A Report on
OFFLINE META-REINFORCEMENT LEARNING WITH ADVANTAGE WEIGHTING

Jeevana Kruthi
Sanjana Kasarla
Aparna Sakshi

Table of Contents

01 **PRIMER**

Background on Meta RL

02 **MODELS**

Baseline models for MACAW

03 **Experiment**

Ablation studies

04 **Results**

Work Done

- Related Work: Offline Meta-Reinforcement Learning with Advantage Weighting - Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, Chelsea Finn (ICML 2021). In this paper Meta-Actor Critic with Advantage Weighting (MACAW), an optimization-based meta-learning algorithm is proposed.

- We reproduced the MACAW algorithm on the Offline Meta-RL setting for two datasets- Cheetah Direction and Cheetah Velocity.

- We also did a comparative study between (MAML + AWR) and the MACAW algorithm proposed in the paper.

01

Primer

- Meta Learning
- Meta Reinforcement Learning
- Why Meta Learning

Meta - Learning

Meta-learning is a branch of machine learning which aims to quickly adapt models, to perform new tasks by learning an underlying structure across related tasks.

Meta Reinforcement Learning

Meta Reinforcement Learning is to do meta-learning in the field of reinforcement learning

Fully offline meta-RL setting

During both meta training and meta testing, the agent cannot interact with environment and has a fixed set of data

Why Meta Learning ?

- Humans are good at RL.
- They utilize previous experiences to learns new tasks
- Train a human to learn images of cats and dogs, they can with very few examples learn that animal on the right is different neither cat nor dog, it is horse.
- In meta learning we learn how to learn

Meta - Reinforcement Learning

Revisiting MDP:
(S, A, T, r)
S - state space
A - action space
T - stochastic transition dynamics T:S x A x S -> [0,1]
r - reward at each time step (s_t, a_t, s_{t+1})

Objective: To minimize discounted sum of rewards $R = \sum_t \gamma^t r_t$

Meta - RL setting:
- Tasks $\mathcal{T}_i$ drawn are from distribution $p(\mathcal{T})$
- MDP corresponding to $\mathcal{T}_i$ is (S, A, p_i, r_i)
- Objective of agent is to find a high performing policy for an unseen task $\mathcal{T}'$ with very less amount of experience on $\mathcal{T}'$
- During Meta Training, the agent meta-learns parameters or update rules that enables such rapid adaptation at test-time.

02

Proposed Models

- MAML
- AWR
- MAML + AWR
- MACAW

Model Agnostic Meta Learning (MAML)

Chelsea Finn, Pieter Abbeel, Sergey Levine - (ICML 2017)

The crux of this algorithm is the following two steps:

- **Inner Loop**: Gradient descent update for each task $\mathcal{T}_i$

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta).$$

- **Outer Loop**: Meta update on the initial parameter

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks

Require: α, β: step size hyperparameters

1: randomly initialize θ

2: **while** not done **do**

3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$

4: **for all** $\mathcal{T}_i$ **do**

5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples

6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$

7: **end for**

8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$

9: **end while**

Advantage Weighted Regression (AWR)

