Qraft Technolongies (AXE)

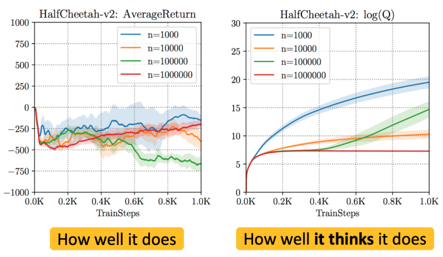
CQL, COMBO

RL Research

1 CQL: Conservative Q-learning (2020)

1.1 Introduction

- Overestimation of values induced by distribution shift (mismatching between training dataset and learned policy) is challenges of offline rl
- Offline Rl needs to answer counterfactual predictions about unseen outcomes without exploration (answering 'what if' questions)
- However, counterfactual predictions for decisions that deviate too much from the behavior in the dataset cannot be made reliably.
- For example Q-learning queries the Q-function at out-of-distribution inputs for computing the bootstrapping target during training.



1.2 Various Method to solve distribution shift

Since we suffer from the curse of distribution shift, a strategy is to be pessimistic with respect to distribution shift (can't do much more than that)

1. Approach 1: Constrain the policy to be close to the behavior policy (policy constraint)

$$\phi := \arg\max_\phi \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi_\phi(\cdot|\mathbf{s})}\left[Q_\theta(\mathbf{s},\mathbf{a})\right] - \alpha D\left(\pi_\phi, \pi_\beta\right) \mathit{or} - \alpha D\left(d^{\pi_\phi}(\mathbf{s},\mathbf{a}), d^{\pi_\beta}(\mathbf{s},\mathbf{a})\right)$$

$$D\left(\pi_\phi(\mathbf{a}\mid\mathbf{s}), \pi_\beta(\mathbf{a}\mid\mathbf{s})\right) \leq \varepsilon$$

   - BCQ: D=KL
   - BEAR: D=MMD
   - BRAC: D=KL/MMD/Wasserstein

2. Approach 2: Directly Modify Q-function to be pessimistic(CQL)
    - Key Idea behind of CQL: Learn lower bounds on Q-values
    - Offline RL via Lower-Bounded Value Function

$$\text{Normal value function:} \quad Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}\left[\sum_{t'=t}^{\infty} \gamma^t r(s_{t'}, a_{t'})\right]$$

errors can be positive or negative!

$$\text{Conservative value function:} \quad Q^{\pi}_{\text{CQL}}(s_t, a_t) \leq \mathbb{E}_{\pi}\left[\sum_{t'=t}^{\infty} \gamma^t r(s_{t'}, a_{t'})\right]$$

guaranteed to be pessimistic, never overestimate value of unseen action

Standard actor-critic algorithm

1. Learn $\hat{Q}^{\pi}$ using offline dataset $\mathcal{D}$.
2. Optimize policy: $\pi \leftarrow \arg\max_{\pi} \mathbb{E}_{\pi}[\hat{Q}^{\pi}]$.

CQL algorithm

1. Learn $\hat{Q}^{\pi}_{\text{CQL}}$ using offline dataset $\mathcal{D}$.
2. Optimize policy: $\pi \leftarrow \arg\max_{\pi} \mathbb{E}_{\pi}[\hat{Q}^{\pi}_{\text{CQL}}]$.

1.3 Method of CQL

1. Conservative Off-Policy Evaluation, finding $\hat{Q}^{\pi}$
    (a) Make lower bound for $\hat{Q}^{\pi} \leq Q^{\pi}$

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \underbrace{\alpha \mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})]}_{\text{Minimize large Q-values}} + \frac{1}{2} \underbrace{\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\left(Q(\mathbf{s}, \mathbf{a}) - \hat{\mathcal{B}}^{\pi}\hat{Q}^{k}(\mathbf{s}, \mathbf{a})\right)^2\right]}_{\text{Standard Bellman Error}}$$

    (b) Improve bound for $\hat{V}^{\pi}(\mathbf{s}) := \mathbb{E}_{\pi(\cdot|\mathbf{s})}\left[\hat{Q}^{\pi}(\mathbf{s}, \mathbf{a})\right] \leq V^{\pi}(\mathbf{s})$

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \alpha \cdot \left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})] - \mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \mathbf{a}\sim\hat{\pi}_{\beta}(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})]\right) + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s}, \mathbf{a}) - \hat{\mathcal{B}}^{\pi}\hat{Q}^{k}(\mathbf{s}, \mathbf{a})\right)^2\right].$$

