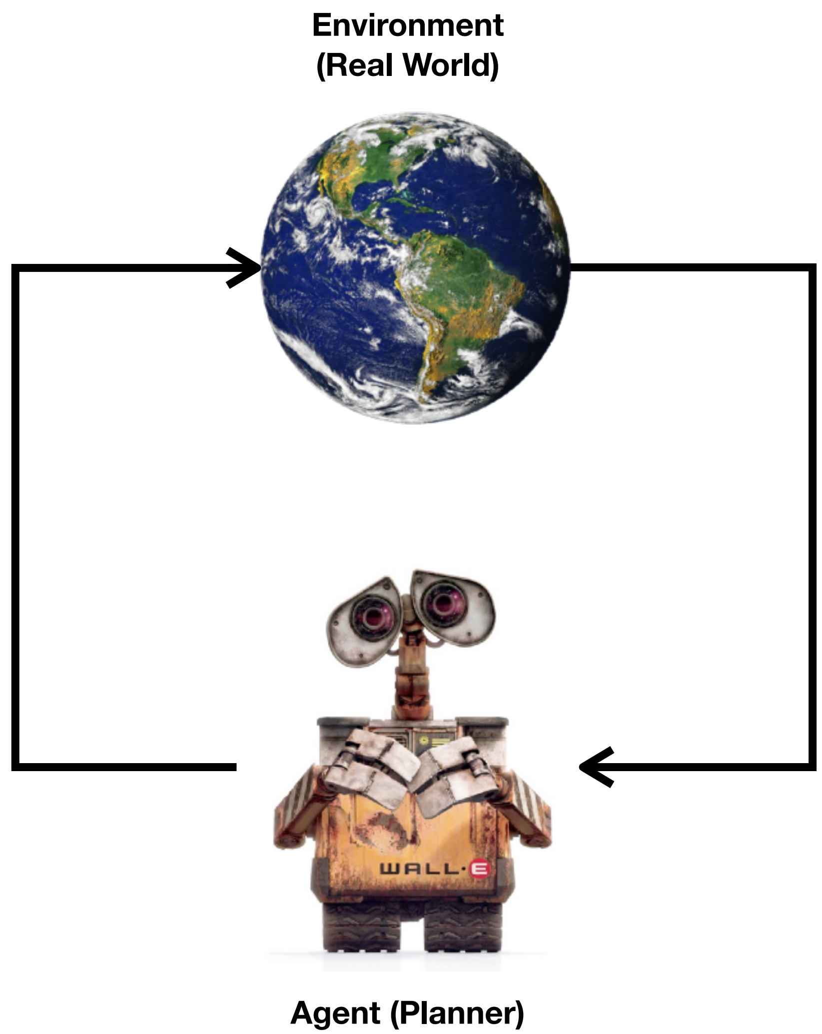
Model-based RL & Decision-Aware Model Learning

Amir-massoud Farahmand

Canada CIFAR AI Chair, Vector Institute
Department of Computer Science, University of Toronto
academic.sologen.net & @sologen

2005-2011 2011-2014 2014 2014-2018

Model-free RL Agent



$A_t \sim \pi(\cdot \,|\, X_t)$

Action

Policy

X_t, R_t

State

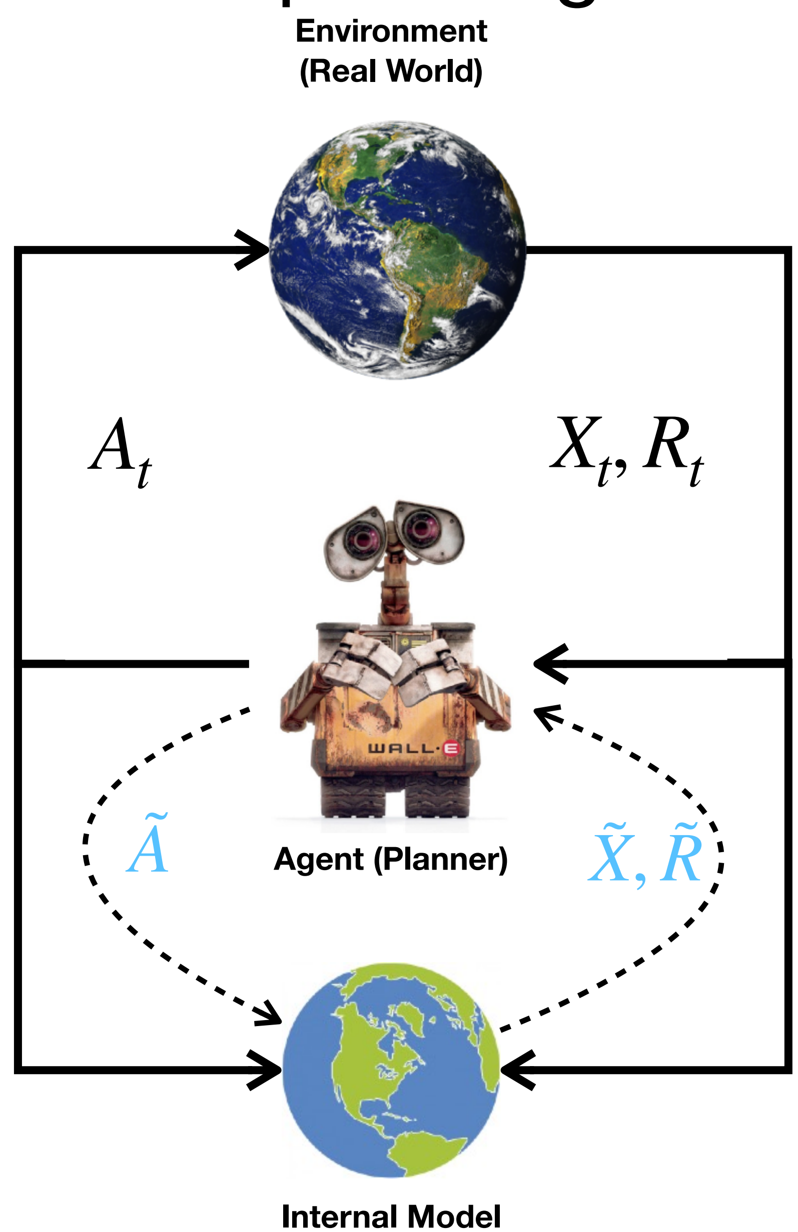
Reward

$$X_{t+1} \sim \mathcal{P}^*(\cdot|X_t, A_t)$$
$$R_t \sim \mathcal{R}(\cdot|X_t, A_t)$$

Bertsekas & Tsitsiklis, 1996 — Busoniu, Babuska, De Schutter, & Ernst, 2010 — Szepesvari, 2010 — Sutton & Barto, 2018.

Model-based RL Agent

- Learn a model of the environment
- Use the learned model for planning



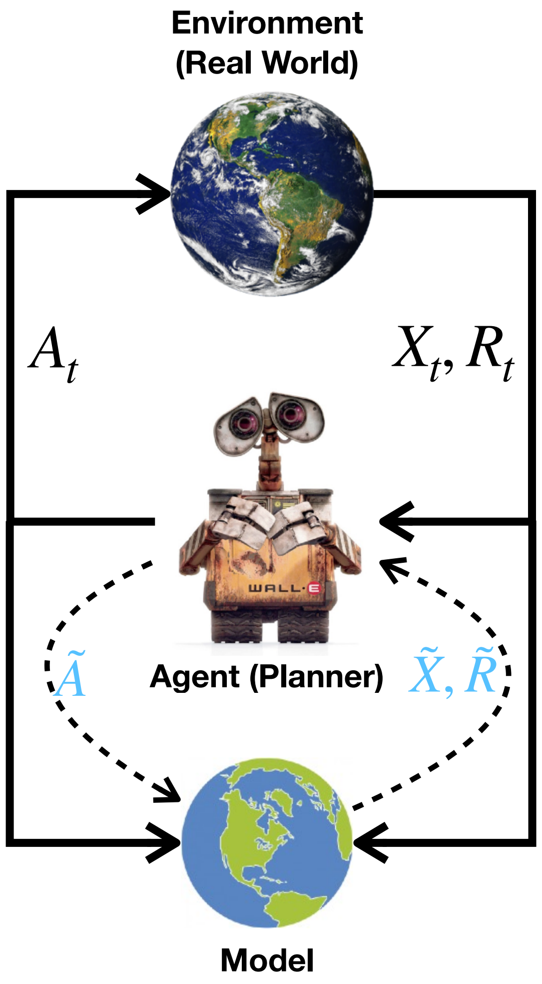
$$X_{t+1} \sim \mathcal{P}^*(\cdot|X_t, A_t)$$
$$R_t \sim \mathcal{R}(\cdot|X_t, A_t)$$

$$\tilde{X}_{i+1} \sim \hat{\mathcal{P}}(\cdot|\tilde{X}_i, \tilde{A}_t)$$
$$\tilde{R}_i \sim \hat{\mathcal{R}}(\cdot|\tilde{X}_i, \tilde{A}_i)$$

Sutton, ICML, 1990 — Peng, Williams, Adaptive Behavior, 1993 — Sutton, Szepesvári, Geramifard, and Bowling, UAI, 2008. — Parr, Li, Taylor, et al., ICML, 2008 — Deisenroth, Fox, Rasmussen, IEEE PAMI, 2015 — Levine, Finn, Darrell, Abbeel, JMLR, 2016 — Oh, Guo Lee, Lewis, Sing, NIPS, 2015 — Ha, Schmidhuber, NeurIPS, 2018.

Dyna Architecture: A Prototypical MBRL Algorithm

// MDP $(\mathcal{X}, \mathcal{A}, \mathcal{R}^*, \mathcal{P}^*)$
Draw initial state $X_1 \sim \nu_{\mathcal{X}}$
for each time step t **do**
  Take action $A_t \sim \pi(\cdot|X_t)$, receive $X_t' \sim \mathcal{P}^*(\cdot|X_t, A_t)$ and $R_t \sim \mathcal{R}^*(\cdot|X_t, A_t)$.
  Update model $\hat{\mathcal{P}}$ and $\hat{\mathcal{R}}$
  Update value function and/or policy using the new sample from the real world
  for p times **do**
    Draw simulated/imaginary sample $\tilde{X}_i \sim \tilde{\nu}_{\mathcal{X}}$
    Take action $\tilde{A}_i \sim \pi(\cdot|X_t)$, receive $\tilde{X}_i' \sim \hat{\mathcal{P}}(\cdot|\tilde{X}_i, \tilde{A}_i)$
    Update value function and/or policy using the new sample from the model
  end for
  $X_{t+1} \leftarrow X_t'$
end for



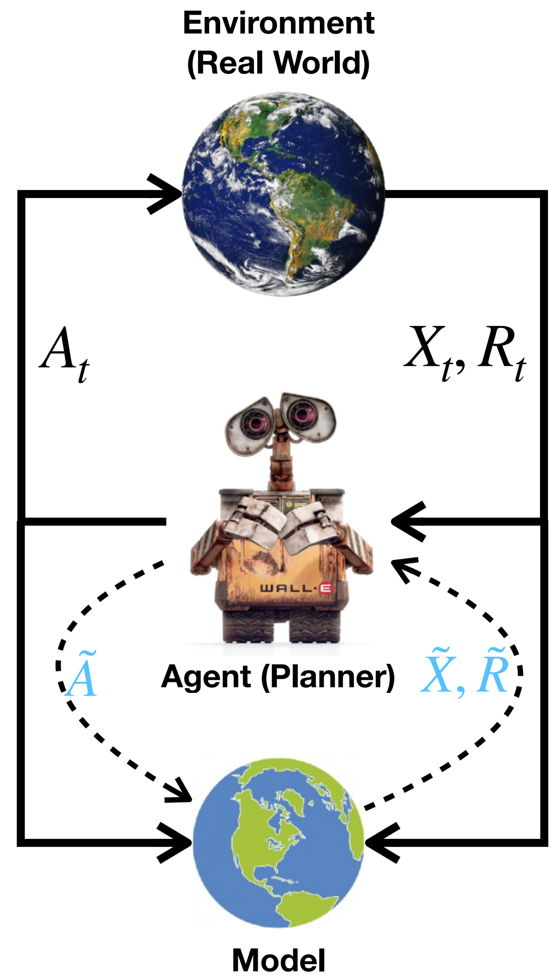
Dyna Architecture:
Finite State/Action Space

MLE

// MDP $(\mathcal{X}, \mathcal{A}, \mathcal{R}^*, \mathcal{P}^*, \gamma)$
// α: Learning rate for TD(0)
Draw initial state $X_1 \sim \nu_{\mathcal{X}}$
for each time step t **do**
  Take action $A_t \sim \pi(\cdot|X_t)$, receive $X_t' \sim \mathcal{P}^*(\cdot|X_t, A_t)$ and $R_t \sim \mathcal{R}^*(\cdot|X_t, A_t)$.
  $\hat{\mathcal{P}}(x'|x,a) \leftarrow \frac{\#\{X_i'=x' | (X_i=x, A_i=a)\}}{\#\{(X_i=x, A_i=a)\}}$
  $Q(X_t, A_t) \leftarrow Q(X_t, A_t) + \alpha \left(R_t + \gamma \sum_{a \in \mathcal{A}} \pi(a|X_t') Q(X_t', a') - Q(X_t, A_t)\right)$
  for p times **do**

TD

    Draw simulated/imaginary sample $\tilde{X}_i \sim \tilde{\nu}_{\mathcal{X}}$
    Take action $\tilde{A}_i \sim \pi(\cdot|\tilde{X}_t)$, receive $\tilde{X}_i' \sim \hat{\mathcal{P}}(\cdot|\tilde{X}_i, \tilde{A}_i)$ and $\tilde{r}_i \leftarrow \hat{r}(\tilde{X}_i, \tilde{A}_i)$.
    $Q(\tilde{X}_i, \tilde{A}_i) \leftarrow Q(\tilde{X}_i, \tilde{A}_i) + \alpha \left(\tilde{r}_i + \gamma \sum_{a \in \pi} \pi(a|\tilde{X}_i') Q(\tilde{X}_i', a') - Q(\tilde{X}_i, \tilde{A}_i)\right)$
  end for
  $X_{t+1} \leftarrow X_t'$
end for



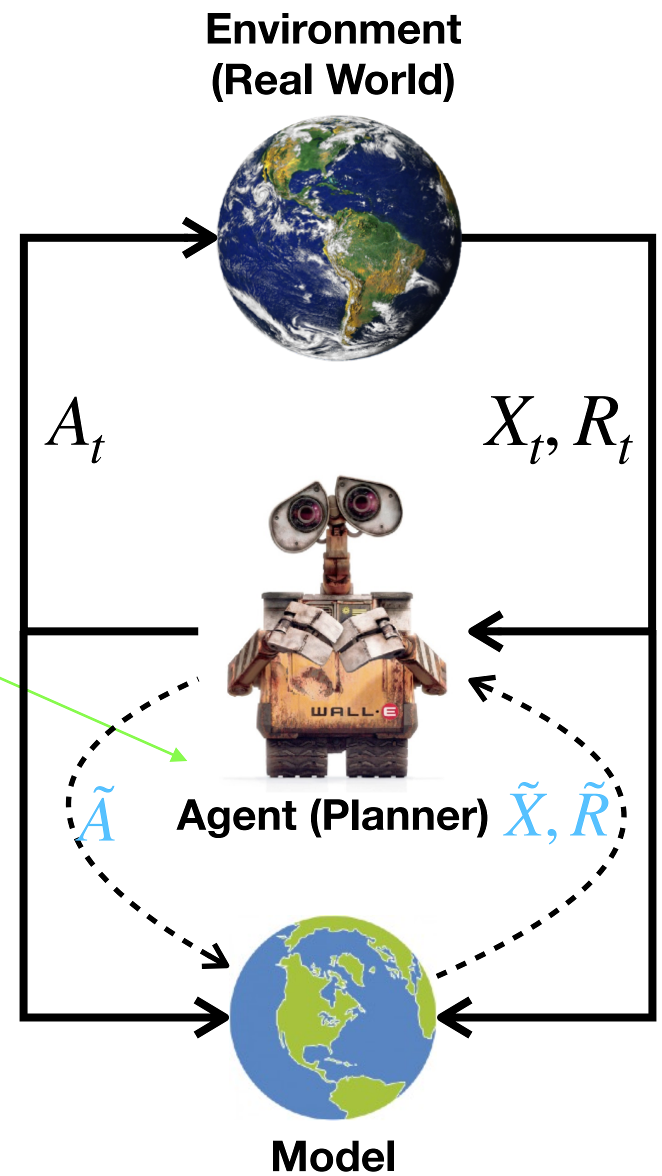
Algorithm 1 Generic Model-based Reinforcement Learning Algorithm