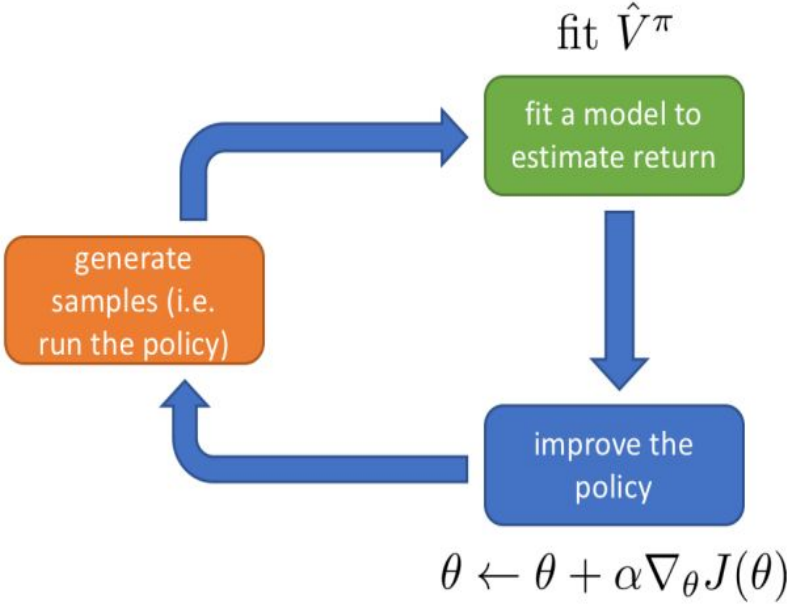
Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning - (Xue Bin Peng, Aviral Kumar, Grace Zhang, Sergey Levine**)**

Algorithm 1 Advantage-Weighted Regression

$\pi_1 \leftarrow$ random policy
$\mathcal{D} \leftarrow \emptyset$
for iteration $k = 1, ..., k_{\max}$ **do**
  add trajectories $\{\tau_i\}$ sampled via π_k to $\mathcal{D}$
  $V_k^{\mathcal{D}} \leftarrow \arg\min_V \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}} \left[\left|\left| \mathcal{R}_{\mathbf{s},\mathbf{a}}^{\mathcal{D}} - V(\mathbf{s}) \right|\right|^2 \right]$
  $\pi_{k+1} \leftarrow \arg\max_\pi \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}} \left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta} \left(\mathcal{R}_{\mathbf{s},\mathbf{a}}^{\mathcal{D}} - V_k^{\mathcal{D}}(\mathbf{s}) \right) \right) \right]$
end for

Does this algorithm look familiar?

Revisiting Actor Critic Algorithm



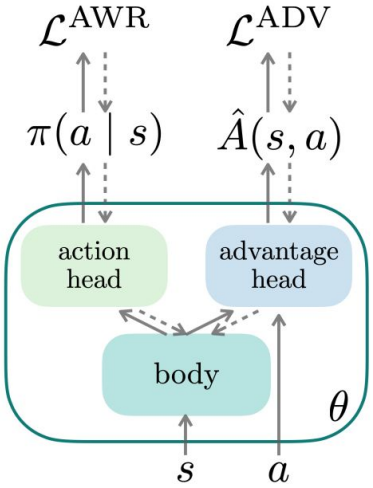
AWR is a off-policy actor critic algorithm based on reward-weighted regression which uses exponentiated advantage given by:

$$\exp\left(\frac{1}{\beta} A^{\mathcal{D}}(\mathbf{s}, \mathbf{a})\right)$$

$$A^{\mathcal{D}}(\mathbf{s}, \mathbf{a}) = \mathcal{R}^{\mathcal{D}}_{\mathbf{s},\mathbf{a}} - V^{\mathcal{D}}(\mathbf{s})$$

MAML + AWR

- Works good for Offline Meta RL setting but suffers from the problem of universality as the gradient of the AWR objective does not contain full information of both the regression weight and the regression target. That is, one cannot recover both the advantage weight and the action from the gradient.
- MACAW algorithm not only includes weighted advantage from AWR but also includes an action advantage term.



$$\theta_i' \leftarrow \theta - \alpha_1 \nabla_\theta \mathcal{L}_\pi(\theta, \phi_i', D_i^{\text{tr}}), \text{ where } \mathcal{L}_\pi = \mathcal{L}^{\text{AWR}} + \lambda \mathcal{L}^{\text{ADV}}$$

$$\mathcal{L}^{\text{ADV}}(\theta, \phi_i', D) \triangleq \mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a} \sim D} \left[\left(A_\theta(\mathbf{s}, \mathbf{a}) - \left(\mathcal{R}_D(\mathbf{s}, \mathbf{a}) - V_{\phi_i'}(\mathbf{s}) \right) \right)^2 \right]$$

