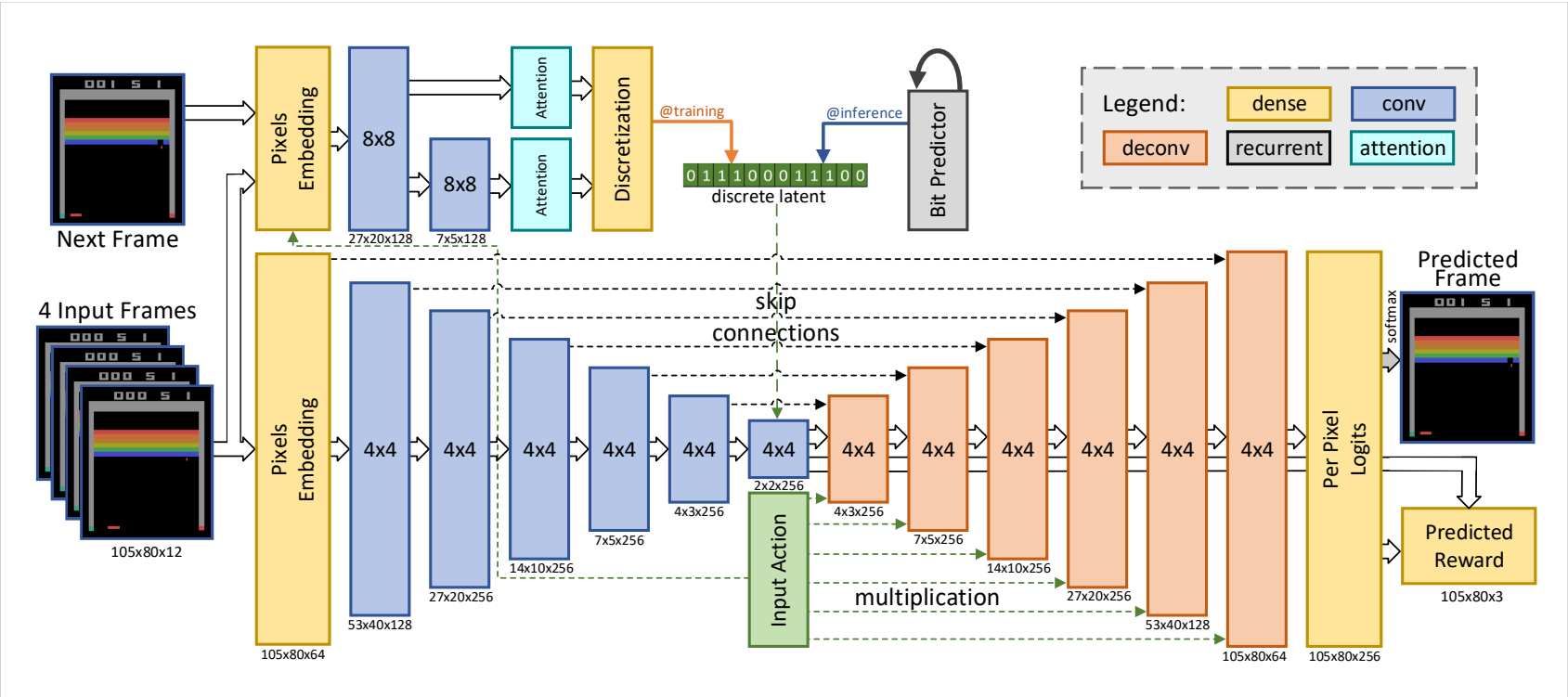
What r_θ and p_θ will focus on?[2]



[2][Kaiser et al., 2019]

Discussion

What r_θ and p_θ will focus on?[2]



Can we train r_θ and p_θ to directly optimize J?

[2][Kaiser et al., 2019]

Learning models that optimize returns

Incorporate model as a constraint:

$$\begin{aligned} &\text{maximize} \;\; J(\pi_Q) \\ &\quad \text{s.t.} \;\; Q(s,a) = B^\theta Q(s,a) \quad \forall s, a, \\ &\text{where} \;\; \pi_Q(a|s) = \frac{\exp\left(Q(s,a)\right)}{\sum_{a'} \exp\left(Q(s,a')\right)}. \end{aligned}$$

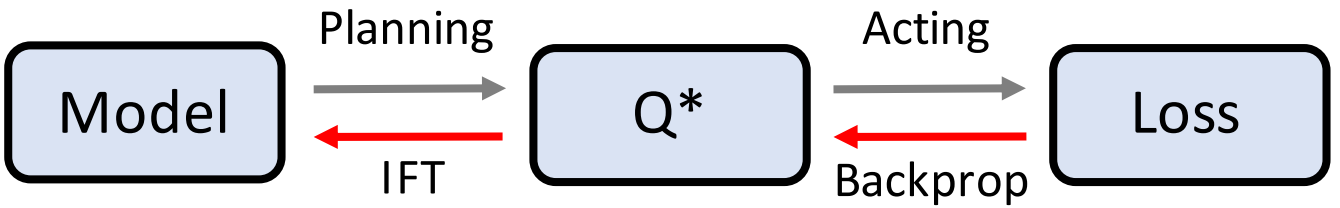
Q is used to act in the true MDP
but tries to satisfy Bellman equation given by model θ.

Optimal Model Design[3]

Suppose there exists an implicit function $\varphi(\theta) = Q^*$ such that for Q^* the constraint is satisfied:

$$Q^*(s, a) = B^\theta Q^*(s, a)$$

We have the following computation graph:



[3][Bacon et al., 2019]

Implicit function theorem

Theorem (informal)

Let $f : \Theta \times W \to W$. *Suppose* $\varphi(\theta) = w^*$ *such that* $f(\theta, w^*) = \mathbf{0}$, *then:*

$$\frac{d\varphi(\theta)}{d\theta} = -\left(\frac{\partial f(\theta, w^*)}{\partial w}\right)^{-1} \cdot \frac{\partial f(\theta, w^*)}{\partial \theta}.$$

Implicit function theorem

Theorem (informal)

Let $f : \Theta \times W \to W$. *Suppose* $\varphi(\theta) = w^*$ *such that* $f(\theta, w^*) = \mathbf{0}$, *then:*

$$\frac{d\varphi(\theta)}{d\theta} = -\left(\frac{\partial f(\theta, w^*)}{\partial w}\right)^{-1} \cdot \frac{\partial f(\theta, w^*)}{\partial \theta}.$$

1d proof:

