Meta Learning MDPs with Linear Transition Models

Robert Müller [1]   Aldo Pacchiano [2]

[1]Technical University of Munich
[2]Microsoft Research, NYC

Some ways to do Meta (Reinforcement) Learning:

- Learn an initialisation that is quickly adapted (Finn et al., 2017)
- Learn a recurrent model (Wang et al., 2017)
- Learn a bias (Cella et al., 2020)

Setup

Definition (MDP with linear transition core)

- MDP $\mathcal{M}$ (S, A, P, r, N, H)
- A priori given feature maps $\phi(s_t, a_t) \in \mathbb{R}^d$ and $\psi(s_{t+1}) \in \mathbb{R}^{d'}$
- There exists an unknown matrix $\boldsymbol{M}^* \in \mathbb{R}^{d \times d'}$ (the *transition core*), such that $\forall (s_t, a_t) \in S \times A, s_{t+1} \in S$:

$$P(\tilde{s}|s, a) = \phi(s, a)^T \boldsymbol{M}^* \psi(\tilde{s}).$$

Regret as Performance Metric

$$R_T(\mathcal{M}) = \sum_{n=1}^{N} \left[V^*(s_0) - \left(\sum_{h=1}^{H} r(s_{n,h}, a_{n,h}) \right) \right] .$$

Finding the transition core

Notation

$\boldsymbol{K}_\psi = \sum_{\tilde{s}\in S} \psi(\tilde{s})\psi(\tilde{s})^T$, $\boldsymbol{V}_n = \sum_{n'\leq n, h\leq H} \phi_{n',h}\phi_{n',h}^T$ and $\boldsymbol{V}_n^\lambda = \lambda \boldsymbol{I} + \boldsymbol{V}_n$.

Ridge regression problem in n-th episode for a fixed bias matrix $\boldsymbol{W} \in \mathbb{R}^{d\times d'}$

$$\boldsymbol{M}_n = \arg\min_{\boldsymbol{M}} \sum_{n',h}^{n,H} \|\psi_{n',h}^T \boldsymbol{K}_\psi^{-1} - \phi_{n',h}^T \boldsymbol{M}\|_2^2 + \lambda \|\boldsymbol{M} - \boldsymbol{W}\|_F^2 .$$

Solution of the biased ridge regression:

$$\boldsymbol{M}_n = \boldsymbol{W} + \left(\boldsymbol{V}_n^\lambda\right)^{-1} \sum_{n',h}^{n,H} \phi_{n,h} \left(\psi_{n',h}^T \boldsymbol{K}_\psi^{-1} - \phi_{n',h}^T \boldsymbol{W}\right) .$$

Feature Regularity Assumptions

1. $\|\phi(s,a)\|_2^2 \leq C_\phi \ \forall (s,a) \in S \times A,$
2. $\|\Psi K_\psi^{-1}\|_{2,\infty} \leq C'_\psi$
3. $\|\Psi^T v\|_2 \leq C_\psi \|v\|_\infty \ \forall v \in \mathbb{R}^S$
4. $\|M^*\|_F^2 \leq C_M d$

Regret Single Task BUC-MatrixRL

Optimistic radius:

$$\beta_n^{\boldsymbol{W}}(\delta) := C'_\psi \sqrt{2d' \log\left(\frac{1}{\delta}\right) + d'd \log(D)} + \sqrt{\lambda}\|\boldsymbol{W} - \boldsymbol{M}^*\|_F \qquad (1)$$

Theorem (Regret BUC-Matrix RL (Yang and Wang, 2019))

Under regularity assumptions, choosing the ellipsoid radius $\beta_n^{\boldsymbol{W}}(\delta)$ as in 1 BUC-MatrixRL abides with probability at least $1 - 1/(NH)$ after NH steps the following bound on the regret:

$$R_T(\boldsymbol{M}^*) \leq \left(C'_\psi \sqrt{d'd \log(TD)} + \sqrt{\lambda}\|\boldsymbol{W} - \boldsymbol{M}^*\|_F\right) 2C_\psi H \sqrt{C_{\phi,\lambda} T d \ln(D)}$$