    (c) Trains the Q-function using the following objective

$$\hat{Q}^{\pi}_{\text{CQL}} := \arg\min_{Q} \alpha \cdot \left(\underbrace{\mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})]}_{\text{minimize Q-values}} - \underbrace{\mathbb{E}_{\mathbf{s}\sim\mathcal{D}, \mathbf{a}\sim\hat{\pi}_{\beta}(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})]}_{\text{maximize Q-values under data}}\right) + \frac{1}{2}\underbrace{\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q - \hat{\boldsymbol{B}}^{\pi}Q\right)^2\right]}_{\text{standard Bellman error}}$$

2. Conservative Q-learning for offline Rl utilize conservative Q-function
    Since the policy $\hat{\pi}^k$ is typically derived from the Q-function, we could instead choose $\mu(\mathbf{a} \mid \mathbf{s})$ to approximate the policy that would maximize the current Q-function iterate, thus giving rise to an online algorithm.

$$\min_{Q}\max_{\mu}\ \alpha\left(\overset{\text{minimize big Q-values}}{\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]}-\overset{\text{maximize data Q-values}}{\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]}\right)$$
$$+\frac{1}{2}\,\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\underset{\text{standard Bellman error}}{\left(Q(\mathbf{s},\mathbf{a})-\hat{\mathcal{B}}^{\pi_k}\hat{Q}^k(\mathbf{s},\mathbf{a})\right)^2}\right]+\mathcal{R}(\mu)\quad(\text{CQL}(\mathcal{R}))$$

- Variants of CQL

(a) $\mathcal{R}=\mathcal{H}(\mu)$

$$\alpha\mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[\log\sum_{\mathbf{a}}\exp(Q(\mathbf{s},\mathbf{a}))-\mathbb{E}_{\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})]\right]$$

(b) $\mathcal{R}(\mu)=-D_{\mathrm{KL}}(\mu,\rho)$

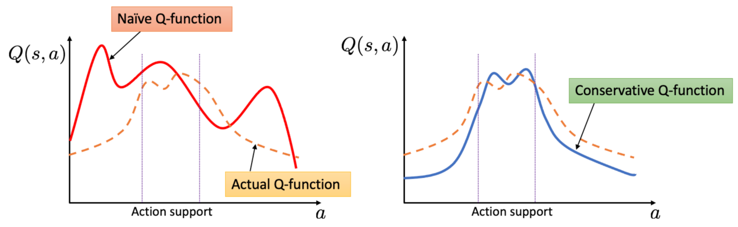
$$\alpha\mathbb{E}_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s})}\left[\mathbb{E}_{\mathbf{a}\sim\rho(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\frac{\exp(Q(\mathbf{s},\mathbf{a}))}{Z}\right]-\mathbb{E}_{\mathbf{a}\sim\pi_\beta(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})]\right]$$

(c) $\mathcal{R}(\mu)=D_f(\mu,\hat{P})$

$$\alpha\mathbb{E}_{\mathbf{s}\sim d^{\pi}\beta(\mathbf{s})}\left[\sqrt{\frac{\mathrm{var}_{\hat{P}(\mathbf{a}|\mathbf{s})}(Q(\mathbf{s},\mathbf{a}))}{d^{\pi_\beta}(s)|\mathcal{D}|}}\right]$$

1.4 Advantage

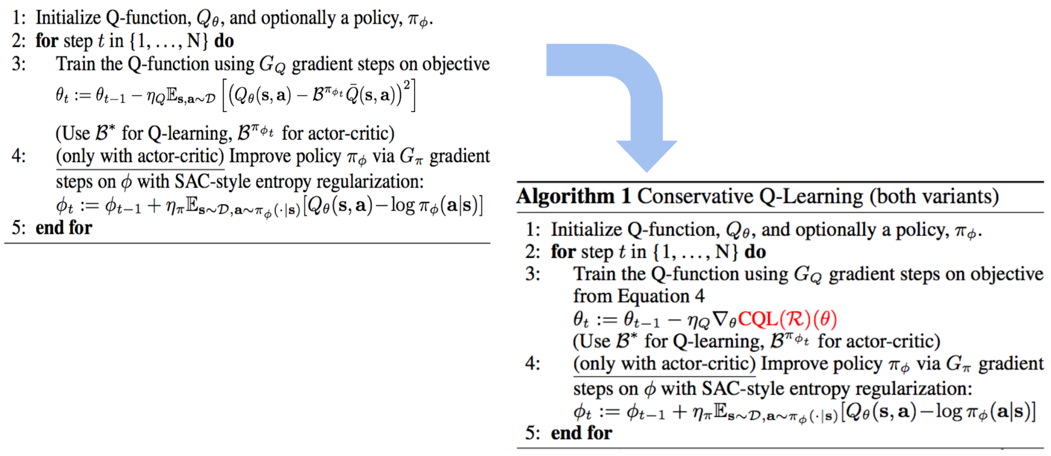
- Since the regularizer can be estimated using samples in the dataset, and so there is no need for explicit behavior policy estimation, CQL can use dataset where tha data is collected from multiple behavior policy which is impossible for previous works (BCQ, BEAR, BRAC)