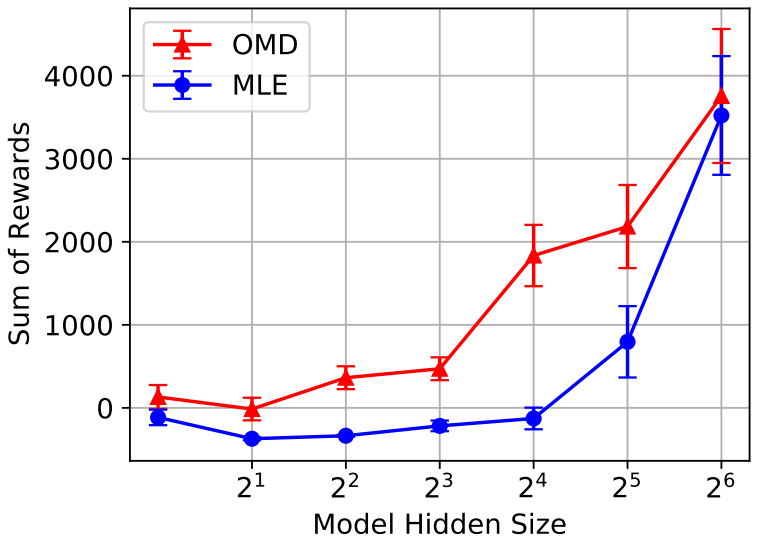
$$df = \mathbf{0}$$

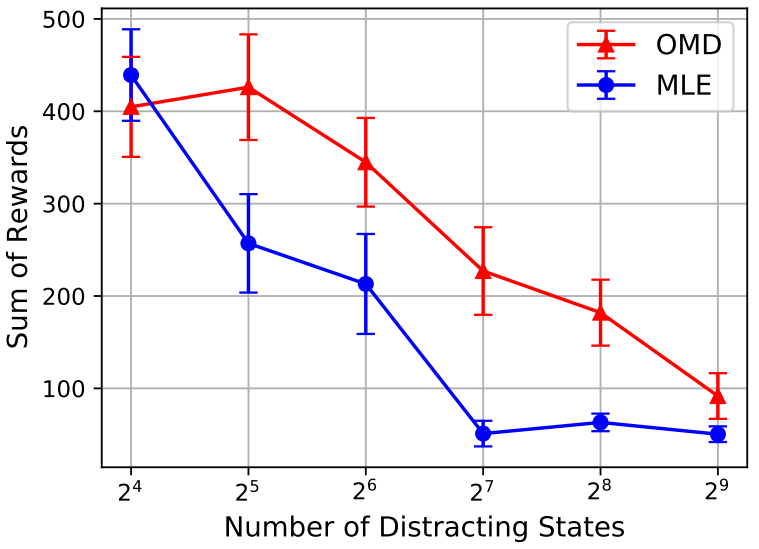
$$\frac{\partial f(\theta, w^*)}{\partial \theta} d\theta + \frac{\partial f(\theta, w^*)}{\partial w} dw^* = \mathbf{0}$$

$$\frac{\partial f(\theta, w^*)}{\partial w} dw^* = -\frac{\partial f(\theta, w^*)}{\partial \theta} d\theta$$

$$\frac{dw^*}{d\theta} = -\left(\frac{\partial f(\theta, w^*)}{\partial w}\right)^{-1} \cdot \frac{\partial f(\theta, w^*)}{\partial \theta}$$

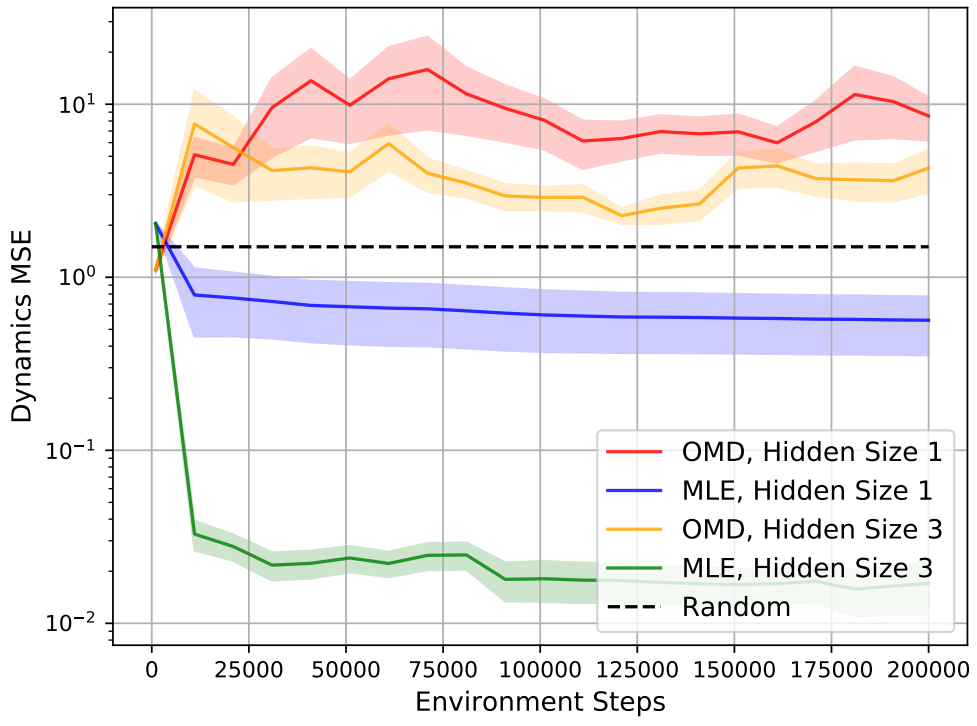
When should we prefer OMD?





Control-oriented MBRL is preferrable *under the model misspecification*.

OMD model likelihood



The learned dynamics $p_\theta(s'|s,a)$ does not resemble real next states, but produce a useful update for Q.

Further details

Theoretical analyses:

- Q^*-equivalent models[4] are OMD solutions;
- OMD enjoys a tighter Q^* approximation bound.

Surprises:

- Approximations of the IFT inverse term[5];
- K inner loop steps (even 1 is OK).

[4] [Grimm et al., 2020]
[5] [Lorraine et al., 2020]

Discussion

Summary:

- End-to-end control-oriented model learning with IFT;
- Bi-level optimization:
  get Q^* in the inner loop, optimize θ in the outer loop;
- Search over simpler models (preferable under the model misspecification).

Takeaways:

- Optimize what you really care about;
- Try IFT in your research!

Bacon, P.-L., Schäfer, F., Gehring, C., Anandkumar, A., and Brunskill, E. (2019).
A lagrangian method for inverse problems in reinforcement learning.
In *Optimization in RL workshop at NeurIPS 2019*.

Grimm, C., Barreto, A., Singh, S., and Silver, D. (2020).
The value equivalence principle for model-based reinforcement learning.
Advances in Neural Information Processing Systems, 33.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. (2019).
Model-based reinforcement learning for atari.
arXiv preprint arXiv:1903.00374.

Lorraine, J., Vicol, P., and Duvenaud, D. (2020).
Optimizing millions of hyperparameters by implicit differentiation.
In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR.

Nikishin, E., Abachi, R., Agarwal, R., and Bacon, P.-L. (2021).
Control-oriented model-based reinforcement learning with implicit differentiation.
arXiv preprint arXiv:2106.03273.

Rajeswaran, A., Mordatch, I., and Kumar, V. (2020).
A game theoretic framework for model based reinforcement learning.
In *International Conference on Machine Learning*, pages 7953–7963. PMLR.

OMD problem statement

$$\text{maximize } J(\pi_Q) \triangleq \mathbb{E}_{\pi_Q}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right]$$

$$\text{s.t. } Q(s,a) = B^\theta Q(s,a) \quad \forall s, a$$

$$\text{where } B^\theta Q(s,a) \triangleq r_\theta(s,a) + \gamma\alpha\mathbb{E}_{s'\sim p_\theta(s'|s,a)}\log\sum_{a'}\exp\frac{1}{\alpha}\left(Q(s',a')\right)$$

$$\pi_Q(a|s) = \frac{\exp\left(\frac{1}{\alpha}Q(s,a)\right)}{\sum_{a'}\exp\left(\frac{1}{\alpha}Q(s,a')\right)}$$

Computational graph and it's derivatives:

$$\theta \xrightarrow{\varphi} Q^* \xrightarrow{\exp} \pi \xrightarrow{\text{act}} J; \qquad \frac{\partial J(\theta)}{\partial\theta} = \underbrace{\frac{\partial J(\pi)}{\partial\pi}}_{\text{PG}} \cdot \underbrace{\frac{\partial\pi(Q^*)}{\partial Q^*}}_{\text{softmax}} \cdot \underbrace{\frac{\partial\varphi(\theta)}{\partial\theta}}_{\text{IFT}}$$

OMD with function approximation

$Q(s,a) = B^\theta Q(s,a) \ \ \forall s, a$ is impractical for non-tabular MDPs. Replace the constraint:

$$\text{subject to } \frac{\partial L(\theta, w)}{\partial w} \triangleq \frac{\partial}{\partial w} \mathbb{E}_{s,a}[Q_w(s,a) - B^\theta Q_w(s,a)]^2 = \mathbf{0}$$

OMD with function approximation

$Q(s,a) = B^\theta Q(s,a) \;\; \forall s,a$ is impractical for non-tabular MDPs. Replace the constraint:

$$\text{subject to } \frac{\partial L(\theta, w)}{\partial w} \triangleq \frac{\partial}{\partial w} \mathbb{E}_{s,a}[Q_w(s,a) - B^\theta Q_w(s,a)]^2 = \mathbf{0}$$