// MDP $(\mathcal{X}, \mathcal{A}, \mathcal{R}^*, \mathcal{P}^*, \gamma)$
// K: Number of interaction episodes
// $\mathcal{M}$: Space of transition probability kernels
// $\mathcal{G}$: Space of reward functions
Initialize a policy π_0
for $k = 0$ to $K - 1$ **do**
  Generate training set $\mathcal{D}_n^{(k)} = \{(X_i, A_i, R_i, X_i')\}_{i=1}^n$ by interacting with the true environment (potentially using π_k), i.e., $X_i' \sim \mathcal{P}^*(\cdot|X_i, A_i)$ and $R_i \sim \mathcal{R}(\cdot|X_i, A_i)$.
  $\hat{\mathcal{P}} \leftarrow \mathrm{argmin}_{\mathcal{P} \in \mathcal{M}} \mathrm{Loss}_{\mathcal{P}}(\mathcal{P}; \cup_{i=0}^k \mathcal{D}_n^{(i)})$
  $\hat{r} \leftarrow \mathrm{argmin}_{r \in \mathcal{G}} \mathrm{Loss}_{\mathcal{R}}(r; \cup_{i=0}^k \mathcal{D}_n^{(i)})$
  $\pi_{k+1} \leftarrow \mathsf{Planner}(\hat{\mathcal{P}}, \hat{\mathcal{R}})$
end for
return π_K

Issues in MBRL

Choice of Planner

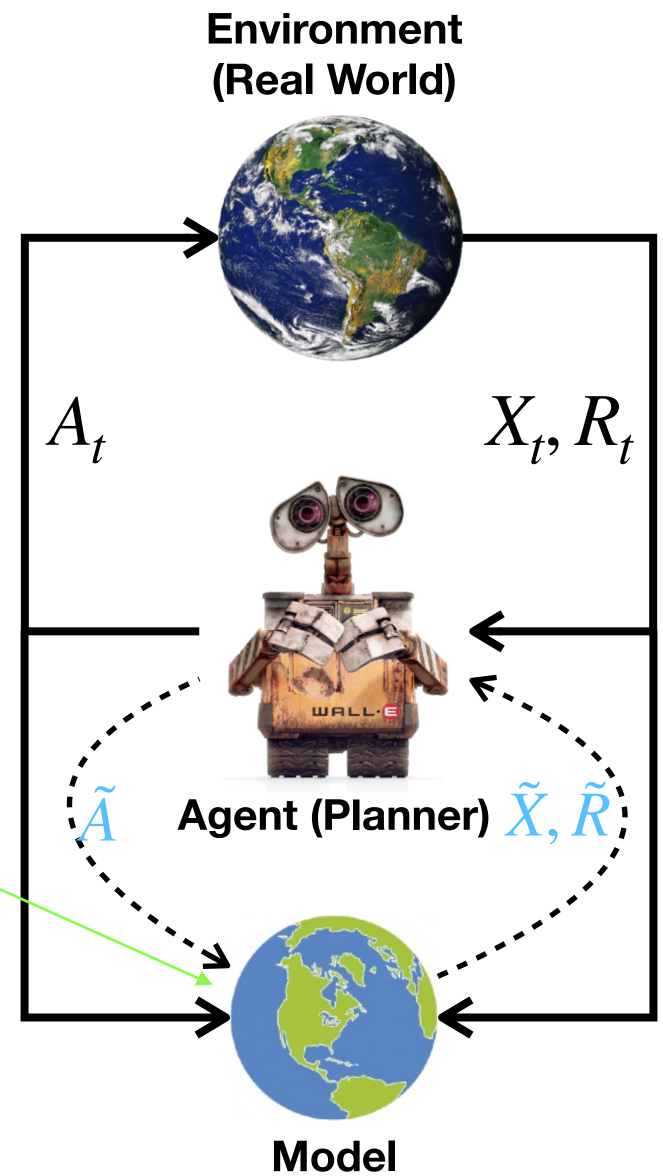
Planner
- Value-based
- Policy Search



Policy Search

- PILCO: Marc P. Deisenroth, Dieter Fox, and Carl E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," IEEE Trans. on PAMI, 2015.
- GPS: - Sergey Levine and Pieter Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," NIPS, 2014.

Model Learning



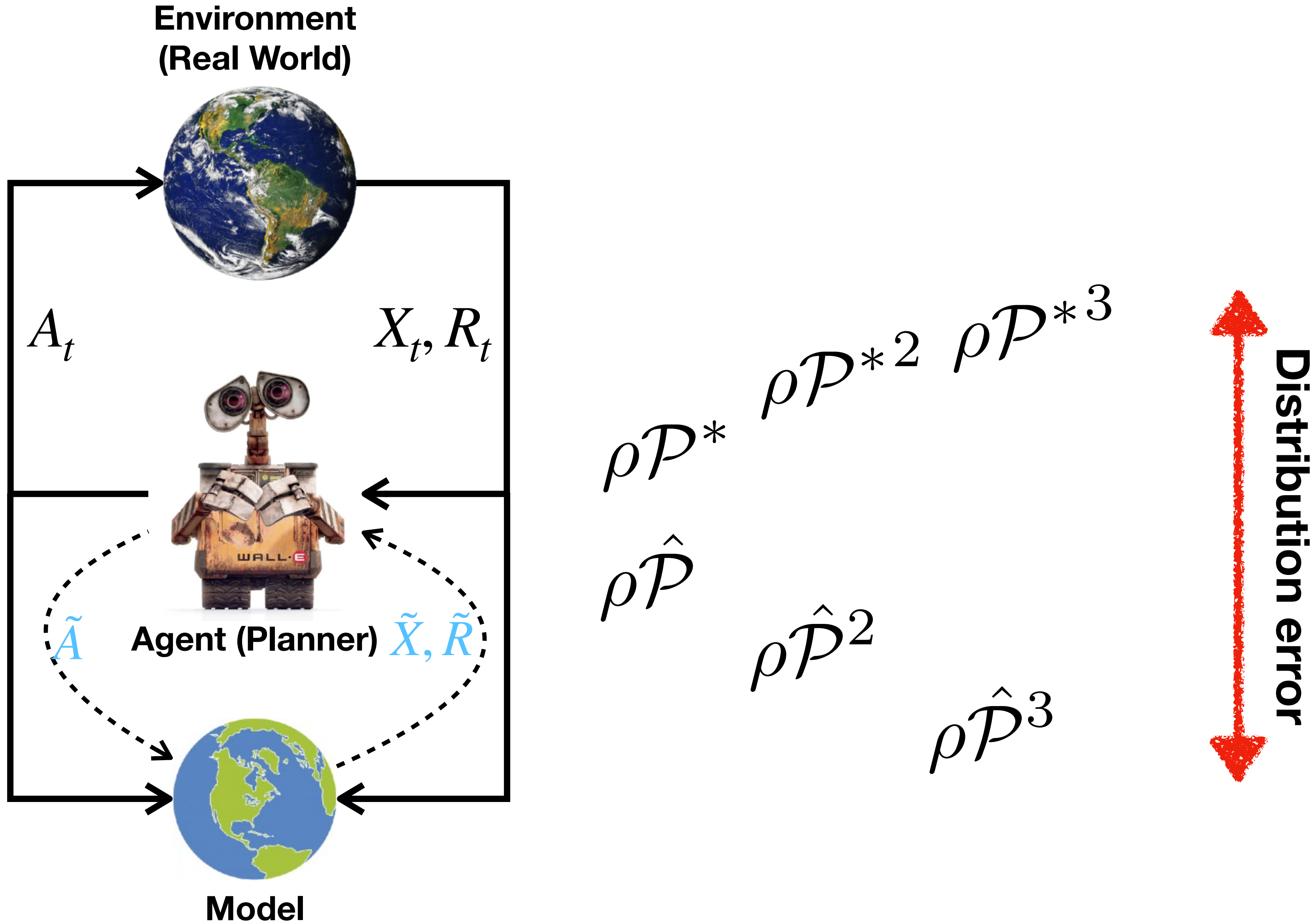
Model Learning

- MLE
- Bayesian
- Decision-Aware Model Learning

Decision-Aware Model Learning

- AMF, André M.S. Barreto, and Daniel N. Nikovski, "Value-aware model learning for reinforcement learning," AISTATS, 2017.
- David Silver, Hado van Hasselt, Matteo Hessel, et al., "The Predictron: End-to-end learning and planning," ICML, 2017.
- Junhyuk Oh, Satinder Singh, and Honglak Lee, "Value prediction network," NIPS, 2017.
- Joshua Joseph, Alborz Geramifard, John W Roberts, Jonathan P How, and Nicholas Roy, "Reinforcement learning with misspecified model classes," ICRA, 2013.

Distribution Mismatch

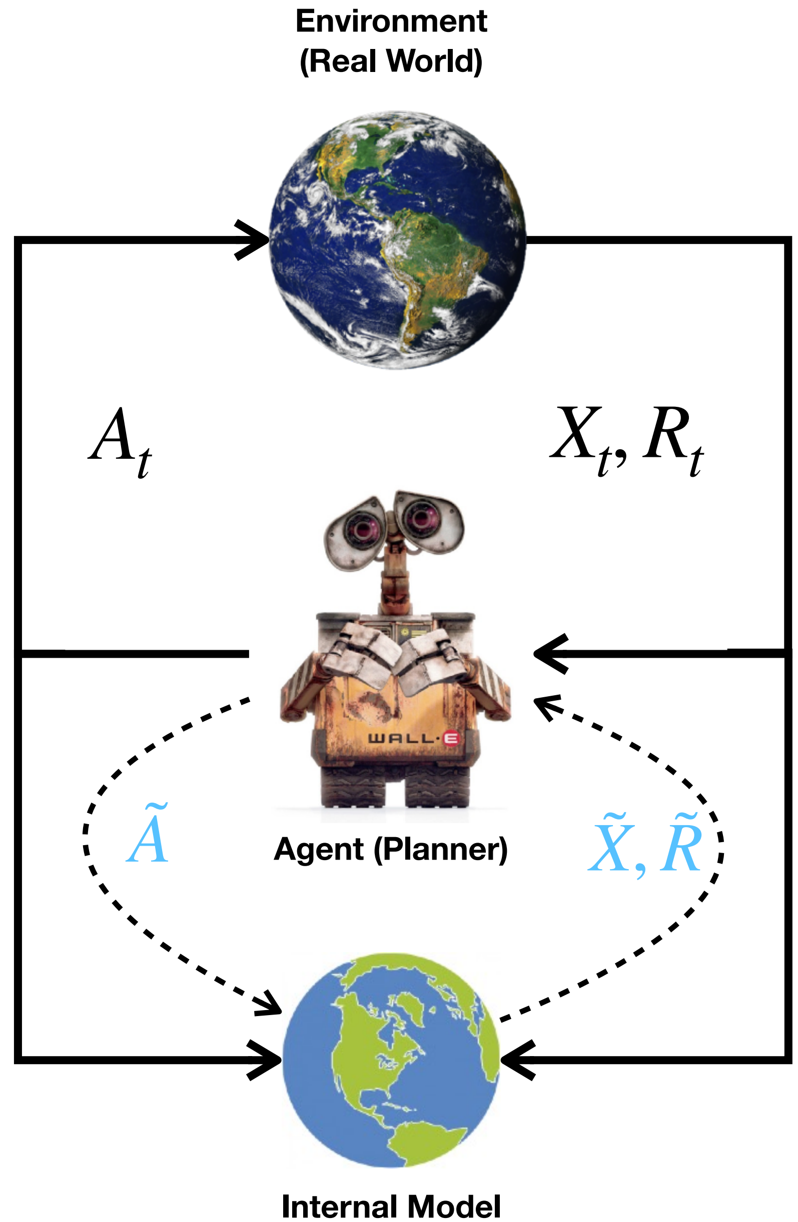


Distribution Mismatch in MBRL

- Erin Talvitie, "Self-correcting models for model-based reinforcement learning," AAAI, 2017.
- Erik Talvitie, "Model regularization for stable sample rollouts," UAI, 2014.
- Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell, "Improving multi-step prediction of learned time series models," AAAI, 2015.

Model-based RL Agent

- Learn a model of the environment
- Use the learned model for planning



$$X_{t+1} \sim \mathcal{P}^*(\cdot|X_t, A_t)$$
$$R_t \sim \mathcal{R}(\cdot|X_t, A_t)$$

$$\tilde{X}_{i+1} \sim \hat{\mathcal{P}}(\cdot|\tilde{X}_i, \tilde{A}_t)$$
$$\tilde{R}_i \sim \hat{\mathcal{R}}(\cdot|\tilde{X}_i, \tilde{A}_i)$$

Sutton, ICML, 1990 — Peng, Williams, Adaptive Behavior, 1993 — Sutton, Szepesvári, Geramifard, and Bowling, UAI, 2008. — Parr, Li, Taylor, et al., ICML, 2008 — Deisenroth, Fox, Rasmussen, IEEE PAMI, 2015 — Levine, Finn, Darrell, Abbeel, JMLR, 2016 — Oh, Guo Lee, Lewis, Sing, NIPS, 2015 — Ha, Schmidhuber, NeurIPS, 2018.

How should we learn a good model for model-based RL?

The conventional approach to model learning might be an overkill!

Conventional Approaches to Model Learning

Learn a predictive model that captures **all aspects** of the environment as much as possible.

- Maximum Likelihood Estimate (MLE)
- Bayesian Inference
- Maximum Entropy

Not all aspects are equally needed!

Images: Magnolia Avenue, St. Augustine, Florida (Jon Dawson)

Artist Robot

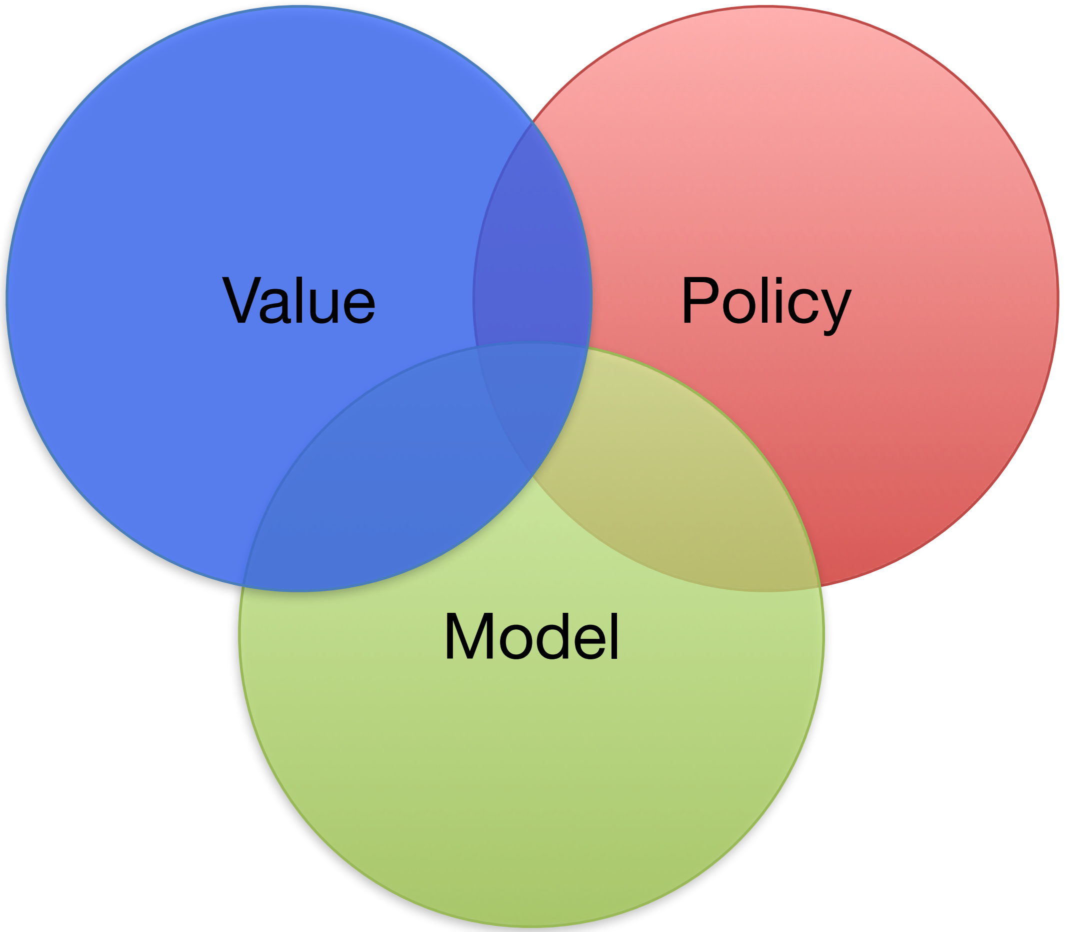
Cleaning Robot

The world might be the same, but the tasks are not!

The conventional approach to model learning might be an overkill!

Images: Artist Robot (Johan Scherft); Cleaning Robot (Mark H. Evans)

How to incorporate information about the decision problem/ task into the model learning process itself?



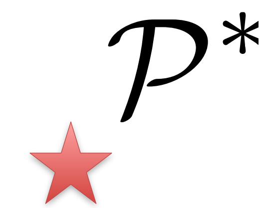
We have to pay attention to the interaction of model and the value function or policy.

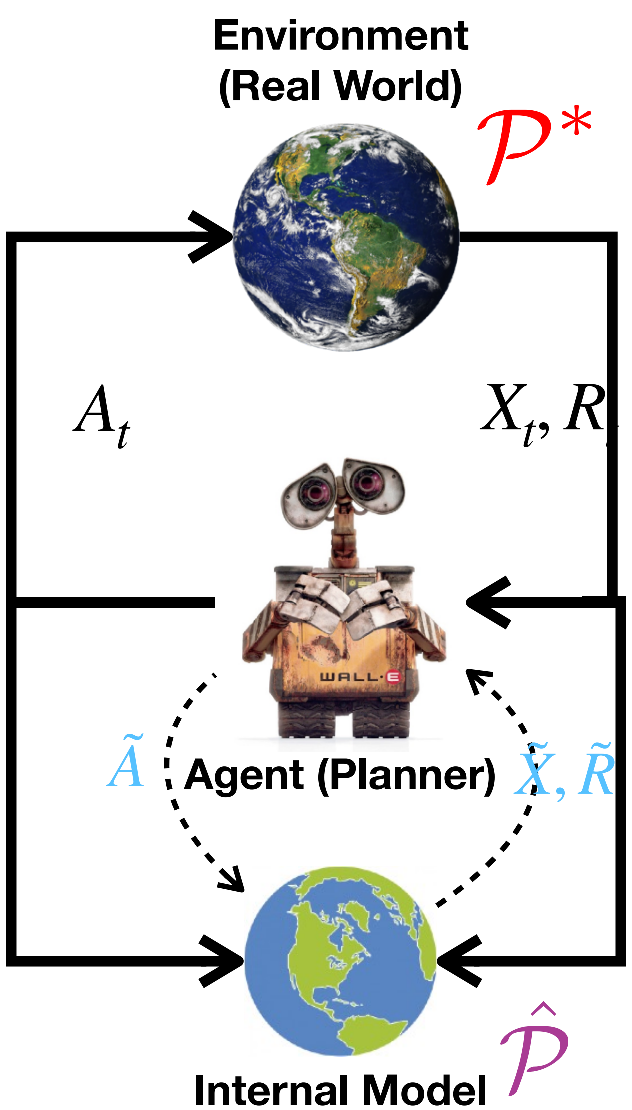
Decision-Aware Model Learning

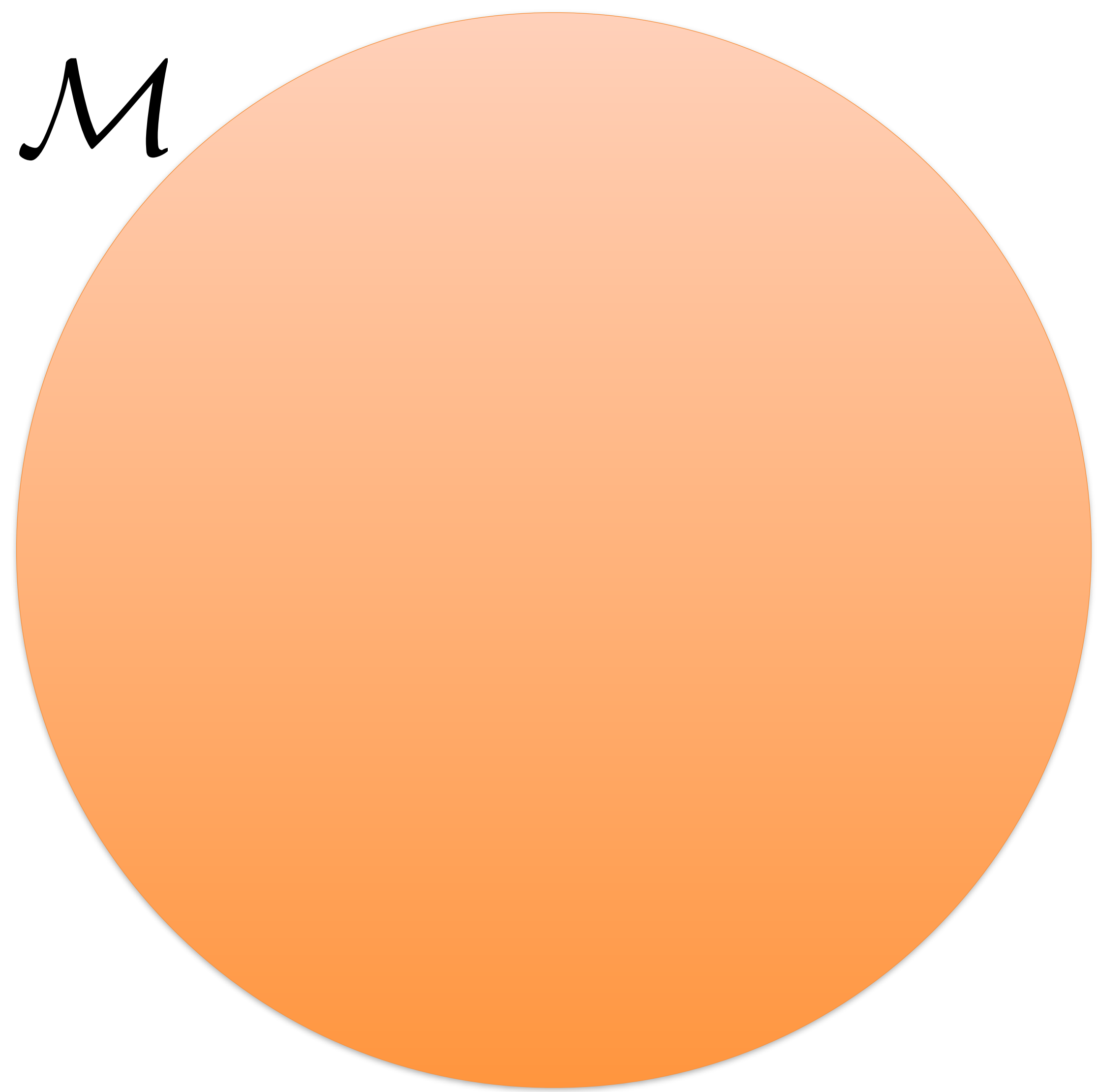
AMF, Barreto, Nikovski, “Value-Aware Loss Function for Model Learning in Reinforcement Learning,” European Workshop on Reinforcement Learning (EWRL), 2016.
AMF, Barreto, Nikovski, “Value-Aware Loss Function for Model-Based Reinforcement Learning,” Artificial Intelligence and Statistics (AISTATS), 2017.
AMF, “Iterative Value-Aware Model Learning,” Neural Information Processing Systems (NeurIPS), 2018.
Abachi, Ghavamzadeh, **AMF**, “Policy-Aware Model Learning for Policy Gradient Methods,” preprint, 2020.

Let us try to design a **decision-aware** model learning method!

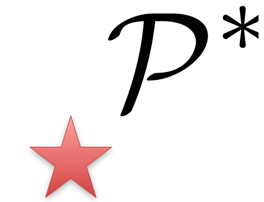
- True model of the environment: $\mathcal{P}^*$
- We are given a dataset $\mathcal{D}_n = \{(X_i, A_i, X_i')\}_{i=1}^n$ with $Z_i = (X_i, A_i) \sim \nu(\mathcal{X} \times \mathcal{A})$ and $X_i' \sim \mathcal{P}^*(\cdot|X_i, A_i)$
- Policy of the MBRL: $\pi \leftarrow \mathsf{Planner}(\hat{\mathcal{P}})$
- How to estimate a model of the environment $\hat{\mathcal{P}}$ such that π is a high-performing policy?

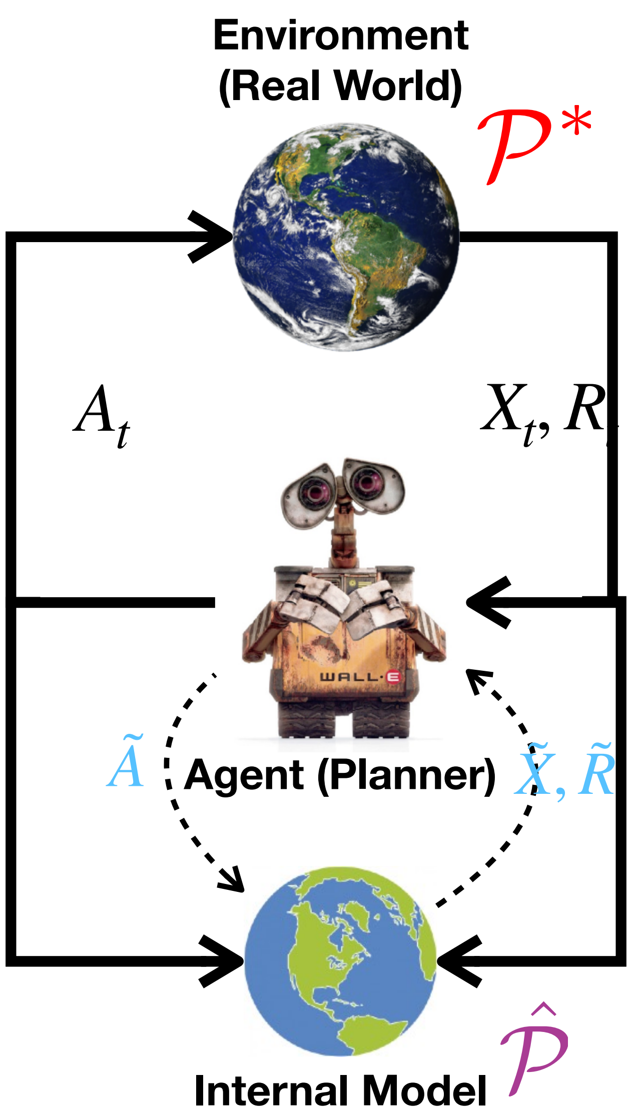


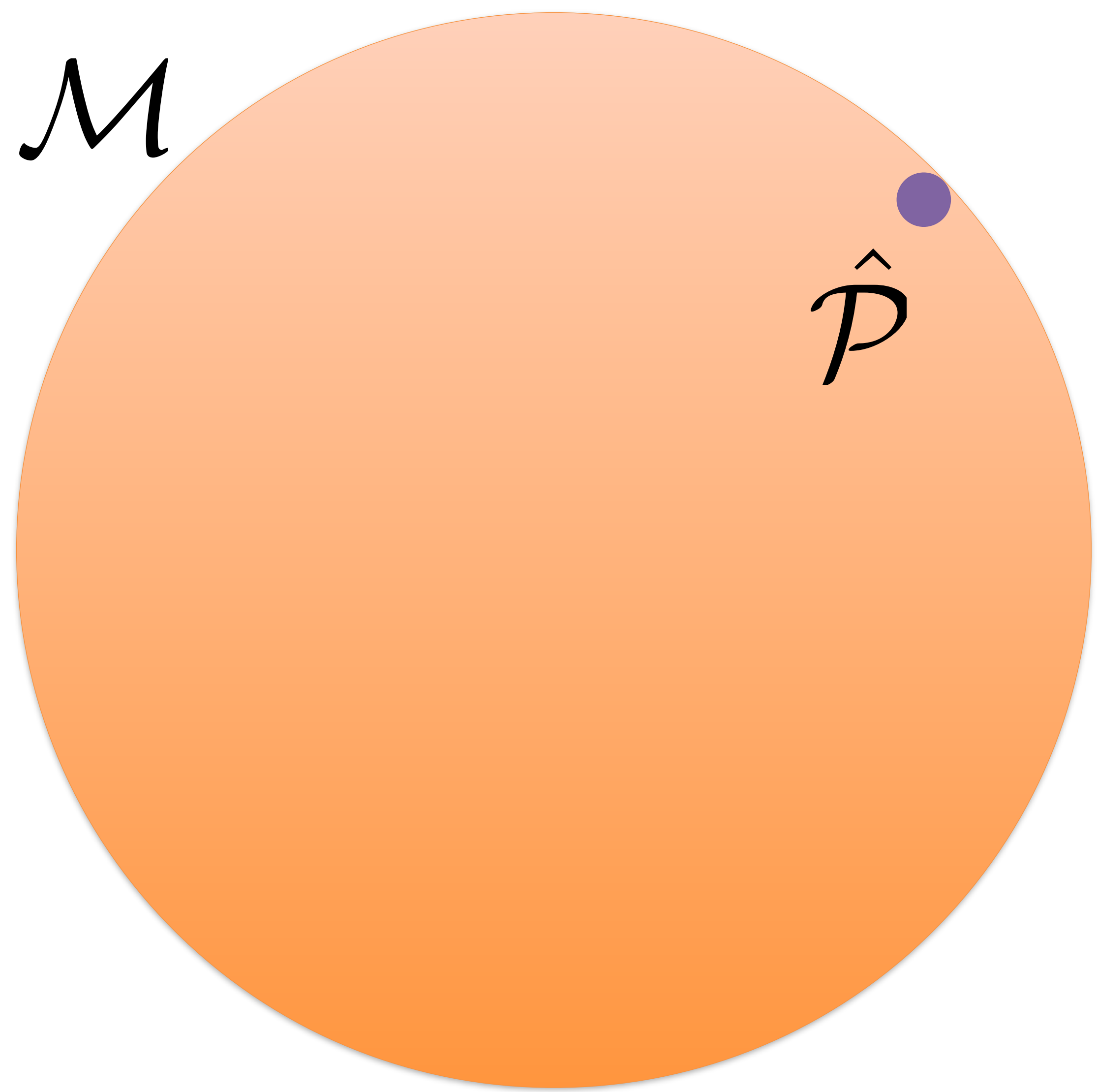




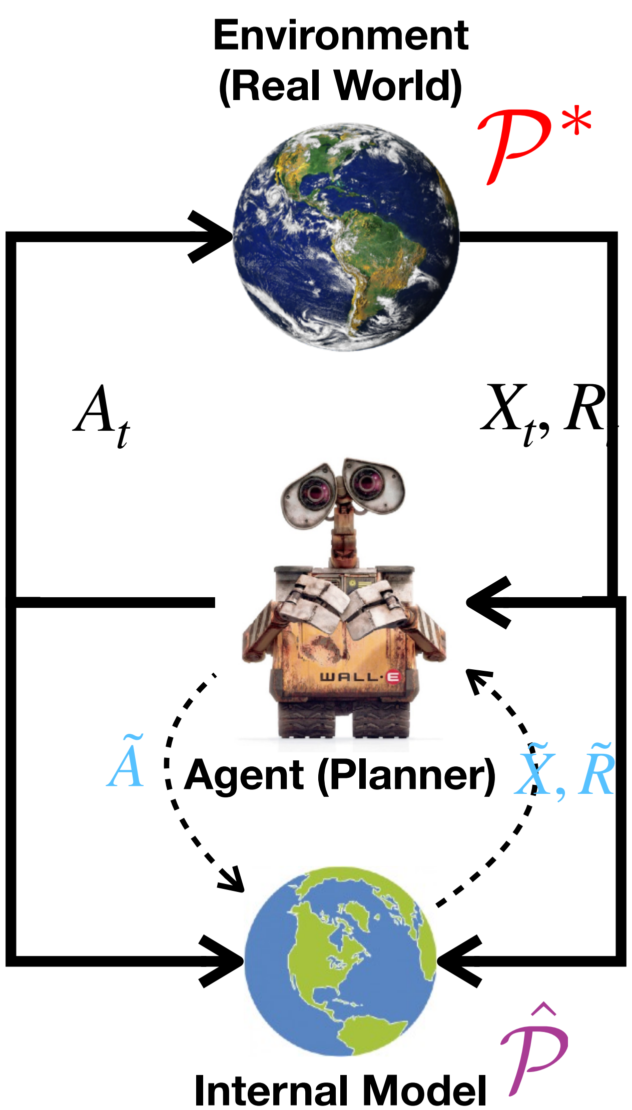
- True model of the environment: $\mathcal{P}^*$
- We are given a dataset $\mathcal{D}_n = \{(X_i, A_i, X'_i)\}_{i=1}^n$ with $Z_i = (X_i, A_i) \sim \nu(\mathcal{X} \times \mathcal{A})$ and $X'_i \sim \mathcal{P}^*(\cdot|X_i, A_i)$
- Policy of the MBRL: $\pi \leftarrow \mathsf{Planner}(\hat{\mathcal{P}})$
- How to estimate a model of the environment $\hat{\mathcal{P}}$ such that π is a high-performing policy?

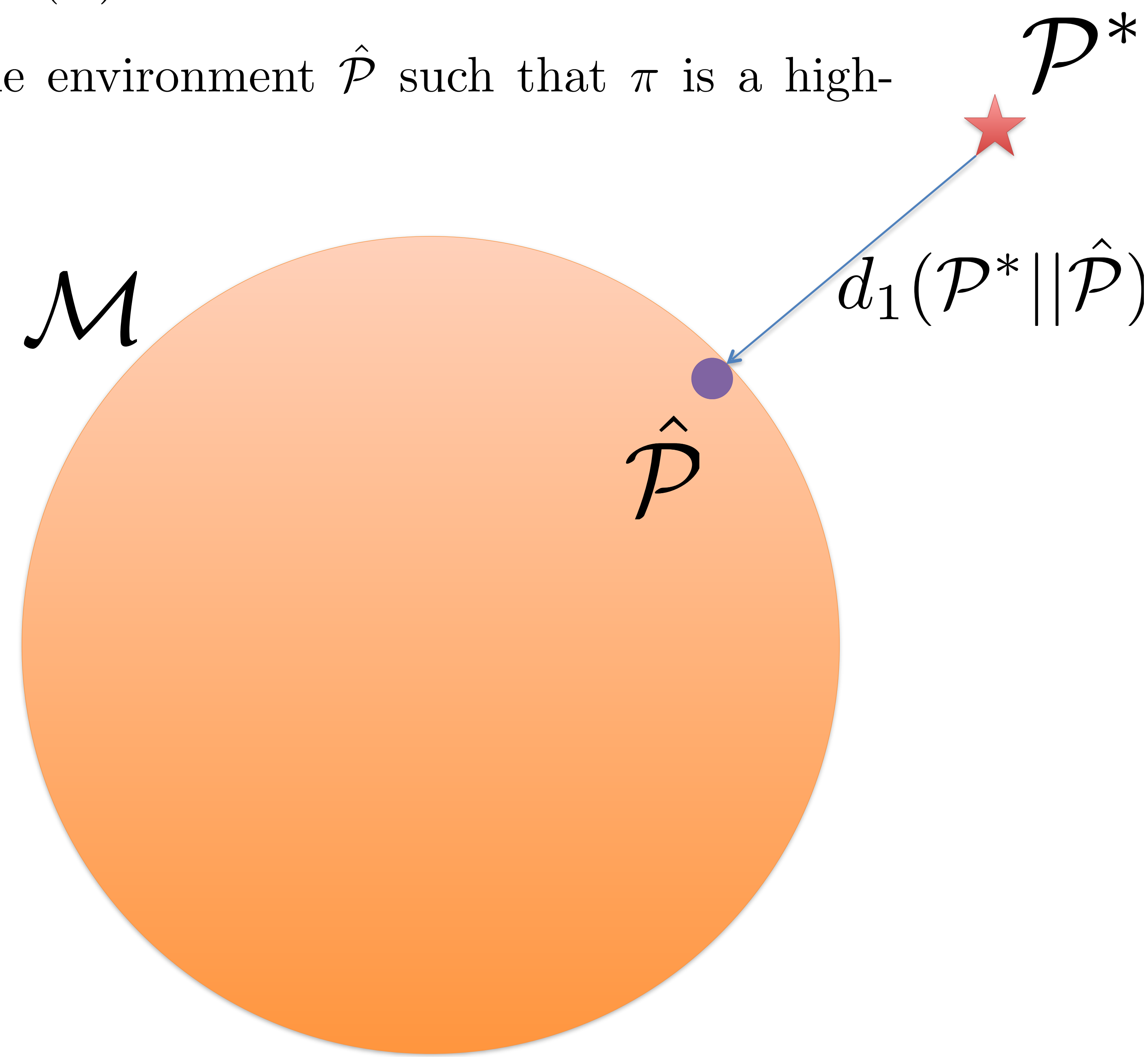




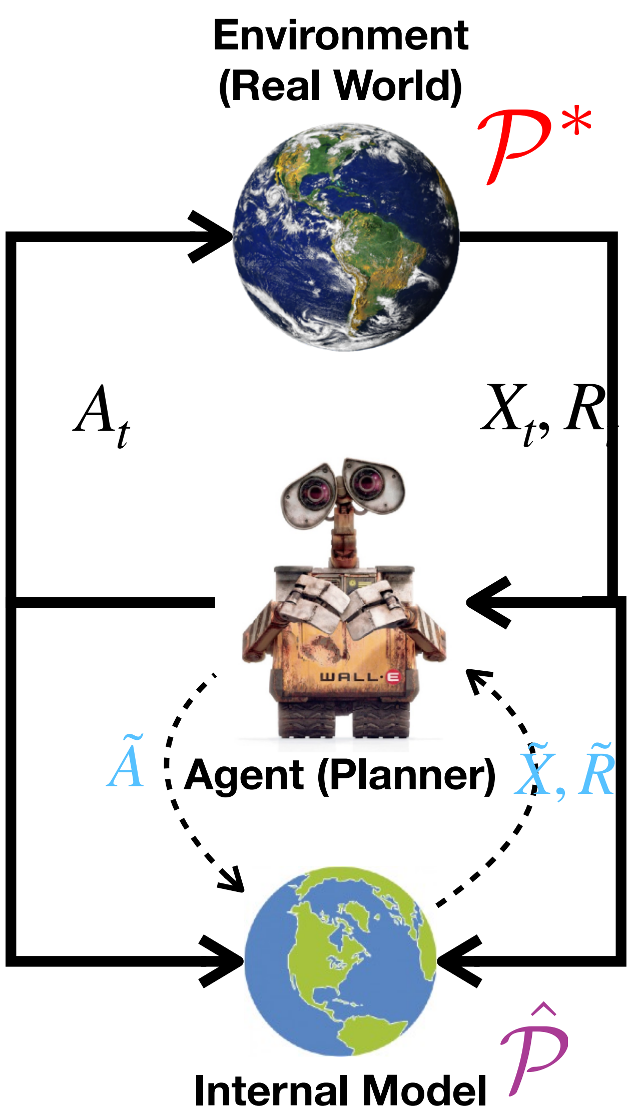


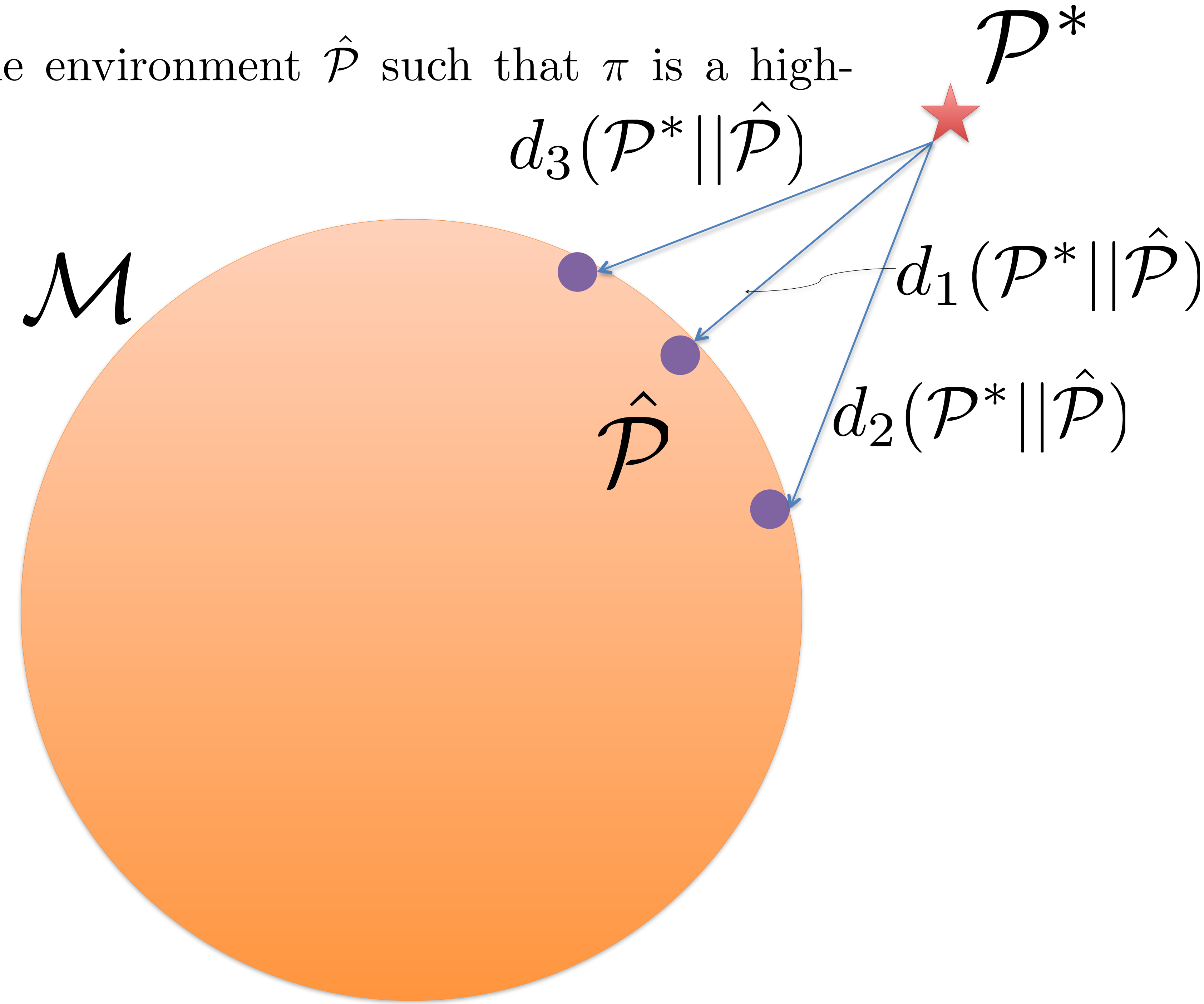
- True model of the environment: $\mathcal{P}^*$
- We are given a dataset $\mathcal{D}_n = \{(X_i, A_i, X'_i)\}_{i=1}^n$ with $Z_i = (X_i, A_i) \sim \nu(\mathcal{X} \times \mathcal{A})$ and $X'_i \sim \mathcal{P}^*(\cdot|X_i, A_i)$
- Policy of the MBRL: $\pi \leftarrow \mathsf{Planner}(\hat{\mathcal{P}})$
- How to estimate a model of the environment $\hat{\mathcal{P}}$ such that π is a high-performing policy?





- True model of the environment: $\mathcal{P}^*$
- We are given a dataset $\mathcal{D}_n = \{(X_i, A_i, X_i')\}_{i=1}^n$ with $Z_i = (X_i, A_i) \sim \nu(\mathcal{X} \times \mathcal{A})$ and $X_i' \sim \mathcal{P}^*(\cdot|X_i, A_i)$
- Policy of the MBRL: $\pi \leftarrow \mathsf{Planner}(\hat{\mathcal{P}})$
- How to estimate a model of the environment $\hat{\mathcal{P}}$ such that π is a high-performing policy?

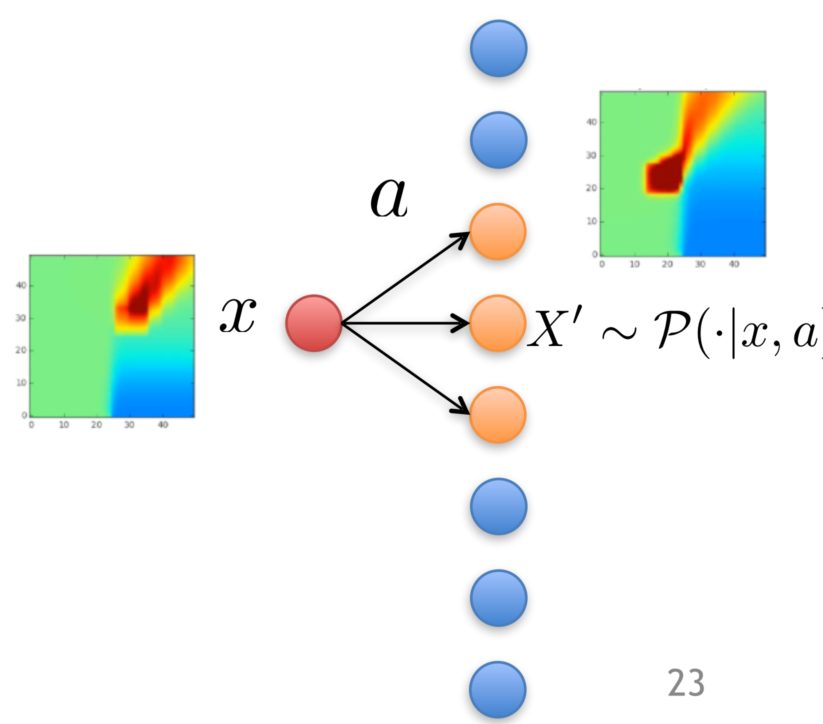




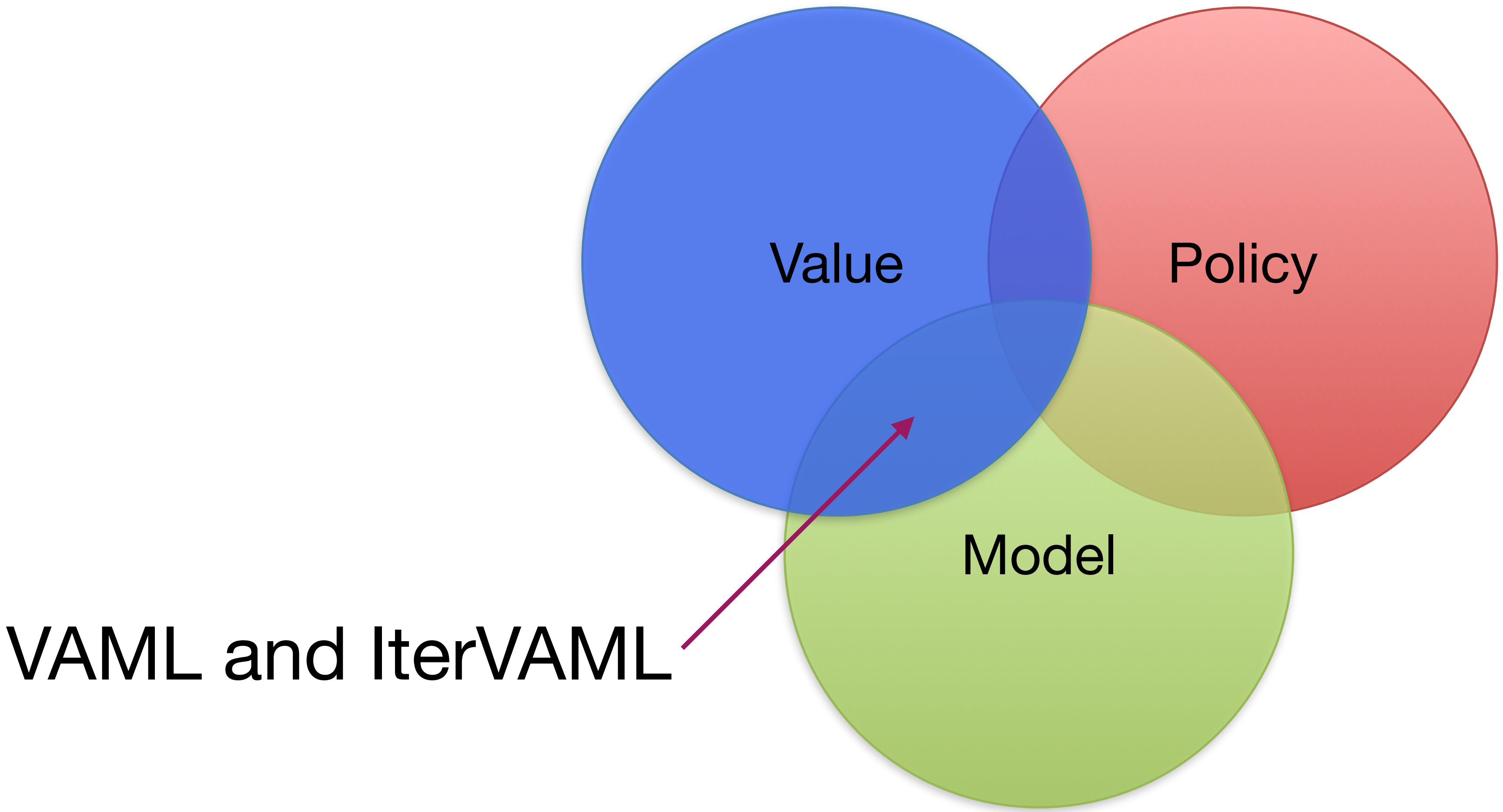
What kind of Planner?

- Variety of Planners
  - Value-based, Policy Gradient, TRPO, etc.
- Let's focus on Bellman operator-based ones:

$$T^*_{\mathcal{P}} Q(x,a) = r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) \max_{a' \in \mathcal{A}} Q(x',a')$$



Value-Aware Model Learning (VAML)



AMF, Barreto, Nikovski, "Value-Aware Loss Function for Model Learning in Reinforcement Learning," European Workshop on Reinforcement Learning (EWRL), 2016.
AMF, Barreto, Nikovski, "Value-Aware Loss Function for Model-Based Reinforcement Learning," Artificial Intelligence and Statistics (AISTATS), 2017.
AMF, "Iterative Value-Aware Model Learning," Neural Information Processing Systems (NeurIPS), 2018.

Value-Aware Model Learning

Goal:

Finding a model that "preserves" the effect of the **Bellman operator** as much as possible.

$$T^*_{\hat{\mathcal{P}}} Q \approx T^*_{\mathcal{P}^*} Q$$

Bellman operator w.r.t. the learned model

Bellman operator w.r.t. the true model

$$T^*_{\mathcal{P}} Q(x, a) = r(x, a) + \gamma \int \mathcal{P}(\mathrm{d}x' | x, a) \max_{a' \in \mathcal{A}} Q(x', a')$$

Value-Aware Model Learning

Let us construct a new loss function …

$$T^*_{\hat{\mathcal{P}}} Q \approx T^*_{\mathcal{P}^*} Q$$

Value-Aware Model Learning

$$T^*_{\mathcal{P}^*} Q(x,a) = r(x,a) + \gamma \int \mathcal{P}^*(\mathrm{d}x'|x,a) \max_{a' \in \mathcal{A}} Q(x',a')$$

$$T^*_{\hat{\mathcal{P}}} Q(x,a) = r(x,a) + \gamma \int \hat{\mathcal{P}}(\mathrm{d}x'|x,a) \max_{a' \in \mathcal{A}} Q(x',a')$$

$$\Big\Downarrow \; T^*_{\hat{\mathcal{P}}} Q \approx T^*_{\mathcal{P}^*} Q$$

$$\begin{aligned} c(\hat{\mathcal{P}}, \mathcal{P}^*; V)(x,a) &= \left| \left\langle \mathcal{P}^*(\cdot|x,a) - \hat{\mathcal{P}}(\cdot|x,a) \, , \, V \right\rangle \right| \\ &= \left| \int \left[\mathcal{P}^*(\mathrm{d}x'|x,a) - \hat{\mathcal{P}}(\mathrm{d}x'|x,a) \right] V(x') \right| \end{aligned}$$

Maximum Likelihood Estimator

Let P_1, P_2 be defined over $\mathcal{X}$ (just to simplify). Note that

$$\|P_1 - P_2\|_1 \leq \sqrt{2\mathsf{KL}(P_1||P_2)}. \qquad \text{(Pinsker)}$$

So we may find $\hat{P}$ that minimizes $\mathsf{KL}(P^*||\hat{P})$:

$$\hat{P} \leftarrow \underset{P \in \mathcal{M}}{\operatorname{argmin}} \sum_{x \in \mathcal{X}} P^*(x) \log \frac{P^*(x)}{P(x)}$$

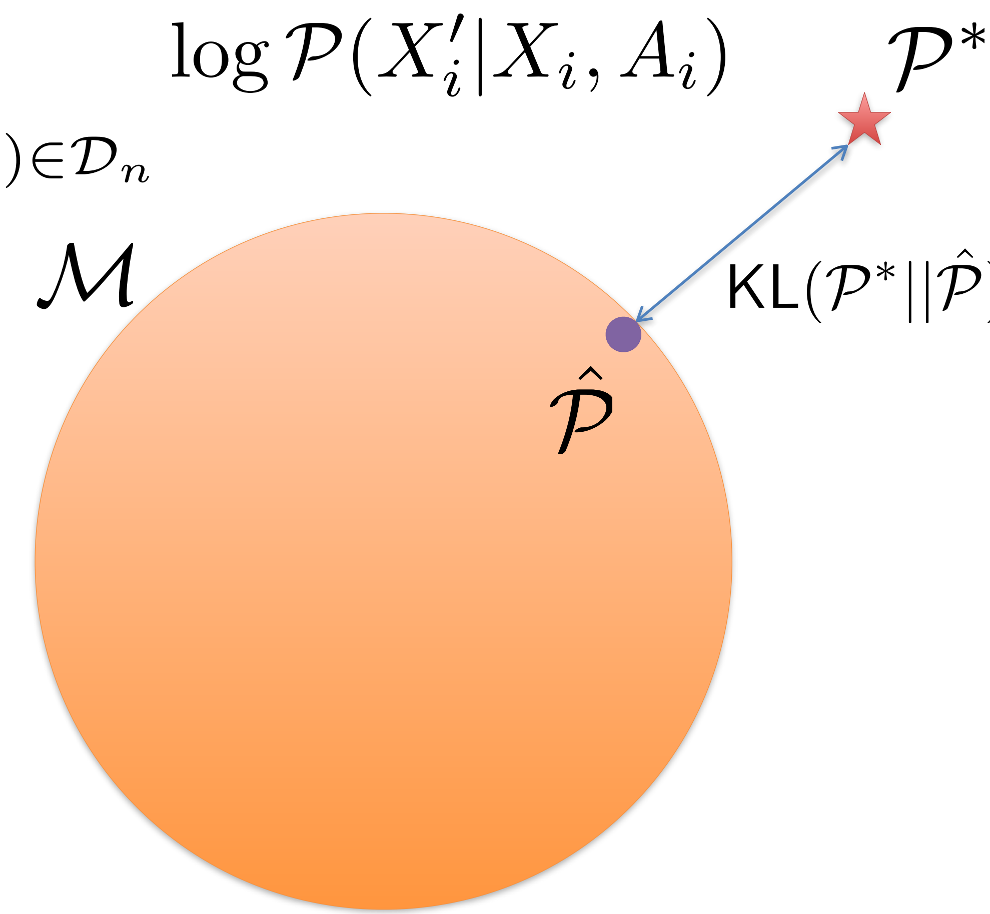
Or its empirical version: Given $\mathcal{D}_n = \{X_i\}_{i=1}^n$ with $X_i \sim P^*$, define the empirical measure $P_n^*(\cdot) = \frac{1}{n}\sum_{i=1}^n \delta_{X_i}(\cdot)$.
The Maximum Likelihood Estimator (MLE) is

$$\hat{P} \leftarrow \underset{P \in \mathcal{M}}{\operatorname{argmin}} \mathsf{KL}(P_n^*||P) \equiv \underset{P \in \mathcal{M}}{\operatorname{argmax}} \frac{1}{n} \sum_{X_i \in \mathcal{D}_n} \log P(X_i).$$

VAML vs. MLE

$$\left|\left\langle \mathcal{P}^*(\cdot|x,a) - \hat{\mathcal{P}}(\cdot|x,a)\,,\, V \right\rangle\right| \leq \underbrace{\left\|\mathcal{P}^*(\cdot|x,a) - \hat{\mathcal{P}}(\cdot|x,a)\right\|_1}_{\leq \sqrt{2\mathsf{KL}\left(\mathcal{P}^*(\cdot|x,a)||\hat{\mathcal{P}}(\cdot|x,a)\right)}} \|V\|_\infty$$

$$\hat{\mathcal{P}} \leftarrow \operatorname*{argmin}_{\mathcal{P}\in\mathcal{M}} \mathsf{KL}(\mathcal{P}^*_n||\mathcal{P}) = \operatorname*{argmax}_{\mathcal{P}\in\mathcal{M}} \frac{1}{n} \sum_{(X_i, A_i, X_i') \in \mathcal{D}_n} \log \mathcal{P}(X_i'|X_i, A_i)$$

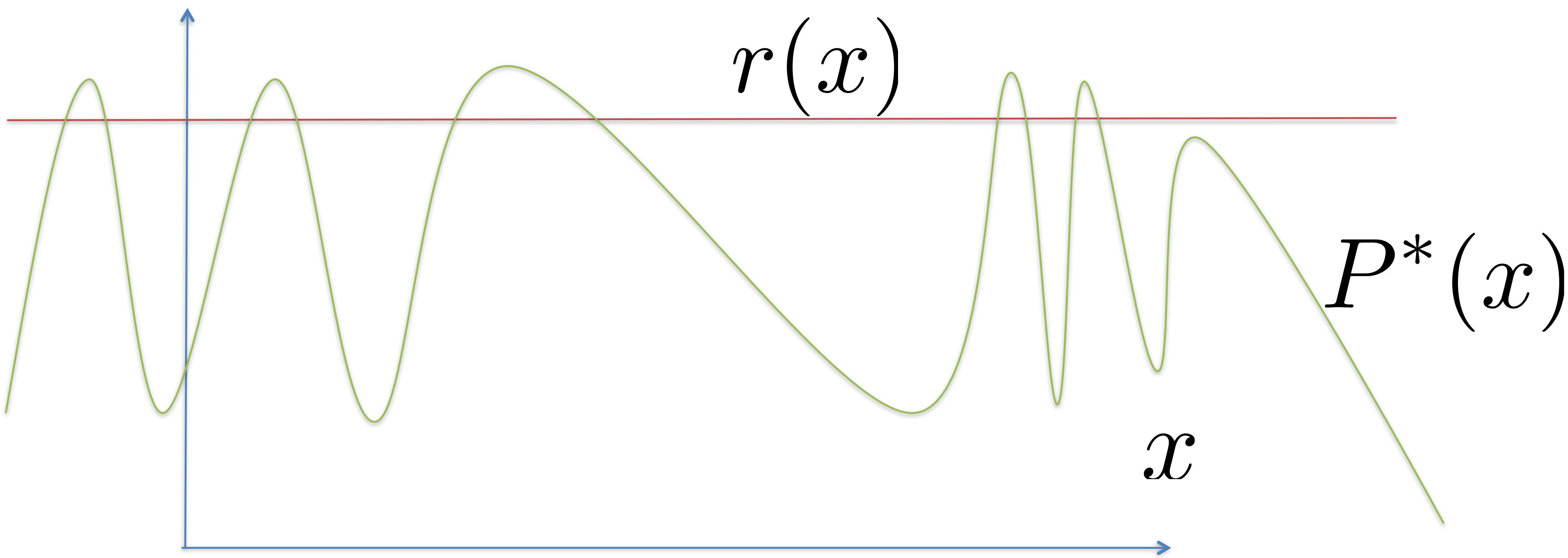


MLE ignores any possible information about the decision problem.

Joseph, Geramifard, Roberts, How, Roy, ICRA, 2013 — Silver, van Hasselt, Hessel, et al., ICML, 2017 — Farquhar, Rocktaeschel Igl, Whiteson, ICLR, 2018 — Oh, Singh, Lee, NIPS, 2017

$$\hat{P}r \approx P^* r$$

$$\text{i.e., } \int \hat{P}(\mathrm{d}x')r(x') \approx \int P^*(\mathrm{d}x')r(x')$$



- No need to accurately (in the KL sense) estimate the true model.
- Any model is sufficient.
- MLE is an overkill for this reward (value) function.


$$c^2(\hat{\mathcal{P}}, \mathcal{P}^*; V)(x,a) = \left| \int \left[\mathcal{P}^*(\mathrm{d}x'|x,a) - \hat{\mathcal{P}}(\mathrm{d}x'|x,a) \right] V(x') \right|^2$$

Pointwise to expectation

$$c_{2,\nu}^2(\hat{\mathcal{P}}, \mathcal{P}^*; V) = \int \mathrm{d}\nu(x,a) \left| \int \left[\mathcal{P}^*(\mathrm{d}x'|x,a) - \hat{\mathcal{P}}(\mathrm{d}x'|x,a) \right] V(x') \right|^2$$

$$c^2_{2,\nu}(\hat{\mathcal{P}}, \mathcal{P}^*; V) = \int \mathrm{d}\nu(x,a) \left| \int \left[\mathcal{P}^*(\mathrm{d}x'|x,a) - \hat{\mathcal{P}}(\mathrm{d}x'|x,a) \right] V(x') \right|^2$$

Unknown!

- **Value-Aware Model Learning (VAML):** Suppose that Planner uses a value function space $\mathcal{F}$ to represent the value function. We learn a model in $\mathcal{M}$ that is uniformly good for any function in $\mathcal{F}$.

- **Iterative VAML:** Learn models by benefiting from how Approximate Value Iteration (AVI)-based Planner generates value functions and uses models.

$$c^2_{2,\nu}(\hat{\mathcal{P}}, \mathcal{P}^*; V) = \int \mathrm{d}\nu(x,a) \left| \int \left[\mathcal{P}^*(\mathrm{d}x'|x,a) - \hat{\mathcal{P}}(\mathrm{d}x'|x,a) \right] {\color{red}V}(x') \right|^2$$

Unknown!

Suppose that Planner uses a value function space $\mathcal{F}$ to represent the value function. We learn a model in $\mathcal{M}$ that is uniformly good for any function in $\mathcal{F}$.

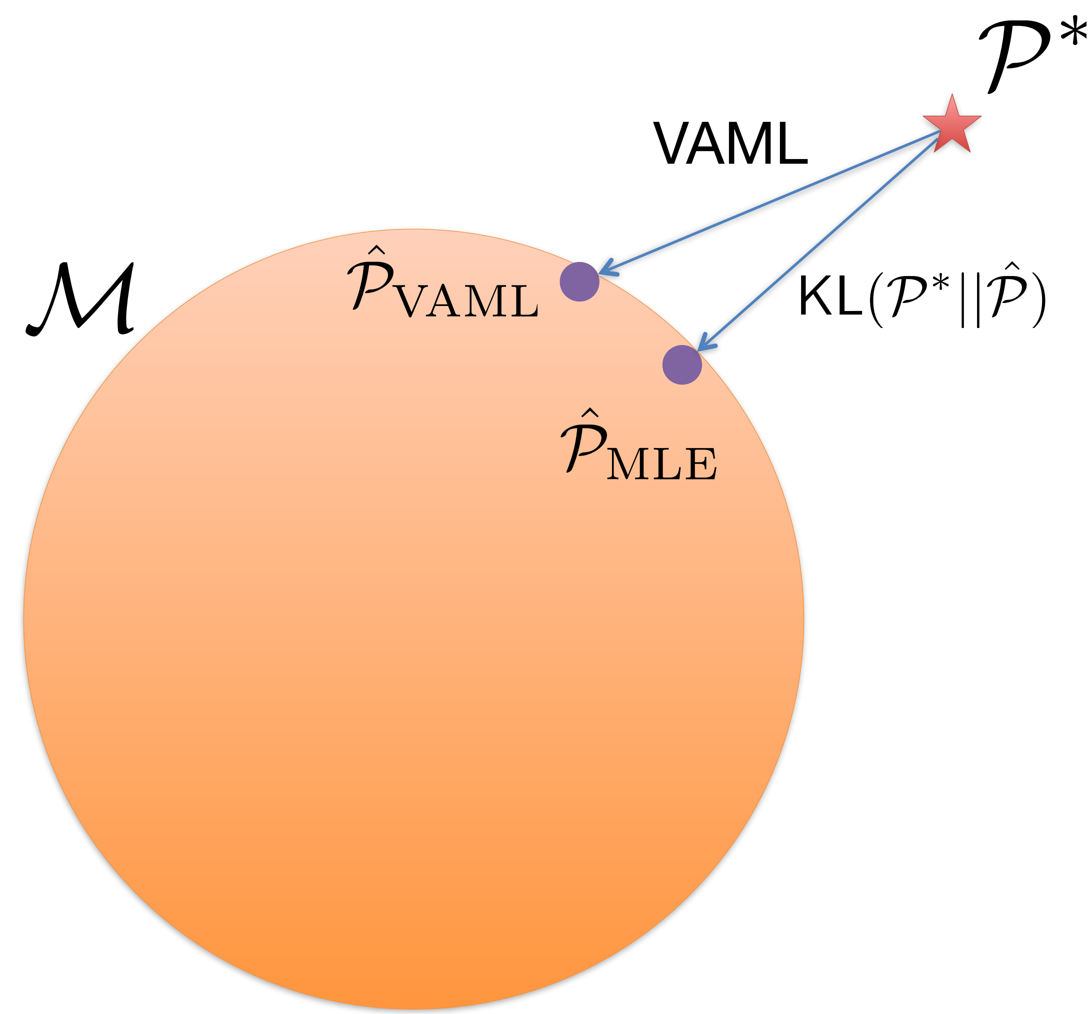
$$c^2_{2,\nu}(\hat{\mathcal{P}}, \mathcal{P}^*) = \int \mathrm{d}\nu(x,a) \sup_{V \in \mathcal{F}} \left| \int \left[\mathcal{P}^*(\mathrm{d}x'|x,a) - \hat{\mathcal{P}}(\mathrm{d}x'|x,a) \right] V(x') \right|^2$$

Value-Aware Model Learning (VAML)

$$\hat{\mathcal{P}}_{\mathrm{VAML}} \leftarrow \operatorname*{argmin}_{\hat{\mathcal{P}} \in \mathcal{M}} \frac{1}{n} \sum_{(X_i, A_i, X_i') \in \mathcal{D}_n} \sup_{V \in \mathcal{F}} \left| V(X_i') - \int \hat{\mathcal{P}}(\mathrm{d}x' | X_i, A_i) V(x') \right|^2$$

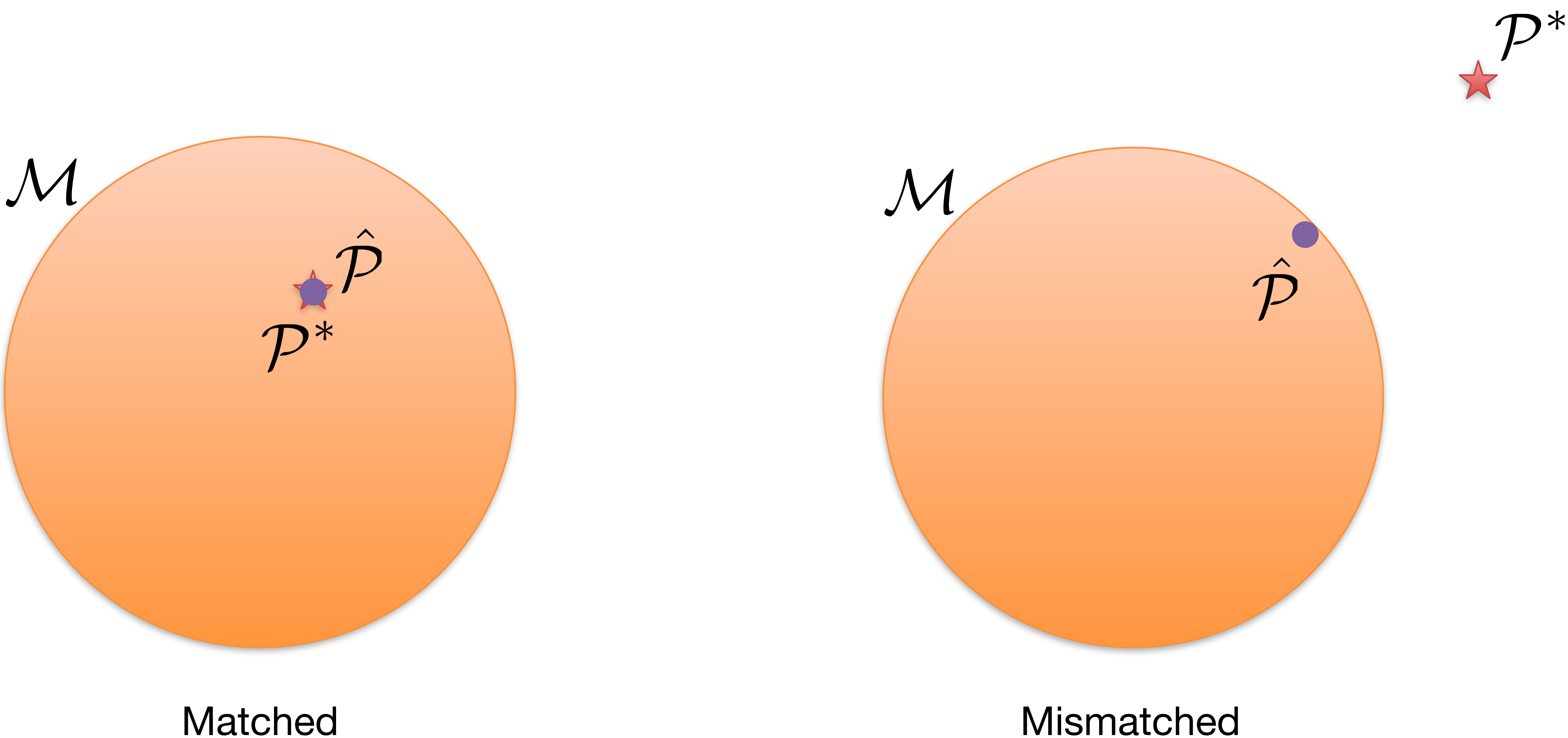
Remarks:

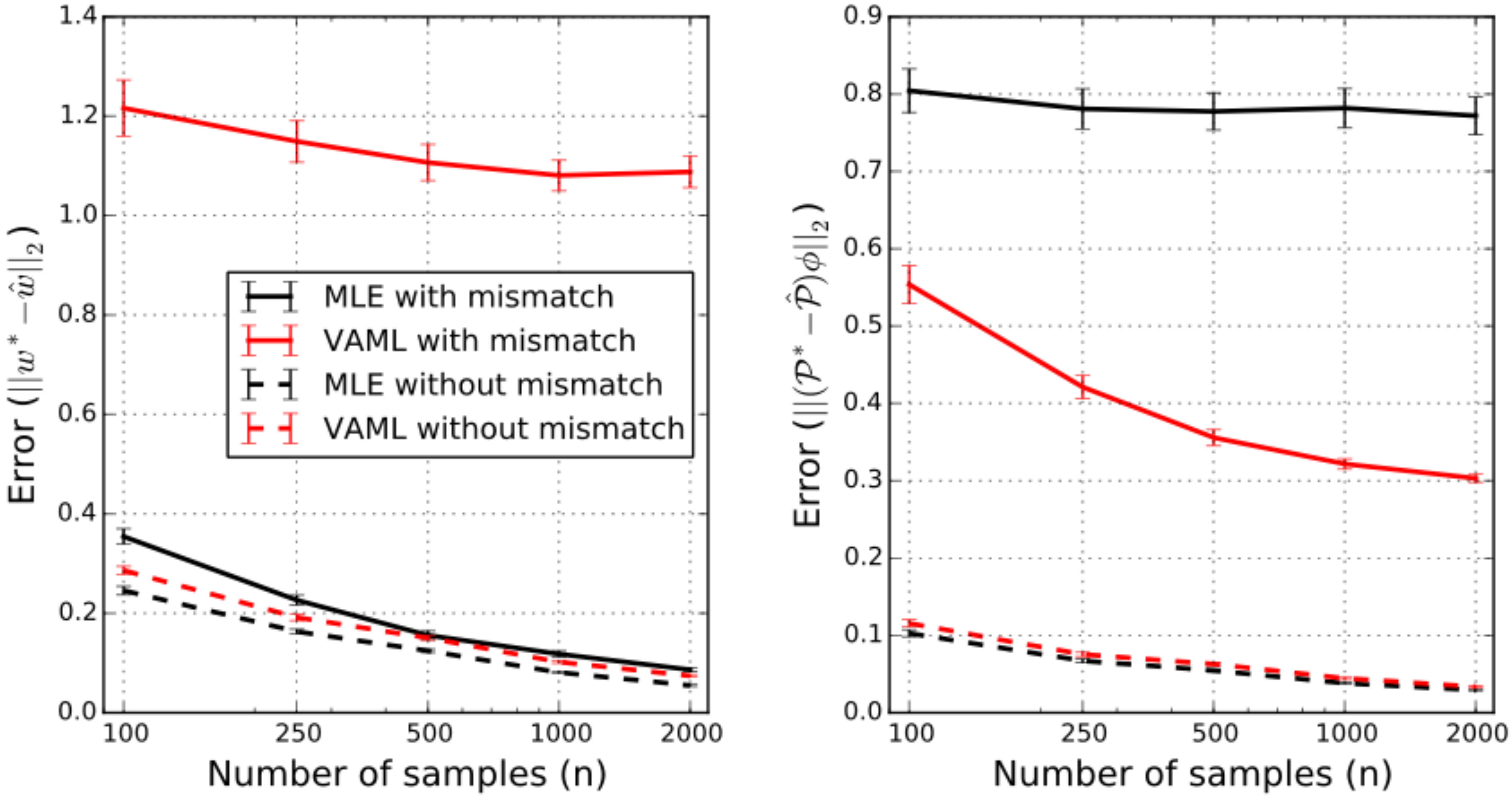
- For linear value function spaces, the inner optimization problem can be solved efficiently.
- We have finite-sample error upper bound for VAML.



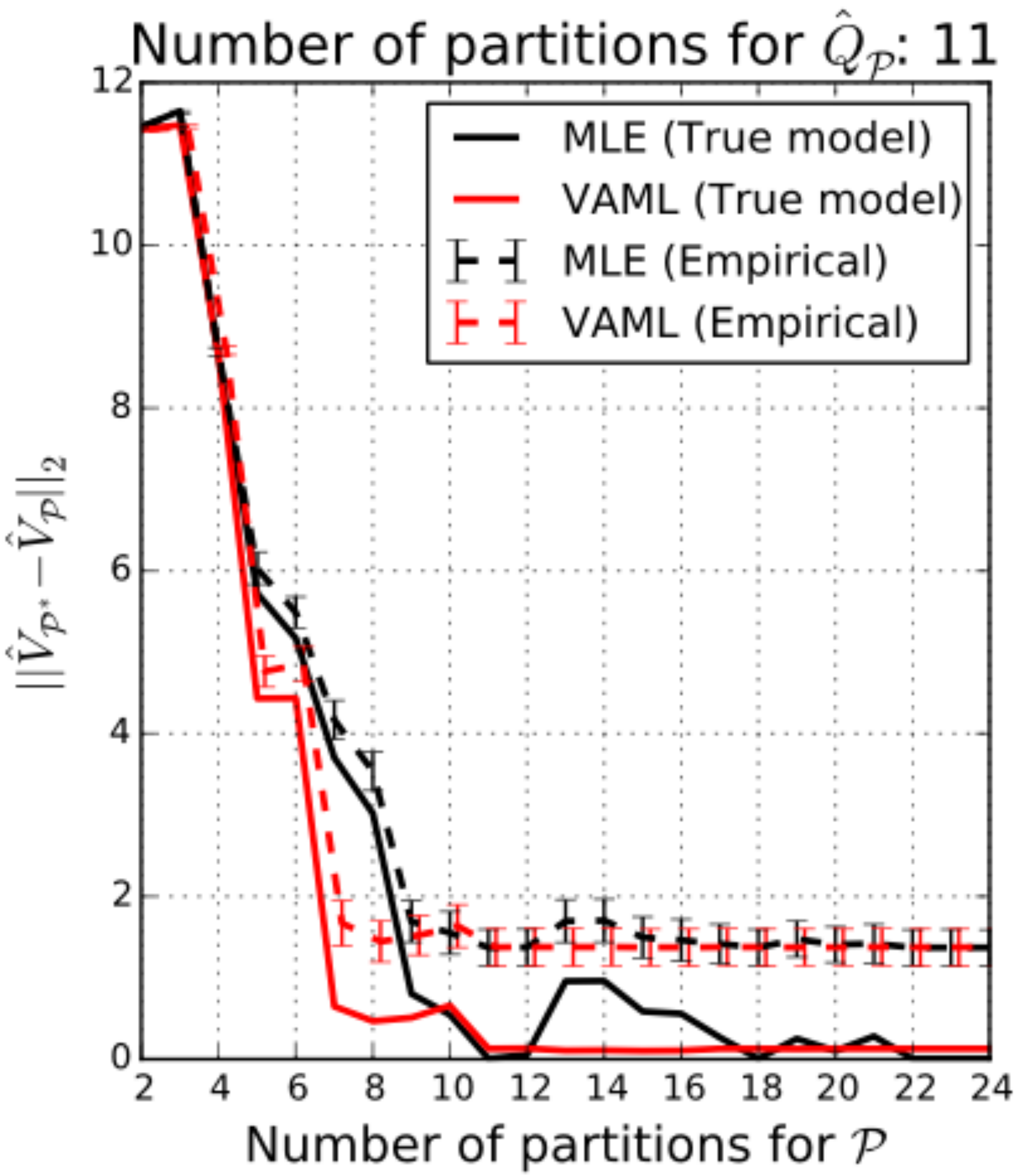
AMF, Barreto, Nikovski, AISTATS, 2017

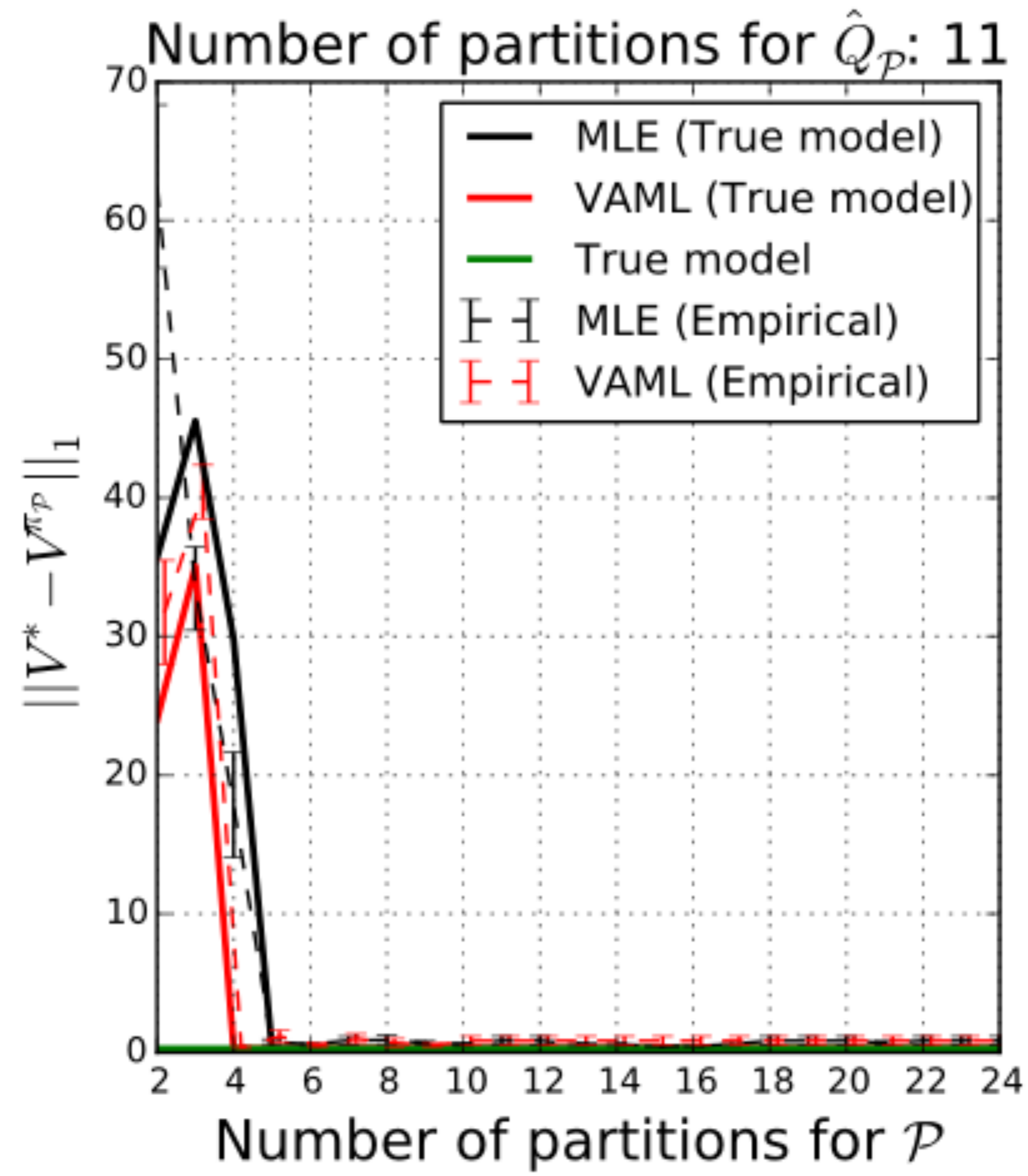
$$\mathbb{E}\left[\sup_{V\in\mathcal{F}}\left|(\hat{\mathcal{P}}_Z - \mathcal{P}^*_Z)V\right|^2\right] \le \inf_{\mathcal{P}\in\mathcal{M}} \mathbb{E}\left[\sup_{V\in\mathcal{F}}\left|(\mathcal{P}_Z - \mathcal{P}^*_Z)V\right|^2\right] + O\left(B^\alpha\sqrt{\frac{\log(1/\delta)}{n}}\right)$$





Domain: 10-dim Gaussian/Exponential
Model: Gaussian





Domain: Finite-state random walk MDP
Model: State aggregation

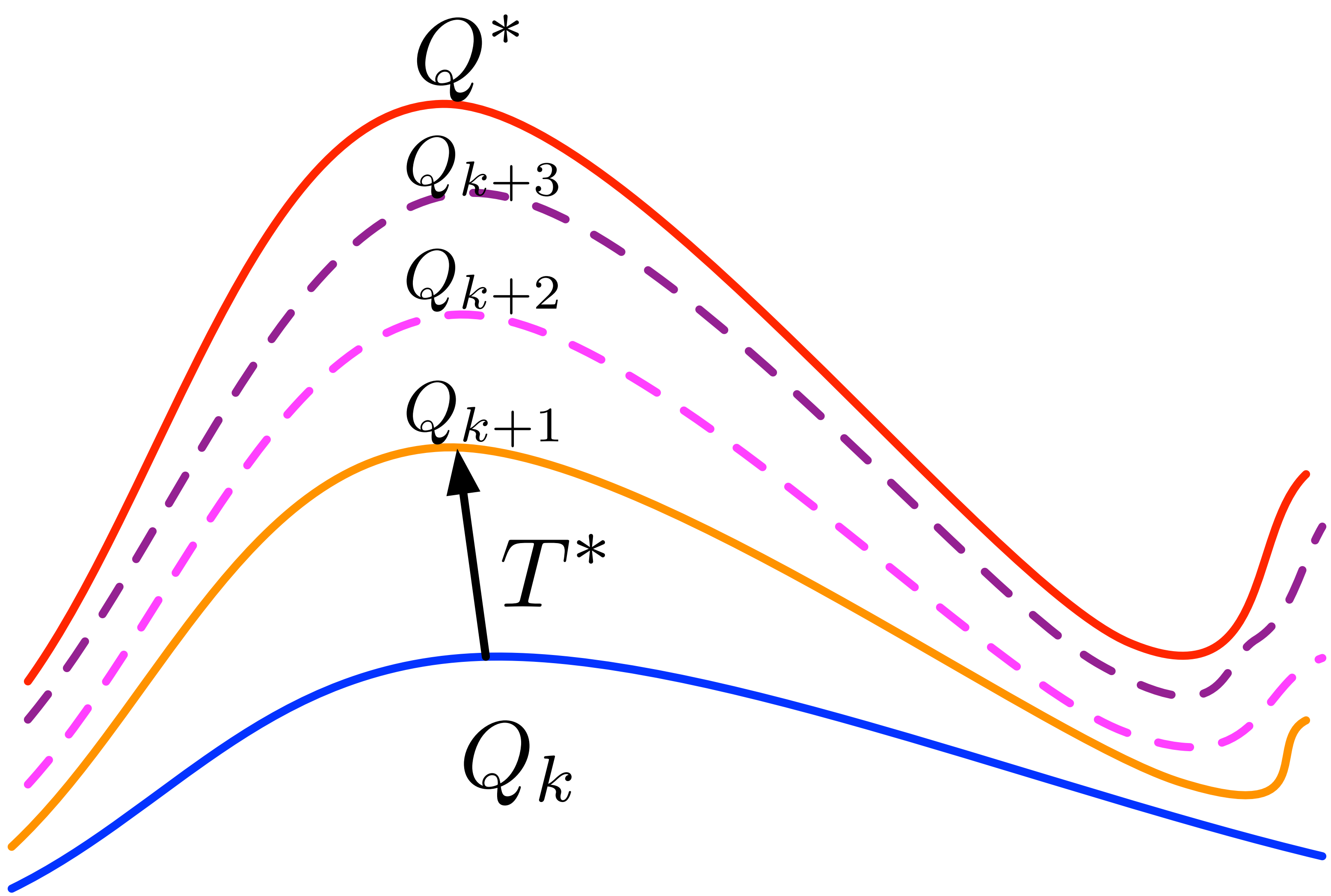
$$\hat{\mathcal{P}}_{\mathrm{VAML}} \leftarrow \operatorname*{argmin}_{\hat{\mathcal{P}} \in \mathcal{M}} \frac{1}{n} \sum_{(X_i, A_i, X_i') \in \mathcal{D}_n} \sup_{V \in \mathcal{F}} \left| V(X_i') - \int \hat{\mathcal{P}}(\mathrm{d}x' | X_i, A_i) V(x') \right|^2$$

Solving VAML optimization problem might be difficult for arbitrary function space!

Iterative VAML

AMF, "Iterative Value-Aware Model Learning," Neural Information Processing Systems (NeurIPS), 2018

Value Iteration



$$Q_{k+1} \leftarrow T^*_{\mathcal{P}^*} Q_k \triangleq r + \gamma \mathcal{P}^* V_k$$

$$V_k(x) \triangleq \max_a Q_k(x, a)$$

Iterative VAML

$$Q_0 \leftarrow r$$

$$Q_1 \leftarrow T^*_{\mathcal{P}^*} V_0 = r + \gamma \mathcal{P}^* V_0$$

$$Q_2 \leftarrow T^*_{\mathcal{P}^*} V_1 = r + \gamma \mathcal{P}^* V_1$$

$$\vdots$$

$$Q_{k+1} \leftarrow T^*_{\mathcal{P}^*} V_k = r + \gamma \mathcal{P}^* V_k$$

$$\hat{\mathcal{P}} V_0 = \mathcal{P}^* V_0$$

$$\hat{\mathcal{P}} V_1 = \mathcal{P}^* V_1$$

$$\hat{\mathcal{P}} V_k = \mathcal{P}^* V_k$$

$$\hat{\mathcal{P}} V_k \approx \mathcal{P}^* V_k$$

Iterative VAML

$$Q_0 \leftarrow r$$

$$Q_1 \leftarrow T^*_{\mathcal{P}^*} V_0 = r + \gamma \mathcal{P}^* r$$

$$Q_2 \leftarrow T^*_{\mathcal{P}^*} V_1 = r + \gamma \mathcal{P}^* V_1$$

$$\vdots$$

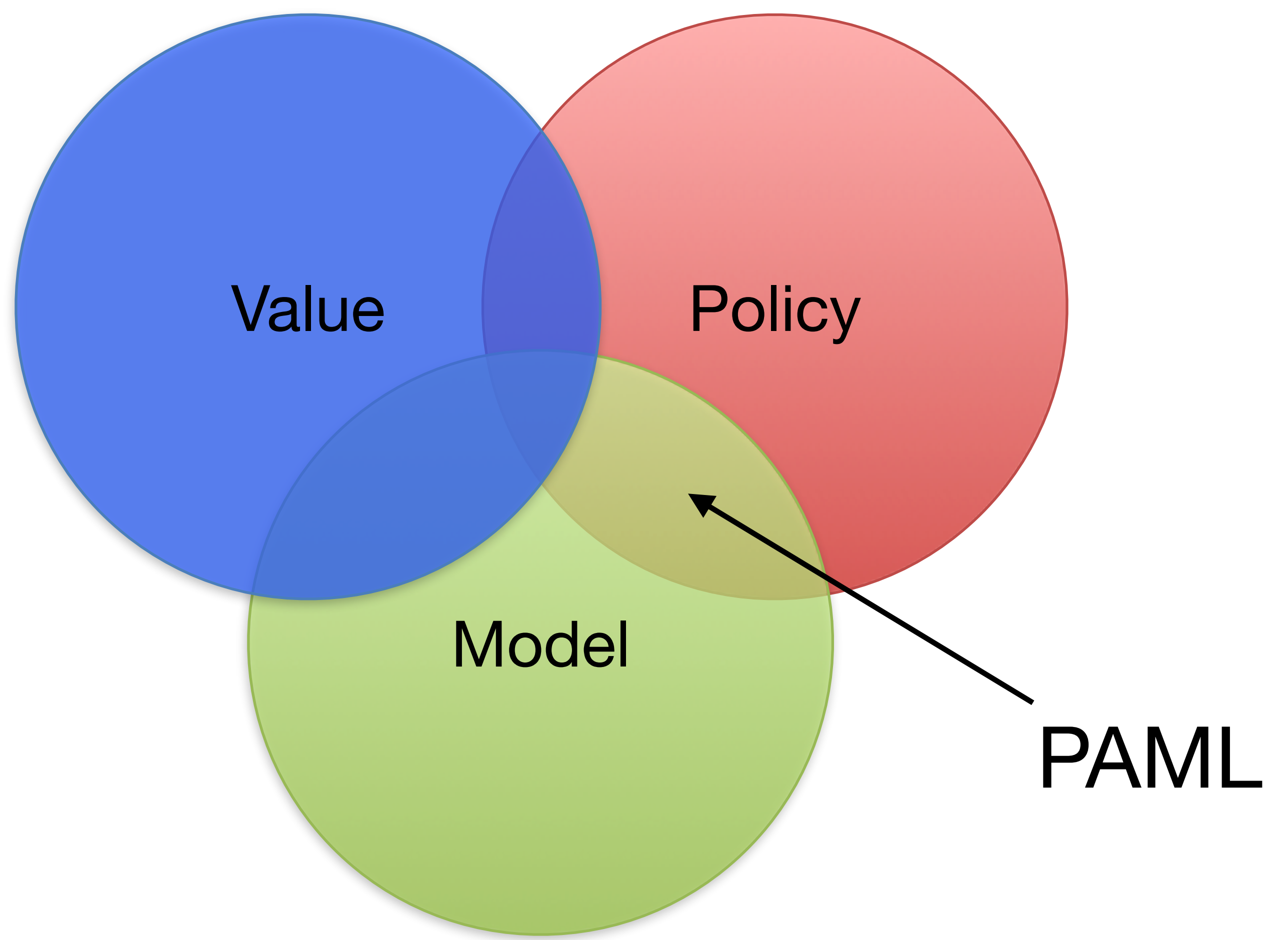
$$Q_{k+1} \leftarrow T^*_{\mathcal{P}^*} V_k = r + \gamma \mathcal{P}^* V_k$$

$$\boxed{\hat{\mathcal{P}} V_k \approx \mathcal{P}^* V_k}$$

$$\hat{\mathcal{P}}^{(k)} \leftarrow \mathop{\mathrm{argmin}}_{\mathcal{P} \in \mathcal{M}} \left\| (\mathcal{P} - \mathcal{P}^*) \hat{V}_k \right\|_2^2 = \int \left| (\mathcal{P} - \mathcal{P}^*)(\mathrm{d}x'|z) \max_{a'} \hat{Q}_k(x', a') \right|^2 \mathrm{d}\nu(z)$$

$$\hat{Q}_{k+1} \leftarrow T^*_{\hat{\mathcal{P}}^{(k)}} \hat{Q}_k$$

Policy-Aware Model Learning (PAML)



Abachi, Ghavamzadeh, **AMF**, "Policy-Aware Model Learning for Policy Gradient Methods," preprint, 2020.

Policy Gradient

Policy parameterized by $\theta \in \Theta$.
Performance objective of an agent starting from an initial probability distribution $\rho \in \bar{\mathcal{M}}(\mathcal{X})$ and following policy π_θ in an MDP $\mathcal{P}$:

$$J_\rho(\pi_\theta; \mathcal{P}) \triangleq \int \mathrm{d}\rho(x) V_{\mathcal{P}}^{\pi_\theta}(x).$$

Policy Gradient:

$$\theta_{k+1} \leftarrow \theta_k + \eta \nabla_\theta J_\rho(\pi_{\theta_k}; \mathcal{P})$$

Policy Gradient

$$\nabla_\theta J(\pi_\theta) = \frac{\partial J(\pi_\theta)}{\partial \theta} = \sum_{k\geq 0} \gamma^k \int \mathrm{d}\rho(x) \int \mathcal{P}^{\pi_\theta}(\mathrm{d}x'|x;k) \sum_{a'\in\mathcal{A}} \frac{\partial \pi_\theta(a'|x')}{\partial \theta} Q^{\pi_\theta}(x',a')$$

$$= \frac{1}{1-\gamma} \int \rho_\gamma(\mathrm{d}x;\mathcal{P}^{\pi_\theta}) \sum_{a\in\mathcal{A}} \pi_\theta(a|x) \frac{\partial \log \pi_\theta(a|x)}{\partial \theta} Q^{\pi_\theta}(x,a).$$

$$\rho^\pi_\gamma(\cdot) = \rho_\gamma(\cdot;\mathcal{P}^\pi) \triangleq (1-\gamma) \sum_{k\geq 0} \gamma^k \int \mathrm{d}\rho(x) \mathcal{P}^\pi(\cdot|x;k).$$

Discounted future-state stationary distribution

Policy-Aware Model Learning

Goal:

Finding a model that computes the **Policy Gradient** as accurate as possible.

$$\nabla_\theta J_\rho(\pi_{\theta_k}; \hat{\mathcal{P}}) \approx \nabla_\theta J_\rho(\pi_{\theta_k}; \mathcal{P}^*)$$

PAML vs MLE

$$\left\| \nabla_\theta J(\pi_\theta) - \nabla_\theta \hat{J}(\pi_\theta) \right\|_p \leq \frac{\gamma}{(1-\gamma)^2} Q_{\max} B_p \times \begin{cases} c_{\mathrm{PG}}(\rho, \nu; \pi_\theta) \sqrt{2\mathsf{KL}_{1(\nu)}(\mathcal{P}^{\pi_\theta} || \hat{\mathcal{P}}_{\pi_\theta})}, \\ 2\sqrt{2\mathsf{KL}_\infty(\mathcal{P}^{\pi_\theta} || \hat{\mathcal{P}}_{\pi_\theta})}. \end{cases}$$

Minimized by PAML

Minimized by MLE

$$\pi_\theta(a|x) = \frac{\exp\left(\phi^\top(a|x)\theta\right)}{\int \exp\left(\phi^\top(a'|x)\theta\right) \mathrm{d}a'}$$

$$c_{\mathrm{PG}}(\rho, \nu; \pi) \triangleq \left\| \frac{\mathrm{d}\rho^\pi_\gamma}{\mathrm{d}\nu} \right\|_\infty$$

$$\mathsf{KL}_\infty(\mathcal{P}_1^\pi || \mathcal{P}_2^\pi) = \sup_{x \in \mathcal{X}} \mathsf{KL}(\mathcal{P}_1^\pi(\cdot|x) || \mathcal{P}_2^\pi(\cdot|x)), \quad \mathsf{KL}_{1(\nu)}(\mathcal{P}_1^\pi || \mathcal{P}_2^\pi) = \int \mathrm{d}\nu(x) \mathsf{KL}(\mathcal{P}_1^\pi(\cdot|x) || \mathcal{P}_2^\pi(\cdot|x)).$$

Convergence for Model-based PG

Projected PG: $\theta_{t+1} \leftarrow \text{Proj}_{\Theta}\left[\theta_t + \eta \nabla_\theta \hat{J}_\mu(\pi_{\theta_k})\right]$

Theorem 1. *After T steps of the projected PG algorithm, we have*

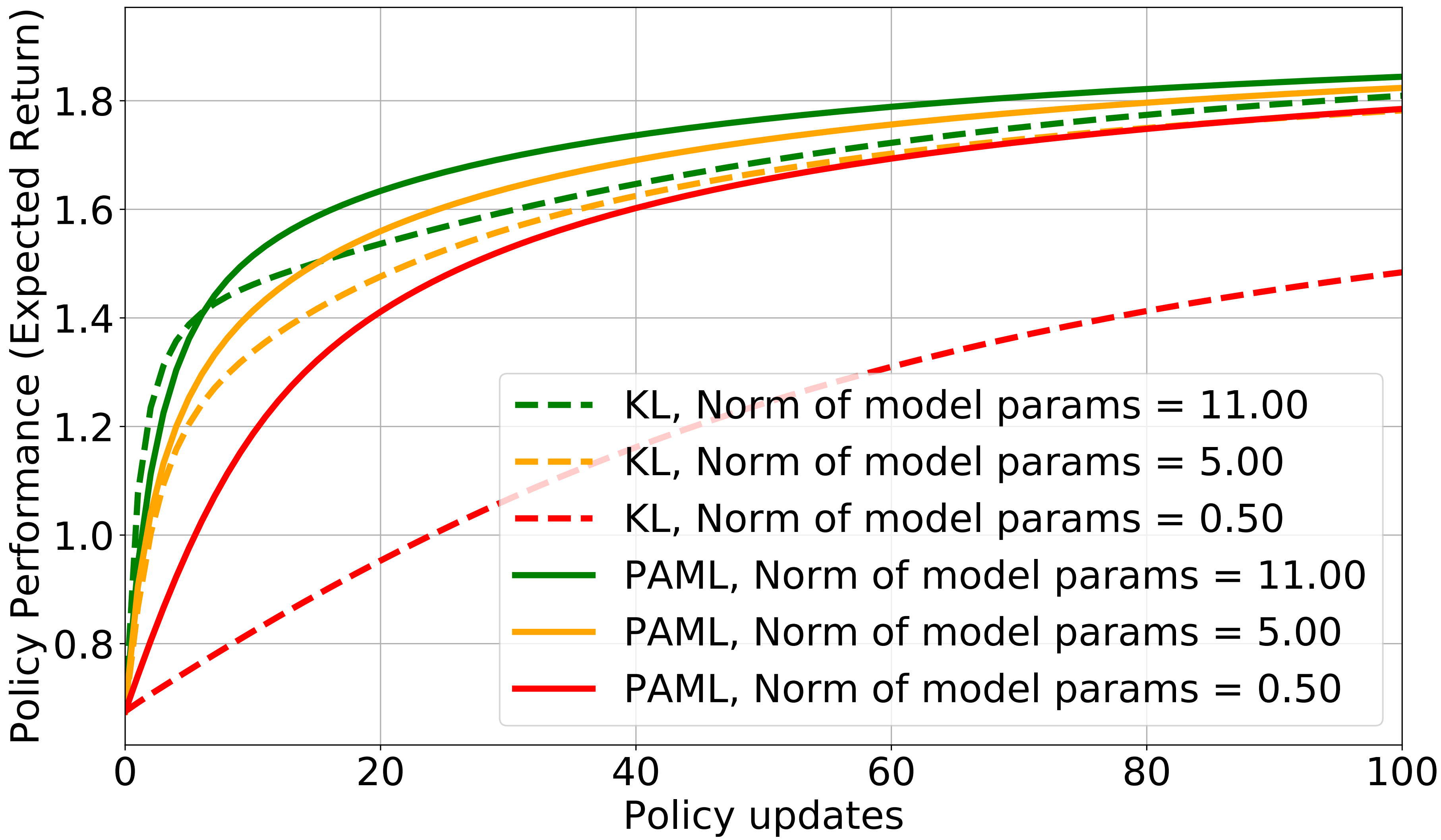
$$\mathbb{E}_{t\sim Unif(1,\ldots,T)}\left[J_\rho(\bar{\pi}) - J_\rho(\pi_{\theta_t})\right] \leq \mathcal{O}\left(\frac{\varepsilon_{PAE}}{1-\gamma} + \frac{\varepsilon_{model}}{1-\gamma} + \frac{1}{(1-\gamma)\sqrt{T}}\right).$$

with

- $L_{PAE}(\theta; \rho_\gamma^{\bar{\pi}}) \leq \varepsilon_{PAE}$ *(policy approximation error)*
- $\|\nabla_\theta J_\mu(\pi_\theta) - \nabla_\theta \hat{J}_\mu(\pi_\theta)\|_2 \leq \varepsilon_{model}$ *(model error)*

$$L_{\mathrm{PAE}}(\theta, w; \nu) \triangleq \mathbb{E}_{X\sim\nu}\left[\Big|\sum_{a\in\mathcal{A}}\big(\bar{\pi}(a|X) - \pi_\theta(a|X) - w^\top \nabla_\theta \pi_\theta(a|X)\big)Q^{\pi_\theta}(X,a)\Big|\right]$$

Experiment: Closed-loop Performance with Exact Gradients



Integral Probability Metric & Model Learning

Given two probability distributions $\mu_1, \mu_2 \in \bar{\mathcal{M}}(\mathcal{X})$ defined over the set $\mathcal{X}$ and a space of functions $\mathcal{F} : \mathcal{X} \to \mathbb{R}$, the Integral Probability Metric (IPM) distance is defined as $d_{\mathcal{F}}(\mu_1, \mu_2) = \sup_{f \in \mathcal{F}} \left| \int f(x) \left(\mathrm{d}\mu_1(x) - \mathrm{d}\mu_2(x)\right) \right|$.

- Total Variation distance: $\mathcal{F}$ is the space of bounded measurable function. (Also recall that $\|\mu_1 - \mu_2\|_{\mathrm{TV}} \leq \sqrt{2\mathsf{KL}(\mu_1 || \mu_2)}$).
- 1-Wasserstein distance: $\mathcal{F}$ is the space of 1-Lipschitz functions. Special case of VAML (Asadi et al., 2018).
- VAML: $\mathcal{F}$ is the space of value functions.
- IterVAML: $\mathcal{F}$ is the most recent value function, i.e., $\mathcal{F} = \{V_k\}$.
- PAML:
  1. $\mathcal{F}$ has a single function $f(x) = \mathbb{E}_{A \sim \pi_\theta(\cdot|x)} \left[\nabla_\theta \log \pi_\theta(A|x) Q^{\pi_\theta}(x, A) \right]$.
  2. Comparison is not between $\mathcal{P}^*$ and $\hat{\mathcal{P}}$, but between $\rho_\gamma(\cdot; \mathcal{P}^{*\pi_\theta})$ and $\rho_\gamma(\cdot; \hat{\mathcal{P}}^{\pi_\theta})$.

IPM: Müller, 1997 — Asadi et al., 2018.

Other DAML Approaches

Several methods in the RL literature might be interpreted as doing some form of DAML, though sometimes it is not explicitly mentioned.
Some examples:

- Joseph et al., ICRA, 2013.
- Predictron (Silver et al., 2017)
- VPN (Oh et al., 2017)
- TreeQN (Farquhar et al., 2018)
- Gradient-Aware Model-based Policy Search (D'Oro et al., 2020)
- muZero (Schrittwieser et al., 2019)
- Value-targeted regression (Ayoub et al., 2020)
- Value equivalence viewpoint (Grimm et al., 2020)
- A few others in non-RL context (Tulabandhula and Rudin, 2013; Kao and Van Roy, 2014; Elmachtoub and Grigas, 2017, Donti et al., 2017)

Take-Home Message

We should incorporate the structure of the **decision problem** into model learning.

- **"Pure" Reinforcement Learning** (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning** (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Unsupervised/Predictive Learning** (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

Unsupervised learning should be guided by the decision problem that we want to solve

References

- Abachi, Ghavamzadeh, and Farahmand, “Policy-Aware Model Learning for Policy Gradient Methods,” preprint, 2020.
- Ayoub, Jia, Szepesvari, Wang, and Yang, "Model-based reinforcement learning with value-targeted regression," International Conference on Machine Learning (ICML), 2020.
- Bertsekas and Tsitsiklis, Neuro-Dynamic Programming, 1996.
- Busoniu, Babuska, De Schutter, and Ernst, Reinforcement Learning and Dynamic Programming Using Function Approximators, 2010.
- Deisenroth, Fox, and Rasmussen, “Gaussian processes for data-efficient learning in robotics and control,” IEEE PAMI, 2015.
- Donti, Amos, and Kolter, "Task-based end-to-end model learning in stochastic optimization," Advances in Neural Information Processing Systems (NIPS), 2017.
- D’Oro, Metelli, Tirinzoni, Papini, and Restelli, "Gradient-aware model-based policy search," AAAI Conference on Artificial Intelligence (AAAI), 2020.
- Adam N. Elmachtoub and Paul Grigas, "Smart “Predict, then Optimize”," Management Science, forthcoming (arXiv in 2017).
- Ernst, Geurts, and Wehenkel, “Tree-based batch mode reinforcement learning,” Journal of Machine Learning Research (JMLR), 2005.
- Farahmand, Ghavamzadeh, Szepesvari, and Mannor, “Regularized fitted Q-iteration for planning in continuous-space Markovian Decision Problems,” ACC, 2009.
- Farahmand, Regularization in Reinforcement Learning, PhD Dissertation, University of Alberta, 2011.

References

- Farahmand and Precup, “Value pursuit iteration,” NIPS, 2012.
- Farahmand, Barreto, Nikovski, “Value-Aware Loss Function for Model-based Reinforcement Learning,” AISTATS, 2017.
- Farahmand, “Iterative Value-Aware Model Learning,” NeurIPS, 2018.
- Farquhar, Rocktaeschel Igl, and Whiteson, “TreeQN and ATreec: Differentiable tree planning for deep reinforcement learning,” ICLR, 2018.
- Gordon, “Stable function approximation in dynamic programming,” ICML, 1995.
- Joseph, Geramifard, Roberts, How, and Roy, “Reinforcement learning with misspecified model classes,” ICRA, 2013.
- Ha and Schmidhuber, “Recurrent world models facilitate policy evolution,” NeurIPS, 2018.
- Kao and Van Roy, "Directed principal component analysis, "Operations Research, 2014.
- Levine, Finn, Darrell, and Abbeel, “End-to-end training of deep visuomotor policies,” JMLR, 2016.
- Luo, Xu, Li, Tian, Darrell, and Ma, "Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees," International Conference on Learning Representations (ICLR), 2019.
- Mnih, Kavukcuoglu, Silver, et al., “Human-level control through deep reinforcement learning,” Nature, 2015.
- Munos and Szepesvari, “Finite-Time Bounds for Fitted Value Iteration,” JMLR, 2008.

References

- Oh, Lee, Lewis, and Sing, “Action-conditional video prediction using deep networks in Atari games,” NIPS, 2015.
- Oh, Singh, and Lee, “Value prediction network,” NIPS, 2017
- Peng, Williams, “Efficient learning and planning within the dyna framework,” Adaptive Behavior, 1993.
- Parr, Li, Taylor, Painter-Wakefield, and Littman, “An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning,” ICML, 2008.
- Schrittwieser, Antonoglou, Hubert, Simonyan, Sifre, Schmitt, Guez, Lockhart, Hassabis, Graepel, et al., "Mastering Atari, Go, Chess and Shogi by planning with a learned model," arXiv preprint arXiv:1911.08265, 2019.
- Silver, van Hasselt, Hessel, et al., “The Predictron: End-to-end learning and planning,” ICML, 2017.
- Sutton, “Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,” ICML, 1990.
- Sutton, Szepesvári, Geramifard, and Bowling, “Dyna-style planning with linear function approximation and prioritized sweeping,” UAI, 2008.
- Sutton and Barto, Reinforcement Learning: An Introduction, 2nd edition, 2018.
- Szepesvari, Algorithms for Reinforcement Learning, 2010.
- Tosatto, Pirotta, D’Eramo, and Restelli, “Boosted fitted Q-iteration,” ICML, 2017.
- Tulabandhula and Rudin, "Machine learning with operational costs," Journal of Machine Learning Research (JMLR), 2013.