Meta Actor-Critic with Advantage Weighting (MACAW)

Algorithm 1 MACAW Meta-Training

Input: Tasks $\{\mathcal{T}_i\}$, offline buffers $\{D_i\}$
Hyperparameters: learning rates $\alpha_1, \alpha_2, \eta_1, \eta_2$, training iterations n, temperature T
Randomly initialize meta-parameters θ, ϕ
for n steps **do**
  for task $\mathcal{T}_i \in \{\mathcal{T}_i\}$ **do**
    Sample disjoint batches $D_i^{\text{tr}}, D_i^{\text{ts}} \sim D_i$
    $\phi_i' \leftarrow \phi - \eta_1 \nabla_\phi \mathcal{L}_V(\phi, D_i^{\text{tr}})$
    $\theta_i' \leftarrow \theta - \alpha_1 \nabla_\theta \mathcal{L}_\pi(\theta, \phi_i', D_i^{\text{tr}})$
  end for
  $\phi \leftarrow \phi - \eta_2 \sum_i \left[\nabla_\phi \mathcal{L}_V(\phi_i', D_i^{\text{ts}})\right]$
  $\theta \leftarrow \theta - \alpha_2 \sum_i \left[\nabla_\theta \mathcal{L}^{\text{AWR}}(\theta_i', \phi_i', D_i^{\text{ts}})\right]$
end for

Algorithm 2 MACAW Meta-Testing

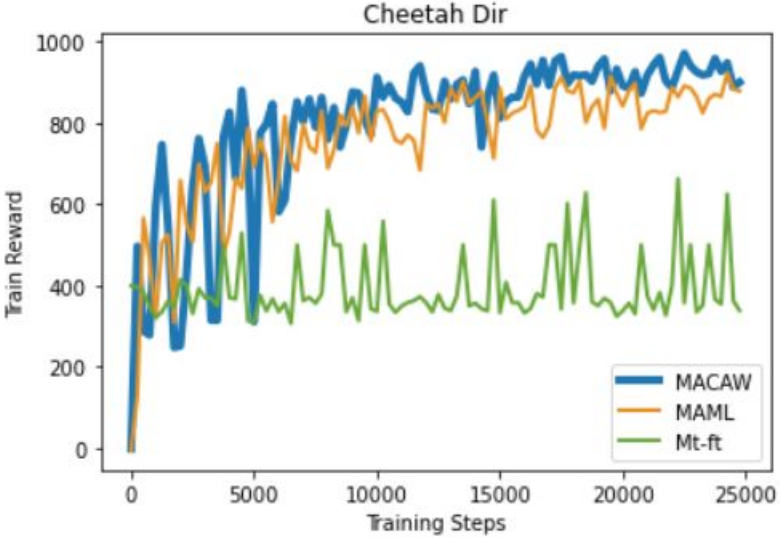
Input: Test task $\mathcal{T}_j$, offline experience D, meta-policy π_θ, meta-value function V_ϕ
Hyperparameters: learning rates α_1, η, adaptation iterations n, temperature T
Initialize $\theta_0 \leftarrow \theta$, $\phi_0 \leftarrow \phi$.
for n steps **do**
  $\phi_{t+1} \leftarrow \phi_t - \eta_1 \nabla_{\phi_t} \mathcal{L}_V(\phi_t, D)$
  $\theta_{t+1} \leftarrow \theta_t - \alpha_1 \nabla_{\theta_t} \mathcal{L}_\pi(\theta_t, \phi_{t+1}, D)$
end for

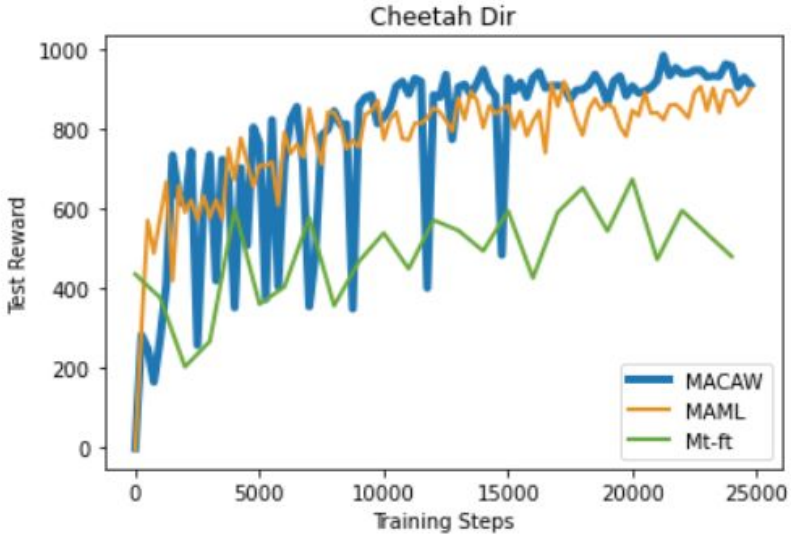
03

Experiments

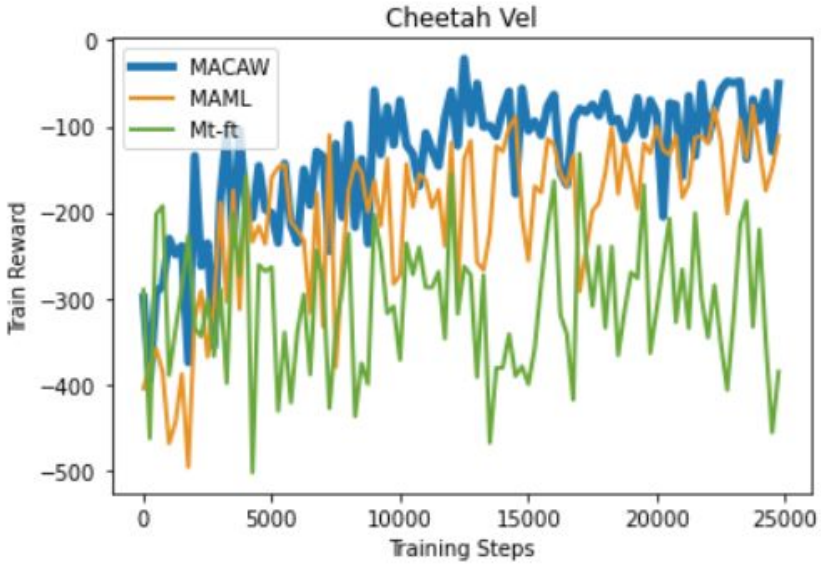
- Datasets - Cheetah Direction, Cheetah Velocity
- Weight Transform
- Other Variations

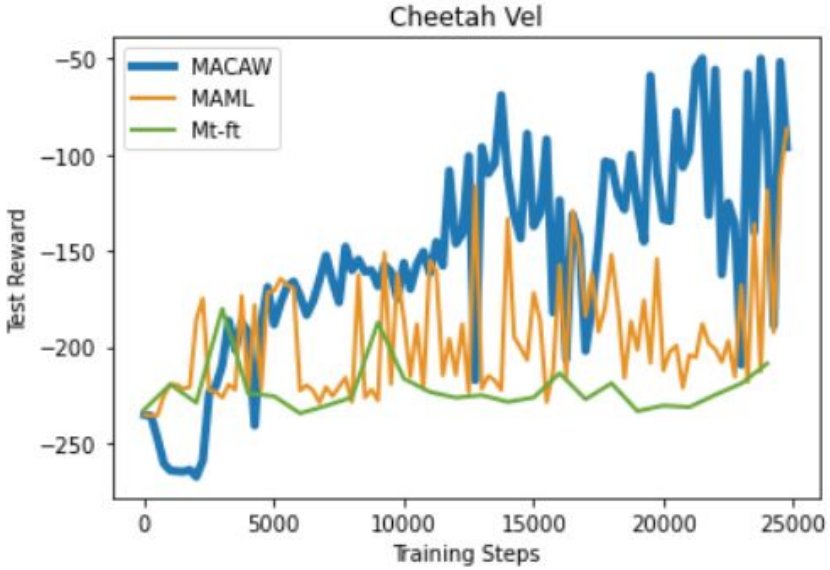
MACAW ON CHEETAH-DIRECTION DATASET





MACAW ON CHEETAH-VELOCITY DATASET

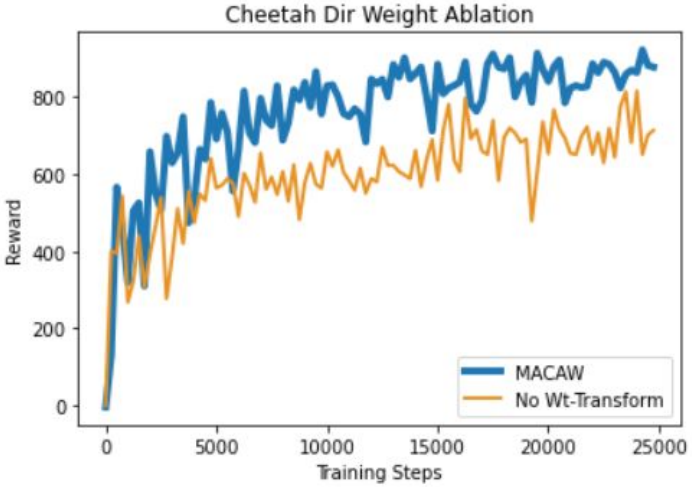


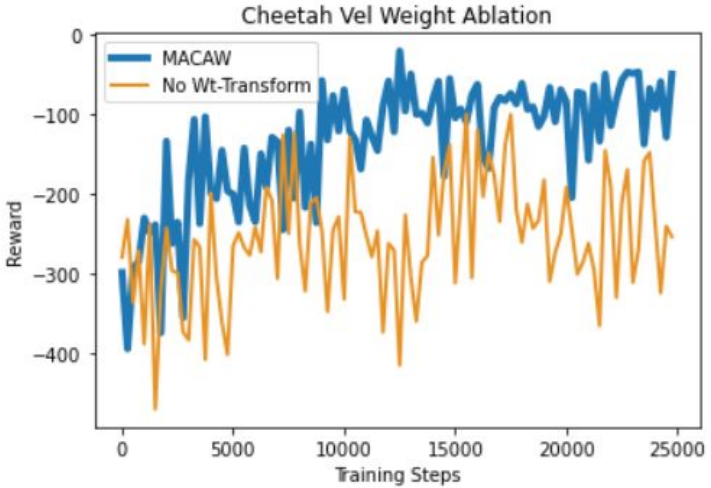


WEIGHT TRANSFORMATION LAYER

- Weight transform layer that increases the expressiveness of the Model
- Extent the idea of bias transformation layer[1] to weights of the network