Interpreting the Single Task BUC-MatrixRL regret

1. Uninformed transition core $W = \mathbf{0} \in \mathbb{R}^{d \times d'}$ recovers Yang and Wang (2019)
2. Oracle provided transition core $\boldsymbol{W} = \boldsymbol{M}^*$. Recalling the definition of D as $1 + \frac{nHC_\phi}{\lambda d}$, it is clear that the regret goes to 0 as $\lambda \to \infty$

Considered Meta RL Setting

Interaction Protocol

1. Interact with train tasks $\mathcal{T}_{train}$
2. Be evaluated on test task distribution $\mathcal{T}_{test}$

Meta Transfer Regret as Performance Metric

$$\mathrm{Mtr}_T(\mathcal{T}_{test}) = \mathbb{E}_{\mathcal{M}\sim\mathcal{T}_{test}} Regret(T, \mathcal{M}) \ .$$

Useful quantities

$$\mathrm{Var}_W(\mathcal{T}) = \mathbb{E}_{\boldsymbol{M}\sim\mathcal{T}}\left[\|\boldsymbol{M} - \boldsymbol{W}\|_F^2\right]$$
$$\mathrm{Mad}_W(\mathcal{T}) = \mathbb{E}_{\boldsymbol{M}\sim\mathcal{T}}\left[\|\boldsymbol{M} - \boldsymbol{W}\|_F\right] \ .$$

Optimal Meta Transfer Regret

Theorem (Meta Transfer Regret BUC-MatrixRL)

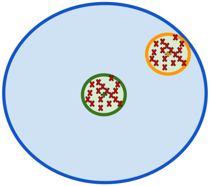
Under regularity assumptions we have with probability at least $1 - 1/(NH)$ *for a task distribution* $\mathcal{T}$ *the following* Mtr *after* T *steps per task (where we absorb constant factors into* C*):*

$$\begin{aligned}\mathrm{Mtr}_{T}(\mathcal{T}) \leq & CC_{\psi}HC'_{\psi}d\sqrt{d' TC_{\phi,\lambda}\log(TD)\ln(D)} \\ & + CC_{\psi}H\sqrt{\mathrm{Var}_{\boldsymbol{W}}\lambda TC_{\phi,\lambda}d\ln(D)}\end{aligned}$$

Interpreting MTR

1. $lim_{\lambda \to \infty} \mathrm{Mtr}_T(\mathcal{T}) \leq CC_\psi H \sqrt{\mathrm{Var}_{\boldsymbol{W}} T^2 C_\phi}$
2. Choosing the regularisation strength $\frac{1}{T\mathrm{Var}_{\boldsymbol{W}}}$ yields a $\sqrt{\log(1 + \mathrm{Var}_{\boldsymbol{W}})}$ dependence
3. Let $\lambda = \frac{1}{T\mathrm{Var}_{\boldsymbol{W}}}$ and $\boldsymbol{W} = \bar{\boldsymbol{M}}$. Then: $\lim_{Var_{\boldsymbol{W}}(\mathcal{T}) \to 0} \mathrm{Mtr}_T(\mathcal{T}) = 0$
4. Oracle BUC-MatrixRL improves against individual task learning, whenever the variance of the task distribution is much lower than its offset from the origin:

$$\mathrm{Var}_{\bar{\boldsymbol{M}}} = \mathbb{E}_{\boldsymbol{W} \sim \mathcal{T}} \|\boldsymbol{M} - \bar{\boldsymbol{M}}\|_F^2 \ll \mathbb{E}_{\boldsymbol{M} \sim \mathcal{T}} \|\boldsymbol{M}\|_F^2 = \mathrm{Var}_0 \;.$$



What to do without an oracle?