- The only change introduced in CQL is a modified training objective for the Q-function as highlighted above. This makes it simple to use CQL directly on top of any standard deep Q-learning or actor-critic implementations.
- Once a conservative estimate of the policy value Q^{π}_{CQL} is obtained, CQL simply plugs this estimate into an actor-critic or Q-learning method, as shown above, and updates π towards maximizing the conservative Q-function.

1.5 Algorithm

- Algorithms can use the $\mathrm{CQL}(\mathcal{H})$ (or $\mathrm{CQL}(\mathcal{R})$ in general) objective from the CQL framework for training the Q -function Q_θ
- The actor-critic algorithm, a policy π_ϕ is trained as well. The algorithm modifies the objective for the Q-function (swaps out Bellman error with $\mathrm{CQL}(\mathcal{H})$) or $\mathrm{CQL}(\rho)$ in a standard actor-critic or Q-learning setting.



```
1: Initialize Q-function, $Q_\theta$, and optionally a policy, $\pi_\phi$.
2: for step $t$ in $\{1, \ldots, \mathrm{N}\}$ do
3:     Train the Q-function using $G_Q$ gradient steps on objective
       $\theta_t := \theta_{t-1} - \eta_Q \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\left(Q_\theta(\mathbf{s},\mathbf{a}) - \mathcal{B}^{\pi_{\phi_t}}\bar{Q}(\mathbf{s},\mathbf{a})\right)^2\right]$
       (Use $\mathcal{B}^*$ for Q-learning, $\mathcal{B}^{\pi_{\phi_t}}$ for actor-critic)
4:     (only with actor-critic) Improve policy $\pi_\phi$ via $G_\pi$ gradient
       steps on $\phi$ with SAC-style entropy regularization:
       $\phi_t := \phi_{t-1} + \eta_\pi \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi_\phi(\cdot|\mathbf{s})}[Q_\theta(\mathbf{s},\mathbf{a}) - \log \pi_\phi(\mathbf{a}|\mathbf{s})]$
5: end for
```

Algorithm 1 Conservative Q-Learning (both variants)

```
1: Initialize Q-function, $Q_\theta$, and optionally a policy, $\pi_\phi$.
2: for step $t$ in $\{1, \ldots, \mathrm{N}\}$ do
3:     Train the Q-function using $G_Q$ gradient steps on objective
       from Equation 4
       $\theta_t := \theta_{t-1} - \eta_Q \nabla_\theta \mathrm{CQL}(\mathcal{R})(\theta)$
       (Use $\mathcal{B}^*$ for Q-learning, $\mathcal{B}^{\pi_{\phi_t}}$ for actor-critic)
4:     (only with actor-critic) Improve policy $\pi_\phi$ via $G_\pi$ gradient
       steps on $\phi$ with SAC-style entropy regularization:
       $\phi_t := \phi_{t-1} + \eta_\pi \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi_\phi(\cdot|\mathbf{s})}[Q_\theta(\mathbf{s},\mathbf{a}) - \log \pi_\phi(\mathbf{a}|\mathbf{s})]$
5: end for
```

2 COMBO: Conservative Offline Model-Based Policy Optimization (2021)

2.1 Goal

1. The principal challenge in practice with prior offline model-based algorithms is the strong reliance on uncertainty quantification, which can be chalenging for complex datasets or deep neural network models
2. Develop a model-based offline RL algorithm that enables optimizing a lower bound on the policy performance, but without requiring uncertainty quantification.
3. Achieve this by extending conservative Q-learning, which does not require explicit uncertainty quantification, into the model-based setting.
4. COMBO employs an actor-critic method where the value function is learned using both the offline dataset as well as synthetically generated data from the model

2.2 Preliminaries

- $d_{\mathcal{M}}^\pi(\mathbf{s}) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathcal{P}\left(s_t = \mathbf{s} \mid \pi\right)$ = discounted state visitation distribution of a policy π
- $d_{\widehat{M}}^\pi(\mathbf{s})$ = the discounted marginal state distribution when executing π in the learned model $\widehat{\mathcal{M}}$.
- $\mathcal{P}\left(s_t = \mathrm{s} \mid \pi\right)$ = the probability of reaching state s at time t by rolling out π in $\mathcal{M}$.
- $d_{\mathcal{M}}^\pi(\mathbf{s},\mathbf{a}) := d_{\mathcal{M}}^\pi(\mathbf{s})\pi(\mathbf{a} \mid \mathbf{s})$ = the state-action visitation distribution

- the dataset $\mathcal{D}$ is sampled from $d^{\pi_\beta}(\mathbf{s},\mathbf{a}) := d^{\pi_\beta}(\mathbf{s})\pi_\beta(\mathbf{a} \mid \mathbf{s})$.
- $d(\mathbf{s},\mathbf{a})$ = sampled-based version of $d^{\pi_\beta}(\mathbf{s},\mathbf{a})$
- **Model-Free (Including CQL)**

  To capture the the long term behavior of a policy without a model, we define the action value function as

$$Q^\pi(\mathbf{s},\mathbf{a}) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t,\mathbf{a}_t) \mid \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\right]$$

  where future actions are sampled from $\pi(\cdot \mid \mathbf{s})$ and state transitions happen according to the MDP dynamics. Consider the following Bellman operator:

$$\widehat{\mathcal{B}}^\pi Q(\mathbf{s},\mathbf{a}) := r(\mathbf{s},\mathbf{a}) + \gamma Q(\mathbf{s}',\mathbf{a}')$$

  - **Policy Evaluation** : The Q function associated with the current policy π is approximated conservatively by repeating the following optimization:

$$Q^{k+1} \leftarrow \arg\min_Q \beta\left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\cdot|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[Q(\mathbf{s},\mathbf{a})]\right)$$
$$+\frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \widehat{\mathcal{B}}^\pi Q^k(\mathbf{s},\mathbf{a})\right)^2\right]$$

    where $\mu(\cdot \mid s)$ is a wide sampling distribution such as the uniform distribution over action bounds. CQL effectively penalizes the Q function at states in the dataset for actions not observed in the dataset. This enables a conservative estimation of the value function for any policy, mitigating the challenges of over-estimation bias and distribution shift.

  - **Policy Improvement**: After approximating the Q function as Q^π, the policy is improved as

$$\pi \leftarrow \arg\max_{\pi'} \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi'(\cdot|\mathbf{s})}\left[\hat{Q}^\pi(\mathbf{s},\mathbf{a})\right]$$

- **Model-Based (Concentrating MBPO)**

  MBPO follows the standard structure of actor-critic algorithms, but in each iteration uses an augmented dataset $\mathcal{D} \cup \mathcal{D}_{\text{model}}$ for policy evaluation. Here, $\mathcal{D}$ is the offline dataset and $\mathcal{D}_{\text{model}}$ is a dataset obtained by simulating the current policy using the learned dynamics model. Specifically, at each iteration, MBPO performs k -step rollouts using $\widehat{T}$ starting from state $\mathbf{s} \in \mathcal{D}$ with a particular rollout policy $\mu(\mathbf{a} \mid \mathbf{s})$, adds the model-generated data to $\mathcal{D}_{\text{model}}$, and optimizes the policy with a batch of data sampled from $\mathcal{D} \cup \mathcal{D}_{\text{model}}$ where each datapoint in the batch is drawn from $\mathcal{D}$ with probability $f \in [0,1]$ and $\mathcal{D}_{\text{model}}$ with probability $1-f$.

2.3 Algorithm

1. **Conservative Policy Evaluation** to obtain a conservative estimate of Q^π

   Penalize the Q-values(Using CQL) by repeating the following recursion:

$$\hat{Q}^{k+1} \leftarrow \arg\min_Q \beta\left(\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\rho(\mathbf{s},\mathbf{a})}[Q(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[Q(\mathbf{s},\mathbf{a})]\right)$$
$$+\frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim d_f}\left[\left(Q(\mathbf{s},\mathbf{a}) - \widehat{\mathcal{B}}^\pi \hat{Q}^k(\mathbf{s},\mathbf{a})\right)^2\right]$$

Penalize the Q-values evaluated on data drawn from a particular state-action distribution that is more likely to be out-of-support while pushing up the Q-values on state-action pairs that are trustworthy.

Choosing sampling distribution between $\rho(\mathbf{s}, \mathbf{a})$ and d_f

Model-based algorithms allow ample flexibility for these choices while providing the ability to control the bias introduced by these choices.

(a) **For** $\rho(\mathbf{s}, \mathbf{a})$ we make the following:

$$\rho(\mathbf{s}, \mathbf{a}) = d^{\pi}_{\widehat{\mathcal{M}}}(\mathbf{s})\pi(\mathbf{a} \mid \mathbf{s})$$

(b) **For** $d_f := d^{\mu}_{f}$, we make the following equation using f-interpolation between the offline dataset and synthetic rollouts from the model:

$$d^{\mu}_{f}(\mathbf{s}, \mathbf{a}) := f d(\mathbf{s}, \mathbf{a}) + (1 - f) d^{\mu}_{\widehat{\mathcal{M}}}(\mathbf{s}, \mathbf{a})$$

Under such choices of ρ and d_f, we push down (or conservatively estimate) Q-values on state-action tuples from model rollouts and push up Q-values on the real state-action pairs from the offline dataset. When updating Q-values with the Bellman backup, we use a mixture of both the model generated data and the real data, similar to Dyna. Note that in comparison to CQL and other model-free algorithms, COMBO learns the Q-function over a richer set of states beyond the states in the offline dataset. This is made possible by performing rollouts under the learned dynamics model, denoted by $d^{\mu}_{\widehat{\mathcal{M}}}(\mathbf{s}, \mathbf{a})$.

2. **Policy Improvement Using a conservative Critic** :After learning a conservative critic $\hat{Q}^{\pi}$, we improve the policy as:

$$\pi' \leftarrow \arg\max_{\pi} \mathbb{E}_{\mathbf{s}\sim\rho, \mathbf{a}\sim\pi(\cdot|\mathbf{s})} \left[\hat{Q}^{\pi}(\mathbf{s}, \mathbf{a})\right]$$

where $\rho(\mathbf{s})$ is the state marginal of $\rho(\mathbf{s}, \mathbf{a})$. When policies are parameterized with neural networks, we approximate the arg max with a few steps of gradient descent. In addition, entropy regularization can also be used to prevent the policy from becoming degenerate if required

Algorithm 1 COMBO: Conservative Model Based Offline Policy Optimization

```
Require: Offline dataset $\mathcal{D}$, rollout distribution $\mu(\cdot|\mathbf{s})$, learned
    dynamics model $\widehat{T}_\theta$, initialized policy and critic $\pi_\phi$ and $Q_\psi$.
1: Train the probabilistic dynamics model $\widehat{T}_\theta(\mathbf{s}', r|\mathbf{s}, \mathbf{a}) = \mathcal{N}(\mu_\theta(\mathbf{s}, \mathbf{a}), \Sigma_\theta(\mathbf{s}, \mathbf{a}))$ on $\mathcal{D}$.
2: Initialize the replay buffer $\mathcal{D}_{\text{model}} \leftarrow \varnothing$.
3: for $i = 1, 2, 3, \cdots,$ do
4:     Perform model rollouts by drawing samples from $\mu$ and $\widehat{T}_\theta$
       starting from states in $\mathcal{D}$. Add model rollouts to $\mathcal{D}_{\text{model}}$.
5:     Conservatively evaluate current policy by repeatedly solving
       eq. 4 to obtain $\hat{Q}^{\pi^i_\phi}_\psi$ using data sampled from $\mathcal{D} \cup \mathcal{D}_{\text{model}}$.
6:     Improve policy under state marginal of $d_f$ by solving eq. 5
       to obtain $\pi^{i+1}_\phi$.
7: end for
```

3. **Practical Implementation Details.**

    - Our practical implementation largely follows MOPO, with the key exception that we perform conservative policy evaluation as outlined in this section, rather than using uncertainty-based reward penalties.
    - Following MOPO, we represent the probabilistic dynamics model using a neural network, with parameters θ, that produces a Gaussian distribution over the next state and reward:

$$\widehat{T}_{\theta}\left(\mathbf{s}_{t+1}, r \mid \mathbf{s}, \mathbf{a}\right)=\mathcal{N}\left(\mu_{\theta}\left(\mathbf{s}_{t}, \mathbf{a}_{t}\right), \Sigma_{\theta}\left(\mathbf{s}_{t}, \mathbf{a}_{t}\right)\right)$$

    - The model is trained via maximum likelihood. We set the ratio $f = 0.5$ to have an equal split between model rollouts and data from the offline dataset.

2.4 Conclusion

- COMBO removes the need of uncertainty quantification as widely used in previous model-based offline RL works which can be challenging and un- reliable with deep neural networks
- COMBO achieves a tighter lower-bound on the true policy value compared to prior model-free offline RL methods and guarantees a safe policy improvement.
- COMBO achieves the best generalization performances in 3 tasks that require adaptation to unseen behaviors.