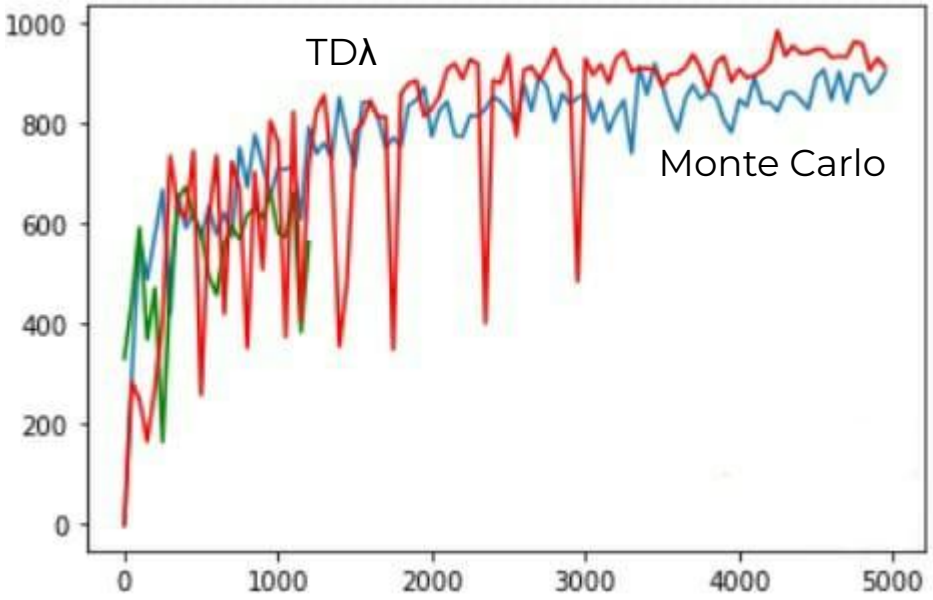
$$w = W^{wt} z, \textbf{ where } W^{wt} \in R^{(d^2+d)c}$$





We observe a significant increase in the reward,speed and model becomes more stable

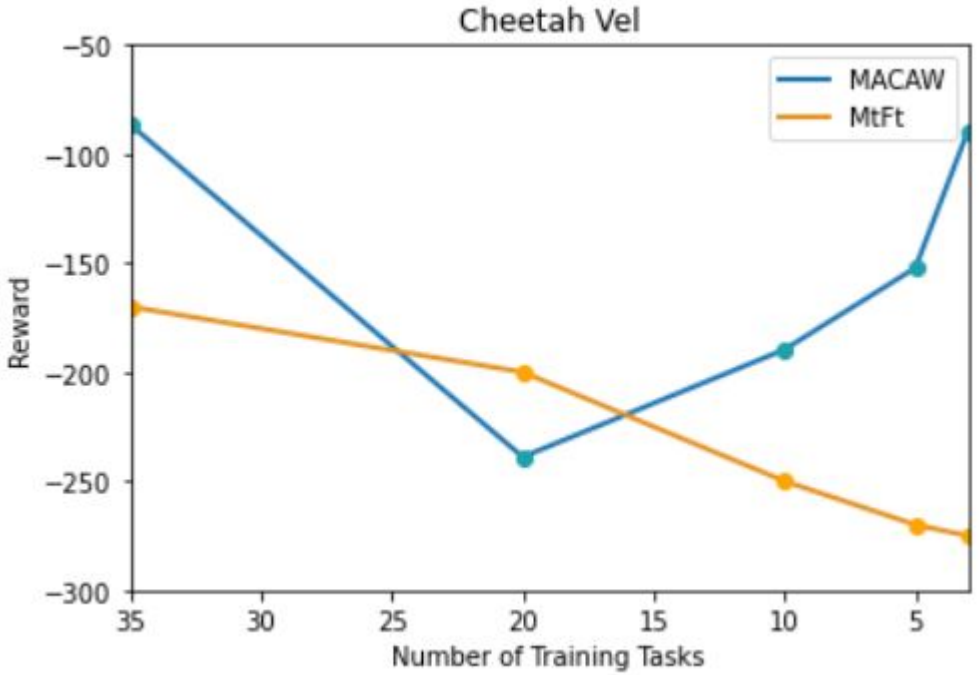
MACAW Experiments



Comparison of MonteCarlo update and TDλ update on cheetah direction dataset

Reducing training data

- Performance of MACAW and Multitask AWR is compared by varying the number of training tasks
- Macaw performs better even on small amount of data



Thank You