The overall expression for θ gradient:

$$\frac{\partial J(\theta)}{\partial \theta} = -\frac{\partial J(w_*)}{\partial w} \cdot \left(\frac{\partial^2 L(\theta, w^*)}{\partial w^2} \right)^{-1} \cdot \frac{\partial^2 L(\theta, w^*)}{\partial \theta \partial w}$$

Characterization of OMD solution set

Definition (Value equivalence [Grimm et al., 2020])

The models with parameters θ and θ' are Q^*-equivalent if

$$B^{\theta}Q^*(s,a) = B^{\theta'}Q^*(s,a) \;\; \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{1}$$

The set of all Q^*-equivalent models (there are many!) forms an equivalence class Θ_{Q^*}.

Proposition

If log-sum-exp temperature $\alpha \to 0$ and θ are any model parameters from the equivalence class Θ_{Q^*}, then (Q^*, θ) is an OMD solution.

Q* approximation bound

Theorem 2. (Q^* approximation error) *Let Q^* be the optimal action-value function for the true MDP. Let $\hat{Q}_{\text{OMD}}$ and $\hat{Q}_{\text{MLE}}$ be the fixed points of the Bellman optimality operators for approximate OMD and MLE models respectively. Then,*

- *If the MLE dynamics $\hat{p}$ and reward $\hat{r}$ have the bounded errors $\max_{s,a} \|p(\cdot|s,a) - \hat{p}(\cdot|s,a)\|_1 = \epsilon_p$ and $\max_{s,a} |r(s,a) - \hat{r}(s,a)| = \epsilon_r$, and the reward function is bounded $r(s,a) \in [0, r_{\max}] \quad \forall s, a$, we have*

$$\max_{s,a} \left| Q^*(s,a) - \hat{Q}_{\text{MLE}}(s,a) \right| \leq \frac{\epsilon_r}{1-\gamma} + \frac{\gamma \epsilon_p r_{\max}}{2(1-\gamma)^2};$$

- *If the Bellman optimality operator induced by the OMD model $\hat{\theta}$ has the bounded error $\max_{s,a} \left| B\hat{Q}_{\text{OMD}}(s,a) - B^{\hat{\theta}}\hat{Q}_{\text{OMD}}(s,a) \right| = \epsilon$, we have*

$$\max_{s,a} \left| Q^*(s,a) - \hat{Q}_{\text{OMD}}(s,a) \right| \leq \frac{\epsilon}{1-\gamma}.$$

Pseudo code

Algorithm 1 Model Based RL with Optimal Model Design

Input: Initial parameters w and θ, empty replay buffer $\mathcal{D}$.
repeat
  Set s to the current state, sample an action a using softmax over $Q_w(s, a)$.
  Take the action a, observe $r = r(s, a)$, $s' \sim p(s'|s, a)$, add (s, a, s', r) to $\mathcal{D}$.
  for $i = 1$ **to** K **do**
    Sample (s, a) from $\mathcal{D}$, apply the model to get $r = r_\theta(s, a)$, $s' \sim p_\theta(s'|s, a)$.
    Update Q_w parameters w to minimize $L(\theta, w)$.
  end for
  Update model parameters θ according to (13).
until the maximum number of interactions is reached

Implicit differentiation in JAX

```
@partial(custom_vjp, nondiff_argnums=(0, 3))
def root_solve(f, w0, p, solver):
  return solver(f, w0, p)

def fwd(f, w0, p, solver):
  sol = root_solve(f, w0, p, solver)
  return sol, (sol, p)

def rev(f, solver, res, g):
  sol, p = res
  _, dp_vjp = vjp(lambda y: f(y, sol), p)
  if USE_IDENTITY_INVERSE:
    vdp = dp_vjp(-g)[0]
  else:
    _, dsol_vjp = vjp(lambda w: f(p, w), sol)
    vdsoli = cg(lambda v: dsol_vjp(v)[0], g)
    vdp = dp_vjp(-vdsoli[0])[0]
  return jnp.zeros_like(sol), vdp

root_solve.defvjp(fwd, rev)
sol = root_solve(f, w0, p, solver)
# solver returns sol: f(p, sol) = 0
# sol is differentiable w.r.t. p
```