Estimating the bias

Theorem (MTR with bias estimator)

BUC-MatrixRL incurs after T interactions in G previous tasks, using a bias estimator $\hat{\boldsymbol{W}}_{G,n,h}$, step size $\lambda = \frac{1}{T Var_{\hat{w}_{G,n,h}}}$ under regularity assumptions, with probability at least $1 - 1/(NH)$ at most the following meta transfer regret:

$$\mathrm{Mtr}_T(\mathcal{M}_{G+1}) \leq C C_\psi H d C'_\psi \sqrt{C_{\phi,\lambda} d' T \log\left(T + \frac{T^3 C_\phi \left(Var_{\bar{M}} + \epsilon_{G,T}(\mathcal{T})\right)}{d}\right)}$$

A low bias estimator

Combining previous estimators with normalisation factor Z:

$$\hat{\boldsymbol{W}}_{G,n,h} = \sum_{g=1}^{G-1} \frac{T}{Z} \hat{\boldsymbol{M}}_{g,T} + \frac{nH+h}{Z} \hat{\boldsymbol{M}}_{G,n,h},$$

Resulting estimation error:

$$\sqrt{\epsilon_{G,T}(\mathcal{T})} \leq H_{\mathcal{T}}(G+1, \bar{\boldsymbol{M}}) + \max_{g \in [G]} \frac{\beta^0_{g,T}(1/NH)}{\lambda^{1/2}_{min}(V^\lambda_{g,T})}$$

A high bias estimator

- Performing global ridge regression with global features $\tilde{\boldsymbol{V}}_{G,n,h}$:

$$\hat{\boldsymbol{W}}_{G,n,h} = (\tilde{\boldsymbol{V}}^{\lambda}_{G,n,h})^{-1}\left[\sum_{g=1}^{G-1}\sum_{n,h}^{N,H}\phi_{g,n,h}\psi_{g,n,h}\boldsymbol{K}_{\psi}^{-1} + \sum_{n',h'}^{n,h}\phi_{G,n',h'}\psi_{G,n',h'}\boldsymbol{K}_{\psi}^{-1}\right]$$

- Resulting estimation error:

$$\sqrt{\epsilon_{G,T}(\mathcal{T})} \leq H_{\mathcal{T}}(G+1,\bar{\boldsymbol{M}}) + 2(G+1)\max_{g\in[G+1]}\tilde{H}(G+1,\boldsymbol{M}_g) + \underbrace{\frac{dC_M}{\lambda+\nu_{\min}} + C'_{\psi}\sqrt{\frac{2}{\lambda+\nu_{\min}}\log\left(NH + \frac{GN^2H^2C_{\phi}}{\lambda d}\right)}}_{\frac{\beta^0(1/(GNH))}{\lambda+\nu_{\min}}}$$

- Minimal singular value: $\nu_{\min} = \lambda_{\min}\left(\tilde{\boldsymbol{V}}_{G,n,h}\right)$
- $\tilde{H}(G+1,\boldsymbol{M}_g)$ is a weighted version of the estimation error $\boldsymbol{M}_g$:

$$\tilde{H}(G,\boldsymbol{M}_g) = H(g,\boldsymbol{M}_g)\sigma_{\max}\left(\boldsymbol{V}_{g,T}\tilde{\boldsymbol{V}}^{-1}_{G,N,H}\right),$$

Conclusion

- Meta Learning via learning a bias improves against single task learning for certain task families
- Tradeoff between more training tasks and alignment of them
- Assumes known features: future work to explore the impact of feature learning (Raghu et al., 2019)

References I

Cella, L., Lazaric, A., and Pontil, M. (2020). Meta-learning with stochastic linear bandits. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1360–1370. PMLR.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. (2019). Rapid learning or feature reuse? towards understanding the effectiveness of maml. *ArXiv*, abs/1909.09157.

Wang, J. X., Kurth-Nelson, Z., Soyer, H., Leibo, J. Z., Tirumala, D., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. M. (2017). Learning to reinforcement learn. *ArXiv*, abs/1611.05763.

Yang, L. F. and Wang, M. (2019). Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound.